Mircea Paul Mureşan

Multimodal Perception and Measurement Strategies for Autonomous Systems:

Foundations and Advances

U.T.PRESS
Cluj-Napoca, 2026
ISBN 978-606-737-838-2

Mircea Paul Mureșan

Multimodal Perception and Measurement Strategies for Autonomous Systems: Foundations and Advances

U.T.PRESS
Cluj - Napoca, 2026
ISBN 978-606-737-838-2

Editura U.T.PRESS
Str. Observatorului nr. 34
400775 Cluj-Napoca
Tel.: 0264-401.999
e-mail: utpress@biblio.utcluj.ro
www.utcluj.ro/editura

Recenzia: Prof.dr.ing. Florin Oniga
Prof.dr.ing. Sergiu Nedevschi

Pregătire format electronic on-line: Gabriela Groza

ISBN 978-606-737-838-2

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my PhD thesis supervisor, Prof. Dr. Eng. Sergiu Nedevschi, for his continuous guidance, support, and mentorship throughout the many years of my academic journey. His dedication to research, high standards, and trust in my work have profoundly shaped both my scientific development and my professional path.

I am also deeply thankful to Prof. Dr. Eng. Radu Danescu for his valuable advice, encouragement, and collaboration in numerous research activities. His professionalism and openness to discussion have contributed significantly to the completion of this work.

My appreciation extends to all colleagues and friends from the Image Processing and Pattern Recognition Research Center, whose support, collaboration, and stimulating academic environment made this journey both meaningful and rewarding.

Finally, I would like to express my heartfelt gratitude to my parents, Liliana and Marinel, for their unconditional love, patience, and encouragement. Their constant support made the completion of this work possible.

“if you’re going to try, go all the
way.
otherwise, don’t even start.

….

if you’re going to try,
go all the way.
there is no other feeling like
that.
you will be alone with the gods
and the nights will flame with
fire.”

— Charles Bukowski, “Roll the Dice”

TABLE OF CONTENTS

Preface

This book is based on my PhD thesis at the Technical University of Cluj-Napoca and brings together several years of research on perception for autonomous systems and its main original contributions.

Autonomous systems are increasingly deployed in industrial, public, and everyday environments. Autonomous vehicles, aerial drones, and humanoid or mobile robots rely on robust perception capabilities to understand and interact with complex and dynamic surroundings. Within the processing pipeline of such systems, perception plays a central role, supporting higher-level modules such as planning and control and directly influencing the safety and efficiency of autonomous operation.

The methods presented in this book address key perception problems encountered in autonomous systems, including depth perception, multi-object tracking, and multi-sensor fusion. These topics are approached with an emphasis on real-world challenges such as sensor imperfections, dynamic environments, and computational constraints. The proposed approaches are designed to be applicable across multiple autonomous platforms and to operate reliably under diverse conditions.

A central focus of this work is the development of perception methods that are robust, fault-tolerant, and suitable for real-time deployment without intensive hardware acceleration techniques. Since autonomous systems are often implemented on embedded platforms with limited computational and energy resources, the approaches described in this volume aim to balance accuracy, computational complexity, memory footprint, and power consumption.

The purpose of this book is twofold: to provide a concise synthesis of representative methods from the state of the art and to present the original contributions developed during my doctoral research. These contributions extend existing approaches through novel algorithmic designs, data association strategies, and sensor fusion architectures, validated in the context of research projects focused on autonomous driving and advanced robotic systems.

I sincerely wish my readers a pleasant and enriching reading experience, and I hope that the knowledge acquired through this book will prove valuable and useful in the future.

Chapter 1. INTRODUCTION

1.1 Context

The fourth industrial revolution is rapidly extending its influence across almost all domains of human activity. Advances in hardware miniaturization, increased connectivity among digital devices, the deployment of 5G wireless networks, and progress in software and machine learning have enabled computing systems to be embedded in industrial, public, private, and domestic environments. This transformation is further characterized by the growing use of robotic and automated systems capable of self-configuration, self-optimization, automated inspection and measurement, interaction with the environment, and cooperation with human operators. In certain scenarios, these systems are also able to function in a fully autonomous manner.

One of the most challenging and actively researched topics today is the development of fully autonomous systems, such as autonomous vehicles, humanoid robots, and unmanned aerial vehicles, that can operate reliably in a wide range of environments. Such systems have the potential to enhance safety while also improving efficiency in everyday activities and in human interaction with the surrounding ecosystem. In general, an autonomous system is composed of three fundamental modules: perception, planning, and control. The perception module allows the system to build a contextual understanding of its environment and to localize itself within it. The planning module enables informed decision-making to achieve specific objectives, such as navigating from an initial position to a target destination while avoiding obstacles and hazardous situations, and while satisfying constraints related to safety, legality, and comfort. Finally, the control module is responsible for executing the planned actions.

Environmental perception represents a critical component of the processing pipeline of an autonomous system. Developing robust perception capabilities remains a significant challenge due to the inherent complexity and unpredictability of real-world environments, as well as the stochastic behaviour exhibited by the entities operating within them. A wide range of complementary sensors, mathematical methods, and machine learning techniques are employed to address the challenging problem of real-time perception while satisfying the

stringent requirements imposed by complex and dynamic real-world environments. The integration of these approaches enables perception systems to achieve both robustness and accuracy under diverse operating conditions.

For many years, vehicle manufacturers have incorporated advanced driver assistance systems (ADAS) that relied primarily on a single sensing modality, most commonly a camera, to warn drivers of potential hazards. With the transition toward autonomous driving, however, such systems require extensive sensor networks composed of multiple, complementary modalities. By exploiting redundant information from cameras, LiDAR, and RADAR sensors, together with increasingly accurate algorithms, autonomous vehicles are able to navigate unpredictable environments, even under adverse weather conditions. Companies such as Zoox, Volkswagen, and Tesla are actively developing fleets of sensor-rich autonomous vehicles intended for use as autonomous taxis, with the goals of reducing traffic congestion, minimizing accidents caused by human error, and lowering air pollution. These systems also promise improved cost efficiency by eliminating the need for human drivers and by reducing reliance on personal vehicles through increased availability.

Intelligent systems are likewise being widely adopted in industrial and medical domains. In smart manufacturing environments, robots can automate production processes, while autonomous ground vehicles perform logistics and material handling tasks. Additionally, computer vision-based non-destructive testing enables early defect detection, ensuring product quality before market release. Automated visual inspection provides objective, repeatable analyses while reducing human error. In medicine, intelligent perception systems can support clinicians in interpreting medical images, especially in repetitive tasks or fatigue-prone situations.

Across these application domains, intelligent systems demonstrate significant potential to enhance safety, efficiency, and quality of life. Achieving reliable performance in such scenarios requires the development of high-quality perception systems that are accurate, robust, and capable of operating in real time.

1.2 Motivation and Challenges

For autonomous operation, an intelligent robot must be capable of estimating distances and maintaining the identity and trajectories of

surrounding objects within the scene. This capability enables the prediction of future states and actions of traffic participants, allowing the system to make decisions that are optimal for achieving its objectives while avoiding unsafe or undesirable situations.

Intelligent systems perceive their environment through a variety of sensors, among which cameras, LiDAR, and RADAR are the most commonly employed in autonomous driving applications. Each sensing modality provides complementary information that contributes to a more comprehensive understanding of the surrounding environment.
Camera-based solutions have received significant attention due to the rich visual information they provide. Cameras are optical sensors capable of capturing, storing, and transmitting visual data, and a wide range of camera types is available depending on application requirements. Visual information may be acquired in various formats, including grayscale, color, and multispectral representations (e.g., ultraviolet, near-infrared, far-infrared, and thermal), and at different spatial resolutions. Line-scan cameras, which acquire data as a continuous line of pixels, are particularly well suited for inspection tasks involving elongated objects moving on conveyor belts. In contrast, area-scan cameras are the most widely used in machine vision applications due to their versatility, robustness, and ease of integration, as they capture entire scenes in the form of two-dimensional pixel arrays.

The camera lens is a critical component of a vision system, as it determines the sensor's field of view. Fixed focal length lenses, commonly used in robotic navigation, provide a constant angular field of view that depends on the sensor size and can range from a few degrees to nearly 180 degrees in the case of fisheye lenses, enabling surround perception. In addition to appearance information, cameras can also be used to infer depth through techniques such as monocular depth estimation or stereo reconstruction.

Stereo vision-based depth estimation employs two cameras to reconstruct the surrounding environment. This approach has garnered significant attention from both researchers and industry, as cameras—unlike many other sensors—can also provide semantic information about a scene while generally being more cost-effective. Consequently, numerous 3D reconstruction methods have been developed to meet diverse application requirements. For example, applications that demand real-time performance and minimal computational resources often rely on local stereo algorithms. While these methods offer low

latency and high frame rates, the quality of the resulting 3D reconstruction is typically limited. To enhance reconstruction quality, global and semi-global stereo algorithms have been introduced. These methods perform more sophisticated aggregation and optimization steps to produce higher-fidelity results. However, semi-global and global approaches generally require hardware acceleration to operate in real time, often leading to substantial power consumption, particularly when implemented on GPUs.

Monocular depth estimation (MDE), as its name suggests, uses a single camera to generate a 3D reconstruction of the environment. Existing approaches can be broadly categorized into structure-from-motion, handcrafted feature methods, and deep learning-based techniques. Advances in neural networks have significantly improved the reliability of monocular depth estimation for autonomous systems. MDE offers several advantages over stereo vision, primarily due to its reliance on a single camera. This reduces system cost and eliminates the need for complex calibrations or temporal alignment procedures while still achieving high-quality results. Although humans can estimate depth with one eye, relying on accumulated experience and knowledge of object shapes, scales, and perspective, current computer vision systems cannot replicate this level of understanding. As a result, some structures in the environment may be reconstructed inaccurately or omitted entirely, limiting the reliability of monocular depth perception for safety-critical applications.

LiDAR (Light Detection and Ranging) sensors constitute another important technology for acquiring three-dimensional information about the surrounding environment. LiDAR systems emit laser pulses into the scene, typically using a rotating mirror, to measure the distance between the sensor and nearby objects. The emitted pulses are reflected by surfaces in the environment, and the sensor estimates distance by measuring the time required for the reflected light to return to the receiver. The collection of these measurements forms a point cloud representation that can be exploited by mobile robots. A wide variety of laser scanners is commercially available, each presenting specific advantages and limitations. For instance, 4-layer LiDAR sensors are capable of detecting objects at relatively long distances; however, they offer a limited field of view and generate sparse point clouds. In contrast, higher-resolution LiDARs with 16, 32, or 64 layers provide denser point clouds but typically over shorter sensing ranges. A major limitation of LiDAR technology is its reduced accuracy in adverse

weather conditions such as rain, snow, or fog. Moreover, although LiDARs provide accurate distance measurements, they generate significantly sparser representations than dense stereo-based approaches and cannot directly estimate object velocity, as RADAR sensors can. In addition, LiDAR systems remain considerably more expensive than camera-based solutions.

RADAR sensors have long been employed in the automotive domain for applications including collision avoidance, blind-spot detection, and adaptive cruise control. RADAR is particularly attractive for autonomous systems because, in addition to providing three-dimensional spatial information, it can directly measure object velocity through the Doppler effect. Furthermore, RADAR sensors are capable of detecting objects under adverse weather conditions, even when targets are partially occluded or not in direct line of sight. Nevertheless, RADAR systems exhibit certain limitations, such as reduced sensitivity to objects made of porous materials like plastic or wood. Additionally, to prevent excessive reporting, RADAR processing pipelines may suppress certain detections, which can result in the omission of static objects.

It is evident that each sensing modality used for environmental perception presents specific failure modes. However, fusing redundant and complementary information from multiple sensors can yield a more robust, reliable, and accurate representation of the environment. Despite these benefits, multi-sensor fusion introduces additional challenges beyond the independent interpretation of each data source. In particular, issues related to temporal synchronization, data association, and efficient fusion under varying environmental and weather conditions must be carefully addressed.

To achieve high-quality results, all inputs to the sensor fusion module must be of the highest possible quality. Moreover, in order to ensure robustness to individual sensor failures and to enhance the overall reliability of the perception system, the fusion architecture should not be centered on any single sensing modality. To obtain stable and noise-free estimates of object positions, and to maintain object identities even under partial or full occlusions, the integration of an object tracking module is essential. By leveraging tracked object information, higher-level components within the autonomous system's processing pipeline can transform raw detections into actionable and decision-relevant information. The challenges encountered in multi-object tracking can be broadly categorized into two main groups: sensor-related issues and

data association problems. Sensor-related challenges may include, for example:

- The number of objects within the field of view (FOV) of the sensor may be unknown and in different states.
- Objects enter and leave the sensor FOV, therefore it is necessary to have good object management and object identity management.

Since object detectors are inherently imperfect, they are susceptible to two primary types of errors: missed detections, which may arise from adverse environmental conditions, object characteristics, or occlusions, and false detections, in which a reported detection does not correspond to a real object. If not properly addressed, both types of errors can lead to severe and potentially dangerous consequences. The core difficulty of the data association problem lies in the absence of explicit information linking a given detection to the real-world object that generated it. Consequently, the challenges associated with data association can be broadly divided into two main categories:

- The origin uncertainty – there is no knowledge about how the new measurements relates to previous sensor data
- Motion uncertainty- objects can have multiple motion patterns, which may change in consecutive frames.

Inadequate handling of the data association problem can result in poor tracking performance. The challenges outlined above have been extensively investigated by the research community, with numerous approaches proposed for object tracking across a wide range of applications and sensor modalities, each addressing sensor-specific limitations. Furthermore, when multiple sensors are combined to enhance tracking performance within a given domain—for example, integrating camera images with LiDAR point clouds to improve three-dimensional object tracking—errors arising from inaccurate spatial alignment, particularly in the presence of moving objects, can further degrade tracking accuracy. Existing solutions in the literature can be broadly classified into feature engineering–based approaches and data-driven methods, such as those relying on neural networks, each offering distinct advantages and limitations.

Beyond autonomous vehicles, many other industries stand to benefit from the advancements enabled by intelligent systems and the fourth industrial revolution. Domains characterized by repetitive inspection tasks, high attentional demands, and susceptibility to human error due

to subjective interpretation are particularly well suited for automation, which can reduce error rates and improve overall process reliability. This book introduces a set of novel methods designed to address the challenges discussed above, with the goal of enhancing the capabilities of intelligent systems in the context of autonomous driving and advanced driver assistance systems. The proposed approaches build upon the state of the art by leveraging techniques from machine learning, computer vision, feature engineering, and geometric modelling. A central objective of this work is to improve result quality while carefully considering computational efficiency, memory usage, and energy consumption. This focus is motivated by the fact that perception systems are typically deployed on resource-constrained computing platforms integrated into mobile, low-power devices such as vehicles, drones, and robotic systems. Moreover, excessive energy consumption can have negative environmental implications. Consequently, all methods presented in this book are designed to operate in real time, minimize resource usage, and simultaneously improve overall system performance.

Environmental perception in autonomous systems represents a particularly suitable research topic from an academic perspective. Accordingly, this book focuses on contributing to several core components that are fundamental to the perception stack of self-driving vehicles. The primary areas addressed include depth perception, data association and tracking, and sensor fusion. Existing state-of-the-art methods in these domains often exhibit limitations, either in terms of output quality or resource efficiency, with many approaches requiring substantial memory and computational power. In certain cases, power consumption is excessively high due to the deployment of large data-driven models executed on GPUs, which raises serious environmental concerns. As highlighted by a report from MIT Technology Review [1], training and deploying complex artificial intelligence models can generate environmental pollution on a scale order of magnitude greater than that produced annually by an average individual.

Additionally, several methods reported in the literature are evaluated under idealized input conditions that are rarely encountered in real-world scenarios, thereby limiting their practical applicability.

The approaches presented in this book were developed over several years of research and have been validated within multiple research projects focused on autonomous driving and advanced driver assistance systems. The proposed methods are explicitly designed for

real-world deployment and real-time operation. Furthermore, they exhibit low computational and energy requirements, making them well suited for execution on embedded platforms. Some of the contributions presented in this work combine deep learning architectures with classical computer vision techniques in order to achieve more robust, reliable, and resource-efficient perception solutions.

1.3 Book Structure

Chapters 2, 3, and 4 of this work follow a similar structural organization. Each chapter begins with an introductory section that outlines the general aspects of the topic under discussion and presents the prerequisites or preliminary remarks necessary for a clear understanding of the chapter content. This section also describes the main processing steps involved in the addressed task and highlights the key challenges associated with it. The introduction is followed by a review of state-of-the-art methods, presented in a structured and logical manner with respect to the individual stages of the corresponding processing pipeline. Subsequently, a set of enhancements beyond the reviewed literature are introduced, and each contribution is accompanied by a dedicated evaluation assessing its impact and performance.

Chapter 2 focuses on the problem of depth perception for automotive applications using dense local stereo reconstruction methods. The chapter begins by outlining the essential preconditions required for implementing a reliable stereo reconstruction system, followed by a discussion of the primary challenges encountered during the reconstruction process. The constraints that must be enforced at different stages of the pipeline are then presented. A review of representative stereo reconstruction approaches from the literature follows, emphasizing the specific innovations introduced at various stages of their processing pipelines. The chapter then details the enhanced methods corresponding to each stage of the proposed pipeline. In the cost computation stage, a novel method is presented to better capture non-fronto-parallel surfaces and mitigate perspective distortion effects through an intensity-invariant binary descriptor designed to characterize surface properties. This descriptor is incorporated into a new cost computation and aggregation strategy

aimed at generating dense disparity maps while reducing artifacts such as bloating. Additionally, a set of original geometric constraints are described that can filter invalid disparities within the cost volume. A second major enhancement of this chapter is a fault-tolerant stereo reconstruction approach capable of operating even when one camera of the stereo rig fails to acquire valid data. This approach is further enhanced through a novel fusion strategy that integrates monocular depth estimation with stereo reconstruction, guided by semantic segmentation. Finally, a two-stage refinement process is introduced to eliminate erroneous disparity values. Each of the enhanced approaches are comprehensively evaluated against state-of-the-art approaches in terms of reconstruction quality and computational performance, conducted on the KITTI benchmark.

Chapter 3 addresses the problem of multi-object tracking. Similar to the previous chapter, it begins with an introductory section that presents the main tracking-by-detection paradigms and outlines the typical processing steps involved in such approaches. The introduction also discusses the key challenges associated with multi-object tracking and introduces the mathematical foundations required to understand the proposed methods. The second section provides a detailed review of existing tracking approaches, organized according to the stages of the tracking pipeline. The third section presents improvements to what was presented from the literature. It begins with solutions for multi-object tracking of 3D LiDAR detections, where the challenges specific to tracking 3D LiDAR objects are analyzed, and a novel tracking pipeline is presented, including two original data association functions that combine features from multiple sensing modalities. Additional enhancements related to 3D LiDAR tracking are also discussed. Subsequent sections focus on 2D object tracking in a tracking-by-detection framework using thermal imagery. These include a novel gating strategy suitable for flat-field correction scenarios, as well as new data association functions that combine feature-engineered and data-driven components. The chapter further introduces an original approach for multi-object tracking, segmentation, and validation in thermal images, where objects are tracked at both instance and bounding-box levels, and semantic information is incorporated into the data association process. Multiple motion models are employed to

predict object trajectories, with the appropriate model selected using optical flow and object dimension cues. The chapter also describes the datasets developed for training the data-driven components and includes a qualitative and quantitative evaluation of all presented methods, including runtime analysis.

Chapter 4 addresses the problem of sensor fusion in autonomous systems. The introductory section presents fundamental concepts, including the JDL model and commonly used sensors in autonomous vehicle applications, and outlines the main challenges in designing robust fusion frameworks. This is followed by a review of existing fusion architectures, fusion levels, data registration methods, and data fusion strategies. The enhancements section begins with an overview of an original sensor fusion architecture and the sensors integrated within it. Spatio-temporal data alignment is then examined through a novel method for aligning sparse point clouds using ego-vehicle motion, followed by temporal fusion to densify the point cloud and improve 3D object segmentation. A new approach for spatio-temporal alignment of object detections, applicable to generic bounding-box outputs, is also presented. In addition, a custom object detection method for point clouds acquired from a 4-layer LiDAR sensor is introduced to compensate for the absence of object-level outputs from this sensor.
Subsequently, an original data association strategy for linking trifocal and reference objects and an object-level fusion architecture combining model-based and data-driven techniques are described, together with a validation mechanism to ensure semantic consistency. The chapter concludes with the presentation of a multi-sensor testing platform used for real-time testing of the implemented algorithms.

This monograph presents a unified framework for robust, real-time multimodal perception in resource-constrained autonomous systems. By integrating complementary sensors, data association mechanisms, and hybrid model-based and learning-based methods into a fault-tolerant architecture, the work advances scalable perception for autonomous driving and robotic applications. It is worth noting that the enhanced methods presented in each chapter originate from the author's doctoral research and were initially developed and validated within the PhD thesis Multimodal Measurement Approaches for Autonomous Systems.

Chapter 2. Depth Perception

2.1 Preconditions of Dense Stereo Reconstruction

2.1.1 Generalities

Projecting a three-dimensional point onto an image plane inherently results in the loss of depth information. Passive stereo vision [2] seeks to recover the distance to objects in a scene by employing multiple cameras that observe the same environment from different viewpoints. The spatial displacement between cameras in a stereo acquisition setup induces a relative shift in the positions of corresponding features in the captured images. This positional difference, referred to as disparity, can be exploited to infer the depth information lost during the projection of a 3D point onto the image plane.

Given a pair of stereo images, dense stereo reconstruction assigns a disparity value to each pixel, corresponding to its displacement in the complementary view. The core challenge of this process—rendered particularly complex by the diversity of real-world scenarios—is the reliable establishment of correspondences between pixels in the two input images. Dense stereo matching [3] produces a depth map that encodes depth information for every pixel and serves as the basis for three-dimensional reconstruction. In contrast, feature-based stereo correspondence methods aim to identify matches for a limited set of distinctive features. These approaches, commonly referred to as sparse stereo matching, result in sparse depth maps [4]. Hybrid methods that combine dense and sparse techniques have also been proposed, using sparse correspondences to initialize or stabilize dense reconstruction results [5].

Prior to the reconstruction process, the stereo image pair is rectified. Image rectification reduces the search space for stereo correspondence by constraining the matching process to a single dimension along the epipolar lines. Using the calibration parameters of both cameras, rectification transforms the image planes such that corresponding epipolar lines become collinear and aligned with one of the image axes [6]. Once corresponding points have been identified in the two images, three-dimensional reconstruction is performed through a triangulation process. A typical stereo reconstruction pipeline generally consists of four main stages, and the overall performance of the system is directly influenced by the effectiveness of each individual step. The four steps which were also presented in [7] are:

1. Computation of matching costs.
2. Matching costs spatial aggregation.
3. Disparity calculation with or without optimization.
4. Disparity map refinement.

The similarity between two image locations can be evaluated using a matching cost once the displacement between corresponding pixels has been determined. To increase robustness to noise and to reduce the influence of outliers, pixel-wise costs are typically aggregated over a rectangular support region. In some approaches, additional optimization procedures are applied in the third stage of the stereo reconstruction pipeline to improve the quality of the resulting disparity maps. Based on how this optimization stage is formulated, dense stereo reconstruction methods are commonly categorized as local, semi-global, or global approaches, each exhibiting distinct advantages and limitations. In the final stage, the estimated disparity map can be further refined by applying left–right consistency checks and by filling small regions where reconstruction has failed.

Stereo reconstruction remains one of the most actively researched topics in computer vision. Figure 2.1 illustrates an intuitive representation of binocular reconstruction, where a spatial point $P(X, Y, Z)$ is projected onto the image planes at points $P_0$and P_1. The optical centers of the two cameras, denoted by C_{source} and $C_{matched}$, are coplanar with point P. The baseline, defined as the line connecting the two optical centers, has a length denoted by b, measured in pixels, while the focal length is represented by f. Through image rectification, the

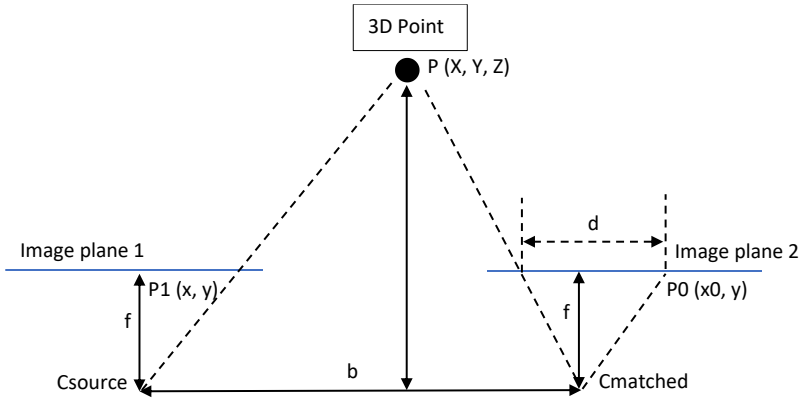

Figure 2.1. Intuitive depiction of stereo triangulation of two image pairs

correspondence search space is significantly reduced, as potential matching pixels are constrained to lie along the same epipolar line.
The source image plane is referred to as *Image Plane 1*, while the image plane in which correspondences are sought is denoted as *Image Plane 2*. Owing to the rectification process, the corresponding points $P_1$and P_2share the same vertical (y) coordinate, and the disparity value is defined as the absolute difference between their horizontal coordinates, $d = | x - x_0 |$. The resulting depth value Z is inversely proportional to the disparity, as expressed in Equation (2.1).

$$Z = \frac{b*f}{d} \tag{2.1}$$

Stereo 3D reconstruction constitutes a low-level processing function that is typically employed as a preprocessing step to enhance the robustness of higher-level perception components. Consequently, it is essential that stereo correspondence yields accurate results in real time, as its performance directly influences the effectiveness and reliability of subsequent stages within the perception pipeline of an autonomous system.

2.1.2 Calibration and Rectification

Through the calibration process, the intrinsic and extrinsic parameters of each camera in a stereo system are estimated. Intrinsic parameters describe the internal characteristics of a camera, including focal length, radial and tangential lens distortions, principal point location, and pixel size. Extrinsic parameters define the camera's position and orientation with respect to a global reference frame. This calibration procedure is typically performed once for each camera, as these parameters remain constant unless the system is physically altered or damaged.

To estimate the intrinsic parameters, a high-contrast black-and-white chessboard pattern is commonly observed from multiple viewpoints and orientations. It is considered good practice to place the pattern on a planar, rigid surface, with a matte background to avoid reflections and image saturation. Additionally, presenting the pattern along circular arcs is often recommended, as this configuration improves the accuracy of focal length estimation.

The minimum measurable distance is constrained by distortions introduced by the camera and lens system. Once the required calibration images have been acquired, specialized calibration software

can be employed to estimate the camera parameters, with several established solutions available in the literature [8, 9]. The accuracy of the estimated parameters is commonly assessed by computing the reprojection error, with values closer to zero indicating more precise calibration results.

Image rectification [10] is a critical step in binocular reconstruction, as it transforms the stereo image pair to appear as if they were captured by a canonical camera configuration. This transformation reduces the correspondence search space from two dimensions to one, thereby significantly simplifying the stereo matching process. Stereo reconstruction can be formulated either in the canonical case, which is computationally simpler, or in the non-canonical case, which requires more complex computations. The core principle of rectification is the definition of a new, canonical camera system and the estimation of its parameters while respecting the geometric constraints imposed by such a configuration.:

1. Coplanar image planes
2. Parallel optical axes
3. Co-linearity of the horizontal axes (x-axes) of the rectified image planes
4. Equal focal distances of the rectified cameras
5. Equal coordinates of the principal points in the two rectified images.

These are the required characteristics of a canonical system. Additionally, other constraints are imposed:

6. The optical centers of the cameras must remain unchanged. Otherwise, the rectification process becomes a difficult task.

A post-condition of the rectification algorithm is the following:

7. The projection of any point in the scene must have the same vertical coordinate in both images

Once the parameters defining the canonical (rectified) camera system have been computed, the images acquired by the original, unrectified setup must be transformed to appear as if they were captured using this newly defined configuration. This transformation is achieved by mapping points from each original image into the world coordinate system, expressed in normalized coordinates, and subsequently reprojecting them onto the rectified image planes.

The rectification process can be divided into two stages: an offline stage, during which the parameters required for image transformation

are computed, and an online stage, in which each image pair in a sequence is rectified using these parameters. To enable efficient real-time rectification, lookup tables are commonly employed to accelerate the transformation process.

I. Offline step

First, by taking into account constraint number 4, the new focal length can be computed as the average of the two original focal lengths, as expressed in Equation (2.2).

$$fxCan = \frac{flx+frx}{2}$$
$$fyCan = \frac{fly+fry}{2} \quad (2.2)$$

The focal length is typically estimated in metric units. To express it in pixel units, the obtained value must be divided by the physical pixel dimensions along the horizontal and vertical axes. In practice, calibration tools such as the Caltech calibration toolbox directly provide the focal length in pixel units. Furthermore, by considering constraint number 5, the coordinates of the new principal point can be computed as either the image center or as the average of the original principal point coordinates, as shown in Equation (2.3).

$$oxCan = \frac{olx+orx}{2}$$
$$oyCan = \frac{oly+ory}{2} \quad (2.3)$$

Once the new canonical parameters have been determined, the corresponding canonical intrinsic matrix can be computed, as shown in Equation (2.4).

$$ACan = \begin{bmatrix} fxCan & y & oxCan \\ 0 & fyCan & oyCan \\ 0 & 0 & 1 \end{bmatrix} \quad (2.4)$$

The third step of the rectification process involves computing the new coordinate axes. By taking into account constraints 3 and 6, the new OX axis can be defined as the baseline of the original system, that is, the line connecting the two optical centers, as expressed in Equation (2.5).

$$\overrightarrow{ox} = O_{right} - O_{left} \quad (2.5)$$

In practice, the same result can be obtained by computing the difference between the translation vectors of the two cameras, $T_{cw}^{Right} - T_{cw}^{Left}$. The new OY axis is then determined using the vector cross product defined in Equation (2.6).

$$\overrightarrow{oy} = \overrightarrow{ozOld} \; X \; \overrightarrow{ox} \tag{2.6}$$

In this formulation, oz_{old} denotes the OZ axis of the left camera, while ox represents the newly computed OX axis. To determine the new OY axis, one of the original OZ axis must be selected, after which the OY axis is defined as the direction orthogonal to the plane spanned by the chosen OZ axis and the new OX axis. In the proposed approach, the OZ axis of the left camera is selected as the reference. The new OZ axis is then computed as the direction orthogonal to the plane formed by the new OY and OX axes. This axis coincides with the OZ axis used as reference in the computation of the OY axis, as shown in Equation (2.7).

$$\overrightarrow{oz} = \overrightarrow{oy}x \; \overrightarrow{ox} \tag{2.7}$$

Each of the three axes is normalized using the norm of its corresponding vector. The fourth step of the rectification process then involves computing the new rotation matrix. The canonical rotation matrix is constructed by assembling the three previously computed and normalized axis vectors, as shown in Equation (2.8)

$$R_{wcCan=[ox,oy,oz]^T} \tag{2.8}$$

The transformation matrices are computed in accordance with the formulations presented in Equation (2.9).

$$\begin{aligned} T_{left} &= A_{can} * R_{wcCan} * \; R_{wcLeft}^{T} * \; A_{ol}^{-1} \\ T_{right} &= A_{can} * R_{wcCan} * \; R_{wcRight}^{T} * \; A_{or}^{-1} \end{aligned} \tag{2.9}$$

The rectification matrix is responsible for mapping a point from the original image to its normalized coordinates in the world reference frame and subsequently reprojecting it onto the rectified image plane. In this process, the translational component between the camera and world reference systems is neglected, as the optical centers remain unchanged. The multiplication by A^{-1} transforms the image point into normalized coordinates within the camera reference frame. The point is

then rotated into the normalized world coordinate system through multiplication with R_{wc}. To express the point in the canonical reference frame, an additional rotation using R_{wcCan} is applied. Finally, the point is reprojected onto the rectified image plane by multiplication with the canonical intrinsic matrix A_{can}. Prior to reprojection, undistortion operations are performed while the point is represented in the world coordinate system.

These undistortion steps compensate for both tangential and radial lens distortions. Tangential distortion arises when the centers of curvature of the lens elements are not perfectly aligned, while radial distortion causes image regions to appear curved or bulged. The final stage of rectification involves computing the new position of each pixel using the transformation matrix. To ensure computational efficiency, this operation is implemented using lookup tables. Since the transformed pixel coordinates are generally non-integer values, two lookup tables are maintained for each image, and the final pixel values are obtained through bilinear interpolation.

II. Online step

In the online stage, pixel intensities in the rectified image are computed using bilinear interpolation. This operation is necessary because certain pixels in the rectified image may not have a direct correspondence in the original image. The analytical formulation of this interpolation process is presented in Equation (2.10).

$$I_r(u_r, v_r) = I_o(u_o', v_o') * (1 - du_o) * (1 - dv_o) + I_o(u_o' + 1, v_o') * du_o * (1 - dv_o) + I_o(u_o', v_o' + 1) * (1 - du_o) * dv_o + I_o(u_o' + 1, v_o' + 1) * dv_o * du_o \quad (2.10)$$

An additional optimization aimed at achieving faster and more efficient processing concerns image scaling. The scaling factor is automatically incorporated into the intrinsic matrix of the original image. Conceptually, scaling the image toward the optical center corresponds to modifying the values of the canonical focal length and the principal point. Efficient scaling is achieved by multiplying the focal length and principal point coordinates by the scaling factor and updating the intrinsic matrices accordingly, no further modifications are required. These adjustments are performed during the offline stage of the rectification process.

Undistortion refers to the operation that compensates for radial and tangential distortions introduced by camera lenses. This operation can likewise be integrated into the offline stage of the rectification algorithm, specifically during the computation of the lookup tables, as defined by Equation (2.11).

$$\begin{pmatrix} x_{img} \\ y_{img} \end{pmatrix} = \begin{pmatrix} f_x(x^N * \delta x^r + \delta x^t) \\ f_y(x^N * \delta x^r + \delta x^t) \end{pmatrix} + \begin{pmatrix} o_x \\ o_y \end{pmatrix} \quad (2.11)$$

Although image rectification provides substantial benefits for stereo reconstruction, it also introduces a notable drawback. Because the intensity of each rectified pixel is obtained through bilinear interpolation of the four nearest neighboring pixels in the original image, a blurring effect is inevitably introduced. This blurring increases the likelihood of false correspondences, particularly at object boundaries, where edges become less sharp and matching ambiguity is amplified. As a result, accurately identifying the correct pixel correspondences becomes more challenging.

To mitigate this effect, a reserve of accuracy is typically employed by initially processing images at a higher resolution and subsequently downsampling them to the resolution required for stereo reconstruction. This strategy, as described in [10] and [11], helps preserve edge information and reduces the adverse impact of interpolation-induced blurring.

2.1.3 Stereo Depth Estimation Challenges

To develop an efficient and robust stereo correspondence solution, it is essential to account for the factors most likely to degrade the quality of disparity maps. The primary challenges encountered in dense stereo matching when processing real-world data are summarized below. Illustrative examples demonstrating the effects of these issues can be found in [12].

- Photometric Variations

  Lighting sources, such as sunlight or artificial illumination, can introduce complex effects, including textureless regions and color inconsistencies between the two images of a stereo pair, which hinder reliable correspondence estimation.

- Image Sensor Noise

  Due to their manufacturing characteristics, certain sensors, such as CMOS devices, can exhibit significant noise under varying

lighting conditions. This noise may affect different pixels in the two stereo images, leading to failures in the stereo matching process.

- Specularities

  Images acquired by cameras may contain highlights and reflections, commonly referred to as specularities, which are often disregarded by many computer vision methods. Such specular effects can introduce textureless regions and lead to incorrect matches, ultimately resulting in erroneous correspondences in stereo reconstruction algorithms.

- Foreshortening

  This effect arises when objects are observed at steep viewing angles. Because the two cameras of a stereo rig capture the scene from slightly different viewpoints, the projection of a surface may appear smaller in one image than in the other. This discrepancy can impair stereo correspondence, as matching algorithms typically assume blocks of identical size in both images, implicitly presuming similar object appearances across views.

- Perspective distortions and slanted surfaces

  Most stereo approaches assume that surfaces are fronto-parallel, an assumption that is frequently violated in practice, as disparities within a matching window are often non-uniform. Ideally, the matching process should also account for perspective effects. However, this remains a challenging problem, since determining whether a region is fronto-parallel and depth-homogeneous would require prior knowledge of the true depth at every point within the matching window.

- Repetitive structures and un-textured regions

  In both situations, the stereo algorithm fails to recover the correct disparity because it is confounded by identical matching costs obtained for multiple disparity hypotheses. Such ambiguities can be detected by analyzing the cost volume and identifying periodic patterns in the matching costs.

- Transparency

  This effect leads to certain features being more prominently highlighted in one image than in the other.

- Occlusions

  Employing multiple cameras for scene reconstruction generally leads to improved reconstruction quality. However, a notable

limitation is the presence of pixels that are visible in one image but not in the other, as the cameras do not observe identical portions of the scene.

If these cases are not properly addressed, they can lead to failures in stereo correspondence and prevent accurate scene reconstruction.

2.1.4 Stereo Depth Estimation Constraints

To reduce the search space for stereo correspondence and to mitigate potential sources of error during the matching process, a set of constraints is commonly imposed [13]. These constraints are fundamental to binocular reconstruction and are briefly outlined below.

The **epipolar constraint** states that a point in the source image must lie on the same epipolar line in the matched image. In the case of parallel cameras, as illustrated in Figure 1, rectification ensures that epipolar lines are horizontal and coincide with image rows, each containing the corresponding feature point. As a result, the correspondence search space is reduced from two dimensions to one.

The **geometric constraint** pertains to the relative spatial arrangement of corresponding pixels. Specifically, a corresponding pixel in the right image must be located to the left of its counterpart in the left image. Consequently, only pixels in the right image with an x-coordinate less than or equal to that of the pixel in the left image are considered valid candidates for matching.

The **uniqueness constraint** enforces a one-to-one correspondence between pixels in the left and right images. Formally, if pixels $k$and $l$are corresponding points in the left and right images, respectively, then for any pixel $p < k$ in the left image, its corresponding pixel $q$in the right image must satisfy $q < l$.

Finally, the **continuity constraint** reflects the assumption that disparity values typically vary smoothly across the image, with abrupt changes occurring primarily at object boundaries. Thus, the disparity of a matched pixel and that of its immediate neighbors are expected to be similar unless the pixels lie near an edge. For instance, given closely spaced matched pixel pairs (k, l), (i, j), and (m, n), with $k < i < m$ and $l < j < n$, the disparity $d(i, j)$ should not differ significantly from $d(k, l)$ or $d(m, n)$.

The **consistency constraint** is particularly useful for detecting occlusions—points visible in one image but not in the other—as well as incorrectly established correspondences. It states that if a disparity

value is obtained for a point during left-to-right matching, the same value should be recovered when performing right-to-left matching for that point. Disparities that do not satisfy this condition are considered unreliable and are typically discarded.

The **disparity limitation constraint** restricts the search for correspondences to a predefined disparity range, thereby reducing the computational search space and improving efficiency.

The **similarity constraint** assumes that corresponding pixels in the left and right images exhibit identical or sufficiently similar intensity values within a defined threshold. When window-based correlation methods are employed, this constraint implies that the corresponding windows should exhibit strong similarity.

Finally, the **smoothness constraint** reflects the assumption that disparity values vary gradually in regions lacking strong texture, with abrupt changes being relatively rare.

Additional details regarding stereo constraints can be found in [13, 14, 15]. Taking all these constraints into account is essential for accurate environment reconstruction using stereo vision. They can be modelled in various ways and play a crucial role in reducing errors and ambiguities in the resulting disparity maps.

2.2 Review of Dense Stereo Reconstruction Methods

2.2.1 Matching Cost Computation

In the context of stereo matching, cost functions are designed to quantify the degree of similarity between pairs of pixels. All stereo correspondence methods rely on some form of matching cost to evaluate the likelihood that two pixels correspond to the same scene point. In certain approaches, multiple cost functions are combined in order to improve robustness and matching reliability [17]. The simplest cost formulations assume constant intensity values for corresponding pixels, implicitly relying on radiometric consistency. Such assumptions are valid only under ideal conditions, typically in well-textured regions. As noted in [18], the challenges associated with designing robust matching cost functions stem from two independent factors:

1. Data uncertainty, caused by miss-calibrated cameras or images that contain objects with low texture or reduced signal to noise ratio.

2. The structural ambiguity, which is caused by the presence of repetitive patterns and the inability of the cameras to capture the entire light spectrum [19]

The difficulty of designing robust matching cost functions increases significantly when dealing with surfaces that exhibit challenging real-world characteristics [20], among which we specifically mention the following:

1. Bi-directional reflectance which can cause matching pixels to have different colors in each frame.
2. Occlusions of objects caused by the different points of view of the cameras from the stereo system. The depth assignment to occluded pixels must rely on more mechanism not just the matching cost function.

To determine the correct disparity value, the first step consists of generating a set of potential matches between a target pixel in one image and multiple candidate pixels in the second image that lie along the same scanline, or epipolar line [21]. This process results in a three-dimensional structure of size height × width × maxDisparities, commonly referred to as the cost volume, as it encodes the matching costs between pixels in the two images. The height and width correspond to the spatial dimensions of the input images, while maxDisparities denotes the predefined range of disparity hypotheses considered during matching. Each value within the cost volume represents the similarity cost between a pixel pair according to a chosen metric, and these values vary depending on the employed matching function. A central challenge in stereo correspondence is the design of a matching cost function that reliably assigns the optimal cost to the true pixel correspondence.

2.2.1.1 Intensity based descriptors

Intensity-based metrics operate under the assumption that corresponding pixels in the left and right images exhibit very similar intensity values. One of the simplest and most commonly used such metrics is the absolute difference (AD) [22], [23], defined in Equation (2.12). Under this assumption, correctly matched pixel pairs are expected to yield very small absolute difference values.

$$C_{AD}(p, d) = |I_L(p) - I_R(p - d)| \tag{2.12}$$

In Equation (2.12), p denotes the pixel location, d represents the disparity value, which ranges from 0 to *maxDisparities*, and I denotes the pixel intensity in the left and right images. Since matching based on a single pixel is often prone to errors, an improvement consists of considering a local window centered around the target pixel and performing the matching using the intensity values within this support region [24]. Typically, the window size is small, such as 3×3 or 5×5 pixels, with the pixel of interest located at the center.

The absolute differences computed for all pixels within the window are then summed to obtain the final matching cost. This approach defines the sum of absolute differences (SAD) metric [25], [26], which is analytically expressed in Equation (2.13). In this formulation, the symbol q denotes the pixels belonging to the neighborhood N_p.

$$C_{SAD}(p,d) = \sum_{q \in N_p} |I_L(q) - I_R(q-d)| \tag{2.13}$$

A closely related approach is the sum of squared differences (SSD) [27], shown in Equation (2.14), in which the absolute difference is replaced by the squared intensity difference.

$$C_{SSD}(p,d) = \sum_{q \in N_p} \big(I_L(q) - I_R(q-d)\big)^2 \tag{2.14}$$

The normalized cross-correlation (NCC) is another intensity-based matching metric that incorporates the standard deviation of the intensities within the matching window, thereby accounting for potential Gaussian noise effects [28]. The analytical formulation of this metric is provided in Equation (2.15).

$$C_{NCC}(p,d) = \frac{\sum_{q \in N_p} I_L(q) I_R(q-d)}{\sqrt{\sum_{q \in N_p} [I_L(q)^2] \sum_{q \in N_p} [I_R(q-d)^2]}} \tag{2.15}$$

2.2.1.2 Non-Parametric descriptors

Although intensity-based descriptors are simple to understand and straightforward to implement, they generally perform poorly in real-world environments and are more suitable for controlled settings [22]. Radiometric variations—such as those caused by non-diffuse surfaces, textureless regions, or illumination changes due to sunlight, reflections, or glare—can introduce significant brightness inconsistencies between stereo image pairs. These effects often lead intensity-based descriptors to produce inaccurate pixel correspondences.

To address these limitations, a class of non-parametric descriptors has been introduced, which derives representations of image patches that are invariant to linear intensity changes [29]. Some of these non-parametric descriptors can be obtained by modifying previously discussed intensity-based measures. For instance, subtracting the mean intensity of the matching window from the individual pixel values transforms the cost functions defined in Equations (2.13) and (2.14) into the formulations presented in Equations (2.16) and (2.17), respectively.

$$C_{ZSAD}(p,d) = \sum_{q\in N_p} |I_L(q) - M_L - I_R(q-d) + M_R| \tag{2.16}$$

$$C_{ZSSD}(p,d) = \sum_{q\in N_p} (I_L(q) - M_L - I_R(q-d) + M_R)^2 \tag{2.17}$$

The terms M_L and M_R are the mean values in the patch window. The NCC descriptor also has a non-parametric variant presented in (2.18).

$$C_{ZNCC}(p,d) = \frac{\sum_{q\in N_p} I_L(q) I_R(q-d)}{\sqrt{\sum_{q\in N_p}[(I_L(q)-M_L)^2] \sum_{q\in N_p}[(I_R(q-d)-M_R)^2]}} \tag{2.18}$$

Another widely used non-parametric descriptor is the Rank Transform (RT). This matching cost formulation has gained considerable attention due to its high computational efficiency and its suitability for parallel implementation on dedicated hardware platforms [30]. The Rank Transform computes the matching cost by counting the number of pixels within a support window whose intensity values are lower than that of the central pixel. This operation is performed independently for corresponding patches in the left and right images, and the resulting values are then subtracted. A smaller difference indicates a higher degree of similarity between the two patches. The analytical formulation of the Rank Transform is provided in Equation (2.19), where *CARD* denotes the cardinality operator, which counts the number of elements in a set.

$$C_{RT}(p,d) = CARD\big[q \in N_p \big| I_L(q) < I_L(p)\big] - CARD[q \in N_p | I_R(q) - I_R(p)] \tag{2.19}$$

Other matching cost formulations with more complex computational expressions have also been proposed in the literature; however, due to their higher computational cost, they are less commonly used in practice. Notable examples include the Birchfield–Tomasi (BT) metric

[31], which provides sub-pixel accuracy in correspondence estimation, and the Mutual Information (MI)–based cost function [22], which additionally accounts for the entropy of the image intensities.

2.1.2.3 Binary Descriptors

These descriptors have attracted significant interest due to their computational simplicity, suitability for implementation on a wide range of hardware platforms, and invariance to additive and multiplicative intensity variations. Moreover, the computational cost associated with extracting such descriptors is relatively low. The Census Transform (CT) [29] is a representative binary descriptor that captures local structural information by comparing pixel intensities within a matching window and encoding the comparison outcomes into a binary string. The step function employed in this comparison process is defined in Equation (2.20), where p_1 and p_2 denote two pixels within the patch used for correspondence estimation. Typically, one of these pixels is chosen as the center of the matching window.

$$\xi(p1, p2) = \begin{cases} 0, p1 \leq p2 \\ 1, p1 > p2 \end{cases} \tag{2.20}$$

The final formulation for computing the Census descriptor of an image patch is given in Equation (2.21), where the operator $\otimes$ denotes concatenation and N_p is the neighborhood of the central pixel p.

$$CT(p) = \otimes\, q \in N_p \xi(p, q) \tag{2.21}$$

Numerous variants of the Census Transform have been proposed to address different application scenarios. For instance, if the center pixel is corrupted by noise in one of the images, the resulting binary descriptor may become unreliable for correspondence estimation. To mitigate this issue, an alternative formulation compares pairs of pixels that are symmetric with respect to the center of the matching window [32]. In this manner, even if one pixel is affected by noise, the overall binary descriptor can remain informative. The analytical expression of the center-symmetric Census Transform is provided in Equation (2.22), where o_i denotes the coordinates of the pixels within the matching window.

$$CT_{csym}(p) = \otimes\, o_i \in N_p \xi(p + o_i, p - o_i) \tag{2.22}$$

To reduce computational complexity when performing dense Census Transform calculations and to mitigate the influence of noisy pixels, sparse Census variants have been introduced. These variants construct the final binary descriptor by considering only a subset of pixels within the matching window. A general analytical formulation of sparse Census descriptors is provided in Equation (2.23), and examples of 3D reconstruction approaches employing such descriptors can be found in [33] and [34]. In Equation (2.23), N denotes the total number of pixels in the matching window, while b_i is a binary indicator that takes the value 1 if the corresponding pixel is included in the descriptor and 0 otherwise.

$$CT_{sparse} = \sum_{i=0}^{N-1} b_i 2^i \tag{2.23}$$

A comprehensive analysis of the Census Transform descriptor is presented in [35], where the trade-off between matching window size and accuracy is systematically investigated. The authors evaluate various Census-based descriptors under both noisy and ideal conditions, considering dense and sparse Census masks. Their results indicate that the highest matching accuracy is achieved with Census window sizes of 16×16 and 24×24; however, the computational cost increases exponentially with window size. In contrast, sparse Census variants, while slightly suboptimal in terms of accuracy, offer a favorable balance between computational efficiency and performance, even when larger windows are employed.

Further insights are provided in [36], where the influence of individual pixels on the overall matching score is examined. This study demonstrates that, in sparse Census masks, pixels located near the center of the matching window tend to contribute less to matching performance than those positioned closer to the window boundaries. Moreover, the authors report that center-symmetric Census masks outperform traditional Census descriptors in both speed and accuracy. Similar conclusions are drawn in additional studies [37], which also identify the center-symmetric Census as a generally superior choice for stereo matching applications. Beyond Census-based methods, several other binary descriptors have been proposed to address specific challenges. For example, the ORB descriptor introduced in [38] provides rotation invariance, while DAISY [39] is designed for stereo matching scenarios involving large baselines.

The final matching cost between descriptors computed in the left and right images is typically evaluated using the Hamming distance, which

measures the number of differing bit positions between two binary strings. The formulation for computing the matching cost for a pixel at position p using the Hamming distance is given in Equation (2.24). In this expression, $\oplus$ denotes the XOR operation, d represents the disparity value, CT refers to the Census Transform applied to the left and right images, and $Count$ denotes a function that counts the number of ones resulting from the XOR operation.

$$C(p) = Count(CT_L(p) \oplus CT_R(p + d)) \tag{2.24}$$

2.1.2.4 Learning-based Descriptors

With the increasing popularity of neural networks, a wide range of data-driven techniques have been developed to learn highly discriminative features for stereo correspondence tasks [40]. One of the earliest deep learning–based solutions for feature extraction is presented in [41], where Siamese neural network architectures are employed to learn feature representations and compute similarity measures between image patches. In this framework, similar features are assigned low matching costs, while dissimilar features incur higher costs. In a related study, [42] investigates several deep learning architectures for image patch comparison and reports that concatenating left and right image patches prior to feature extraction leads to superior matching accuracy. However, this strategy also results in a substantial increase in computational complexity and runtime.

Further efforts to improve efficiency are described in [43], where the authors propose multiple optimizations for the cost computation stage. Specifically, they learn a probability distribution over the entire disparity range, enabling the model to capture correlations between different disparity hypotheses. Additionally, the two branches of the neural network are merged using a dot-product layer applied to the convolutional feature maps, reducing computational overhead while maintaining competitive performance. Beyond neural network–centric approaches, evolutionary methods such as genetic algorithms have also been explored for identifying optimal stereo descriptors [44], demonstrating alternative strategies for descriptor optimization.

Overall, learning-based descriptor methods often provide improved matching accuracy compared to traditional feature-engineered approaches in stereo correspondence. Nevertheless, these benefits come at the cost of increased computational complexity, longer

execution times, and a strong dependence on hardware acceleration, typically requiring GPUs to achieve real-time performance. Consequently, these methods tend to consume significant amounts of power. The trade-off between accuracy and computational efficiency remains a central challenge in stereo matching. In autonomous driving applications, the perception pipeline includes multiple processing modules, many of which already rely on GPU resources to ensure acceptable system-wide performance. In such resource-constrained edge computing environments, it is often impractical to further burden the GPU for marginal accuracy gains. Instead, improving feature-engineered descriptors that can run efficiently on CPUs offers a more balanced solution, enabling robust performance while maintaining reasonable computational and energy requirements.

2.2.2 Spatial Aggregation of Matching Costs

Cost aggregation represents the second stage of the stereo reconstruction pipeline. Spatial aggregation involves combining the matching costs stored in the cost volume generated in the previous step in order to reduce noise and improve the reliability of stereo correspondence. Aggregation can be performed either through spatial aggregation or weighted aggregation, and these two strategies may be applied independently or jointly in a combined formulation. The analytical expression corresponding to this aggregation step is given in Equation (2.25).

$$C^W(d) = Z^W \sum_{p \in W} w^p C(p, d) \tag{2.25}$$

We denote by $C^W(d)$ the aggregated cost corresponding to disparity level d over an aggregation window W. The term Z^W represents a normalization factor, while w^p denotes an optional weighting term applied during aggregation; in the case of unweighted aggregation, $w^p = 1$. The term $C(p, d)$ refers to the matching cost of pixel p at disparity level d.

According to the study in [45], a favorable trade-off between computational efficiency and matching accuracy is achieved when using an aggregation window of size 5×5, and it is further recommended that the window size should not exceed 7×7. More advanced aggregation strategies have been proposed to further improve robustness. For instance, the approach described in [46] employs a cross-based aggregation scheme that includes neighboring pixels as

long as their intensities remain sufficiently similar. In [47], the authors introduce a more sophisticated aggregation method capable of enabling local block-matching stereo algorithms to achieve performance comparable to semi-global approaches. This method leverages the implicit assumption in stereo correspondence that all pixels within a matching block share the same disparity—an assumption that is frequently violated on slanted surfaces or under perspective distortion. To address this issue, the aggregation process selectively incorporates the most reliable costs from a larger support region, while imposing a small penalty when the optimal disparity originates from a neighboring, non-fronto-parallel region. The analytical formulation of this aggregation strategy is presented in Equation (2.26).

$$C_d = \sum_i \max_{s \in D_s} P(s,d) c_{i,s} \tag{2.26}$$

Einecke *et al.* extend this idea further and, in [48], propose an aggregation scheme that combines multiple support regions of varying shapes and sizes in order to better capture the disparities of thin structures, such as poles. By employing this strategy, the proposed method mitigates noise typically introduced by small aggregation windows while simultaneously reducing the bloating (or fattening) effect often associated with large support regions.

In a related approach presented in [49], aggregation is performed using a weighted scheme, where disparity costs are assigned weights based on their spatial distance from the center of the aggregation window. This weighting strategy emphasizes contributions from pixels closer to the reference location, thereby improving robustness.

Given the widespread adoption of deep learning techniques, several studies have explored the use of convolutional neural networks (CNNs) for the aggregation stage of the stereo pipeline. One such method is described in [50], where edges are detected at multiple scales using CNN-extracted features and subsequently employed to define adaptive aggregation window boundaries.

2.2.3 Disparity Calculation With or Without Optimization

Many stereo correspondence constraints are frequently violated in challenging real-world scenarios. For instance, under the uniqueness constraint, an ideal stereo matching solution assumes a one-to-one correspondence between pixels in the left and right images. In practice,

this assumption is often invalid due to occlusions, noise, and ambiguous matches. To address such limitations, an additional stage—referred to as optimization—is incorporated into the stereo processing pipeline [51]. In this stage, the matching problem is formulated as an energy minimization task, where correct disparity assignments correspond to low energy values, while disparities arising from violated constraints are penalized with high energy. However, this formulation has been shown to be NP-complete [52], rendering it impractical for real-time applications. To make the optimization step feasible for intelligent systems, the problem is reformulated as an energy function that includes a regularization term, as expressed in Equation (2.27).

$$E(D) = E_{Data}(D) + E_{Smooth}(D) \tag{2.27}$$

The term $E_{\text{Data}}(D)$ denotes the data fidelity component and comprises the aggregated costs obtained from the cost volume following the aggregation stage. The term $E_{\text{Smooth}}(D)$ represents the smoothness component, which is incorporated to enforce the smoothness constraint in the reconstructed three-dimensional scene. For clarity and ease of interpretation in the subsequent discussion, Equation (2.27) can be reformulated as shown in Equation (2.28), where p and q denote neighboring disparity positions, C represents the cost volume, and $V_{p,q}$ defines the smoothness penalty between two adjacent positions.

$$E(D) = \sum_{p \in I} C_p\left(d_p\right) + \sum_{(p,q) \in N} V_{p,q}(d_p, d_q) \tag{2.28}$$

To address the second term of Equation (2.28), which poses significant computational challenges, several optimization techniques have been proposed, including graph cuts [51][52], belief propagation [53], and total variation methods [54]. Among these approaches, Semi-Global Matching (SGM) [55] has gained the greatest popularity due to its ability to produce high-quality disparity estimates in real time. This balance between accuracy and computational efficiency makes SGM particularly well suited for intelligent systems applications that require reliable three-dimensional reconstruction.

Semi-Global Matching

The Semi-Global Matching (SGM) method [55] was introduced to simplify the energy formulation and thereby reduce the computational complexity of the stereo optimization problem. SGM belongs to the class of global optimization approaches, as it enforces smoothness

constraints across the image, while remaining computationally tractable. The method achieves this balance by approximating the two-dimensional energy function through the aggregation of multiple one-dimensional paths, typically along 4, 8, or 16 directions.

The SGM energy formulation comprises a data term, a penalty term designed to handle slanted surfaces and perspective distortions, and a smoothness term that accounts for depth discontinuities. The complete energy function used in SGM is presented in Equation (2.29).

$$E(D) = \sum_p C(p, D_p) + \sum_{q \in N_p} P1 * T[|D_p - D_q| = 1] + \sum_{q \in N_p} P2 * T[|D_p - D_q| > 1] \quad (2.29)$$

In this formulation, D denotes the set of disparity values, C represents the cost volume, and N_p defines the neighborhood of point p. The function $T(\cdot)$ maps the truth value of the expression within the brackets to either 1 or 0. The variables D_p and D_q correspond to the selected disparities at positions p and q, respectively. The constant P_1 represents a penalty applied to neighboring pixels when the disparity change is small (i.e., a difference of one pixel), while the final term imposes a larger penalty for more significant disparity variations. For practical implementation, this energy formulation can be rewritten as shown in Equation (2.30).

$$L_r(p, d) = C(p, d) + \min(L_r(p - r, d), L_r(p - r, d - 1) + P1, L_r(p - r, d + 1) + P1, \min L_r(p - r, i) + P2) \quad (2.30)$$

In this formulation, $C(p, d)$ denotes the cost volume value associated with the pixel at position $p$and disparity d, while r specifies the direction of a one-dimensional aggregation path. The final optimized cost is obtained by combining the costs accumulated along all paths defined by r, as expressed in Equation (2.31).

$$C_{fin}(p, d) = \sum_r L_r(p, d) \quad (2.31)$$

One of the most challenging aspects of Semi-Global Matching is the selection of the penalty parameters P_1 and P_2. The parameter P_1 penalizes small disparity variations, which are typically associated with slanted surfaces or perspective effects, whereas P_2 penalizes large disparity changes that usually occur at depth discontinuities. In [55],

the authors suggest that these penalties may be selected empirically, with the constraint that P_2 must be significantly larger than P_1.

Further analysis is provided in [56], where four distinct strategies for choosing penalty values are examined. The authors conclude that the optimal penalty selection depends on the matching cost function employed, and they associate each of the proposed strategies with specific matching descriptors. Equation (2.32) presents the analytical formulation of an adaptive P_2 penalty that accounts for local image gradients, where P_2' is an empirically chosen constant. This gradient-adaptive formulation of the P_2penalty was first introduced in [56].

$$P2 = \frac{P_2'}{|I_{bp} - I_{bq}|} \tag{2.32}$$

A commonly adopted heuristic in penalty selection is to set the second penalty P_2 to approximately the square or at least twice the value of P_1. In addition to heuristic strategies, learning-based approaches have also been explored to automatically determine optimal penalty values for SGM; an example of such a method is presented in [57].

The primary limitation of SGM is that achieving real-time performance—typically exceeding 10 frames per second—requires substantial hardware optimization. Consequently, numerous SGM implementations have been developed and optimized for specific platforms, including CPUs [58], GPUs [59], and FPGAs [30]. While the incorporation of optimization techniques significantly improves stereo matching results, it does not entirely resolve the challenges related to reconstruction quality and accuracy in three-dimensional scene estimation.

Winner Takes All

The winner-takes-all (WTA) strategy selects a single disparity value for each pixel position by choosing the minimum cost from the constructed cost volume. Originally introduced by Pollard [60], this method relies heavily on the quality of the underlying matching cost function. In textureless regions, WTA often produces erroneous results due to the high ambiguity of matching costs. Despite these limitations, the approach remains widely used, particularly in local and semi-global stereo methods, where it serves to generate an initial disparity estimate. Its continued popularity is largely due to its simplicity of implementation and high computational efficiency, which is largely

independent of image size. Alternative strategies for disparity selection have also been proposed. For instance, the *loser-takes-nothing* approach introduced by Li *et al.* [61] aims to iteratively eliminate excessively large matching costs arising from poor correspondences. This method employs a steepest-descent optimization scheme to determine the winning disparity; however, it is computationally expensive and therefore inefficient in practice. In standard WTA formulations, the optimal disparity for each pixel is commonly obtained using the argmin operator, as expressed in Equation (2.33).

$$disp(p) = \underset{d}{\operatorname{argmin}}\, C_{fin}(p, d) \tag{2.33}$$

After computing disparity maps using both the left and the right images as reference views, a left–right consistency check can be applied to validate the reliability of the estimated disparity values. It is worth noting that this consistency verification can also be performed directly on the cost volume, without explicitly generating both disparity maps. This optimized approach can improve computational efficiency, as it eliminates the need to execute the stereo algorithm twice and to compute two separate disparity maps.

2.2.4 Disparity Map Refinement

According to [7], the final major stage in the stereo matching pipeline is the refinement step. During this phase, the initially generated disparity map is re-evaluated, and disparity values deemed unreliable are either removed or, when possible, replaced with estimates of higher confidence. Common refinement strategies include filtering the disparity map using edge-aware techniques, such as median [62], bilateral [63], or guided filters [64], which aim to preserve depth discontinuities while reducing noise.

Alternative refinement approaches enforce consistency constraints directly on the disparity map. For instance, the method proposed in [65] propagates disparity values along line segments with similar color characteristics. In this framework, highly reliable disparity values serve as seeds and are propagated within these segments, thereby imposing a smoothness constraint across neighboring pixels. The resulting disparity map is refined using small-kernel bilateral filtering to prevent streaking artifacts. However, relying on a limited set of seed pixels may lead to information loss. To address this, anisotropic image processing techniques have been proposed. In [66], neighboring pixels around an

unreliable pixel are weighted according to intensity similarity, and their disparity values are combined to replace the erroneous estimate. Learning-based methods have also been explored for disparity refinement. In [67], a multi-scale neural network processes stereo image pairs at different resolutions, reducing inference complexity and improving runtime performance. Other learning-based approaches aim to replicate the behavior of classical filtering techniques [68]. For example, the method presented in [69] models the effect of bilateral filtering through a learning framework, with the authors reporting improved accuracy compared to traditional filter-based refinement methods.

2.2.5 Fault Tolerance Methods for 3D Reconstruction Solutions

A common challenge in stereo vision systems arises when one of the frames is incorrectly acquired or becomes corrupted. Several studies have addressed error mitigation under such conditions. A representative approach is presented in [70], where missing stereoscopic video frames are recovered through a combination of temporal change detection between successive left-view images, disparity estimation, and frame-difference projection. This disparity-based frame-difference projection method demonstrates robust performance in scenarios involving frame drops.

In related work, Chung *et al.* [71] exploit motion vectors extracted from color images to conceal missing right-view frames. Their approach employs 3D image warping to establish inter-view correspondences and reconstructs the missing image using both motion vectors and intensity differences between matched pixels. Furthermore, Miclea *et al.* [72] identify three critical failure points within the stereo reconstruction pipeline and propose a neural network–based solution to address them. The proposed model is formulated as a regression task and leverages previously acquired RGB images, disparity maps, and semantic segmentation outputs to generate an accurate final disparity map.

2.3 Review of Monocular Depth Estimation Approaches

Acquiring images with a single camera inherently results in the loss of three-dimensional information, making depth estimation from a single image a highly challenging task. Early approaches to monocular depth perception relied on exploiting visual cues such as known object

dimensions, texture variations, and defocus effects. In a seminal study, Horn *et al.* [73] utilized color gradients in acquired images to infer depth information. Following a feature-engineering paradigm, Kong [74] formulated monocular depth estimation as an intrinsic image decomposition problem. In this work, object contours were inferred from shadows and albedo, and temporal consistency was enforced by linking results across consecutive frames using optical flow.

In [75], the author proposed a learning-based monocular depth estimation approach that incorporates prior knowledge about the sizes of known objects present in each frame. The image structure is decomposed into a set of features obtained through transformations such as Fourier and wavelet analysis, which are then used to infer absolute scene depth. Given that object recognition in unconstrained environments is prone to errors, the authors in [76] combined texture features, geometric context, and motion-boundary-based monocular cues with spatio-temporal co-planarity constraints to improve depth inference from monocular videos. In a related approach, [77] classifies scene elements into four categories—sky, ground, planar, and cubic—and assigns approximate depth values to objects based on their class membership.

Compared to feature-engineered methods, deep learning approaches have achieved substantially higher accuracy in dense monocular depth estimation. Laina *et al.* [78] proposed an encoder–decoder convolutional neural network for depth estimation from a single image, where the encoder is based on a modified ResNet-50 architecture with the fully connected and final pooling layers removed, and the decoder up samples the output using convolutional layers. As network depth is strongly correlated with depth estimation accuracy—owing to the ability of larger receptive fields to capture broader contextual information—many monocular depth estimation models have been designed with over 100 layers [80, 81]. To further enhance depth prediction quality, Mancini *et al.* [82] introduced a fusion strategy that concatenates RGB images with optical flow information, enabling the network to learn depth cues from motion-aware inputs.

Building upon prior work, Alhashim and Wonka [83] proposed a tightly connected encoder–decoder architecture characterized by a lightweight decoder composed of only two convolutional layers and bilinear upsampling. Despite its simplicity, this approach achieves competitive performance on established benchmarks such as KITTI. Finally, [84]

introduced a semi-supervised monocular depth estimation framework consisting of two networks: a pose estimation network and a symmetric depth estimation network. Both networks process RGB and semantic images to produce intermediate depth estimates, which are subsequently fused to generate the final depth map. A comprehensive survey of monocular depth estimation methods, including the approaches discussed above, can be found in [85].

Although the primary focus of this work is not the advancement of monocular depth estimation techniques, this brief review of the state of the art is included because a monocular depth estimation method [86] is employed to enhance the accuracy of local stereo approaches and to enable the development of a fault-tolerant stereo reconstruction system.

2.4 Advanced Depth Perception Methods

2.4.1 Improving Local Block Matching Algorithms, the Slanted Block Matching Approach

One of the primary reasons block-matching stereo methods fail to accurately reconstruct certain surfaces lies in the implicit assumption that all features within a matching block share the same three-dimensional depth. This assumption is frequently violated in the presence of slanted or non-fronto-parallel surfaces. To address this limitation, an enhanced block-matching approach, originally presented in [224], is discussed in this section. The presented method, referred to as Slanted Block Matching (SBM), captures surface orientation by warping the matching descriptor block to better align with the underlying surface geometry. This operation is implemented efficiently through the use of lookup tables, enabling fast execution.

Following the matching stage, the resulting costs are aggregated using support regions of varying shapes and sizes, which helps mitigate errors introduced by single-pixel matching. Since conventional descriptor computation does not account for surface orientation, matching inaccuracies may arise and subsequently propagate through later stages of the stereo pipeline, ultimately degrading the quality of the resulting disparity map. To further suppress outliers, each estimated disparity value is constrained using a set of local consistency rules. Finally, the disparity map is refined by removing small speckle

regions caused by erroneous matches, yielding a cleaner and more reliable reconstruction.

2.4.1.1 Matching Descriptors

In this work, binary descriptors are employed due to several advantageous properties, including their computational efficiency, invariance to additive and multiplicative intensity variations, and relatively low processing cost. For these reasons, a 7×9 weighted center-symmetric Census descriptor is adopted. The center-symmetric Census, defined in Equation (2.34), exhibits increased robustness to noise compared to the classical Census Transform, which relies exclusively on comparisons with the central pixel of the support window. In the classical formulation, corruption of the central pixel can compromise the entire descriptor, whereas in the center-symmetric variant, the influence of a noisy pixel remains localized and does not affect the full metric.

Furthermore, instead of encoding each comparison result using a single bit, the presented approach stores two bits per comparison. This modification enables a more nuanced weighting of comparison confidence. Specifically, when the compared pixel intensity is significantly smaller or larger than the reference value, the comparison result is stored twice, whereas values that fall within a defined interval relative to the reference are encoded as *01*. This encoding scheme reduces the adverse impact of uncertain comparisons when computing the Hamming distance between descriptors from the two images. As a result, mismatches have a less detrimental effect on the final matching outcome compared to classical Census-based approaches. The analytical formulation of the modified center-symmetric Census descriptor (MCST_CENSUS) is presented in Equation (2.35).

$$MCST\_CENSUS(u,v) = \sum_{i=0}^{\frac{N}{2}-1} \sum_{j=0}^{M} \varepsilon(I_{(u_i,v_j)}, I_{(u_{N-i},v_{M-j-1})}) \oplus \sum_{j=0}^{M/2} \varepsilon(I_{(u_{N/2},v_j)}, I_{(u_{N/2},v_{M-j-1})}) \quad (2.34)$$

$$\varepsilon(X,Y) = \begin{cases} 00,\ X - t \geq Y \\ 01,\ X - t < Y\ AND\ X + t \geq Y \\ 11,\ X + t < Y \end{cases} \quad (2.35)$$

In Equation (2.34), N and M denote the dimensions of the descriptor patch, while the $\bigotimes$ symbol represents the bitwise concatenation

operation. In Equation (2.35), the parameter t denotes a small intensity threshold introduced to compensate for minor intensity variations between corresponding pixels. In the described implementation, the value of t is set to 1. To effectively capture slanted surfaces, seven lookup tables are constructed for the right-image descriptors, each corresponding to a different degree of tilt applied to the descriptor block. Because the positional derivatives (offsets) for each pixel are precomputed and stored within these lookup tables, all seven right-image descriptor variants can be generated concurrently, resulting in very high computational efficiency. An intuitive illustration of the slanted descriptor windows is provided in Figure 2.2.

The seven lookup tables were selected through an empirical process. Initially, multiple tables corresponding to different degrees of inclination were generated. Subsequently, those that did not yield measurable improvements in reconstruction quality were progressively discarded, using the KITTI 2012 dataset as the reference benchmark.

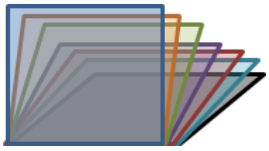

Figure 2.2. Tilted MCST CENSUS descriptor blocks

An intuitive example of a lookup table for a 3×3 patch is illustrated in Figure 2.3. Each entry stores the offset relative to the central pixel, where the first component denotes the row offset and the second denotes the column offset. In the presented approach, seven 7×9 lookup tables are constructed using the modified weighted center-symmetric Census descriptor. Of these, six tables are designed to model slanted surface configurations, while the remaining table corresponds to the fronto-parallel surface case.

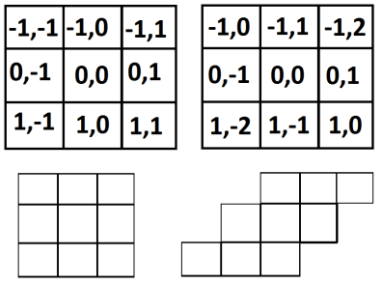

Figure 2.3. Intuitive illustration of a 3x3 lookup table for frontal (left) and slanted (right) situations.

2.4.1.2 Shifted Window Hamming Distance Computation

When binary descriptors are employed, the matching cost at a given pixel location is typically computed using the Hamming distance. In the classical formulation, for each pixel in the left image, candidate correspondences are searched along the corresponding epipolar line in the right image across a predefined range of disparities. The disparity associated with the minimum Hamming distance is then selected as the winning match.

In contrast, the described system extends this strategy by evaluating the correspondence of a left-image pixel against all seven descriptor representations of the right image across the same disparity range. Thus, instead of considering a single right-image descriptor per disparity, the method evaluates multiple descriptor variants corresponding to different surface orientations. The final matching score is obtained by selecting the minimum Hamming distance among the fronto-parallel case and the slanted-surface cases, to which a small penalty is added for the slanted configurations. This penalty discourages unnecessary selection of slanted hypotheses while preserving their benefit when appropriate. Empirically, a penalty value of 2 was found to yield the best performance on the stereo benchmark. The analytical formulations of these operations are provided in Equations (2.36), (2.37), and (2.38).

$$SI(i,j,d) = \sum_{k=1}^{6} \min(Hamming(IL(i,j), SRI_k\,(i,j,d))) \quad (2.36)$$

$$Frontal(i,j,d) = Hamming(IL(i,j), FPRI(i,j,d)) \quad (2.37)$$

$$FS(i,j,d) = Min(Frontal(i,j,d), SI(i,j,d) + Penalty) \quad (2.38)$$

In the above formulations, SI denotes the slanted-image score, defined as the minimum matching cost obtained across all six slanted right-image descriptors (SRI). The term *Frontal* represents the matching cost computed for the fronto-parallel configuration of a pixel located at position (i, j) with disparity d, while $FPRI$ denotes the corresponding fronto-parallel right-image descriptor.

The final matching score produced by the enhanced depth method for a pixel at position (i, j) and disparity d is denoted by $FS(i,j,d)$. This score is computed as the minimum between the frontal matching cost and the slanted matching cost augmented with a small penalty. The introduction of this penalty follows a rationale similar to that adopted

in semi-global matching approaches [55], where fronto-parallel surfaces are implicitly favored unless strong evidence suggests a slanted configuration.

The resulting value FS is stored in the cost volume $C(p,d)$, where $p$denotes the pixel position and $d$the associated disparity, as defined in Equation (2.39).

$$C(p,d) = FS(p,d) \tag{2.39}$$

After computing the Hamming distance, a multi-block aggregation step is applied to the cost volume in order to suppress potential outliers and reduce the bloating effect in the resulting disparity map. The adopted aggregation strategy is inspired by the approach presented in [48] and is further refined through experimental adaptation of the support regions.

Specifically, aggregation is performed using blocks of varying shapes and sizes—namely 1×155, 155×1, 17×17, and 7×7—to effectively capture structures of different orientations and extents. The aggregated cost is denoted by A, and the analytical formulation of the aggregation scheme is provided in Equation (2.40).

$$A = \max(val_{1x155}, val_{155x1})\, val_{17x17} val_{7x7} \tag{2.40}$$

The term $\text{val}_{a\times b}$ denotes the value obtained after aggregating the cost volume using a support window of dimensions $a \times b$, where a and b correspond to the vertical and horizontal sizes of the aggregation block, respectively.

2.4.1.3 Winner Takes All

In the winner-takes-all stage of the stereo correspondence pipeline, the disparity corresponding to the minimum cost value in the cost volume $C$is selected for each pixel position p, as expressed in Equation (2.41).

$$D(p) = argmin_d(C(p,d)) \tag{2.41}$$

The disparity map produced by stereo correspondence can be adversely affected by various external factors, including surface reflectance, textureless regions, and repetitive patterns. To mitigate these effects, several constraints are imposed on the cost volume during the winner-takes-all stage in order to filter out, or at least reduce,

erroneous disparity estimates. A disparity value is accepted only if all imposed constraints are satisfied.

As a first step, the three smallest cost values are identified for each pixel location. The initial constraint targets the detection of textureless regions by analyzing the structure of the cost volume. Specifically, if the three lowest local minima are equal, the corresponding region is considered ambiguous, indicating insufficient texture for reliable matching. This situation is illustrated intuitively in Figure 2.4 and formally described in Equation (2.42).

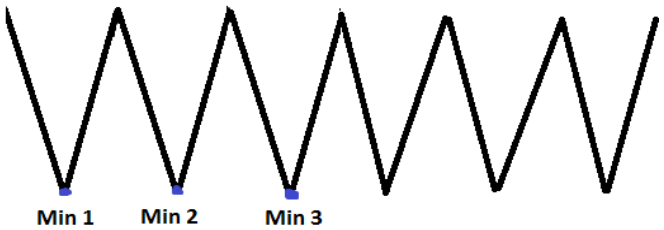

Figure 2.4. Representation of the periodicity in the cost volume

$$PeriodicityFlag = \begin{cases} 1, \ min1 = min2 \ \ AND \ min1 = min3 \\ 0, otherwise \end{cases} \quad (2.42)$$

The second constraint provides a measure of confidence in the estimated disparity value. This confidence constraint is computed by considering the three smallest costs in the cost volume at a given pixel position $p$and evaluating whether the ratio between the third smallest cost and the minimum cost falls below a predefined confidence threshold. The analytical formulation of this constraint is presented in Equation (2.43). In the implementation described, the confidence threshold is empirically set to 15.

$$ConfidenceFlag = \begin{cases} 1 \ , \ \ \frac{min3}{min1} \leq Confidence1 \ AND \ |min1 - min2| \leq Confidence2 \\ \\ 0, otherwise \end{cases} \quad (2.43)$$

To further enhance detection precision, sub-pixel interpolation is applied. In this work, the symmetric V interpolation method proposed

in [33] is employed. The sub-pixel disparity estimation is computed according to Equation (2.44), where the values M_1, M_2, and M_3 correspond to the winning disparity and its immediate neighboring disparities. To mitigate residual matching errors, an additional refinement stage is subsequently applied to the resulting disparity map.

$$Disp_{final} = Disp_{integer} + \begin{cases} 0.5 - 0.25 \cdot \left(\frac{(M_3 - M_1)^2}{(M_2 - M_1)^2} + \frac{M_3 - M_1}{M_2 - M_1}\right), if M_2 > M_3 \\ -\left(0.5 - 0.25 \cdot \left(\frac{(M_2 - M_1)^2}{(M_3 - M_1)^2} + \frac{M_2 - M_1}{M_3 - M_1}\right)\right), if M_2 \leq M_3 \end{cases} \quad (2.44)$$

2.4.1.4 Stereo Refinement

The application of refinement steps aims to eliminate residual outliers and improve the overall quality of the disparity map. The first refinement stage consists of a background fill-in procedure, which is applied in the presence of occlusions and inconsistencies identified through the left–right (LR) consistency check. The analytical conditions used to classify disparity values as correct, mismatch, or occlusion are defined in Equation (2.45). Based on this classification, the appropriate refinement action can be determined. For instance, when a disparity is identified as a mismatch, a background fill-in operation is applied, whereas in the case of an occlusion, the disparity value is discarded, resulting in an unreconstructed region.

$$\text{Condition} \begin{cases} correct, if \left| d - D^R(pd) \right| < 1 for\ d = D^L(p) \\ missmatch, if\ |d - D^R(pd)| < 1\ for\ any\ other\ d \\ occlusion, otherwise \end{cases} \quad (2.45)$$

To further eliminate residual speckles caused by incorrect correspondences, a novel speckle removal strategy is introduced. Conventional speckle removal techniques, which are typically based on region-growing approaches, remove only those speckles whose size falls below a predefined threshold, leaving larger erroneous regions unaddressed. In contrast, the enhanced method applies an erosion operation even when the speckle exceeds this size threshold, thereby progressively reducing the extent of the erroneous region.

The refinement process begins with a segmentation stage. Two disparity threshold values are employed for this purpose: a strong threshold T_2 and a weak threshold T_1. These thresholds define an upper

and a lower segmentation bound, respectively. Disparity values that lie between these bounds are included in the segmentation region. Let $v$denote the current disparity value; the corresponding upper and lower limits are computed according to Equations (2.46) and (2.47).

$$lower = v - T1 \tag{2.46}$$

$$upper = v + T2 \tag{2.47}$$

By employing two distinct thresholds, the segmentation process remains robust, particularly in the vicinity of object boundaries. The weak threshold T_1 is set to a lower value than the strong threshold T_2, allowing for controlled inclusion of neighboring disparity values while preserving edge integrity. Following segmentation, a morphological closing operation with an 11×11 structuring element is applied to the segmented regions. If a detected region corresponds to a small speckle, it is effectively removed through the erosion component of the closing operation. The resulting output is then incorporated into the filtered disparity map. Subsequently, a fill operation is performed to eliminate holes introduced by the morphological processing. For each pixel, a search is conducted along all eight directions to identify the nearest two disparity values exhibiting the smallest difference. This search is performed over a predefined number of k positions, ensuring reliable interpolation of missing disparity values.

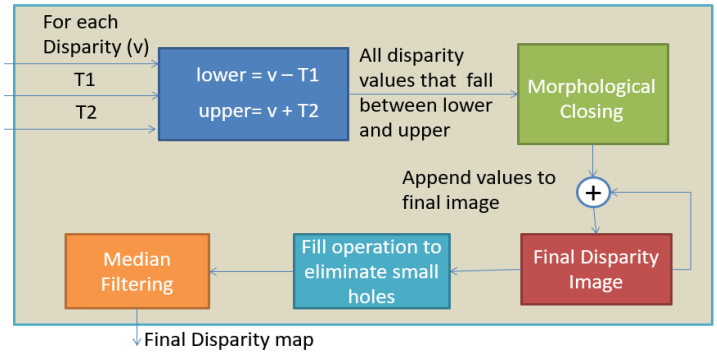

Figure 2.5. Disparity refinement process for speckle removal

The final stage of the presented refinement scheme applies median filtering with kernel sizes of 1×9 and 9×1 to suppress any remaining noise. In the implemented system, the values of the weak and strong thresholds are set to 3 and 1, respectively, while the search distance

used during the fill operation is 49 pixels. No background interpolation is incorporated into the refinement process; consequently, the resulting disparity map does not achieve full density. An intuitive illustration of the overall refinement procedure is presented in Figure 2.5.

2.4.1.5. Evaluation of the Block Matching Stereo Approach

For the evaluation of the proposed block-matching approach, experiments were conducted using the online KITTI benchmark [87], which consists of real-world road scenes and traffic scenarios. The proposed method was compared against several classical stereo approaches employing different cost functions, including the standard Census 7 × 9, Modified Census Transform, sparse Census, and weighted center-symmetric Census descriptors. In addition, the ranking of the proposed method relative to state-of-the-art approaches, as reported on the KITTI benchmark platform, is also presented. For all methods summarized in Table I, multi-block aggregation and the proposed refinement procedures were applied. The observed differences in reconstruction quality are primarily attributed to the LUT-based warping strategy used for descriptor computation. The comparison of disparity map density with respect to classical methods was performed on the training set using the publicly available ground-truth data. A quantitative comparison with classical rectangular descriptor-based approaches is reported in Table 2.1.

All experiments were carried out on a system equipped with an Intel i5-2500 CPU operating at 3 GHz. No hardware acceleration was employed. Limited parallelization was achieved using OpenMP for selected portions of the implementation. The evaluation was performed using an error threshold of 2 pixels.

Figure 2.6 illustrates a real-world example in which the Slanted Block Matching (SBM) approach outperforms the classical block-matching method. In this illustration, the bottom image corresponds to the left intensity image, the middle image shows the disparity map produced by the proposed approach, and the top image depicts the result obtained using a sparse Census method combined with multi-block matching aggregation. In the sparse Census approach, every second pixel is skipped, resulting in a sparse descriptor pattern, in contrast to the dense representation produced by the 7 × 9 Census descriptor.

Table 2.1: Evaluation with respect to the classical matching cost functions using KITTI Data-Set.

Method	Density (%)	Out-All Error (%)
Census	99.67	11.865
WCS-Census	99.98	11.24
MCT	99.68	11.21
SBM	**99.97**	**8.75**
Sparse Census	99.51	13.29

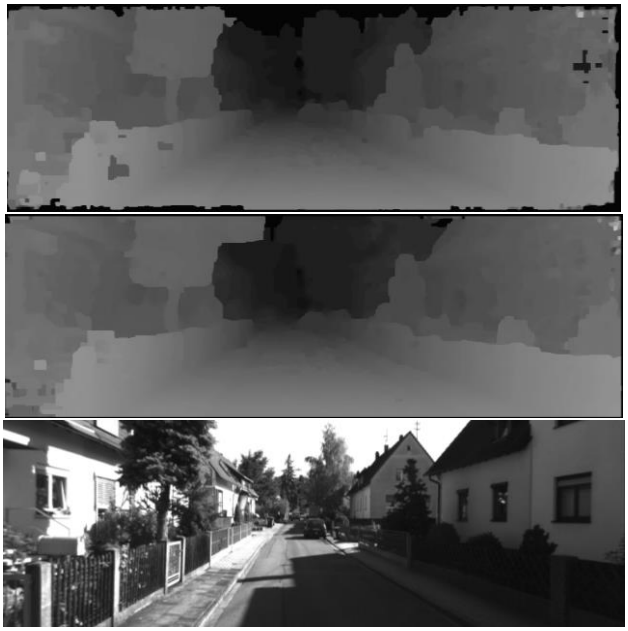

Figure 2.6. The first image is the disparity map obtained using a sparse census, MBM, method, the second image depicts the result of SBM and the bottom image represents the left intensity image

Figure 2.7 presents a qualitative comparison between the proposed method and the multi-block matching (MBM) approach using a dense 7×9 Census descriptor. The bottom image corresponds to the intensity image from the KITTI dataset, the middle image depicts the disparity map generated by the proposed Slanted Block Matching (SBM) algorithm, and the top image shows the result obtained with the MBM Census-based method. The comparison highlights that the proposed

approach achieves a more accurate reconstruction of the slanted surface on the right side of the scene. It should be noted that the same

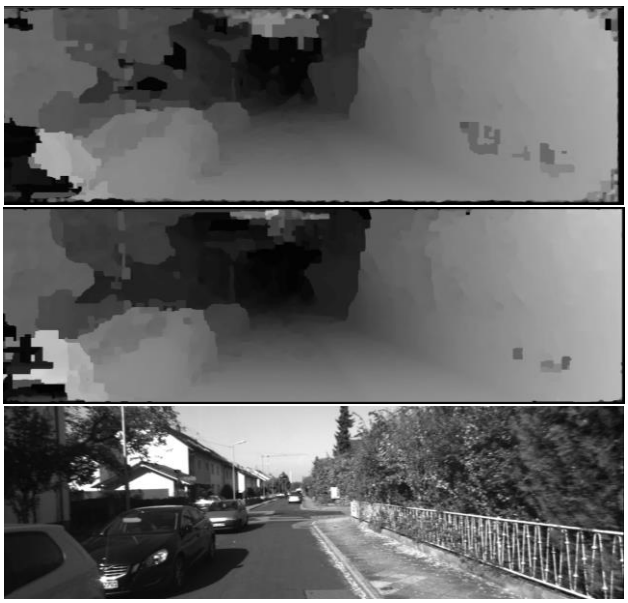

Figure 2.7. The top image depicts the disparity map obtained using the census MBM approach, the second image is the result of our approach (SBM) and lastly is the left intensity image.

aggregation scheme is applied consistently across all evaluated images.

Table 2.2 shows the comparison with other methods present in the KITTI dataset from 2016 (when the current approach was developed and tested). This overview shows that our local stereo correspondence method achieves good results, even surpassing some semi-global approaches. The error metric shown is out all, the percentage of erroneous pixels out of all the pixels in the image, including the occluded ones.

The proposed algorithm achieves a runtime of approximately 0.3 seconds per frame when evaluated on images from the KITTI dataset at their original resolution. The execution time decreases proportionally when operating on reduced image sizes. While the implementation can be further optimized to achieve higher processing speeds, runtime optimization was not within the scope of this work, as the primary focus was on improving the quality and robustness of the stereo matching results.

The low memory footprint of the proposed approach is justified by the fact that it does not require storing the entire disparity space for cost volume optimization. Furthermore, the method operates independently on each frame and does not rely on retaining information from previous frames when computing the current disparity map, which further contributes to its memory efficiency.

Table 2.2: Comparison with existing methods from the KITTI benchmark.

Position	Method	Performance on Kitty dataset	
		Density	**Out-All error**
32	wSGM[88]	97.03%	8.72%
33	AARBM[47]	85.80%	8.70%
34	DispNetC[89]	100%	8.11%
35	**SBM**	**99.97**	**8.75%**
36	AABM[90]	100%	8.77%
37	rSGM[58]	97.22%	9.24%

The only additional memory overhead compared to classical block-matching (BM) methods arises from storing the seven weighted center-symmetric Census descriptor images—one corresponding to the fronto-parallel case and six corresponding to slanted surface configurations—for the right intensity image. The overall memory consumption scales linearly with the size of the input images.

An alternative and intuitive way to assess the quality of a stereo reconstruction algorithm is through visual inspection of the three-dimensional point cloud generated from the disparity map. In the presence of matching errors, such visualizations typically reveal artifacts such as spikes protruding from the reconstructed surface or large unreconstructed regions. To this end, a synchronized stereo image pair was captured using a set of Manta cameras. After rectification, the proposed algorithm was applied, and the resulting disparity map was used to generate a three-dimensional point cloud using the Point Cloud Library. The resulting reconstruction is illustrated in Figure 2.8.

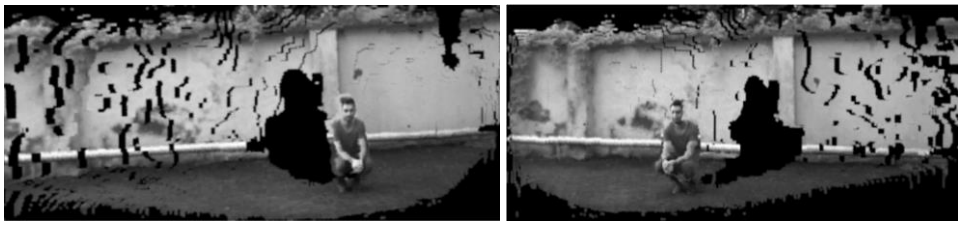

Figure 2.8. Rotation to the right and left of the 3D projected points obtained through the reconstruction process in PCL.

To illustrate the three-dimensional structure of the reconstructed scene, the point cloud was rotated and two snapshots were captured, demonstrating how the scene can be observed from different viewpoints.

For the evaluation of the proposed stereo–monocular fusion approach, the KITTI benchmark [87] was employed. This dataset comprises images from a wide range of driving scenarios and provides ground-truth depth measurements acquired using LiDAR sensors. In addition, for selected scenes, the dataset includes semantic and instance-level segmentation annotations, which further support the evaluation of perception algorithms in complex traffic environments.

2.4.2 Stereo and Mono Depth Estimation Fusion for an Improved and Fault Tolerant 3D Reconstruction

The solution presented in this section constitutes an extension of classical depth perception methodologies by efficiently integrating monocular and stereo approaches to achieve a more robust and reliable depth estimation framework. The method, originally introduced in

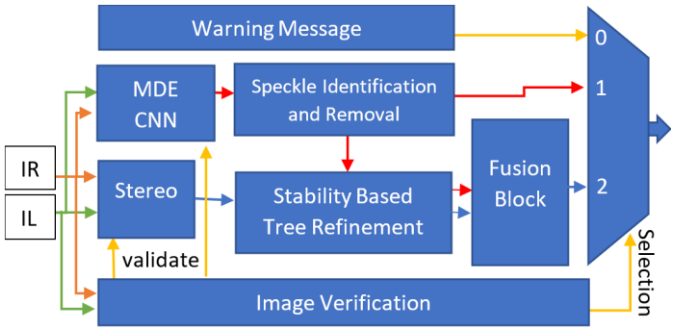

Figure 2.9. Pipeline of the fault tolerant disparity fusion approach

[349], is organized as follows. Section 2.4.2.1 provides an overview of the monocular depth estimation algorithm employed, followed by a description of the disparity refinement strategy. Section 2.4.2.3 is devoted to the fusion of monocular and stereo disparity maps with the objective of producing an enhanced depth representation.

The proposed system is designed to be fault tolerant, allowing it to operate reliably even in the presence of individual camera failures. A block diagram of the overall architecture is presented in Figure 2.6. After the left and right image frames are acquired, each is first validated to determine whether it contains meaningful visual information or consists primarily of noise. If both frames are deemed valid, they are processed by both the stereo and monocular reconstruction modules, and their outputs are subsequently refined and fused to generate a higher-quality depth map. If only a single frame is available, that frame is processed by the monocular depth estimation module, irrespective of whether it originates from the left or right camera. In the event that neither frame is valid, the system issues a warning indicating that depth estimation cannot be performed

2.4.2.1 Monocular Depth Estimation

The monocular depth estimation module used to generate the disparity maps is based on the convolutional neural network (CNN) introduced in [86]. The network follows an encoder–decoder architecture comprising a total of 14 layers and is derived from a slightly modified version of VGG-16. Conceptually, the architecture is structured as an encoder–decoder system, where the encoder progressively downsamples the feature maps to extract increasingly abstract representations, and the decoder performs the corresponding upsampling to recover spatial detail.

The encoder consists of seven convolutional blocks. Within each block, convolutional filtering is first applied using a stride of 1 and appropriate padding in order to preserve spatial resolution, since a rapid reduction in image size would lead to a significant loss of visual information. This operation is followed by a strided convolution with stride 2, which halves the spatial dimensions of the feature maps. All encoder layers use ELU activation functions. The first convolutional layer employs a 7×7 kernel with 32 filters, the second uses a 5×5 kernel with 64 filters, the third and fourth use 3×3 kernels with 128 and 256 filters, respectively, and the final three layers each use 3×3 kernels with 512 filters.

The decoder is likewise composed of seven blocks. Each block first upsamples the incoming feature maps by a factor of two and then concatenates the result with the corresponding encoder feature maps of identical spatial resolution via skip connections. A convolution with stride 1 is subsequently applied to refine the features and recover information lost during downsampling. Depth reconstruction is performed using a bilinear sampler, applied symmetrically by warping the left image with the right disparity map and the right image with the left disparity map.

To enhance depth estimation accuracy, the model operates on the original input image as well as on three additional downscaled versions obtained using scale factors of 2, 4, and 8. The final disparity map is produced by fusing the four scale-specific disparity estimates through a cost function. This loss function comprises three components: a photometric reconstruction term comparing the reconstructed images with the original left and right images, a smoothness term that enforces local regularity of the disparity field, and a consistency term that encourages agreement between the left and right disparity maps. The smoothness term is formulated using the L_1 norm of the disparity gradients and is modulated by the image gradients, reflecting the fact that depth discontinuities often coincide with intensity changes.

The model is implemented in TensorFlow and contains approximately 30 million trainable parameters. Training is performed using the Adam optimizer over 50 epochs with a progressively decreasing learning rate: 10^{-4} for the first 30 epochs, halved for the next 10, and halved again for the final 10 epochs. Data augmentation includes horizontal flipping, random color perturbations, gamma adjustments in the range [0.8, 1.2], and random per-channel intensity variations, followed by pixel value normalization.

Although this architecture was not developed as part of the present work, its key characteristics are summarized here for completeness, as it forms an integral component of the fault-tolerant three-dimensional reconstruction pipeline.

To further enhance the quality of the estimated disparities, refinement strategies are applied. In particular, the tree-based refinement method proposed in [95] is adapted for monocular depth estimation and implemented on the GPU to ensure real-time performance.

2.4.2.2 Speckle Filtering

Speckles of erroneous disparity values may appear in the resulting disparity map due to repetitive patterns or textureless regions. To remove these artifacts and replace them with more plausible estimates derived from neighboring regions, a dedicated speckle removal procedure is employed, based on two successive traversals of the disparity image.

During the first traversal, clusters of similar disparity values are identified and assigned unique labels. This labeling process considers a four-neighbor neighborhood—comprising the left, top-left, top, and top-right pixels. Only disparity values that satisfy the condition defined in Equation (2.48) are grouped into the same cluster, ensuring that only sufficiently similar disparities are aggregated.

$$\left|D_{ij} - first(D_{ij}^{neighbour})\right| < T \tag{2.48}$$

In Equation (2.48), D_{ij} denotes the disparity value of the currently unlabeled pixel, while first($D_{ij}^{\text{neighbour}}$) represents the first disparity value associated with an already labeled neighboring cluster. The parameter $T$is an empirically chosen threshold, set to 15, and $|x|$denotes the absolute value of x. If none of the neighboring pixels satisfies the condition defined in Equation (2.48), a new cluster identifier is assigned to the unlabeled pixel. Conversely, if a neighboring pixel has already been labeled and the condition is satisfied, the unlabeled pixel inherits the same cluster identifier. When multiple previously labeled neighbors satisfy the condition, the current pixel is assigned the smallest cluster identifier, and all such neighboring clusters are marked as equivalent.

Following the establishment of equivalence relations among disparity clusters, a breadth-first search (BFS) algorithm is applied to identify all connected components and to relabel them consistently. Each newly formed cluster resulting from the BFS operation is assigned the minimal label among the merged clusters. This relabeling process also yields, as a byproduct, the area in pixels of each disparity cluster. A second pass through the disparity map is then performed, during which all disparity values belonging to clusters with an area smaller than 30×30 pixels are invalidated.

To fill the resulting invalidated regions, semantic and instance-level segmentation maps are employed. The disparity image is scanned from

left to right and from top to bottom, and for each pixel with an invalid disparity, the closest neighboring pixel with a valid disparity is sought, provided that both pixels belong to the same instance. In cases where instance information is unavailable—such as for road or vegetation regions—the semantic segmentation is used to validate the match. This process is iteratively applied across the entire disparity map until all invalid regions are addressed. Finally, the refined disparity map is smoothed using a 3×3 median filter to remove residual noise.

2.4.2.3 Mono and Stereo Disparity Map Fusion

The fusion of stereo and monocular depth information provides both robustness and fault tolerance. In situations where one of the cameras in a stereo rig fails to acquire valid imagery or becomes unavailable due to hardware malfunction, the system must remain capable of reconstructing the scene, albeit with potentially reduced accuracy. Moreover, stereo vision systems may fail to recover depth in certain regions affected by repetitive patterns, textureless surfaces, or occlusions. In such cases, monocular depth estimation can supply complementary information and help fill unreconstructed areas.

To ensure low computational and memory requirements suitable for embedded platforms, a stereo reconstruction approach based on local block-matching algorithms, as described in the previous section, is employed. The processing pipeline begins with a validation step that determines whether the captured images are suitable for reconstruction. For this purpose, the histogram of the grayscale image is computed over a region of interest located in the lower part of the frame. If more than 70% of the pixel intensities fall below 10 or exceed 230, the image is classified as invalid and excluded from further processing. If only one image is deemed valid, the scene is reconstructed solely using the monocular depth estimation method. If both images pass the validation test, stereo block matching is applied, and monocular depth estimation is also performed independently on each image. In this case, the left image is selected as the reference, and fusion is conducted using the monocular disparity map corresponding to the left view.

The next step evaluates the stereo-derived disparity map to identify unreconstructed regions. Whenever such regions are detected, they are provisionally filled using information from the monocular depth estimation map. Subsequently, the disparity maps produced by stereo

and monocular reconstruction are fused, with the aid of the semantic segmentation of the left image.

The fusion strategy is based on the observation that monocular depth estimation is particularly reliable for certain surface types, such as roads, walls, and objects located close to the ego vehicle. However, as disparity values decrease—that is, as objects become more distant—the monocular depth map becomes increasingly blurred and less accurate, whereas stereo reconstruction remains more reliable. Prior to fusion, the absolute difference between the stereo and monocular disparity values is evaluated, and only values whose difference falls below an empirically defined threshold of 10 are considered for combination. In addition, only monocular disparity values greater than 60 are used, as smaller values are deemed unreliable.

The actual fusion is performed through a weighted combination based on semantic class information, as defined in Equation (2.49). In this formulation, disp denotes the stereo disparity, w_{class} is a class-dependent weight determined experimentally, and maxDisp represents the maximum disparity. The weights for selected semantic classes are defined as follows: $w_{\text{road}} = 0.8$, $w_{\text{building}} = 0.3$, $w_{\text{car}} = 0.4$, $w_{\text{pavement}} = 0.6$, $w_{\text{sign}} = 0.3$, $w_{\text{bicycle}} = 0.1$, and $w_{\text{pedestrian}} = 0.1$. For all remaining semantic classes provided by the KITTI segmentation—such as grass, sky, fence, tree, and discard—the stereo-derived disparity values are retained unchanged, as monocular estimates for these categories are considered unreliable

$$w_m = w_{class} * \frac{disp}{maxDisp} \tag{2.49}$$

The final disparity value at pixel location $(i, j)$is computed according to Equation (2.50), where $\Delta_{i,j}$denotes the fused disparity. In this formulation, $d_{i,j}^{\alpha}$ represents the disparity obtained from the local stereo reconstruction algorithm, while $d_{i,j}^{\beta}$ corresponds to the disparity estimated by the monocular depth estimation method.

$$\Delta_{i,j} = \left(1 - w_{m_{i,j}}\right) d\alpha_{i,j} + w_{m_{i,j}} d\beta_{i,j} \tag{2.50}$$

2.4.2.4 Evaluation of the Stereo-Mono Fusion Method

The experimental platform used for implementation and evaluation consists of an Intel® Core™ i5-7300HQ CPU, 8 GB of DDR RAM, and an NVIDIA GeForce GTX 1050 GPU with 4 GB of GDDR5 memory. Neural

network training and inference are performed on the GPU, while parts of the CPU code are parallelized using OpenMP. The implementation is developed in C++ and Python and makes use of OpenCV, TensorFlow, and the Point Cloud Library.

For quantitative evaluation, an error threshold of 2 pixels is used in the error maps. Disparity images are assessed using the Out-All metric, which evaluates all pixels with available ground-truth depth, including occluded ones.

Table 2.3 Comparison of the fusion with the individual algorithms

Method	**Disparity Evaluation Error**			
	2px	***3px***	***4px***	***Density***
Mono	18.202	13.940	9.649	100%
Stereo	7.324	6.168	4.888	100%
Proposed Fusion	5.756	4.446	3.000	100%

Figure 2.10 illustrates qualitative results on a KITTI traffic scene, showing the outputs of the stereo algorithm, the monocular depth estimation method, and their fusion, together with the corresponding error maps, where white pixels indicate errors exceeding 2 pixels. Table 2.3 summarizes the quantitative performance on the KITTI 2012 training set for error thresholds of 2, 3, and 4 pixels.

It should be noted that most of the observed errors are concentrated near object boundaries and within certain semantic classes, particularly vegetation—especially bushes—and fences. The results demonstrate that fusing information from the two disparity maps, obtained through fundamentally different reconstruction paradigms, leads to improved overall accuracy by effectively combining the complementary strengths of stereo and monocular methods.

The overall runtime of the proposed system is approximately 100 ms per frame, with processing distributed between the CPU and GPU. The stereo reconstruction module operates exclusively on the CPU, whereas the monocular depth estimation component is executed on the GPU.

Table 2.4 presents a comparison between the proposed approach and several state-of-the-art methods, evaluated using the Out-All error metric with a 2-pixel threshold. Additional disparity maps and their corresponding error visualizations produced by the proposed fusion approach on the KITTI stereo dataset are presented in Figures 2.11 and 2.12. In both examples, the method demonstrates the ability to

reconstruct the scene with high accuracy despite the presence of multiple challenging conditions, including repetitive patterns, shadows, reflections, and unstructured surfaces. Nevertheless, one notable limitation of the current approach, which remains to be addressed in future work, is its reduced accuracy at object boundaries.

Table 2.4 Comparison with state of the art on KITTI

Position	Method	Density	Out-All Error
129	FD-Fusion [91]	100%	5.73 %
130	Proposed Fusion	100%	5.75%
131	OSF [92]	99.98 %	5.79 %
132	CoR [93]	100%	5.88%
133	SPS-St [94]	100%	6.28%

In Figure 2.10, image a presents the semantic segmentation, b is the original color image, c is the error map obtained for mono algorithm, d is the disparity map of the monocular depth estimation approach, f is the disparity map and e represents the error map of the local stereo algorithm, h represents the fused disparity map and g is the error map.

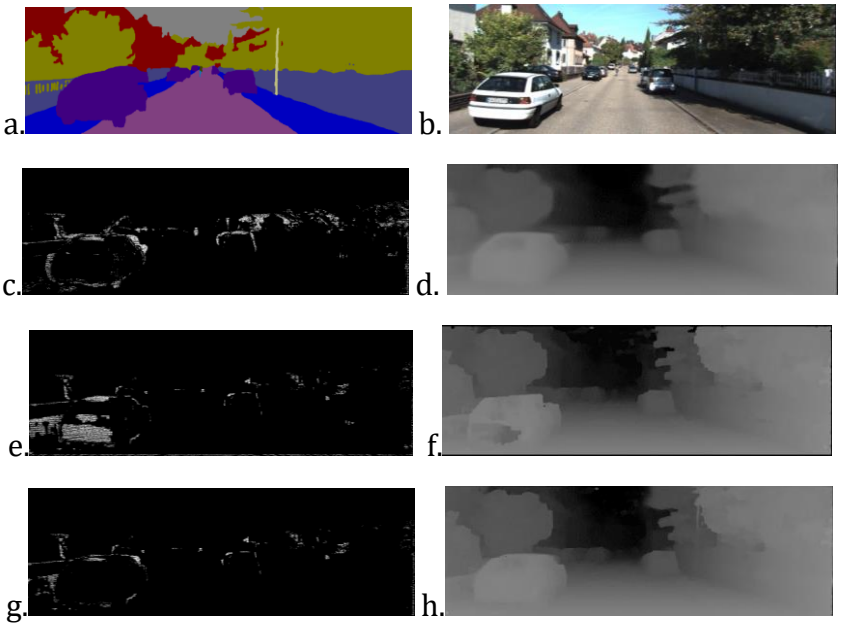

Figure 2.10. Comparison between the mono, stereo and proposed solution disparity maps.

Figure 2.11. Disparity map (middle) of the proposed fusion and its error map(bottom) using a 2-pixel threshold of a KITTI image (top).

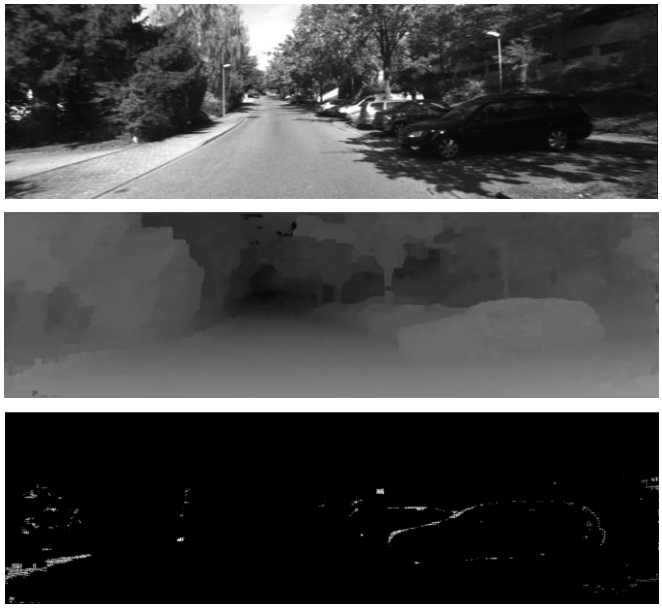

Figure 2.12. Disparity (middle) and error map (bottom) of the proposed method applied on an image illustrating a parking lot.

Chapter 3. Multi-Object Tracking

3.1 Introduction

3.1.1 Generalities

Multi-object tracking (MOT) represents a fundamental component of contemporary autonomous and intelligent systems. In domains such as autonomous driving, robotic inspection, and large-scale automated surveillance, the ability to consistently estimate and maintain the states of multiple entities in dynamic environments is a prerequisite for safe and reliable system operation. In particular, autonomous vehicles require continuous estimates of the kinematic states of surrounding objects—including position, velocity, orientation, and higher-order motion descriptors—in order to support both motion planning and feedback control.

The perception–planning–control pipeline that underlies autonomous systems critically depends on the quality of environment perception. While raw sensor measurements provide partial and noisy observations of the physical world, actionable information can only be obtained through filtering, association, and temporal integration. In cluttered and dynamic environments populated by multiple interacting agents, this requirement becomes especially pronounced, as objects may appear, disappear, occlude one another, or exhibit complex motion patterns.

Within this context, MOT can be formally regarded as the problem of estimating a time-varying, unknown number of object states from a sequence of noisy and incomplete measurements. These measurements are typically obtained from heterogeneous sensing modalities such as cameras, RADAR, and LiDAR, each of which provides only a partial and indirect observation of the underlying scene. Consequently, MOT is inherently a stochastic inference problem, in which uncertainty arises both from sensor noise and from ambiguities in the correspondence between measurements and physical objects.

When the estimation process relies on a single sensing modality, it is commonly referred to as filtering, whereas the combination of information from multiple sensors constitutes sensor fusion. Regardless of the sensing configuration, the objective remains the same: to infer the posterior distribution over object states and object existence, conditioned on all available observations. This task is complicated by

the fact that relevant state variables are not directly observable. For example, although position and velocity are typically included in the state vector, a RADAR sensor provides only range, bearing, and radial velocity measurements, requiring the latent state to be inferred through a probabilistic measurement model.

Two principal paradigms dominate the MOT literature: tracking-by-detection and tracking-before-detection. In tracking-by-detection systems, sensor data are first processed by dedicated detection algorithms that produce object hypotheses in the form of bounding boxes, point clusters, or parametric measurements. These detections are then associated across time and filtered to produce object trajectories. This paradigm has become dominant in vision-based and multimodal perception systems, owing to the availability of high-performance object detectors and learned feature representations.

By contrast, tracking-before-detection is designed for scenarios in which individual targets generate weak or ambiguous sensor returns that cannot be reliably thresholded in a single scan. In this framework, raw sensor measurements are integrated over time prior to making discrete detection decisions. Target hypotheses emerge gradually as evidence accumulates across multiple observations, allowing the system to detect and track objects that would otherwise remain below the detection threshold in individual frames. This approach has a long tradition in RADAR and sonar processing and remains relevant for low-signal or long-range sensing.

Historically, MOT originated in military RADAR systems developed during the Second World War, where the objective was to detect and track multiple hostile aircraft from noisy radar echoes. The fundamental principle—estimating target states from time-of-flight measurements of reflected electromagnetic energy—remains unchanged and continues to underpin modern surveillance, air-traffic control, and automotive perception systems.

The number, quality, and spatial distribution of detections generated for a given object depend strongly on both the sensing modality and the physical characteristics of the target. As a result, different classes of MOT systems may be distinguished based on whether they operate on sparse point measurements, dense image-based detections, or multimodal object hypotheses. In all cases, however, the central challenge remains the same: to robustly associate uncertain measurements with latent object states over time, while accounting for

missed detections, false alarms, occlusions, and changing motion dynamics.

The tracking types that can be defined are:

1. Point Object Tracking [101]

In this class of tracking models, each target is assumed to generate at most a single measurement at any given time instant; consequently, a target either produces one detection or remains unobserved..

2. Extended Object Tracking [102]

Extended Object Tracking [102] refers to a class of tracking models in which a single physical target may give rise to multiple measurements within a single observation cycle. In this setting, the spatial distribution of detections carries information about the geometric properties of the target, such as its shape and physical extent.

This paradigm is particularly relevant for sensing modalities such as LiDAR and RADAR, which frequently generate multiple returns from a single object. For instance, the point cloud produced by a LiDAR sensor for a road vehicle typically exhibits an L-shaped structure corresponding to the visible sides of the vehicle, from which its length and width can be inferred

3. Group Object Tracking [103]

Group Object Tracking [103] addresses scenarios in which multiple individual targets are not tracked separately but are instead modeled and estimated as collective entities. In this framework, detectors may produce multiple observations corresponding to members of a group at each time instant; however, these measurements are clustered and associated with a common group-level representation.

This approach is applicable across different sensing modalities and is particularly suited to the analysis of collective behavior or to situations in which reliable tracking of individual objects is impractical due to high density, severe occlusion, or limited sensor resolution.

4. Tracking with Multipath Propagation [104]

In this class of tracking scenarios, a single physical target may generate multiple detections within the same observation cycle as a result of multipath propagation effects. A representative example arises in over-the-horizon radar (OTHR) systems, where electromagnetic signals transmitted toward the upper atmosphere are reflected by ionospheric layers, enabling the detection of objects that are not in the direct line of sight of the radar platform.

An analogous phenomenon occurs in automotive RADAR sensing, where signals may reflect off ground structures such as manholes or

road surfaces and subsequently illuminate regions beneath or behind a vehicle. These indirect propagation paths can produce multiple spatially separated detections originating from the same physical object, thereby introducing additional ambiguity into the tracking process..

5. Tracking with Unresolved Objects [105]

This paradigm, commonly referred to as tracking with merged measurements, describes situations in which multiple physical targets give rise to a single sensor observation. A typical example arises in automotive radar sensing when two vehicles traveling in close proximity with similar velocities produce overlapping returns that are reported as a single detection, often located spatially between the two vehicles.

Within the scope of this monograph, the focus is placed on point-object tracking in a tracking-by-detection framework. Under this modeling assumption, the fundamental processing stages of a tracking algorithm following the acquisition of uncertain detector outputs can be systematically characterized as follows::

1. Track birth/State prediction for existing tracks
2. Gating
3. Data Association
4. Filtering (State estimation)
5. Track management

Track initiation (birth) denotes the creation of a new track hypothesis when a detection cannot be associated with any previously existing object. State prediction refers to the propagation of the current state estimate of each track through the assumed motion model in order to obtain a prior estimate for the next time step. Gating is employed to restrict the set of candidate detections considered for association with a given track, thereby reducing computational complexity and eliminating implausible matches.

Data association assigns measurements to track hypotheses based on a compatibility or matching score, while the filtering stage computes the posterior probability distribution of each object state given the associated measurements. Finally, track management governs the life cycle of tracks by confirming, maintaining, or terminating them, and by maintaining their trajectory histories once sufficient evidence of persistence has been accumulated.

Despite several decades of intensive research, multi-object tracking remains an open and challenging problem. Although a set of canonical

processing stages has been established, no universally optimal solution exists. Consequently, practical tracking systems must be designed and adapted to the specific sensing modalities, environmental conditions, and application requirements under consideration.

3.1.2 Multi-Object Tracking Challenges

In this subsection, the principal factors that influence the performance of multi-object tracking algorithms are examined. The discussion is framed with reference to an urban driving scenario in which a vehicle equipped with multiple perception sensors operates in a dynamic environment populated by numerous interacting objects.

- The number of objects present within the sensor field of view (FOV) is generally unknown and time-varying. As objects move through the environment, they may enter or leave the FOV, thereby introducing uncertainty regarding the current number of targets. This dynamic process of appearance and disappearance is commonly referred to as object birth and object death, respectively.
- Multi-object tracking is inherently limited to estimating only those objects that lie within the sensor's field of view. The true states of the objects—such as their precise positions and future trajectories—are not directly observable and must therefore be inferred, implying that the object states are intrinsically uncertain.
- Objects located within the sensor's field of view may be partially or completely occluded by other objects, leading to missing or degraded measurements and increased ambiguity in the tracking process.
- When multiple objects are in close spatial proximity, their sensor returns may overlap, resulting in fewer detections than the actual number of physical targets and thereby increasing the difficulty of correct data association..
- Owing to the imperfect nature of detection algorithms, two primary types of errors arise: missed detections, in which an existing object is not observed, and false detections, in which a spurious measurement is generated in the absence of a real object.
    - A missed detection occurs when an object that is present within the sensor's field of view is not reported by the

detector. In safety-critical applications, such failures can have severe consequences. Missed detections may arise due to several factors, including:

- Adverse environmental conditions, such as fog, rain, snow, or poor illumination, which degrade sensor signal quality and reduce detection reliability.
- Target-specific properties, such as material composition, geometry, or surface characteristics, which may significantly reduce sensor reflectivity or visibility; for example, stealth aircraft in defense applications are designed to minimize RADAR detectability.
- Occlusions, whereby objects are partially or fully blocked by other scene elements; in road traffic scenarios, pedestrians are frequently occluded by vehicles or other obstacles.

- False detections, also referred to as *clutter* in the literature [106], correspond to measurements that do not originate from any physical object. Such spurious observations are problematic because an autonomous system may respond to them as if they represented real obstacles or agents. False detections may arise due to several factors, including:
  - Strong background reflections, in which stationary or environmental surfaces reflect sufficient electromagnetic energy to be interpreted by the RADAR as a target.
  - Visual ambiguities, where background structures or textures resemble objects of interest, leading vision-based detectors to produce spurious detections.
  - Extraneous environmental returns, originating from irrelevant scene elements or sensor artifacts, which are incorrectly interpreted as object measurements.

- Beyond the aforementioned challenges, multi-object tracking is fundamentally affected by the data association problem, which arises from the absence of explicit information about the origin of each measurement. In particular, it is unknown whether a given detection corresponds to a previously observed object, to a newly appearing target, or to spurious clutter. Inadequate handling of data association can lead to severe tracking failures

and, in safety-critical applications such as autonomous driving, may result in catastrophic outcomes.

- Motion uncertainty constitutes an additional fundamental challenge in multi-object tracking. In urban environments, objects rarely follow simple or stationary motion patterns, and no single motion model can reliably describe the trajectories of all agents over time. As a result, prediction errors are inevitable and must be explicitly accounted for within the tracking framework.
- Additional challenges arise from the characteristics of the sensing modalities themselves. Each sensor exhibits specific strengths and limitations that directly affect tracking performance, including measurement noise, limited spatial resolution, and modality-specific artifacts such as flat-field non-uniformity in thermal cameras.
- Calibration and acquisition inconsistencies in multi-sensor systems can severely degrade tracking performance. For example, when color information from cameras is used to augment the tracking of three-dimensional objects detected in LiDAR point clouds, inaccurate spatial calibration may lead to erroneous projection of 3D points onto the image plane. Moreover, temporal acquisition effects become significant when the platform or the observed objects are in motion: because a LiDAR scan is acquired sequentially over time, points belonging to a fast-moving vehicle may no longer correspond to its instantaneous image location by the time the scan is completed, thereby introducing additional spatial misalignment and association errors.

3.1.3 Remarks Regarding the Statistics Used in MOT

The tracking methodologies presented in this book are formulated within a filtering framework, in which object states are inferred from measurements that are inherently noisy, incomplete, or corrupted. A comprehensive treatment of this probabilistic perspective on tracking can be found in standard references such as [107] and [108].

Quantities of interest—such as object states and sensor observations—are modeled as random variables. Because these variables can assume a wide range of possible values, their evolution and mutual dependencies must be described using probabilistic inference principles. Among the

available statistical formalisms, Bayesian statistical theory [107] provides a particularly powerful and general framework for object tracking and underpins even the most advanced modern tracking algorithms.

The distribution of a random variable is characterized by a probability mass function in the discrete case and by a probability density function in the continuous case. Let $\Pr\{z\}$ denote the probability mass function of a discrete random variable z; this function must satisfy the axioms listed in (3.1).

$$\Pr\{z\} \geq 0 \; and \; \sum_z \Pr\{z\} = 1 \tag{3.1}$$

The probability density function of a random variable z is denoted as p(z) and the requirements of a probability density function are presented in (3.2).

$$p(z) \geq 0 \; and \; \int p(z)dz = 1 \tag{3.2}$$

In contrast to the discrete case, a probability density function $p(z)$ may exceed unity for certain values of z; the fundamental requirement is instead that its integral over the entire domain equals one. To characterize the dependence between two or more random variables, joint and conditional distributions are employed. The joint probability of two events A and B, denoted $p(A \cap B)$, represents the probability that both events occur simultaneously, that is, the probability of their intersection. The conditional probability $p(A \mid B)$ denotes the probability of event A occurring given that event B has occurred. By contrast, the marginal probability $p(A)$ describes the likelihood of event A independent of any conditioning on other events.

Within Bayesian statistics, two fundamental identities play a central role in formulating the tracking problem in probabilistic terms. The first is the product rule, which relates the joint probability density function of two random variables x and z, denoted $p(x, z)$, to their conditional and marginal densities. Provided that the probability of x is nonzero, the conditional density $p(z \mid x)$ is defined as in (3.3) and (3.4).

$$p(x, z) = p(z|x)p(x) = p(x|z)p(z) \tag{3.3}$$

$$p(z|x) = \frac{p(x,z)}{p(x)} \tag{3.4}$$

By combining the product rule with the definition of conditional probability, Bayes' theorem (3.5) is obtained. This fundamental identity

provides a mechanism for expressing one conditional probability, such as $p(x \mid z)$, in terms of the reverse conditional $p(z \mid x)$ and the associated marginal distributions.

$$p(x|z) = \frac{p(Z|X)p(x)}{p(z)} \tag{3.5}$$

The second fundamental identity of the Bayesian framework is the law of total probability. This principle states that the marginal probability of a random variable can be obtained by marginalizing its joint distribution over all possible outcomes of the associated variables. Let x be a random variable taking values in a set S_x; for the discrete case, the law of total probability is expressed as in (3.6), while its continuous counterpart is given in (3.7).

$$\Pr\{z\} = \sum_{x \in S_x} \Pr\{x, z\} = \sum_{x \in S_x} \Pr\{z|x\} \Pr\{x\} \tag{3.6}$$

$$p(z) = \int_{x \in S_x} p(x, z)dx = \int_{x \in S_x} p(z|x)p(x)dx \tag{3.7}$$

One of the most widely used probability distributions in tracking and sensor fusion is the Gaussian distribution. In many practical filtering and fusion algorithms, the system state is characterized by the mean vector (expected value) and the covariance matrix of a Gaussian density. Even when the true posterior distribution is non-Gaussian, it is often approximated by a Gaussian through its first- and second-order moments in order to obtain a tractable representation.

The probability density function (pdf) of a multivariate Gaussian random variable with mean μ and covariance matrix Q is given in (3.8), where $|\cdot|$ denotes the matrix determinant. The Gaussian pdf is fully specified by its mean vector and covariance matrix:

$$p(x) = \mathcal{N}(x; \mu, Q) = \frac{1}{\sqrt{|2\,\pi\,Q|}} \exp\left(-\frac{1}{2}(x - \pi)^T Q^{-1}(x - \pi)\right) \tag{3.8}$$

In the context of autonomous systems, statistical models are employed to represent the quantities of interest. Let k denote a discrete time index and x_k the system state at time k. The motion or process model describes the temporal evolution of the state, specifying how the previous state x_{k-1} transitions to the current state x_k, as expressed in (3.9).

$$p(x_k|x_{k-1}) = x_k = f_{k-1}(x_{k-1}, q_{k-1}) \tag{3.9}$$

As indicated in (3.9), the formulation of a motion model involves two complementary components. The first captures the deterministic

dynamics of the system, typically expressed through physical or kinematic models. The second accounts for uncertainty arising from unmodeled effects and random perturbations, which are represented by the process noise q_{k-1} governing the variability of the state evolution between successive time steps.

The term process model originates from control theory, where it is used to describe the underlying dynamics of a physical system. In the context of object tracking, it specifies how a vehicle or target is expected to move over time, thereby linking consecutive state estimates and constraining the set of physically plausible trajectories.

To relate sensor observations to the underlying state vector, a measurement (or sensor) model is employed, as expressed in (3.10). This model maps the latent state to the observed measurement space while accounting for measurement noise.

$$z_k = h_k(x_k, r_k) \tag{3.10}$$

In addition, a Markov assumption is introduced, according to which the measurement noise r_k and the process noise q_k are statistically independent of all other noise terms, including those associated with previous time steps. This property is also applied to the system state and measurements: the state at time k is conditionally independent of all past states and observations given the immediately preceding state, since all relevant information is assumed to be encapsulated in the current state vector. Likewise, the current measurement z_k is conditionally independent of all previous measurements given the current state. Because motion and measurement models are only approximations of the true physical processes, they cannot fully capture the complexity of real-world dynamics. For this reason, practical tracking systems often employ multiple motion models to better accommodate diverse and time-varying behaviors.

Once the motion and measurement models have been defined, the objective of the filtering process is to compute the posterior probability density of the state at each time step. Two general strategies can be employed: a non-recursive formulation, which conditions the state on the entire measurement history, and a recursive formulation, which updates the state estimate incrementally. Since the computational complexity of non-recursive methods grows rapidly with time,

recursive filtering approaches are preferred in most practical applications.

The essence of the recursive filtering framework is to compute the posterior state density $p(\mathbf{x}_k \mid \mathbf{z}_{1:k})$ by propagating and updating the previously available posterior $p(\mathbf{x}_{k-1} \mid \mathbf{z}_{1:k-1})$. This iterative estimation process is schematically illustrated in Figure 3.1.

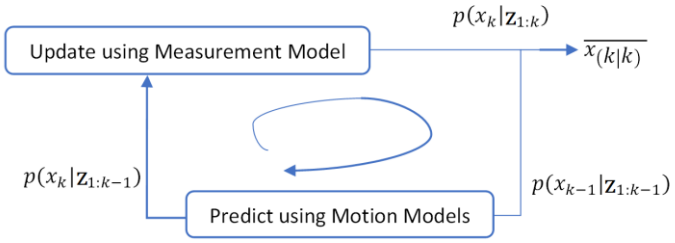

Figure 3.1. Recursive computation of the posterior density x_k

As illustrated in Figure 3.1, the recursive computation of the posterior state density proceeds in two stages: prediction and update. It is assumed that, at time $k-1$, the posterior density $p(\mathrm{x}_{k-1} \mid \mathrm{z}_{1:k-1})$ is available. This distribution compactly summarizes the information about the state up to time $k-1$ given all measurements acquired so far. At the subsequent time step, the posterior density is propagated through the motion model to obtain the predicted (prior) density for time k. This prediction summarizes what is known about the current state based solely on past measurements and the assumed system dynamics. The predicted density is then combined with the new observation z_k via the measurement model in the update step to obtain the posterior filtering density $p(\mathrm{x}_k \mid \mathrm{z}_{1:k})$. From this posterior distribution, a point estimate of the current state, denoted $\bar{\mathrm{x}}_{k|k}$, can be derived according to a chosen estimator.

A key advantage of the recursive formulation is that its computational complexity remains bounded over time, since only the previous posterior and the current measurement are required at each iteration. Because the prediction and update operations constitute recurring elements throughout the remainder of this book, the mathematical foundations of each step are detailed in the following.

The Prediction Step

In the prediction step, the objective is to compute the predicted (prior) density $p(\mathrm{x}_k \mid \mathrm{z}_{1:k-1})$ from the posterior density $p(\mathrm{x}_{k-1} \mid \mathrm{z}_{1:k-1})$ The information about the previous state x_{k-1}, encapsulated by the measurements $\mathrm{z}_{1:k-1}$, is propagated forward in time in order to infer the distribution of the current state x_k.

By expressing the current state in terms of the previous state and applying the law of total probability (3.7), the predictive density can be written in the form given in (3.11).

$$p(x_k|z_{1:k-1}) = \int p(x_k, x_{k-1}|z_{1:k-1})\, dx_{k-1} \tag{3.11}$$

The resulting density can be factorized into two components, as shown in (3.12). Moreover, under the Markov assumption for the state sequence, the current state conditioned on the immediately preceding state is independent of all past measurements. Consequently, the transition density depends only on x_{k-1} and not on the measurement history.

$$\begin{aligned} &\int p(x_k, x_{k-1}|z_{1:k-1})\, dx_{k-1} = \\ &\int p(x_k|x_{k-1}, z_{1:k-1}) p(x_{k-1}|z_{1:k-1}) dx_{k-1} \end{aligned} \tag{3.12}$$

As a result, the past measurements can be omitted from the conditional density, since they do not contribute additional information about the current state beyond that contained in the previous state. Accordingly, the term $p(\mathrm{x}_k \mid \mathrm{x}_{k-1}, \mathrm{z}_{1:k-1})$ reduces to $p(\mathrm{x}_k \mid \mathrm{x}_{k-1})$, which corresponds to the motion model. The resulting expression for the predicted density $p(\mathrm{x}_k \mid \mathrm{z}_{1:k-1})$, given in (3.13), is known as the Chapman–Kolmogorov equation.

$$p(x_k|z_{1:k-1}) = \int p(x_k|x_{k-1}) p(x_{k-1}|z_{1:k-1}) dx_{k-1} \tag{3.13}$$

This equation provides the recursive mechanism by which the predicted state density is computed within the filtering framework

The Measurement Update Step

In the update step, the prior knowledge about the state x_k obtained during the prediction phase is refined using the newly acquired measurement z_k. The current observation is first isolated from the past measurement set, after which Bayes' theorem (3.5) is applied to obtain the posterior expression given in (3.14).

$$p(x_k|z_{1:k}) = p(x_k|z_k, z_{1:k-1}) = \frac{p(y_k|x_k, z_{1:k-1})p(x_k|z_{1:k-1})}{p(z_k|y_{1:k-1})} \quad (3.14)$$

The denominator in this expression can be treated as a normalization constant. Moreover, conditioned on the current state, the present measurement is independent of all past observations, which allows the term $p(\mathrm{z}_k \mid \mathrm{x}_k, \mathrm{z}_{1:k-1})$ to be replaced by the likelihood $p(\mathrm{z}_k \mid \mathrm{x}_k)$, corresponding to the known measurement model. Consequently, the posterior state density is obtained as the product of the likelihood and the predicted density, as shown in (3.15).

$$p(x_k|z_{1:k}) \propto p(z_k|x_k)p(x_k|z_{1:k-1}) \quad (3.15)$$

A principal limitation of the Chapman–Kolmogorov equation is that it is generally difficult to evaluate in closed form, and the resulting predictive density may belong to a different distribution family than the prior. The Kalman filter [109], however, provides an exact and tractable solution to the filtering equations under specific modeling assumptions. Most notably, both the motion and measurement models are required to be linear, and all associated uncertainties must be Gaussian.

3.2 Review of Data Association and Multi Object Tracking Methods

3.2.1 Measurement Validation

Measurement validation, also known as gating, is employed to limit the number of candidate measurement-to-track associations by discarding combinations that are highly unlikely. By restricting the set of admissible associations, gating substantially reduces the computational burden of the data association stage.

The fundamental principle of gating is to define a validation region, or gate, around the predicted position of each object and to consider for association only those measurements that fall within this region. Measurements that do not lie inside any validation gate are typically used to initiate new track hypotheses. At each time step k, a validation gate is characterized by a gate center, corresponding to the predicted measurement, and a gate volume, which defines adaptive bounds of the admissible region [110].

From a statistical perspective, gating can be interpreted as a measurement selection procedure, in which only those observations

that are highly likely to correspond to a given predicted state are retained for further evaluation [108].

When object state and measurement uncertainties are modeled as Gaussian distributions, the validation region is naturally represented by an ellipsoidal gate. In this case, the Mahalanobis distance between the predicted measurement and an observed detection is used to determine whether a given measurement lies within the validation gate of a particular track [111]. The corresponding measurement validation criterion is given in (3.16) [112].

$$\vartheta(k,\gamma) = \{\mathrm{z}: \left[\mathrm{z} - \hat{z}_{(k|k-1)}\right]^{\mathrm{T}} \mathrm{S(k)}^{-1} \left[\mathrm{z} - \hat{z}_{(k|k-1)}\right] \leq \gamma\} \qquad (3.16)$$

In this expression, γ denotes the gating threshold associated with the gate probability P_G, which represents the probability that a measurement lying within the gate originates from the tracked object. The matrix S_k^{-1} corresponds to the inverse of the innovation covariance, whose explicit form is derived in Section 3.2.5 in the context of Kalman filtering. For an n-dimensional measurement space, the volume of the validation region is given by (3.17) [113], where the constant c_n depends solely on the dimensionality of the measurement vector.

$$V_k = c_n \, \gamma^{\frac{n}{2}} \sqrt{|S(k)|} \qquad (3.17)$$

The design of validation gates has been continuously refined to improve data association performance in the presence of clutter and has been adapted to suit different tracking paradigms. For example, in [114] an adaptive locally optimal validation algorithm is proposed within the data association filter framework, in which the gate is determined by maximizing a performance estimation function. In [115], a rectangular sigma-gate is introduced to discard measurements lying outside the validation region, thereby accelerating the probability hypothesis density (PHD) filter while simultaneously improving its accuracy relative to the standard formulation. This approach is based on the premise that measurements closer to the predicted state are more likely to originate from the target and contribute meaningfully to the estimation process, whereas distant measurements have negligible influence and can be excluded to mitigate the effects of clutter.

For maritime surveillance applications, a non-elliptical gate based on detection dynamics was developed in [116], yielding improved target probability estimates at the cost of increased computational complexity. Extending the classical elliptical gating concept, a velocity-based

circular gate was proposed in [117]. In road traffic scenarios, where certain classes of road users exhibit limited abrupt motion changes, [118] introduces a motion-direction gate that selects only measurements consistent with the expected direction of travel. The effective validation region is then defined by the intersection of this directional gate with the ellipsoidal gate given in (3.18), thereby reducing false alarms and enhancing association reliability.

A centralized gating strategy is presented in [119], where the gate center is obtained by combining predicted measurements from multiple motion models at each time step. The performance of this approach can be further improved by estimating the gate size through a mixture of the associated covariance matrices. An alternative strategy employs multiple gate centers, each derived from a predicted measurement and its corresponding innovation covariance, so that all measurements falling within any of the gates are retained for subsequent processing. Although this approach is computationally demanding, it has been successfully applied to multipath data association in [120]. A compromise between these strategies is proposed in [119], where the gate center is computed in a centralized manner, while the gate volume is obtained by forming a weighted combination of innovation covariances.

Ultimately, the selection and design of a gating strategy must be tailored to the specific sensing modality and filtering framework employed in a given tracking system.

3.2.2 Motion and Measurement Models

The performance of a filtering-based tracking system is strongly influenced by the choice of motion and measurement models, which must be carefully adapted to the specific tracking problem under consideration. Motion modeling characterizes the temporal evolution of the object state, describing how the state transitions from $\mathrm{x}_k$to x_{k+1}. Real-world objects exhibit a wide range of motion behaviors: vehicles may travel at constant velocity, accelerate, or execute turns, while pedestrians often display highly variable and less predictable trajectories. Capturing these dynamics is essential for accurate state prediction.

In autonomous driving applications, the decisions of the ego vehicle are directly informed by predicted object trajectories, which makes the fidelity of motion modeling particularly critical. Accurate prediction

also facilitates subsequent stages of the tracking pipeline; for instance, by placing the predicted object state close to the corresponding measurement in the next frame, the data association problem becomes more tractable.

Despite its importance, no universally optimal motion model exists for describing the full diversity of traffic participant behaviors. By definition, motion and measurement models are approximations of reality and therefore inherently imperfect. They must strike a balance between representational accuracy and computational efficiency, since overly complex models may be impractical for real-time operation, while overly simplistic ones can degrade tracking performance. The optimal trade-off between model complexity and estimation accuracy depends on both the sensing configuration and the operational scenario.

3.2.2.1 Motion models

Two principal classes of motion models are considered in this book.:

a. Translational Kinematics

These classes of motion models are grounded in Newtonian mechanics and assume either linear or uniformly accelerated motion. Their principal advantage lies in their simplicity and low computational cost, which makes them particularly attractive for real-time filtering implementations. Common representatives of this class include the constant velocity, constant acceleration, and random walk models.

A key limitation of such models is their restricted ability to represent complex object dynamics, as they typically assume rectilinear motion and neglect rotational or articulated behavior. Consequently, objects are usually modelled as point masses moving through space, which may be insufficient for accurately capturing the kinematics of extended or highly manoeuvrable targets.

b. Rotational Kinematics

The second class of motion models is more expressive and encompasses curvilinear dynamics, including pure rotations as well as combined rotational and translational motion. These models are particularly relevant when object orientation in two or three dimensions must be explicitly represented. Depending on which state variables are assumed to remain constant, several formulations can be distinguished within this category. Prominent examples include the

Constant Turn Rate and Velocity (CTRV) model, commonly used in aerial and vehicular tracking, and the Constant Turn Rate and Acceleration (CTRA) model. Both formulations assume independence between linear velocity and yaw rate.

To address this limitation, the Constant Steering Angle and Velocity (CSAV) model introduces a coupling between velocity and yaw rate through the steering angle. Under the additional assumption of constant velocity, this formulation yields the Constant Curvature and Acceleration model. With the exception of the CTRA model, which assumes motion along a clothoid, most curvilinear models are based on circular trajectories. In practice, many of these models require parameters that cannot be reliably inferred for externally observed vehicles, which restricts their applicability primarily to ego-vehicle modeling [122].

A comprehensive overview of widely used motion models is provided in [123], and the importance of selecting an appropriate model for vehicular tracking performance is demonstrated in [124].

A further challenge arises from the fact that object motion evolves in continuous time, whereas sensors and digital filters operate at discrete time instants. Consequently, continuous-time motion models and noise processes must be transformed into discrete-time representations. This is typically achieved by first formulating the dynamics in continuous time and subsequently sampling them at discrete intervals. Two principal discretization strategies are commonly employed: the Euler discretization and the analytical solution for linear time-invariant systems, also referred to as the exact discretization, which is derived via Taylor expansion [125], [126]. The appropriate choice depends on the application context: Euler discretization is convenient for nonlinear systems or small sampling intervals, whereas the exact solution is more suitable when the time step between successive states is relatively large.

a. Translational Kinematics

1. The constant velocity (CV) model

The state vector of this model comprises the position and velocity of the object. Several equivalent parameterizations are possible; one such representation is given in (3.18).

$$x = [x\ v_x\ y\ v_y]^T \tag{3.18}$$

In this model, the time derivative of position is defined as the velocity, while the time derivative of velocity is assumed to be zero. Accordingly, the velocity is modeled as constant, which motivates the name of the model. Since perfectly constant velocity is not realistic in practice, a stochastic noise term is introduced to allow small deviations from this idealized behavior. As a result, the velocity becomes nearly constant, with its variations represented by a zero-mean random process. The corresponding continuous-time state transition equation is given in (3.19).

$$x(t+T) = A(t+T)x(t) + q(t) = \begin{pmatrix} x(t) + Tv_x \\ v_x \\ y(t) + Tv_y \\ v_y \end{pmatrix} + q(t) \tag{3.19}$$

2. The constant acceleration (CA) model

Another representative of the translational motion model class is the constant acceleration (CA) model. This formulation extends the constant velocity model by augmenting the state vector with acceleration. In this case, the time derivative of the acceleration is assumed to be zero, up to a stochastic perturbation.
Let the state vector be defined as in (3.20), where $p(t)$ denotes position, $v(t)$ velocity, and $a(t)$ acceleration at time t. The corresponding discrete-time state transition equation is given in (3.21).

$$x(t) = [p(t), v(t), a(t)] \tag{3.20}$$

$$x_{k+1} = \begin{bmatrix} 1 & T & \frac{T^2}{2} \\ 0 & 1 & T \\ 0 & 0 & 1 \end{bmatrix} x_k + q(t) \tag{3.21}$$

The CA model is particularly appropriate for scenarios in which a vehicle undergoes sustained acceleration, such as when merging onto a highway. Alternative parameterizations of the state vector and corresponding transition equations for the CA model are discussed in [122].

3. Random Motion (RM) Model

This motion model is employed when a detection is likely to correspond to clutter, a stationary road element such as a traffic sign, or an object exhibiting small, irregular displacements, such as a

pedestrian. Spurious motion may also arise from over-segmented clusters caused by occlusions or shape variations across consecutive frames, which can create the appearance of random movement. To account for such cases, a random-motion (RM) or stationary model with relatively large process noise qis used. In this formulation, the state transition function is effectively the identity mapping, so that the predicted state equals the previous state up to stochastic perturbations. Let the state vector be defined as in (3.22).

$$x(t) = [x(t)\ y(t)\ \theta(t)\ v(t)\ \omega(t)]^T \tag{3.22}$$

The transition function has the same expression as the state vector, to which a large motion noise is added (3.23).

$$x(t+T) = x(t) + q(t) \tag{3.23}$$

b. Rotational Kinematics

1. Coordinated turn (CT) motion model

This motion model is particularly suited to situations in which an object follows a smooth curved trajectory, such as during turning maneuvers [127]. It is especially applicable in radar-based tracking, where both position and velocity of surrounding vehicles can be directly observed. Unlike the constant velocity model, the constant turn (CT) model does not assume a fixed linear velocity. Instead, it postulates a constant yaw rate, corresponding to a fixed steering angle. Under this assumption, the vehicle is constrained to move along a circular arc. A convenient state representation for this model is given in (3.24), where $p_x(t)$ and $p_y(t)$ denote the Cartesian position, $v(t)$ the speed, $\phi(t)$ the heading angle, and $\omega(t)$ the yaw rate.

$$x(t) = \left[p_x(t), p_y(t), v(t), \emptyset(t), \omega(t)\right]^T \tag{3.24}$$

In the time continuous approach, the model is described using the transition function (3.25).

$$\begin{bmatrix} p_x(t) \\ p_y(t) \\ v(t) \\ \emptyset(t) \\ \omega(t) \end{bmatrix} = \begin{bmatrix} v(t)\cos(\emptyset(t)) \\ v(t)\sin(\emptyset(t)) \\ 0 \\ \omega(t) \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} q_c^v(t) \\ q_c^\omega \end{bmatrix} \tag{3.25}$$

The first two equations express that the object moves in the direction defined by the heading $\phi(t)$ with speed $v(t)$. The third equation specifies that the time derivative of $v(t)$ is zero, up to an additive noise term, reflecting small stochastic variations in speed. The evolution of the heading $\phi(t)$ and the yaw rate $\omega(t)$ follows dynamics analogous to those of a constant-velocity model.
To obtain a discrete-time formulation suitable for filtering, either the Euler discretization or the exact analytical solution may be applied to the continuous-time model.

2. Constant turn rate and velocity (CTRV) model

The Constant Turn Rate and Velocity (CTRV) model [128] and the Constant Turn Rate and Acceleration (CTRA) model belong to the class of curvilinear motion models and are capable of predicting vehicle trajectories during both straight-line motion and turning maneuvers. These models are particularly well suited for representing the kinematics of road vehicles whose motion includes smooth changes in direction. A typical state representation for these models is given in (3.26) and comprises the Cartesian position components, the speed (i.e., the magnitude of the velocity), the orientation $\phi(t)$, and the yaw rate $\omega(t)$.

$$x(t) = \left[p_x(t), p_y(t), v(t), \emptyset(t), \omega(t)\right]^T \tag{3.26}$$

The discrete-time process model used to predict the vehicle state at the next time step is given in (3.27). As implied by the model name, both the turn rate and the velocity (or acceleration, in the CTRA case) are assumed to remain constant between successive time instants, subject to additive stochastic perturbations. The terms $\nu_{\phi,k}$ and $\nu_{a,k}$ denote zero-mean Gaussian process noise components that capture deviations from the idealized motion assumptions.

$$x_{k+1} = x_k + \begin{bmatrix} \frac{v_k}{\omega_k}(\sin(\emptyset_k + \omega_k \Delta t) - \sin(\emptyset_k)) \\ \frac{v_k}{\omega_k}(-\cos(\emptyset_k + \omega_k \Delta t) + \cos(\emptyset_k)) \\ 0 \\ \omega_k \Delta t \\ 0 \end{bmatrix} \begin{bmatrix} \frac{1}{2}(\Delta t)^2 \cos(\emptyset_k)\, \nu_{a,k} \\ \frac{1}{2}(\Delta t)^2 \sin(\emptyset_k)\, \nu_{a,k} \\ \Delta t \nu_{a,k} \\ \frac{1}{2}(\Delta t)^2 \nu_{\emptyset,k} \\ \Delta t \nu_{\emptyset,k} \end{bmatrix} \tag{3.27}$$

The CTRV model is often referred to as the bicycle model, as it reduces the four-wheel kinematics of a vehicle to an equivalent two-wheel representation. This abstraction is generally sufficient for capturing the planar motion of road vehicles in tracking applications.

3. Controlled turned rate and acceleration (CTRA) model

In comparison with the CTRV formulation, the CTRA model augments the state vector with an explicit acceleration component [122]. The resulting state representation is given in (3.28).

$$x(t) = \left[p_x(t)\ p_y(t)\ \emptyset(t)\ v(t)\ a(t)\ \omega(t)\right]^T \tag{3.28}$$

The continuous time transition function can be observed in (3.29).

$$x(t+T) = x(t) + \begin{pmatrix} \Delta x(T) \\ \Delta y(T) \\ \omega T \\ aT \\ 0 \\ 0 \end{pmatrix} + q(t) \tag{3.29}$$

In (3.29), the terms $\Delta x(T)$ and $\Delta y(T)$ denote the positional increments that are added to the previous state in order to obtain the updated position; their explicit expressions are given in (3.30) and (3.31). The term q(t) represents the process noise. A detailed derivation of the process noise model for the CTRA formulation is provided in [129].

$$\Delta x(T) = \frac{1}{(\omega)^2}[v(t)\omega + a\omega T)\sin\left(\emptyset(t)\right) + \omega T) + \text{a}\cos(\emptyset(t) + \omega T) - v(t)\omega\sin\emptyset(t) - \text{a}\cos\emptyset(t) \tag{3.30}$$

$$\Delta y(T) = \frac{1}{(\omega)^2}[(-v(t)\omega - a\omega T)\cos(\emptyset(t) + \omega T) + \text{a}\sin(\emptyset(t)\ \omega T)\ v(t)\ \omega\cos\left(\emptyset(t)\right) - \text{a}\sin\left(\emptyset(t)\right)] \tag{3.31}$$

The set of motion models discussed above is not exhaustive. Numerous additional formulations have been proposed in the literature [122, 130, 131, 132, 133], all of which can be broadly categorized into linear and rotational (curvilinear) kinematic models and are derived from the fundamental principles outlined here. It should be emphasized that no single motion model can adequately describe all possible object behaviors. Consequently, improved prediction performance is often achieved by combining multiple motion models within a unified framework, while carefully managing computational complexity to preserve the real-time capabilities of the tracking system.

3.2.2.2 Measurement Models

In contrast to motion models, measurement models are specific to the sensing modalities employed. They describe how the object state x_k is mapped to the corresponding observation z_k, a relationship that can be expressed probabilistically as $p(z_k \mid x_k)$. The deterministic component of this mapping is represented by the function $h_k(\cdot)$, which relates the latent state to the expected measurement.

No universal measurement model exists that is applicable across all sensors; instead, each sensing technology requires a model that reflects its physical measurement process. As with motion models, measurement models incorporate additive noise, as shown in (3.32). This noise is typically assumed to be zero-mean Gaussian with covariance matrix R_k.

$$z_k = h_k(x_k) + r_k \tag{3.32}$$

The measurement function may be either linear or nonlinear, depending on the characteristics of the sensing modality. Autonomous systems typically rely on a diverse set of sensors, including cameras, LiDAR, RADAR, global navigation satellite systems (e.g., GPS), gyroscopes, and accelerometers. In the following, representative measurement models for several of these sensors are presented.

LiDAR

A LiDAR sensor produces a point cloud in which each return corresponds to a three-dimensional spatial location. For simplicity, it is assumed that, after object detection has been performed on the point cloud, each object is represented by its planar position, with vertical motion neglected. Accordingly, the measurement vector is defined as $\mathbf{z}_k = (p_x, p_y)^\top$. If the corresponding state vector is given by $\mathbf{x}_k = (p_x, p_y, v_x, v_y)^\top$, the linear measurement matrix $\mathbf{H}$ that maps the four-dimensional state to the two-dimensional observation space is given in (3.33) [134].

$$z_k = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} x_k + r_k \tag{3.33}$$

RADAR

RADAR sensors are particularly valuable because, through the Doppler effect, they provide direct measurements of an object's radial velocity, that is, the component of the velocity along the line of sight to

the sensor. Rather than delivering Cartesian position estimates, RADAR returns the range ρ(distance from the sensor), the bearing ϕ(angle relative to the forward axis), and the radial velocity $\dot{\rho}$. Using the same state vector $\mathrm{x}_k = (p_x, p_y, v_x, v_y)^\top$ and the measurement vector $\mathrm{z}_k = (\rho, \phi, \dot{\rho})^\top$, the nonlinear measurement function $h(\mathrm{x})$ that maps the state to the RADAR observation space is given by (3.34) [134].

$$h(x_k) = \begin{pmatrix} \sqrt{p_x^2 + p_y^2} \\ \arctan(\frac{p_y}{p_x}) \\ \frac{p_x v_x + p_y v_y}{\sqrt{p_x^2 + p_y^2}} \end{pmatrix} \tag{3.34}$$

It can be seen that, in the RADAR case, the measurement function $h(\cdot)$ is nonlinear. Consequently, unlike the LiDAR formulation, no constant measurement matrix Hexists for this sensor model.

Gyroscope

A gyroscope provides measurements of the system's orientation and angular velocity, in particular the yaw rate. Consider a discrete-time state vector of the form $\mathrm{x}_k = [p_x, p_y, v, \phi, \omega]^\top$, analogous to that used in the constant-turn motion model. In this case, the gyroscope yields a noisy observation of the yaw rate ω. The corresponding relationship between the state and the measurement is given in (3.35).

$$z_k = [0 \quad 0 \quad 0 \quad 0 \quad 1]x_k + r_k \tag{3.35}$$

Wheel speed encoders

Wheel encoders are commonly used to measure motion in mobile robots and ground vehicles [135]. These sensors rely on either optical or magnetic principles to generate digital signals proportional to wheel rotation. Optical encoders consist of a patterned disk and a light source whose interruptions are detected by photodiodes, producing a sequence of pulses as the disk rotates. Magnetic encoders, in contrast, sense variations in a magnetic field induced by the movement of a ferromagnetic element. Owing to their higher speed, accuracy, and reliability, optical encoders are widely employed in industrial and robotic applications.

Two main types of optical encoders are in common use: absolute and incremental. Absolute encoders provide a direct measurement of the

angular shaft position, whereas incremental encoders generate pulses for each incremental rotation of the shaft, thereby enabling the estimation of rotational velocity by counting pulses relative to a reference index [136].

In tracking applications, wheel encoders typically provide a noisy measurement of the vehicle's speed. For a state vector $\mathrm{x}_k = [p_x, p_y, v_x, v_y]^\top$, the corresponding measurement function at time step $k$is given in (3.36).

$$z_k = \sqrt{v_x^2 + v_y^2} + r_k \tag{3.36}$$

A broad variety of measurement models for different sensing modalities has been proposed in the literature [136]; however, a comprehensive survey of these formulations is beyond the scope of this book. In all models considered here, the measurement noise r_k is assumed to follow a zero-mean Gaussian distribution with covariance R_k, that is, $\mathrm{r}_k \sim \mathcal{N}(0, \mathrm{R}_k)$. The measurement covariance matrix R_k is typically derived from the technical specifications of the corresponding sensor.

3.2.2.3 Clutter Model

A complete measurement model must also account for clutter, a term commonly used in the literature to denote false detections. Such spurious measurements may originate from sensor noise or from environmental structures that are incorrectly interpreted as objects by the sensing system. At each discrete time step k, the set of measurements can be viewed as a random permutation of true object-generated detections, denoted $\mathcal{O}_k$, and clutter measurements, denoted $\mathcal{C}_k$. The resulting measurement matrix is therefore formed by permuting the columns corresponding to object detections and clutter, as expressed in (3.37), where $\Pi(\cdot)$ denotes a permutation operator.

$$Z_k = \boldsymbol{\Pi}(O_k, C_k) \tag{3.37}$$

One of the most widely used models for clutter is the Poisson point process [137, 138]. The Poisson distribution is a discrete probability law that describes the number of occurrences of an event within a specified interval or spatial region. In the context of autonomous systems, this region is typically identified with the sensor's field of view. The Poisson model is well suited to clutter modeling because it

satisfies two key assumptions that are generally valid for false detections. First, clutter measurements are assumed to occur independently, which is consistent with the common assumption that the elements of the clutter set $\mathcal{C}_k$ are independent and identically distributed, often uniformly over the field of view. Second, the probability of an event occurring within a given region is assumed to be stationary over time, meaning that the expected clutter rate does not vary. The probability mass function of the Poisson distribution is given in (3.38). In this distribution, the mean and the variance coincide and are both equal to the parameter λ, which represents the expected number of clutter measurements.

$$P(X = x) = \frac{\lambda^x e^{-1}}{x!} \tag{3.38}$$

A Poisson point process may be parameterized either by an intensity function $\lambda_c(\mathrm{c})$ or, equivalently, by the combination of a Poisson rate $\bar{\lambda}_c$ and a spatial probability density function $f_c(\mathrm{c})$, as expressed in (3.39).

$$\begin{cases} \bar{\lambda}_c = \int \lambda_c(c)dc \\ f_c(c) = \frac{\lambda_c(c)}{\bar{\lambda}_c} \end{cases} \tag{3.39}$$

The resulting expression for the clutter distribution under the Poisson point process model, for the clutter set $\mathcal{C}_k = \{\mathrm{c}_k^1, \mathrm{c}_k^2, \dots, \mathrm{c}_k^{m_k^c}\}$, is given in (3.40), where m_k^c denotes the number of clutter measurements at time k. In the general case, the spatial density $f_c(\mathrm{c})$ is not required to be uniform, allowing for nonhomogeneous clutter distributions.

$$p(C_k) = Po\left(m_k^c; \bar{\lambda}_c\right) \prod_{i=1}^{m_k^c} f_c\left(c_k^i\right) \tag{3.40}$$

Further details on the use of the Poisson point process for clutter modeling are provided in [138].

3.2.3 Different Data Association Approaches

One of the most challenging aspects of multi-object tracking is that the correspondence between measurements and objects is unknown. Given a measurement set $\mathcal{Z}_k$ of cardinality m_k, which contains both true detections and clutter, the data association for object $i$with state x_k^i is denoted by θ_k^i. This association variable is defined as in (3.41).

$$\theta_{\mathrm{k}}^{i} = \begin{cases} j, if\ object\ i\ is\ associated\ to\ measurement\ j \\ 0, if\ i\ is\ undetected\ and\ all\ vectors\ are\ clutter \end{cases} \quad (3.41)$$

If n tracks are present at time step k, the corresponding association vector Θ_k is defined as in (3.42).

$$\Theta_{\mathrm{k}} = [\theta_{\mathrm{k}}^{1}\ \theta_{\mathrm{k}}^{2} \ldots . \theta_{\mathrm{k}}^{i} \ldots \theta_{\mathrm{k}}^{n}] \quad (3.42)$$

All tracking approaches considered in this book adopt the point-object assumption. Under this hypothesis, each measurement is assumed to originate from a single source, which may correspond either to a real physical object or to clutter. In addition, each real object is assumed to generate at most one detection at each time step. Under this assumption, two association constraints are imposed to ensure a valid data association. For any association variable θ_k^i belonging to the association vector Θ_k, each object must be either detected or missed at time k, as formalized in (3.43).

$$\theta_{\mathrm{k}}^{i} \epsilon \{0 \ldots\ m_k\}, \forall\ i\ \in \{1 \ldots n\} \quad (3.43)$$

The second constraint stipulates that no single measurement may be associated with more than one tracked object. Under the assumptions that the sensor, motion, and measurement models are linear and Gaussian, the exact posterior density can be expressed as a Gaussian mixture, with one mixture component corresponding to each feasible association at time k, as shown in (3.44). The coefficient $\widetilde{w}^{(}\theta_k^i|h)$ represents a probability mass function that encodes the likelihood of a particular measurement-to-track association. The terms $w_{k|k-1}^{h}$ and $p_{k|k}^{(h|\theta_k)}$ denote, respectively, the weight of the predicted mixture component from the previous iteration and the updated posterior density obtained through the Bayesian recursion.

$$P_{k|k}(x_k) \propto \sum_h \sum_{\theta_k \in \Theta_{\mathrm{k}}} \widetilde{w}^{\theta_k|h}\, w_{k|k-1}^{h} p_{k|k}^{h|\theta_k}(x_k) \quad (3.44)$$

Because the number of possible data associations grows combinatorially, the exact evaluation of all hypotheses becomes computationally intractable. Consequently, practical tracking systems avoid exhaustive enumeration and instead cast data association as an optimization problem, in which only a subset of highly probable hypotheses is retained. The selected subset, denoted θ^*, consists of

those associations in Θ_k that have the largest weights $\tilde{w}^{(\theta_k^i|h)}$. From a mathematical perspective, this problem can be formulated by constructing a cost matrix, whose entry at row i and column j represents the cost of associating measurement i with track j. An accompanying assignment matrix, composed of binary elements, is then used to encode the association decisions: a value of 1 at position (i, j) indicates that measurement i is assigned to track j, whereas a value of 0 indicates no association. The optimization objective is to select the assignment that minimizes the overall cost according to a specified criterion. Accordingly, for each track, the optimal association θ^* is obtained by minimizing the cost function defined in (3.45).

$$\theta^* = \arg\min_{\theta\in\Theta} \sum_{i=1}^{n} -\log(\tilde{w}^{\theta_k^i|h}) \tag{3.45}$$

The cost of associating a measurement with a tracked object can be computed in various ways, depending on the application and sensing modality. Two principal paradigms are commonly distinguished. In feature-based approaches, carefully designed descriptors are extracted from the measurements in order to characterize their similarity to predicted object states. In data-driven approaches, neural networks are employed to learn such features directly from data. Each paradigm offers distinct advantages and limitations with respect to robustness, generalization, and computational complexity.

3.2.3.1 Feature Engineering-Based Tracking Methods

Most online multi-object tracking systems adopt a tracking-by-detection paradigm, in which a detector generates object hypotheses and a data association mechanism links these hypotheses across consecutive frames. When computing similarity costs between detections and existing tracks, appearance and motion cues constitute the primary sources of information.

In appearance-based association, early approaches relied on distance measures between color histograms [139]. A related method was introduced in [140], where similarity was evaluated using the chi-square distance between grayscale histograms and the cosine distance between spatio-temporal locations of detections and tracks. In [141], a more elaborate appearance similarity function was designed by combining multiple cues, including object dimensions and color histograms. Motion similarity was computed using the Euclidean norm,

and the resulting scores were fused with the appearance-based metric. The association problem was then solved using the Hungarian algorithm [142] after converting images to the HSV color space.

In [143], an aggregated local flow descriptor encoding the relative motion patterns of bounding box detections across frames was proposed and combined with additional features for data association. Similarly, [144] introduced a cost function based on multiple cues—such as histogram of oriented gradients (HOG), bounding box width and height, and intersection-over-union (IoU)—to enable robust tracking of objects in thermal imagery. Bertozzi [145] addressed camera motion effects by applying image stabilization, followed by pedestrian detection based on warm, symmetric objects with specific size and aspect ratio constraints. Alternative thermal tracking approaches detect pedestrians as hot regions; for instance, the HotSpot tracker performs pixel-intensity thresholding and tracks detections using a Kalman filter with a global nearest-neighbor association strategy [146, 147].

A weighted combination of positional, size, and appearance cues was proposed in [148]; however, the simplicity of the appearance model can lead to association failures under significant target overlap. Yu et al. [149] exploited edge features and their orientations, transformed into the Fourier domain, to achieve real-time tracking. Another thermal-image-based tracker [150] employed edge features together with a high-dimensional intensity histogram descriptor to compute association costs.

3.2.3.2 Data-Driven Tracking Methods

Recent literature has increasingly favored the use of deep convolutional neural networks (CNNs) for learning discriminative object representations in tracking applications. A common strategy is to employ metric learning, whereby the network is trained to map observations of the same object to nearby points in an embedding space, while representations of different objects are mapped farther apart. Formally, this implies that the distance between a reference sample and a positive instance (the same object observed at a different time) is minimized, whereas the distance to a negative instance (a different object) is maximized. This objective is typically achieved using contrastive loss [151] or triplet loss [152].

To improve training efficiency and convergence, several optimization techniques have been proposed. Hard-negative mining [207] restricts

training to challenging positive and negative samples rather than full images, while group loss [153] leverages sets of samples from the same class within a training iteration to improve discrimination between positive and negative examples.

Within the tracking-by-detection paradigm, metric learning is commonly employed in two principal ways. In the first approach, object detection and feature embedding are performed by separate networks that are trained independently. An example is presented in [154], where object detections are generated using a Faster R-CNN detector fine-tuned on ImageNet, while appearance embeddings are extracted using a network inspired by GoogLeNet. Object affinity is computed using cosine similarity, and training employs a combination of softmax and triplet loss functions.

The second approach performs joint detection and embedding extraction within a single network architecture, enabling the sharing of low-level features and avoiding redundant computation. A representative example is given in [155], where a feature pyramid network (FPN) is employed to generate multi-scale predictions. High-level semantic features are progressively upsampled and fused with lower-level features via skip connections. The network outputs three parallel heads corresponding to object classification, bounding-box regression, and dense appearance embeddings, with triplet loss used for metric learning.

Despite their effectiveness, purely data-driven methods may fail when training datasets do not adequately capture real-world variability. To address this limitation, hybrid approaches combining learned representations with engineered features have been proposed. For instance, in [156], spatio-temporal features learned via CNNs are combined with contextual cues using gradient boosting. The CNN inputs consist of optical flow and normalized LUV color channels, while contextual features encode relative changes in position, size, and velocity.

Broadly, metric-learning-based tracking approaches fall into three categories: (i) detection followed by embedding extraction, (ii) joint detection and embedding, and (iii) hybrid methods combining learned and engineered features.

Several specialized solutions have been developed for robustness in challenging sensing conditions. The TCNN framework [157] maintains appearance stability during online learning through a tree-structured ensemble of CNNs. The SRDCFir tracker [158], an extension of SRDCF

for thermal imagery, introduces a regularization term that constrains filter coefficients to the target region, yielding a more discriminative appearance model. In addition to HOG features [159], SRDCFir incorporates channel-coded intensity and motion features.

An increasingly popular paradigm in visual and thermal tracking relies on similarity verification using Siamese networks [160]. These architectures consist of two identical branches with shared parameters that compute a similarity score between a track and a detection. Liu et al. [161] proposed a multi-layer fusion Siamese network for thermal imagery that integrates optical flow and combines shallow and deep features. Zhang et al. [162] introduced a multi-stage feature fusion network that combines a region proposal network with a spatial transformer for thermal object tracking. While SiamFC [160] achieves real-time performance, its lack of online adaptation limits accuracy. This limitation is addressed in DSiamM [163], which integrates correlation filters to enable online updates.

Other approaches decompose robustness and discrimination into separate processing stages. In [164], multiple Siamese AlexNet architectures [165] are trained independently and their outputs fused to enhance association reliability. In [166], a synthetic thermal dataset is generated from RGB imagery to enable end-to-end training of a Siamese model [167] for thermal feature extraction.

Beyond bounding-box tracking, multi-object tracking and segmentation methods aim to assign a persistent instance mask to each object across frames. SiamMask [168] augments Siamese tracking with a binary segmentation task, producing class-agnostic masks in real time from a single bounding-box initialization. Another approach [169] extends Mask R-CNN with 3D convolutions to incorporate temporal context and an association head that produces embedding vectors for detection matching. MOTSFusion [170] combines tracking, segmentation, and dynamic object fusion by constructing short tracklets using optical flow and merging them into 3D object reconstructions, enabling localization under occlusion or missed detections.

Inspired by advances in natural language processing [171], transformer-based architectures have recently been adopted for detection and tracking. Transformers rely on self-attention and cross-attention mechanisms [172] to model long-range dependencies and can operate directly on input sequences without handcrafted feature extraction [173]. Many transformer-based approaches perform joint detection and tracking in a single stage. For example, [174] introduces a

multitask regression-based framework, while JDE [175] uses a shared backbone for both detection and appearance embedding. TransTrack [176] propagates object identities across frames using a query–key mechanism. In [177], a transformer-based architecture is proposed for 3D object tracking, where self-attention models intra-point-cloud relations and cross-attention fuses multimodal features.

Despite their promising performance, transformer models typically require substantial computational resources, both during training and inference. Their reliance on multi-GPU training and high-performance hardware makes them less suitable for deployment on edge devices or in systems with strict real-time constraints.

Finally, several works pursue end-to-end learning of data association. In [178], camera and LiDAR data are jointly exploited to produce accurate trajectories, with detection and tracking optimized together. A key contribution is the backpropagation of gradients through a linear programming layer during training. The matching cost integrates both appearance and motion cues using a Siamese CNN architecture.

3.2.3.2 Optimization Algorithms for Data Association

After similarity scores between tracks and detections have been computed, the assignment cost must be minimized subject to two constraints. First, each tracked object must be associated either with exactly one detection or with a missed detection. Second, each detection may be associated with at most one track. All data association optimization methods share a common structure: they take as input a cost matrix and return an assignment matrix that encodes a valid set of associations minimizing the overall assignment cost. Two main classes of solvers are commonly distinguished.

The first class seeks a single globally optimal assignment for all tracks, that is, it finds the association θ^* such that $\widetilde{w}^{(\theta^*|h)} \geq \widetilde{w}^{(\theta|h)}, \forall \theta \in \Theta$. Representative algorithms in this category include the Hungarian (Kuhn–Munkres) algorithm [142], the Auction algorithm [179], the Jonker–Volgenant–Castanon (JVC) algorithm [180], and the Ford–Fulkerson method [181].

The second class of solvers computes not just a single solution but the best m association hypotheses for a given set of tracks. The Murty algorithm [182] enumerates and ranks the m lowest-cost assignments in ascending order of total cost. The Gibbs sampling approach [183], a

Markov chain Monte Carlo method, provides a computationally efficient but suboptimal alternative for generating multiple high-probability associations. Its main limitation is the lack of guarantees that the selected m hypotheses include the globally optimal ones; some valid associations may be discarded to improve computational efficiency. Nevertheless, Gibbs sampling has been shown to produce competitive tracking performance in practice.

Numerous extensions and variants of these algorithms have been developed to address different optimization requirements in tracking systems [184, 185, 186].

In [187], the data association problem is formulated as a minimum-cost flow optimization. In this framework, each detection is represented by two nodes in a weighted directed graph, connected by observation edges and transition edges. Observation edges are assigned negative weights proportional to detection confidence, while transition edges encode positive costs that distinguish different object trajectories. A source node connects to detections when tracks are initiated, and a sink node is connected when tracks terminate. This formulation requires access to all frames and is therefore unsuitable for real-time operation.
A real-time variant of this approach is proposed in [188], where a dynamic version of the successive shortest path algorithm is employed. However, this method does not explicitly handle occlusions and relies on a constant-position motion model, which is overly simplistic for complex traffic scenarios.

3.2.4 Data Association Filters

In multi-object tracking, the objective is to process the set of detected objects, represented by the measurement set $\mathcal{Z}_k$, and to estimate the posterior density of each object state. This posterior is expressed as a weighted mixture over all feasible data association sequences, where each weight represents the probability of a particular association hypothesis, and each component corresponds to a state density conditioned on that hypothesis, as shown in (3.46) [138].

$$\begin{aligned} p_{(k|k)} &= \sum_{\theta_{1:k} \in \Theta_{1:k}} w_{(k|k)}^{\theta_{1:k}} p_{(k|k)}^{\theta_{1:k}}(X_k) \\ &= \sum_{\theta_{1:k} \in \Theta_{1:k}} \Pr[\theta_{1:k}|Z_{1:k}]\, p[X_k|\theta_{1:k}, Z_{1:k}] \end{aligned} \tag{3.46}$$

Because the number of feasible data association hypotheses grows rapidly, evaluating the exact posterior density given by (3.46) is computationally intractable for real-time applications such as autonomous systems. Consequently, practical tracking algorithms rely on approximations that control this combinatorial growth. Two fundamental strategies are employed for this purpose: pruning and merging.

In pruning, hypotheses with low posterior probability are discarded. From a computational perspective, this corresponds to setting small mixture weights to zero and renormalizing the remaining weights. In merging, multiple hypotheses are combined into a single representative hypothesis, thereby approximating the mixture by a reduced set of components or even by a single density. This combination is typically performed by minimizing a suitable divergence measure, such as the Kullback–Leibler divergence [189, 190].

In most practical systems, pruning and merging are applied jointly, with their relative emphasis determined by the accuracy and computational constraints of the application. Three representative algorithms are outlined below to illustrate different uses of these two strategies.

Global Nearest Neighbour (GNN)

The Global Nearest Neighbor (GNN) filter [191] follows a greedy strategy in which only the association hypothesis with the largest weight, θ^*, is retained, while all other hypotheses are pruned. The main advantage of this approach is its low computational complexity, which makes it suitable for deployment on resource-constrained embedded platforms. Under the GNN formulation, the posterior density is approximated by a single component corresponding to the sequence of optimal data associations selected up to time k, as expressed in (3.47).

$$p_{(k|k)}^{GNN}(X_k) = p_{(k|k)}^{\theta_{1:k}^*}(X_k) \tag{3.47}$$

At each time step of the GNN recursion, the algorithm begins with a prediction step based on the Chapman–Kolmogorov equation. When Gaussian densities and linear-Gaussian motion models are assumed, this prediction is implemented using the Kalman filter for each track; the Kalman filter and its variants are discussed in the following section.

After the cost matrix has been computed, the update phase starts by determining the optimal data association. For each track, if a detection is assigned, a Bayesian measurement update is performed; if no detection is associated, the posterior density remains proportional to the predicted density.

In addition to its simplicity and low computational cost, the GNN filter performs well in scenarios characterized by high signal-to-noise ratio, high detection probability, low clutter rate, and small measurement uncertainty.

Joint Probabilistic Data Association Filter (JPDA)

In the Joint Probabilistic Data Association (JPDA) algorithm [192], multiple association hypotheses are incorporated by merging their marginal contributions rather than selecting a single hypothesis. This is achieved by first computing the marginal association probabilities for each object across all feasible hypotheses.

The JPDA framework was designed to retain much of the computational efficiency of the GNN filter while exploiting information from multiple competing hypotheses instead of discarding them outright. As a result, the posterior density in JPDA is approximated by a single effective density, obtained by weighting the updates with the marginal association probabilities $\beta_{1:k}$, as shown in (3.48).

$$p_{(k|k)}^{JPDA}(X_k) = p_{(k|k)}^{\beta_{1:k}}(X_k) \tag{3.48}$$

The marginal probability that object $i$is associated with detection j at time k is denoted by $\beta_k^{(i,j)}$and is computed as given in (3.49).

$$\beta_k^{i,j} = \Pr\left(\theta_k^i = j \middle| Z_{1:k-1}\right) = \sum_{\theta_k \in \Theta_k : \theta_k^i = j} w^{\theta_k} \tag{3.49}$$

To mitigate the computational burden associated with summing over all valid association hypotheses, measurement gating is typically applied prior to the evaluation of marginal association probabilities, thereby reducing the number of admissible terms.

The posterior density for object $i$is then obtained as in (3.50), where $p_{k|k-1}^i(\mathrm{x}^i)$ denotes the predicted (prior) density, and $p_{k|k}^{(i,j)}$ represents the posterior density resulting from updating this prior with measurement z_k^j.

$$p^{i}_{(k|k)}(X_k) = \beta_k^{i,0} p^{i}_{(k|k-1)}(X^i) + \sum_{j=1}^{m_k} \beta_k^{i,j} p^{i,j}_{(k|k-1)}(X^i) \qquad (3.50)$$

Finally, for each object, the marginal posteriors are merged into a single density by minimizing the Kullback–Leibler divergence [190]. For linear-Gaussian models, the prediction step remains identical to that of the standard Kalman filter, whereas the update step is modified to account for multiple potential associations. Given the predicted mean and covariance, the updated mean $\mu^{i}_{k|k}$ for object $i$is obtained by first computing an innovation $\varepsilon_k^{(i,j)}$ for each measurement $j$that lies within the validation gate of the track, as defined in (3.51). These innovations are then combined into a single effective innovation ε_k by forming a weighted sum, where the weights are the marginal association probabilities, as shown in (3.52). Finally, the posterior mean $\mu^{i}_{k|k}$ is obtained by adding to the predicted mean the product of the Kalman gain and the effective innovation, as given in (3.53).

$$\varepsilon_k^{i,j} = z_k^j - H\mu^{i}_{(k|k-1)} \qquad (3.51)$$

$$\varepsilon_k = \sum_{j=1}^{m_k} \beta_k^{i,j} \varepsilon_k^{i,j} \qquad (3.52)$$

$$\mu^{i}_{(k|k)} = \mu^{i}_{(k|k-1)} + K_k^i \varepsilon_k \qquad (3.53)$$

The updated covariance is given by (3.54). The term $\beta_k^{(i,0)} \mathrm{P}^{i}_{k|k-1}$ accounts for the probability of a missed detection and represents the contribution of the predicted covariance, that is, the covariance obtained when no measurement is available to update the state.

$$P^{i}_{(k|k)} = \beta_k^{i,0} P^{i}_{(k|k-1)} + (1 - \beta_k^{i,0})\overline{P_k^i} + \widetilde{P_k^i} \qquad (3.54)$$

The term $(1 - \beta_k^{(i,0)})\, \bar{\mathrm{P}}_k^i$ represents the probability that the object is associated with one of the available detections and the corresponding covariance after the predicted covariance has been updated using the measurement information. The expression for $\bar{\mathrm{P}}_k^i$ is given in (3.55).

$$\overline{P_k^i} = P^{i}_{(k|k-1)} - K_k^i \left(H P^{i}_{(k|k-1)} H^T + R\right) \left(K_k^i\right)^T \qquad (3.55)$$

The final term $\widetilde{\mathrm{P}}_k^i$ is small when the gated measurements are highly consistent and clustered, and increases when they are more dispersed. Its analytical form is given in (3.56).

$$\widetilde{P_k^i} = K_k^i \left(\left[\sum_{j=1}^{m_k} \beta_k^{i,j} \varepsilon_k^{i,j} \left(\varepsilon_k^{i,j} \right)^T \right] - \varepsilon_k \varepsilon_k^T \right) \left(K_k^i \right)^T \quad (3.56)$$

The JPDA algorithm is computationally efficient, although not as lightweight as the GNN filter. Its primary advantage lies in its superior performance in environments with lower signal-to-noise ratios and higher ambiguity. A notable limitation, however, is that JPDA may exhibit degraded performance when multiple objects are in close proximity, as the marginalization process can lead to track coalescence. Its computational cost is also higher than that of GNN due to the additional complexity involved in computing and merging marginal association probabilities. Numerous variants of JPDA have been proposed to approximate the marginal probability computations more efficiently, including Cheap JPDA [193], Nearest Neighbor Cheap JPDA [194], Suboptimal JPDA [195], and Fast JPDA [196]. More recent extensions and applications of the JPDA framework are reported in [197, 198].

Multiple Hypothesis tracker (MHT)

In the Multiple Hypothesis Tracking (MHT) algorithm, the central idea is to retain the m highest-weight association hypotheses and discard all others. After updating these $m$hypotheses, a reduction step is applied to ensure that no more than $N_{\max}$ hypotheses are maintained in the posterior mixture, thereby controlling computational complexity. This reduction is typically achieved through pruning and capping strategies.
The motivation behind MHT is that maintaining multiple high-probability hypotheses provides a more faithful approximation of the true posterior density than retaining only a single hypothesis. Accordingly, the posterior is represented as a mixture of hypotheses, as shown in (3.57), where each hypothesis h_k corresponds to a distinct, valid sequence of data associations.

$$p_{(k|k)}^{MHT}(X_k) = \sum_{h_k=1}^{H_k} w_{(k|k)}^{h_k} p_{(k|k)}^{h_k}(X_k) \quad (3.57)$$

Two principal variants of MHT are described in the literature: hypothesis-oriented (HO) MHT and track-oriented (TO) MHT. In practice, TO-MHT achieves performance comparable to HO-MHT and generally surpasses both JPDA and GNN in challenging scenarios.

Moreover, TO-MHT is more memory-efficient and computationally less demanding than its hypothesis-oriented counterpart.
The main drawback of MHT-based approaches is their higher resource consumption relative to GNN and JPDA. In addition, unless the hypothesis cap $N_{\max}$ is sufficiently large, there is no guarantee that the most probable association sequence is retained in the hypothesis set. A detailed treatment of the two MHT variants is beyond the scope of this book; the interested reader is referred to [199, 200, 201] for comprehensive discussions.

3.2.5 Kalman Filters

3.2.5.1 Linear Gaussian Models

The Chapman–Kolmogorov equation is computationally demanding, and after successive measurement updates the resulting posterior density may belong to a different distribution family than the one assumed at initialization. Only in a limited number of cases does the application of the filtering equations yield posterior distributions with closed-form analytical expressions. The general Bayesian filtering equations are summarized in (3.58) and (3.59).

Prediction: $$p(x_k|y_{1:k-1}) = \int p(x_k|x_{k-1})p(x_{k-1}|y_{1:k-1})dx_{k-1} \quad (3.58)$$

Update: $$p(x_k|y_{1:k}) \propto p(y_k|x_k)p(x_k|y_{1:k-1}) \quad (3.59)$$

There exists, however, one important class of models for which the recursive Bayesian filtering equations admit an exact closed-form solution, namely the class of linear Gaussian models. Under these assumptions, the Kalman filter provides an optimal solution to the filtering problem [138, 202].

The Kalman filter [109] computes the filtering recursion analytically. At each time step, it first evaluates the predicted density $p(\mathrm{x}_k \mid \mathrm{z}_{1:k-1}) = \mathcal{N}(\mathrm{x}_k; \hat{\mathrm{x}}_{k|k-1}, \mathrm{P}_{k|k-1})$, and subsequently the posterior density $p(\mathrm{x}_k \mid \mathrm{z}_{1:k}) = \mathcal{N}(\mathrm{x}_k; \hat{\mathrm{x}}_{k|k}, \mathrm{P}_{k|k})$, which incorporates the current measurement. In both steps, only the mean and covariance—given by (3.60) and (3.61), respectively—need to be propagated.

$$\hat{x}_{(k|k-1)} = A_{k-1}\hat{x}_{(k-1|k-1)} + q_{k-1} \quad (3.60)$$

$$P_{(k|k-1)} = A_{k-1}P_{k-1|k-1}A_{k-1}^T + Q_{k-1} \quad (3.61)$$

The process noise q_{k-1} is assumed to be zero-mean. In the update step, the predicted state is refined using the information provided by the current measurement. The posterior mean $\hat{x}_{k|k}$ and covariance $P_{k|k}$ are computed according to (3.62) and (3.63).

$$\hat{x}_{(k|k)} = \hat{x}_{(k|k-1)} + K_k v_k \tag{3.62}$$

$$P_{(k|k)} = P_{(k|k-1)} - K_k S_k K_k^T \tag{3.63}$$

The posterior mean is obtained by correcting the predicted mean with an additive innovation term. Likewise, the posterior covariance is computed by shrinking the predicted covariance according to the amount of information provided by the new observation. The degree of this reduction is governed by the relative uncertainty of the prediction and the measurement. In (3.62)–(3.66), K_k denotes the Kalman gain, v_k the innovation, and S_k the innovation covariance. Their explicit definitions are given in (3.64)–(3.66), where H_k is the measurement model mapping the state space to the observation space, and R_k is the measurement noise covariance.

The Kalman gain is small when the measurement uncertainty is large, ensuring that unreliable observations have limited influence and that the posterior estimate remains primarily driven by the predicted state.

$$K_k = P_{(k|k-1)} H_k^T S_k^{-1} \tag{3.64}$$

$$v_k = y_k - H_k \hat{x}_{(k|k-1)} \tag{3.65}$$

$$S_k = H_k P_{(k|k-1)} H_k^T + R_k \tag{3.66}$$

In the Kalman filter, the posterior mean is the only quantity that depends explicitly on the realized measurement values. By contrast, the posterior covariance is independent of the specific observation and depends only on the assumed noise statistics and model parameters. Consequently, while the state estimate adapts to the incoming data, the uncertainty of that estimate is governed solely by the model and noise characteristics.

3.2.5.2 Non-Linear Models

If either the motion model used for state prediction or the measurement model is nonlinear, the resulting posterior density is no

longer Gaussian. Under such conditions, the standard Kalman filter is no longer applicable. In this subsection, two approaches for handling nonlinearities within the Kalman filtering framework are discussed.

Extended Kalman Filter (EKF)

One way to address nonlinear motion or measurement models is to linearize them around the current state estimate. This is achieved by approximating the nonlinear function with a linear function that is tangent to it at the mean of the underlying Gaussian distribution.
In the Extended Kalman Filter (EKF), this linearization is performed using a first-order Taylor expansion [202]. The nonlinear function is first evaluated at the current mean, and then approximated by a linear function whose slope is given by the Jacobian with respect to the state. Although the full Taylor expansion in (3.67) contains higher-order terms, only the terms up to the Jacobian are retained to obtain a linear approximation, while all higher-order components are neglected.

$$f(x) \cong f(\mu) + \frac{\partial f(\mu)}{\partial x}(x - \mu) \tag{3.67}$$

The equations of the Extended Kalman Filter (EKF) [203] are structurally similar to those of the standard Kalman filter, as shown in (3.68) and (3.69). The key difference is that the linear state-transition and measurement matrices A_{k-1} and H_k are replaced by their corresponding Jacobian matrices, evaluated at the current state estimate.

$$x_k = f(x_{k-1}, u_k) + w_k \tag{3.68}$$

$$z_k = h(x_k) + v_k \tag{3.69}$$

The terms w_k and v_k denote the process and measurement noise, respectively, and are modeled as zero-mean multivariate Gaussian variables with covariance matrices Q_k and R_k. The vector $u_k$represents an optional control input.

A notable drawback of the EKF is that, because the linearization point changes at every time step, the Jacobian matrices must be recomputed continuously, which can be computationally expensive. Further algorithmic and mathematical details of the EKF in the context of object tracking are provided in [202].

Unscented Kalman Filter (UKF)

To address nonlinear motion and measurement models, the Unscented Kalman Filter (UKF) [204] employs the unscented transformation. The objective of the UKF is to approximate a generally non-Gaussian predicted distribution by a Gaussian, that is, to determine a mean vector and covariance matrix that best represent the transformed state.

This is achieved through the use of sigma points. While propagating an entire probability density through a nonlinear function is analytically intractable, propagating a carefully selected set of representative points is comparatively straightforward. These sigma points are deterministically chosen around the state mean and along each state dimension at distances proportional to the standard deviation, hence their name.

Each sigma point is propagated through the nonlinear function, after which the mean and covariance of the transformed points are computed. Although this procedure yields only an approximation of the true transformed distribution, it has been shown to provide sufficiently accurate estimates for tracking applications. A commonly used heuristic for the number of sigma points is $2n + 1$, where n is the dimension of the state vector. The first sigma point $\mathrm{X}^0_{k|k}$ corresponds to the state mean, as given in (3.70), while for each state dimension two additional points, defined in (3.71) and (3.72), are placed symmetrically about the mean.

$$X^0_{K|K} = X^*_{K|K} \tag{3.70}$$

$$X^i_{K|K} = X^*_{K|K} + \sqrt{(\lambda + n_x)P_{(k|k)}} \tag{3.71}$$

$$X^i_{K|K} = X^*_{K|K} - \sqrt{(\lambda + n_x)P_{(k|k)}} \tag{3.72}$$

The parameter λ controls the spread of the sigma points around the mean, with larger values producing a wider dispersion. The quantity n_x denotes the dimensionality of the state vector. Nonlinear noise terms that appear in the system equations can be incorporated directly into the augmented state, allowing sigma points to be generated for both the state and the noise variables.

During the prediction step, the sigma points are propagated through the nonlinear transition function, and the resulting transformed points are used to compute the predicted mean and covariance. These quantities are obtained by weighted averaging of the propagated sigma points, as given in (3.73) and (3.74).

$$X_{k+1|K} = \sum_{i=1}^{n_\sigma} w_i X_{k+1|k,i} \tag{3.73}$$

$$P_{k+1|K} = \sum_{i=1}^{n_\sigma} w_i (X_{k+1|k,i} - x_{k+1|k})\,(X_{k+1|k,i} - x_{k+1|k})^T \tag{3.74}$$

The weighting coefficients are designed to compensate for the dispersion of the sigma points so that the correct mean and covariance can be recovered. These weights are computed according to (3.75) and (3.76). The quantity n_a denotes the dimension of the augmented state vector, which includes both the state variables and the associated noise terms.

$$w_i = \frac{\lambda}{\lambda + n_a}, i = 0 \tag{3.75}$$

$$w_i = \frac{1}{2(\lambda + n_a)}, i = 2, \dots, n_a \tag{3.76}$$

In the measurement update step, when the measurement function is nonlinear, the same sigma-point-based procedure used in the prediction phase can be applied. To improve computational efficiency, the sigma points generated during the prediction step are typically reused for the update. Considering the nonlinear measurement model defined in (3.77), the transformed sigma points are propagated through this function to compute the predicted measurement statistics.

$$z_{k+1} = h(x_{k+1}) + w_{k+1} \tag{3.77}$$

The predicted measurement mean and measurement covariance are obtained from the transformed sigma points according to (3.78) and (3.79). Here, $Z_{k+1|k,i}$ denotes the measurement sigma points, and R is the measurement noise covariance.

$$z_{k+1|k} = \sum_{i=1}^{n_\sigma} w_i Z_{k+1|k,i} \tag{3.78}$$

$$S_{(k+1|k)} = \sum_{i=0}^{n_{(\sigma)}} wi \left(Z_{(k+1|k,i)} - z_{(k+1|k,i)}\right)\left(Z_{(k+1|k,i)} - z_{(k+1|k,i)}\right)^T + R \tag{3.79}$$

Once a measurement becomes available, the state mean and covariance are updated using the same equations as in the standard

Kalman filter. The principal difference in the UKF lies in the computation of the Kalman gain, which requires evaluation of the cross-covariance between the sigma points in the state space and their counterparts in the measurement space, as defined in (3.80) and (3.81).

$$K_{(k+1|k)} = T_{(k+1|k)} S^{-1}_{(k+1|k)} \tag{3.80}$$

$$T_{(k+1|k)} = \sum_{i=0}^{2n_\sigma} w_i \left(X_{k+1|k,i} - x_{k+1|k}\right)\left(Z_{k+1|k,i} - z_{k+1|k}\right)^T \tag{3.81}$$

Owing to its lower computational cost relative to the EKF and its improved accuracy in handling nonlinear dynamics, the UKF is widely used for LiDAR-based object tracking in driving scenarios, where vehicles may exhibit diverse and complex motion patterns [205, 206].

3.2.6 Semantic Segmentation Datasets for Thermal Images

Because this book presents an enhanced multi-object tracking framework based on panoptic segmentation for thermal imagery, and because existing thermal semantic segmentation datasets do not meet the requirements of this approach, a new state-of-the-art thermal dataset was developed. To situate this dataset within the existing body of work, this section provides a concise review of the most relevant thermal semantic segmentation datasets reported in the literature.

Semantic segmentation plays a central role in environment perception, as it enables the pixel-level identification of objects and scene structures. It also provides valuable information for instance segmentation and for improving data association in multi-object tracking systems. Despite its importance, relatively few annotated datasets exist for the thermal domain, particularly in driving scenarios.
The SODA dataset [208] contains 2,168 thermal images captured with an SC260 FLIR camera and annotated with 20 semantic classes across indoor and outdoor environments. The CrosIR dataset [209] provides thermal road-surface segmentation together with pedestrian bounding boxes and tracking identities. Wang et al. [210] introduced a driver-view pedestrian thermal segmentation dataset comprising 1,031 images at a resolution of 720×480, although this dataset is not publicly available.

In [211], a combined RGB–thermal dataset focused on low-illumination and high-noise conditions is presented, containing 7,063 images annotated into two semantic classes: pedestrians and

background. Xiong et al. [212] proposed a thermal semantic-segmentation dataset for driving scenes consisting of 2,010 images with pixel-level annotations for ten semantic classes collected across diverse environments.

Another dataset is reported in [213], which includes 1,569 aligned RGB–thermal image pairs annotated into eight semantic classes, captured at a resolution of 640×480 using an InfRec R500 camera. Additional thermal datasets exist [213, 214], but they do not provide semantic segmentation labels. A broader comparative survey of semantic-segmentation datasets and methods is available in [215].

3.3 Advanced Data Association and Tracking Methods

This section reviews the principal enhancements in 2D and 3D multi-object tracking. For the sake of clarity and continuity, several equations are restated where necessary, even if they were introduced in earlier sections.

3.3.1 Data Association and Tracking of 3D LiDAR Objects

One of the most challenging components of the object-tracking pipeline is data association, which arises from the fact that the correspondences between detections and tracked objects are unknown. In multi-object tracking, the objective is to estimate the posterior density of the state of each object both accurately and efficiently.

Under the assumptions of linear-Gaussian motion and measurement models, the exact posterior density can be expressed as a Gaussian mixture, with one component for each feasible data-association sequence at time k, as shown in (3.82). The term $p_{k|k}^{(\theta_{1:k})}$ denotes the state probability density conditioned on a specific association history, while $w_{k|k}^{(\theta_{1:k})}$ is the corresponding probability mass that represents the likelihood of that association sequence. The summation $\sum_{\theta_{1:k} \in \Theta_{1:k}}$ indicates that the mixture spans all admissible associations within the validation region of the tracked object.

$$p_{(k|k)} = \sum_{\theta_{1:k} \in \Theta_{1:k}} w_{(k|k)}^{\theta_{1:k}} p_{(k|k)}^{\theta_{1:k}}(X_k) \tag{3.82}$$

To achieve real-time performance, the solution adopts a Global Nearest Neighbor (GNN) strategy, in which only the best association θ^* is retained for each track and all other hypotheses are pruned. This

greedy selection yields a computationally efficient approximation that is compatible with the real-time constraints of autonomous driving systems. Under this assumption, the posterior density is approximated by a single-hypothesis density $p_{k|k}^{\mathrm{GNN}}(\mathrm{X}_k)$, as given in (3.83), where $\theta_{1:k}^{*}$ denotes the sequence of optimal associations from time step 1 to time step k.

$$p_{(k|k)}^{GNN}(X_k) = p_{(k|k)}^{\theta_{1:k}^{*}}(X_k) \tag{3.83}$$

To obtain reliable associations in the presence of measurement noise and feature uncertainty, correspondences between tracks and detections are determined using multiple complementary features, which are aggregated into a single association score.

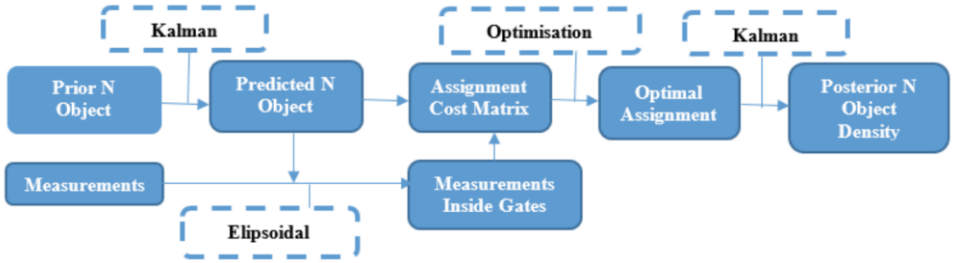

Figure 3.2. The pipeline for multiple objects tracking scenario

State estimation is performed using Kalman prediction and update when the underlying densities are Gaussian; otherwise, nonlinearities are handled through sigma-point sampling within the Unscented Kalman Filter framework. The overall tracking pipeline is illustrated in Figure 3.2. To better capture the kinematics of road users, two motion models are employed: the constant velocity (CV) model and the constant turn rate and velocity (CTRV) model.

3.3.1.1 Data Association Score

The data association score designed for this tracking framework is based on two complementary classes of features. The first class characterizes the appearance of each object, while the second captures the spatial and relational context between a track and its surrounding objects. Each descriptor and the rationale underlying its design are detailed in the following two subsections.

The inputs to the tracking system consist of 3D object bounding boxes extracted from the LiDAR point cloud, together with the associated 3D point measurements, RGB intensity images, and the semantic segmentation of the RGB imagery.

The first data association score presented in the next section, and the corresponding object tracking steps, has been published in [350], while the second data association approach together with its tracking process has been published in [351].

3.3.1.1.1 Aggregated Features Score Method 1

To compute appearance-based similarity between a track and a detection, multiple complementary cues are combined to ensure robust data association. Color and semantic information for each object are obtained by projecting the 3D LiDAR points belonging to each bounding box onto the corresponding RGB image and semantic segmentation map. For color representation, a three-channel reduced color histogram with eight bins per channel is constructed to encode the RGB appearance of each object, resulting in a compact 32-dimensional descriptor. Each projected point contributes a vote to the appropriate bin of the corresponding color channel.

The resulting histogram is then used to compute a color similarity score between LiDAR detection j and track i, as defined in (3.84).

$$CE_{i,j} = RMS\big(LiDAR_j.R, Track_i.R\big) + RMS\big(LiDAR_j.G, Track_i.G\big) + RMS(LiDAR_j.B, Track_i.B) \quad (3.84)$$

The function RMS denotes the root mean square metric, defined in (3.85), where M is the number of histogram bins, and Λ and Ω denote the histogram values of the tracked object and the corresponding detection, respectively.

$$RMS(\Lambda, \Omega) = \frac{1}{M}\sum_{i=0}^{M-1}\big(\Lambda(i) - \Omega(i)\big)^2 \quad (3.85)$$

The 3D points associated with each detection are also projected onto the semantic segmentation image in order to infer the semantic class of the object. Owing to acquisition delays, motion, and sensor synchronization errors, the projected points do not always align perfectly with the true object region, as illustrated in Figure 3.3.

To account for this uncertainty, the three most probable semantic classes together with their associated probabilities are extracted, as defined in (3.86).

$$FC_{i,j} = \sum_{k=0}^{2} \begin{cases} 0, w[k] = -1 \\ |\text{detection}_j \,.\, P(k) - track_i.\, P(w[k])| \end{cases} \tag{3.86}$$

In (3.86), the index $w[k]$ indicates the position at which the semantic class of a detection matches that of the corresponding track. If no such match exists, $w[k] = -1$. The absolute value of a variable $a$is denoted by $|\ a\ |$.

The color and semantic scores computed via (3.84) and (3.86) approach zero when the track and detection are identical or highly similar. Because appearance features derived from Camera–LiDAR fusion may become unreliable under adverse weather or illumination conditions, the proposed framework applies an inverse mapping to camera-based similarity scores. As a result, higher values correspond to greater similarity, while lower values indicate dissimilarity.

When visual information is degraded, the association score is instead computed from geometric features derived from the LiDAR data, including object area, width, length, and the intersection between the predicted track region and the measured detection.

Figure 3.3. Issues that can appear when projecting LiDAR points onto the semantic segmentation image due to lack of synchronization.

To simplify the computation of geometric attributes, all objects are projected onto a 2D top-view plane, and the corresponding parameters are extracted from these planar representations. Objects whose projected areas differ significantly from that of the reference object are discarded a priori, which improves computational efficiency by eliminating implausible associations.

The object size similarity between track $i$and detection j is then computed as in (3.87), where A_T denotes the area of the track and A_D the area of the detection.

$$od_{i,j} = \frac{intersection2DArea}{|AT-AD|} \quad (3.87)$$

The LiDAR-based object orientation is also incorporated, based on the assumption that an object's orientation cannot change abruptly between consecutive frames. The corresponding orientation similarity measure is defined in (3.88).

$$\omega Sim_{i,j} = |detection_j.orientation - track_i.orientation| \quad (3.88)$$

The final data association score between a track and a detection is obtained by summing the similarity measures defined in (3.84), (3.86), (3.87), and (3.88), and weighting this sum by the spatial proximity between the predicted track and the detection. This distance-based weight, denoted wd in (3.89), assigns greater importance to associations between objects that are spatially closer, reflecting their higher likelihood of correspondence.

$$ASc_{i,j} = \frac{\left(\left|\frac{1}{CE_{i,j}+0.0001}\right| + \left|\frac{1}{FC_{i,j}+0.0001}\right| + od_{i,j} + \omega Sim_{i,j}\right)}{wd} \quad (3.89)$$

A positional descriptor is introduced to capture the local relational context between a track and a detection. The underlying assumption is that a detection and its corresponding track should exhibit similar spatial relationships with respect to their neighboring objects.

Accordingly, a positional descriptor PD is computed as the sum of differences in color error (ce) and object dimension (od) between an object and its neighbors within a radius of 10m. Specifically, for a candidate association between track $i$and detection j, the descriptor is first computed between detection j and all tracks located within 10 m of that detection, excluding track i. The same descriptor is then computed between track $i$and all neighboring tracks within the same distance threshold.

If the track and detection correspond to the same physical object, the difference between their positional descriptors should be minimal. This formulation is expressed analytically in (3.90), where n_T and m_T denote the numbers of neighboring tracks within the specified radius around the detection and the track, respectively. Ideally, n_T and m_T should be

equal, indicating that both entities share a consistent local neighborhood structure.

$$PD_{i,j} = \sum_{k=0}^{nT}\left(\left|CE_{i,k}\right| + \left|od_{i,k}\right|\right) - \sum_{k=0}^{mT}(|CE_{j,k}| + |od_{j,k}|) \quad (3.90)$$

The value of the positional descriptor is incorporated into the aggregated similarity score to yield the final association cost. When no neighboring objects are available to compute this descriptor, a small penalty is added to the final score; this penalty was determined empirically and is set to 3.8.

To compute the optimal assignment between tracks and detections based on the features described above, the Hungarian algorithm is employed. The objective is to identify the set of associations that minimizes the total cost defined in (3.91). Here, θ^* denotes the set of optimal associations, and $w^{(\theta_{i,j})}$ represents the aggregated association score between track i and detection j.

$$\theta^* = argmin \sum_{i=0}^{n} -\log(w^{\theta_{i,j}}) \quad (3.91)$$

Supplementary visual results for the first aggregation method are provided in an online demonstration. [1]

3.3.1.1.2 Aggregated Features Score Method 2

An alternative aggregation function is also employed. In addition to the features described above, this formulation incorporates a comparison between the dimensions of the most visible object facades of the detection and the track. The corresponding geometric descriptor is defined in (3.92), where A_D and A_T retain the same meanings as in (3.87), and Df and Tf denote the lengths of the most visible facades of the detection and the track, respectively.

$$\beta = |AD - AT| + |Df - Tf| \quad (3.92)$$

The resulting association score is computed as given in (3.93), where w_d denotes the Euclidean distance between the predicted track position and the detection, and α is a weighting factor set to 0.2. All remaining terms retain the same definitions as in the first aggregated-feature scoring method.

[1] https://www.youtube.com/watch?v=RrUhclmXNWQ

$$AgSc_{i,j}^{I} = \alpha wd_{i,j} + (1-\alpha)(od_{i,j} + \beta + RMS(\Lambda, \Omega) + FC_{i,j}) \quad (3.93)$$

After being placed in the assignment matrix, the aggregated scores are converted into association probabilities. Following the computation of the optimal assignment between tracks and measurements, associations with probabilities below 0.3 are regarded as unreliable. Tracks involved in such low-confidence associations are therefore subjected to a second association stage using an alternative feature aggregation function, as defined in (3.94).

In this second stage, the dimensions of the validation gate are expanded by a factor of two along each axis, thereby increasing the search region for potential correspondences.

$$AgSc_{i,j}^{II} = \alpha wd_{i,j} + (1-\alpha)(od_{i,j} + \beta) \quad (3.94)$$

An optimal assignment algorithm is executed again to determine the associations between tracks and detections for those tracks that exhibited low-confidence matches in the first stage. Associations whose probabilities remain below 0.3 after this second evaluation are treated as unassigned in the current frame.

The combination of a two-stage association strategy with an enlarged validation gate substantially improves tracking performance, particularly in systems operating at frame rates below 10 FPS.

3.3.1.2 The Tracking Process

The lifecycle of a track is represented by a state-transition diagram comprising the states Initialized, Processed, Updated, Drifting, and Absolute Death. Each track evolves through these five states during online operation, depending on the availability and quality of associated measurements. The transitions between states are illustrated in Figure 3.4.

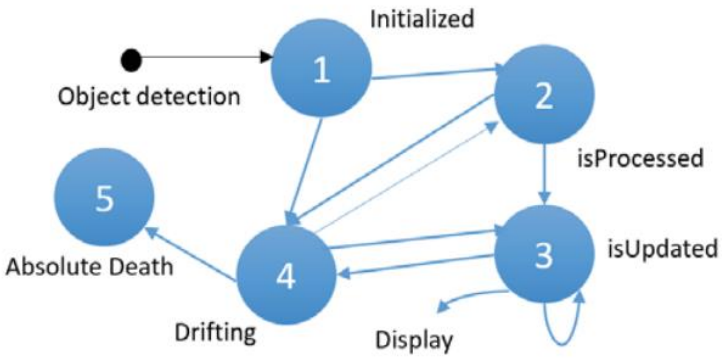

Figure 3.4. Transition between track states

Each tracked object maintains an internal representation that aggregates multiple sources of information, including RGB appearance, geometric attributes, semantic labels, orientation, velocity, spatial localization, object size, and the history of past associations. This information is continuously updated as new measurements become available and serves as the basis for both state estimation and data association.

When a new object hypothesis is generated by the detector, it is first processed by the track manager and assigned to the Initialized state. At this stage, no output is produced for the corresponding object, since it cannot yet be determined whether the detection represents a true physical target or clutter. Tracks in the Initialized state are therefore not displayed. If a track fails to obtain a valid association in the subsequent frame, it transitions into the Drifting state, where it remains temporarily unmatched and is also not rendered.

When a detection is successfully associated with a track based on the similarity score computed by the data association module, the track enters the Processed state. From this state, the object may either be associated again in the following frame, in which case it progresses further along its lifecycle, or it may fail to receive an association and return to the Drifting state, where it remains available for possible future matches. When a track is successfully associated in two consecutive frames, it transitions into the Updated state. A track remains in the Updated state as long as it continues to be matched with incoming detections. Once a track has been updated at least three times while in this state, it is considered stable and is made visible to downstream perception modules.

If no valid association is found for a track during several consecutive frames, the track enters the Drifting state. The number of frames for which a track may remain in this unassociated state while still being retained in the system is referred to as the track history and is directly related to the frame rate of the perception pipeline. At higher frame rates, such as those exceeding 10 FPS, longer history windows can be tolerated, since the probability of reacquiring a valid measurement within a short time interval is high. In the present system, which operates at frame rates below 10 FPS, the history threshold is set to three frames in order to prevent excessive growth of state uncertainty.
This design reflects the fact that, when no measurement is available to update a track, the uncertainty of the predicted state increases over time. At lower frame rates, this uncertainty grows more rapidly, which

reduces the likelihood of correct future associations. However, tracks are not immediately removed when no association is found, as real-world objects may be temporarily occluded or may briefly leave the sensor's field of view.

While in the Drifting state, a track may transition back to Initialized, Processed, or Updated, depending on both the number of frames spent drifting and the number of previously successful associations. If a track enters the Drifting state from the Updated state, it has already accumulated at least three confirmed associations. In this case, if a valid association is obtained within three frames, the track returns directly to the Updated state. If fewer than three confirmed associations exist, or if the track has remained in the Drifting state for between three and eight frames, a successful association leads to the Processed state. If the track has drifted for between nine and fourteen frames before being associated again, it re-enters the Initialized state, reflecting the reduced confidence in its identity.

The terminal state of the lifecycle is Absolute Death. A track transitions into this state when it has remained unassociated for more than fifteen frames while in the Drifting state, or when more than two seconds have elapsed since its last valid association. Once a track enters the Absolute Death state, it is removed from the list of active tracks.

Future object states are predicted using a combination of two motion models: the Constant Turn Rate and Velocity (CTRV) model and the Constant Velocity (CV) model. The use of multiple motion models is motivated by the diversity of motion patterns exhibited by road users and by the limited frame rate of the perception system, which makes reliance on a single kinematic model insufficient.

Under the CTRV model, the state vector includes the two-dimensional position, speed, yaw angle, and yaw rate of the object, reflecting planar vehicle motion. Under the CV model, the state vector consists of the two-dimensional position and Cartesian velocity components. State prediction for the CTRV model is performed using the Unscented Kalman Filter, which is well suited for handling the nonlinearities of curvilinear motion, whereas the CV model is propagated using a standard linear Kalman filter. Since both models employ linear measurement functions, a common update mechanism can be applied.

The interaction and mixing of the multiple motion models are addressed in the section on multi-object tracking and segmentation.

3.3.1.3 Filtering Meta Parameters

Parameters such as the width, height, and orientation of a 3D bounding box are referred to as meta-parameters. These quantities are often subject to pronounced fluctuations, primarily due to imperfect LiDAR sensor synchronization and inaccuracies in point-cloud segmentation during 3D object detection.

To address this issue, a dedicated meta-parameter filtering module is introduced, providing a clear separation between the estimation of positional states and object-dimension attributes. By fusing the outputs of the main tracking module with those of the meta-parameter filter, a more reliable estimate of the object orientation can be obtained. Moreover, following the mixing stage of the interacting multiple-model framework, several iterations may be required before the state becomes fully consistent with the selected motion model. The collaboration between the two modules is therefore essential for mitigating oscillations in orientation that may arise from frequent model switching.

The state vector of the meta-parameter filter comprises the two-dimensional position, the Cartesian velocity components, object width and length, yaw angle, and yaw rate, as defined in (3.95).

$$X_k = [x\ y\ \dot{x}\ \dot{y}\ \psi\ \dot{\psi}\ w\ l] \tag{3.95}$$

The state-transition equations for the individual components of the meta-parameter model are given in (3.96)–(3.100). An intuitive overview of the complete framework, incorporating both the primary tracking module and the meta-parameter filtering component, is provided in Figure 3.5.

$$x = x + \dot{x} * t \tag{3.96}$$

$$y = y + \dot{y} * t \tag{3.97}$$

$$\psi = \psi + \Delta t \dot{\psi} \tag{3.98}$$

$$w_{k+1} = w_{k+1} + (w_{k+1} - w_k) \tag{3.99}$$

$$l_{k+1} = l_{k+1} + (l_{k+1} - l_k) \tag{3.100}$$

The meta-parameter module employs a Kalman filter to perform state prediction and update, following a constant-velocity motion model. As illustrated in Figure 3.5, the combination strategy receives as inputs the

predicted yaw angle from the main tracking module and the corresponding prediction from the meta-parameter filter. The positional discrepancies between these predictions and the LiDAR measurement are then evaluated.

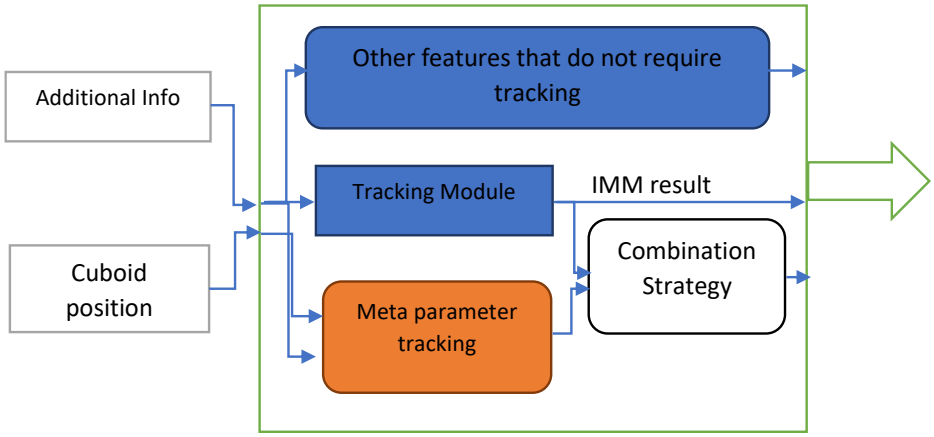

Figure 3.5 Components of the tracking module

The final orientation estimate is computed as a weighted fusion of the filtered yaw values produced by the two modules. The weight assigned to each estimate is inversely proportional to its deviation from the measurement, such that predictions closer to the observed data receive higher influence. The resulting fused orientation ψ_f is defined in (3.101).

$$\psi_f = w_1\psi_{meta} + w_2\psi_{CTRV} \tag{3.101}$$

A variant of the above fusion strategy was also investigated, in which the weighting factors were determined by the maturity level of each module. In this context, maturity was defined by the number of successful associations accumulated by each module over time. Under this scheme, the tracking processes of the main tracker and the meta-parameter filter were performed independently, after which an additional association stage was applied between the resulting track sets.

In practice, this alternative formulation led to inferior performance in terms of both computational efficiency and tracking accuracy. Representative results of the LiDAR-based object tracking system using the second aggregation score method are shown in Figures 3.6 and 3.7, where the cuboids corresponding to tracked objects are highlighted in red in the bottom-right panel.

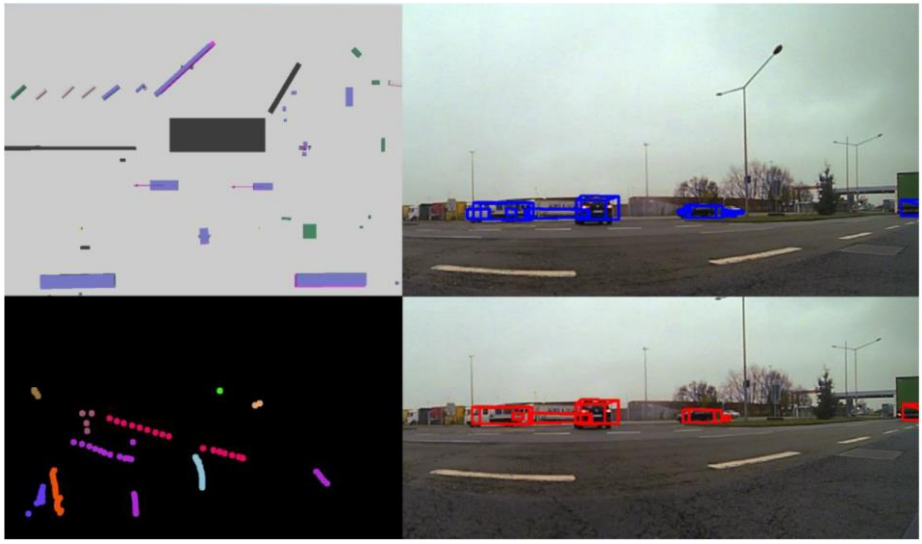

Figure 3.6 Tracking of dynamic and static objects. Top-left: bird's-eye-view 3D representation of the scene. Top-right: projection of 3D detections onto the RGB image (blue). Bottom-right: projected tracks (red). Bottom-left: trajectories of tracked objects, with consistent coloring per object instance.

In the top-right panel of Figure 3.6, the cuboids corresponding to raw measurements are shown in blue, whereas the top-left panel presents a top-view representation of all detected objects, with cuboid colors indicating their semantic classes. The pink cuboids overlaid beneath certain detections denote the corresponding tracked objects. In the present tracking framework, the following semantic categories are considered: *person, rider, car, truck, bus, train, motorcycle, bicycle, pole,* and *traffic sign*.

The motion vectors of the two vehicles passing in front of the ego vehicle are clearly visible. Because the ego vehicle is stationary at the intersection, static objects do not exhibit apparent motion vectors. The bottom-left panel of Figure 3.6 visualizes the object identifiers and track histories, with distinct colors assigned to each track ID.

Figure 3.7 presents results for static-vehicle tracking. In the left panel, tracked objects are displayed in pink, each with an associated motion vector and a unique identifier, while other detections are shown in different colors. The location of the ego vehicle is indicated by the small green and red cross at the center of the image. The occasional overlap between tracked objects and measurements is an artifact of the ADTF

visualization tool. In the right panel, tracks are rendered in red and measurements in blue, projected onto the front-view image.

The material presented in the preceding sections has been previously published and is available in [350] and [351].

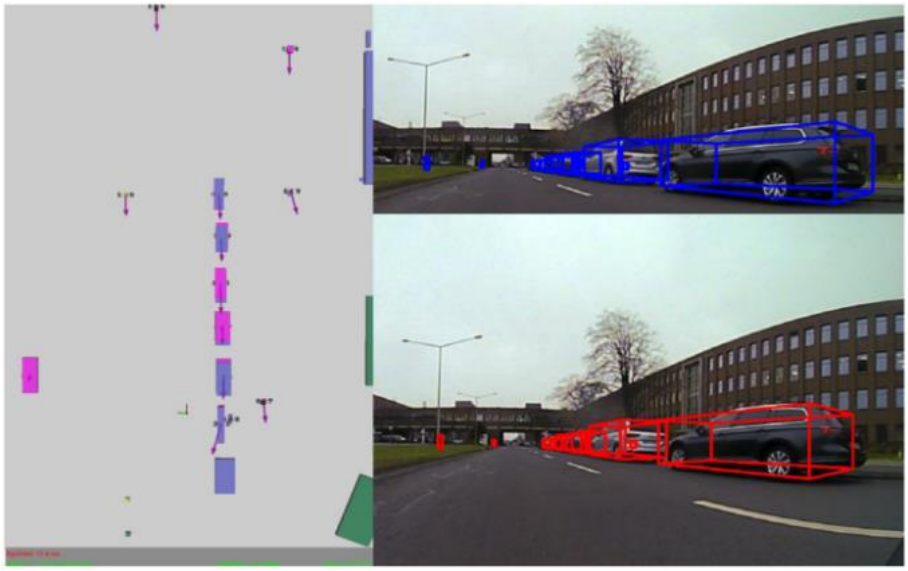

Figure 3.7. Tracking of static objects. Left: top-view 3D representation of measurements and tracks with associated motion vectors. Right (top and bottom): projection of measurements (blue) and tracks (red) onto the RGB image.

3.3.1.4 Evaluation of the Advanced 3D Object Tracking Metods

3.3.1.4.1 Evaluation of Aggregated Features Score Method 1

This section evaluates the performance of the proposed framework using the Multiple Object Tracking Accuracy (MOTA) and Multiple Object Tracking Precision (MOTP) metrics. The experiments were conducted on a system equipped with an Intel i5-2500 CPU running at 3.0 GHz. The average processing time of the tracking pipeline is 80 ms per frame. The main characteristics of the 16-layer LiDAR sensor used for object detection are summarized in Table 3.1. The proposed framework was evaluated on 3,000 frames containing objects from multiple semantic classes. Tracker performance is quantified using MOTA, which aggregates true positives, false positives, false negatives, and identity switches, as defined in (3.102).

Table 3.1 LIDAR Characteristics

Features
Time of flight distance measurement with calibrated reflectivities
16 channels
Measurement range up to 100m
Accuracy +/- 3cm
Dual returns
Field of view (vertical): 30° (+15° to -15°)
Angular resolution (vertical): 2°
Field of view (horizontal/azimuth): 360°
Angular resolution (horizontal/azimuth): 0.1° - 0.4°
Rotation rate: 5 - 20 Hz

The variable t denotes the time index, and GT refers to the ground-truth annotations. Notably, the MOTA value can become negative when the total number of tracking errors exceeds the number of correctly detected objects.

$$MOTA = 1 - \frac{\sum_t FN_t + FP_t + IDSW_t}{\sum_t GT_t} \tag{3.102}$$

In contrast, the MOTP metric, defined in (3.103), measures the average localization error between correctly associated tracks and the corresponding ground-truth objects. It quantifies the mean spatial overlap between true positive tracks and their detections.

$$MOTP = \frac{\sum_{t,i} d_{t,i}}{\sum_i c_t} \tag{3.103}$$

In (3.103), c_t denotes the number of matched track–target pairs in frame t, and $d_{t,i}$ represents the bounding-box overlap between tracked object $i$and its corresponding ground-truth instance. The quantitative results are reported in Table 3.2.

The evaluation sequence was recorded on the Volkswagen campus and includes both clear and rainy weather conditions. The tracking method forms part of a larger perception pipeline and has been configured to meet the operational requirements of that system.

Table 3.2 Tracking results

Metrics	Value
MOTA	86.86 %
MOTP	85.39 %
IDSW (sum)	130
Total Frames	3000
Total Objects	14317

The results demonstrate a high level of tracking accuracy and precision. The largest miss rates are observed for fragmented objects, which exhibit intermittent variations in size, shape, semantic classification, and position. As expected, overall tracking performance is strongly influenced by the quality of the input detections and semantic segmentation.

Table 3.3 presents a comparative evaluation against the classical GNN [191] and JPDA [192] algorithms, adapted to driving scenarios and tested on the same dataset.

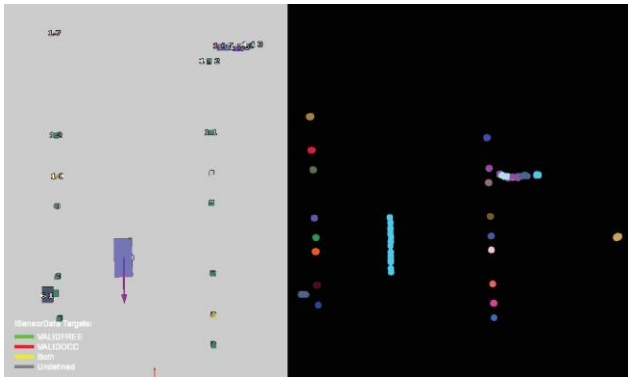

Figure 3.8. Motion vector and target ID trail

Figure 3.8 illustrates the motion vector of an approaching vehicle together with its trajectory in the right-hand panel, where each object identity is indicated by a distinct color. Figure 3.9A shows a different scenario in which the ego vehicle is located at an intersection adjacent to a parking area. In the upper panel of this figure, the semantic segmentation is overlaid on the corresponding intensity image. The lower panel visualizes the trajectories of targets approaching the ego vehicle with approximately the same speed magnitude but in the opposite direction. Figure 3.9B presents the corresponding 3D cuboid

tracks and detections, together with their associated motion vectors, for the same scene shown in Figure 3.9A.

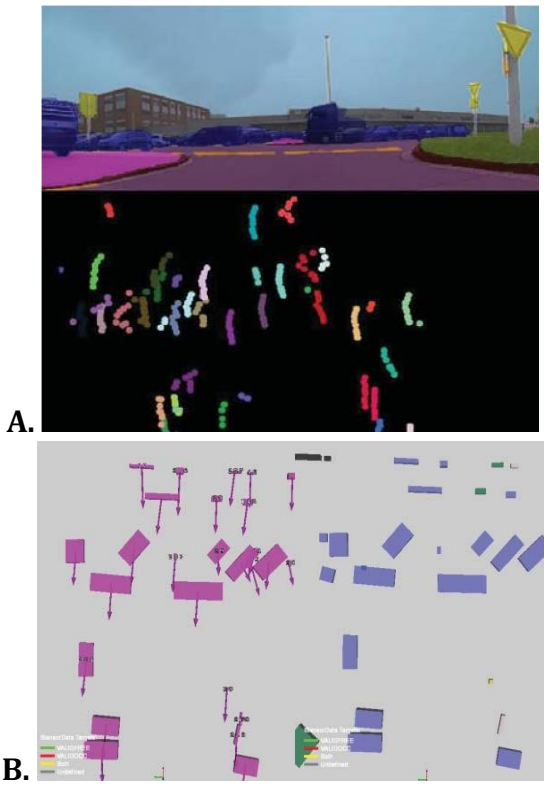

Figure 3.9. A: Semantic segmentation overlaid with target trajectories and object identities. B: 3D cuboid tracks (pink) and detections (blue) for the same scene.

Table 3.3 Tracking comparison.

Name	MOTA (%)	MOTP (%)
Proposed Solution 1	**86.86**	**85.39**
JPDA	78.3	77.5
GNN	72.13	70.84

3.3.1.4.2 Evaluation of Aggregated Features Score Method 2

The same MOTA and MOTP metrics described in the previous subsection are used to evaluate the second aggregation method.

Table 3.4 Performance Characteristics of the tracking using the second data association function

Method	MOTA	MOTP	IDSW	Running time (ms)
Proposed	86.12%	91.01%	75	0.3

The dataset used for the evaluation reported in Table 3.4 consists of real-world driving scenes acquired under diverse and challenging conditions, including varying weather, and is annotated with ground-truth data. The results demonstrate a high level of tracking accuracy and precision. The largest miss rates are observed for large objects that exhibit sporadic fluctuations in size, semantic classification, and spatial localization across consecutive frames. As in the previous evaluation, overall tracking performance is strongly dependent on the quality of the underlying 3D object segmentation.

Table 3.5 Comparison with solutions from the KITTI benchmark.

No.	Solution Name	MOTA	MOTP	Running Time
1.	**Proposed Solution 2**	**77.05%**	**81.65%**	**0.35s**
2.	MDP [227]	76.59%	82.10%	0.9s
3.	CIWT [228]	75.39%	79.25%	0.3s
4.	DCO-X [229]	68.11%	78.85%	0.9s

Table 3.5 presents a comparison of the proposed tracking framework with other methods reported in the literature on the KITTI car dataset, using MOTA, MOTP, and runtime as evaluation criteria. As noted previously, the proposed association and tracking approach supports multi-class object tracking, rather than being restricted to vehicles alone.

3.3.2 Data Association and Tracking of 2D Thermal Camera Objects

Owing to their ability to operate reliably under adverse environmental conditions—such as rain, fog, and snow—thermal cameras have attracted considerable interest in the automotive domain. Unlike RGB cameras, thermal sensors do not require external illumination and are not affected by glare from oncoming headlights. Moreover, their capacity to detect pedestrians and animals at long distances provides additional reaction time for accident avoidance.

However, thermal imagery typically exhibits lower spatial resolution and contains less visual detail than RGB or monochrome images. This limited information content significantly complicates the design of appearance-based data association functions for multi-object tracking, particularly when combined with the inherent ambiguities already present in the tracking pipeline.

In the literature, two principal approaches have been explored to address data association in multi-object tracking: data-driven methods, based on learning and neural networks [157, 159, 160, 161], and feature-engineering approaches [143, 145, 146]. Data-driven methods automatically learn discriminative representations from training data once the network architecture has been defined and trained. A major limitation of such approaches, however, is the risk of track drift, whereby a tracker may become permanently associated with an incorrect detection that resembles patterns seen during training but does not correspond to the true target. This issue is further exacerbated when objects are partially occluded and such cases are underrepresented in the training set.

In feature-engineering approaches, suitable descriptors are selected and explicitly designed to distinguish between objects, and corresponding cost functions are constructed based on these features. An optimal assignment algorithm, such as the Hungarian method [142], is then used to associate tracks with detections by minimizing the overall cost. While this paradigm avoids the pitfalls of model overfitting, it requires careful selection and tuning of features for each sensor modality.

From a computational perspective, feature-engineering methods are generally faster than data-driven solutions and typically do not require GPU acceleration or specialized hardware to achieve real-time performance.

3.3.2.1 Camera Setup

To obtain a robust pedestrian-tracking solution capable of operating under diverse conditions, data were collected across a wide range of weather and illumination scenarios, including daytime and nighttime operation, as well as sun, snow, fog, and other challenging environments.

The sensing platform consists of a FLIR PathFindIR thermal camera equipped with a VOx microbolometer operating in the 8–14 μm spectral

range. The sensor produces images with a native resolution of 320 × 240 pixels and employs a 19 mm focal-length lens, providing a field of view of 36° (horizontal) and 27° (vertical). The camera is hermetically sealed and IP67-rated, allowing operation under adverse environmental conditions without degradation due to dust or water ingress. Its thermal time constant is 12 ms.

Although the camera outputs an analog PAL signal, a digital video converter (EZMaker 7, AVerMedia) is used to acquire the thermal imagery in digital form. During this conversion process, the images are automatically upscaled to a resolution of 640 × 480 pixels.

The camera was mounted on the roof of a vehicle using a magnetic tripod and positioned symmetrically with respect to the vehicle's lateral axis. The mounting configuration is shown in Figure 3.10. The camera was located 2,555 mm behind the front of the vehicle and at a height of 1,788 mm above ground level.

The input to the tracking system consists of a set of object detections represented by bounding boxes, each associated with a semantic class label and a corresponding classification probability. The output of the system is a set of tracked objects, each assigned a unique identity and a filtered trajectory.

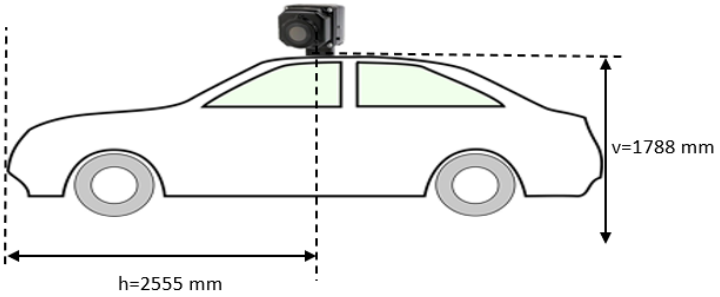

Figure 3.10 Position and mounting of the thermal camera on the vehicle.

The implemented tracking system adopts a point-target tracking paradigm within a tracking-by-detection framework. This choice is motivated by the need for a computationally efficient, real-time solution that can be readily deployed on embedded platforms, such as the NVIDIA Jetson family. In addition, the structure of the available input data—namely, frame-wise object detections—naturally lends itself to a tracking-by-detection formulation. Pedestrian detection in thermal imagery is performed using a YOLO-based architecture augmented with

spatial pyramid pooling. This detector was selected based on its strong classification performance reported in previous work [217].

3.3.2.2 Gating

The presence of clutter makes it difficult to discriminate between true sensor measurements and false alarms. In addition, computing an association score between every track and every detection in a frame is computationally expensive. To reduce the number of candidate associations, a measurement validation gate is therefore employed. This gate is centered on the predicted track position and restricts the association process to detections that fall within a predefined validation region. The validation gate is characterized by two elements: the gate center, given by the predicted state $\bar{\mathrm{X}}_k$, and the gate volume V_k. In the case of ellipsoidal gating, the validation region is defined by (3.104).

$$\left(\chi_k^i - \bar{x}_k\right)^T S_k^{-1}\left(\chi_k^i - \bar{x}_k\right) \leq \gamma \tag{3.104}$$

In (3.104), χ_k^i denotes the i-th measurement that lies within the validation gate, S_k is the innovation covariance (as also defined in [248]), and γ is a constant probability threshold whose value is obtained from the chi-squared distribution. The volume of the validation gate is given analytically in (3.105), where c denotes a scaling factor

$$V_k = c\gamma^{\frac{1}{2}}|S_k|^{\frac{1}{2}} \tag{3.105}$$

An intuitive visualization of the gating process is provided in Figure 3.11. Thermal cameras periodically suspend image acquisition for several milliseconds due to flat-field correction, a process required for sensor recalibration. During these intervals, moving objects may not be detected, which causes their actual positions to deviate from the tracker's predicted states and increases the apparent measurement error.

To account for this effect, the proposed framework adapts the validation gate size by employing two different values of the scaling factor c in (3.105). The thermal camera signals an impending flat-field correction by displaying a small white rectangle in the lower-right corner of the acquired image; detecting this marker enables the system to switch between the two gate configurations. Under normal operating conditions, the formulation in (3.106) is used, whereas after a flat-field

correction, the expanded gate defined in (3.107) is applied to accommodate the increased positional uncertainty.

$$c = 2(w + h) \tag{3.106}$$

$$c = \frac{2wh}{3} \tag{3.107}$$

In these equations, w and h denote the width and height of the track's bounding box, respectively, ensuring that the validation gate scales proportionally with the object's spatial extent. The two formulations used to compute the scaling factor $c$were determined empirically.

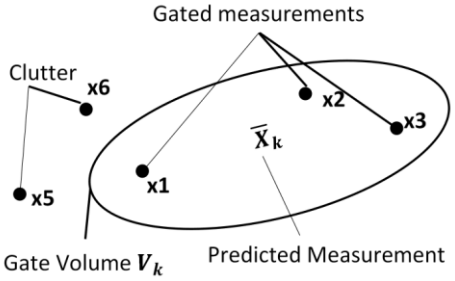

Figure 3.11. Items of the measurement validation gate

After the gating step, only detections that fall within the validation region of a track are considered for association. A similarity cost is then computed between the track and each of the gated detections.

3.3.3 Feature Engineered Solution for Thermal Object Tracking

3.3.3.1 Data Association Overview

The similarity cost between the i-th track and the j-th detection, denoted $\varepsilon(i, j)$, is defined as the sum of two components: an appearance score $a(i, j)$ and a motion score $m(i, j)$, as given in (3.108). Each component is itself a weighted combination of several feature-based similarity measures specifically designed for the thermal imaging domain. The individual similarity terms are described in the following subsections.

$$\varepsilon(i, j) = a(i, j) + m(i, j) \tag{3.108}$$

Feature-engineering approaches offer the advantage of providing explicit insight into the role and purpose of each extracted descriptor. This transparency enables precise control over the contribution of

individual features to the overall association score and facilitates systematic tuning. Moreover, the resulting cost function is interpretable rather than a black box, allowing new features to be incorporated when previously unmodeled effects are identified.

In contrast, neural-network-based solutions typically lack such interpretability and flexibility. When confronted with scenarios that were not adequately represented in the training data, a learned model may fail to generalize or may incorrectly associate tracks with visually similar but unrelated detections.

The principal challenge in designing feature-based similarity functions lies in introducing new terms that address specific failure modes without degrading the performance of existing components. To ensure that each added feature yields a net improvement, all representative evaluation sequences are reprocessed whenever the cost function is modified, thereby verifying that previously resolved cases remain correctly handled.

3.3.3.2 Appearance Score

The appearance score is a fundamental component of multi-object tracking, as it enables the discrimination of distinct objects in close proximity and supports the re-identification of the same object across successive frames. Nevertheless, object appearance may vary over time due to viewpoint changes, partial occlusions, or non-rigid deformations.

In the thermal domain, although infrared emission is independent of ambient illumination, the combination of clothing materials and the emissivity of human skin produces characteristic thermal patterns that are often distinctive for each individual. These patterns, however, may still vary with posture, motion, and environmental conditions.

Consequently, an effective appearance model must simultaneously capture both the uniqueness and the temporal variability of each pedestrian's thermal signature. In the present work, this challenge is addressed by constructing an appearance score that integrates multiple complementary visual descriptors within a weighted aggregation framework. The task of appearance-based discrimination is inherently more difficult in thermal imagery than in RGB imagery, owing to the reduced availability of color and fine-grained texture cues.

The resulting appearance similarity between a track i and a detection j is therefore defined as a combination of several visual features, as formalized in Eq. (3.109):

$$\vartheta(i,j) = w_{hL}hL(i,j) + w_{\mu s}\mu s(i,j) + w_{\sigma s}\sigma s(i,j) + w_{hs}hs(i,j) + w_{ws}ws(i,j) + w_{cs}cs(i,j) + w_{os}os(i,j) \quad (3.109)$$

The individual components of Eq. (3.110) are defined as follows. The term $h_L(i,j)$ denotes the dissimilarity between the uniform Local Binary Pattern (LBP) descriptors computed over the region of interest (ROI) of track $i$and detection j. The quantity $\mu_s(i,j)$ represents the difference in mean pixel intensity within the respective ROIs, while $\sigma_s(i,j)$ corresponds to the difference in intensity variance. The terms $h_s(i,j)$ and $w_s(i,j)$ denote the differences in height and width of the bounding boxes associated with track i and detection j, respectively.

The overlap score $o_s(i,j)$ measures the spatial overlap between the track and the detection, and $c_s(i,j)$ represents the difference between the semantic class confidence stored in track $i$from its most recent associated detection and the class confidence of the current detection j. The components of Eq. (3.109) are combined through a weighted aggregation scheme, where $w_{h_L}, w_{\mu_s}, w_{\sigma_s}, w_{h_s}, w_{w_s}, w_{c_s}$, and w_{o_s} denote the relative weights assigned to the corresponding distance measures. These weights were determined empirically by analyzing their contribution across more than 160 recorded sequences encompassing a wide range of environmental conditions, including variations in temperature, illumination, and time of day. The adopted values are:

$w_{h_L} = 10, w_{\mu_s} = 285, w_{\sigma_s} = 8, w_{h_s} = 10, w_{w_s} = 10, w_{c_s} = 550, w_{o_s} = 95.$

These values are not unique, and small perturbations do not significantly affect the overall performance of the tracking system. For clarity and reproducibility, several weights were rounded to the nearest multiple of five without degrading the accuracy or stability of the proposed tracking framework.

The textural characteristics of each detection are represented by a uniform Local Binary Pattern (LBP) histogram computed over its region of interest. Because the texture of a pedestrian in thermal imagery is expected to vary only gradually across consecutive frames, this descriptor provides a stable and discriminative cue for measuring similarity between a track and a candidate detection. The LBP operator encodes the local texture around each pixel into a binary pattern, as defined in Eq. (3.110), thereby yielding a compact representation of local intensity variations that is well suited for appearance-based association.

$$LBP_{P,R} = \sum_{p=0}^{P-1} s(g_p - g_c)2^p \quad (3.110)$$

Let P denote the number of sampling points located on a circle of radius R. The function $s(x)$ takes the value 0 when $x \geq 0$ and 1 otherwise, while g_p and g_c represent the intensity values of the p-th neighbouring pixel and the central pixel, respectively. In the proposed framework, a 3×3 neighbourhood is employed, which allows all possible binary patterns within the region of interest to be encoded into a 256-bin histogram. To improve computational efficiency, robustness, and memory usage, a uniform Local Binary Pattern (LBP) histogram is adopted. A binary pattern is classified as uniform if it contains at most two bitwise transitions between 0 and 1 when traversed circularly [218]. For a 3×3 neighbourhood, although 256 distinct LBP codes are theoretically possible, only 58 of them are uniform, which leads to 59 distinct labels when a single bin is reserved for all non-uniform patterns. A precomputed lookup table is used to accelerate the feature extraction process by enabling fast mapping of each binary code to its corresponding uniform-pattern bin during histogram voting. An intuitive illustration of the construction of the uniform LBP histogram is provided in Fig. 3.12.

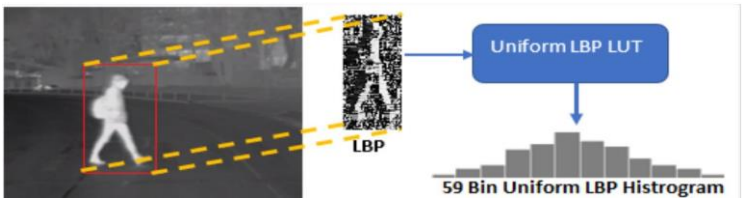

Figure 3.12. Creation of the uniform local binary pattern histogram using the LBP codes extracted for the region of interest.

The term $h_{u\mathrm{LBP}}(i,j)$ is computed by applying a root-mean-square (RMS) distance to the uniform Local Binary Pattern histograms associated with track $i$and detection j, as defined in (3.111).

$$huLbp(i,j) = \sqrt{\frac{1}{59}\sum_{k=1}^{59}(huLbp(i)_k - huLbp(j)_k)^2} \qquad (3.111)$$

A generic similarity function that combines the various extracted features is defined in (3.112). The absolute value of a scalar quantity a is denoted by the operator $|\ a\ |$.

$$F(x,y) = |x - y| \qquad (3.112)$$

Within the appearance similarity measure, two additional descriptors are employed: the mean and the standard deviation of the pixel intensities within the region of interest (ROI). For a given object instance, these quantities are not expected to vary significantly between consecutive frames, since pedestrians do not undergo abrupt changes in temperature that would substantially alter their thermal texture. The standard deviation characterizes the local contrast within the ROI, whereas the mean reflects the overall intensity level, both of which are indicative of the amount of thermal radiation emitted by the pedestrian. By applying the generic similarity function defined in (3.112) to the mean intensity values of the detection and the corresponding track, the similarity term $\mu_s(i,j)$ in (3.113) is obtained.

$$\mu s(i,j) = F(\mu s(i), \mu s(j)) \tag{3.113}$$

The variance-based appearance term $\sigma_s(i,j)$ is computed in an analogous manner to $\mu_s(i,j)$ in (3.113), by applying the generic similarity function $F(\cdot)$ to the variance values $\sigma_s(i)$ and $\sigma_s(j)$ associated with the track and the detection, respectively. The expressions used to compute the mean $\mu_s(i)$ and the variance $\sigma_s(i)$ within the region of interest are given in (3.114) and (3.115). Here, h and w denote the height and width of the ROI, respectively, while Image refers to the portion of the far-infrared image defined by the object's bounding box.

$$\mu s(i) = \frac{1}{M}\left(\sum_{k=0}^{h-1}\sum_{r=0}^{w-1} Image(k,r)\right) \tag{3.114}$$

$$\sigma s(i) = \sqrt{\frac{1}{M}\left(\sum_{k=0}^{h-1}\sum_{r=0}^{w-1}\left(Image(k,r) - \mu s(i)\right)^2\right)} \tag{3.115}$$

The height and width of a pedestrian constitute salient physical attributes that can contribute to the re-identification process. These attributes are captured by the bounding boxes associated with each detection and, given the sufficiently high frame rate of the system, variations of the same object instance across consecutive frames can be reliably observed. While height and width are effective cues for discriminating between pedestrians with distinct physical characteristics, they are not, by themselves, sufficient when multiple objects exhibit similar dimensions. For this reason, these attributes are incorporated as components of the overall appearance-based association score. The height similarity term $h_s(i,j)$ between a track i

and a detection j is computed by applying the generic similarity function F from (3.112) to the corresponding heights of the two instances. The width similarity term $w_s(i,j)$ is defined analogously, using the width of the respective bounding boxes.

The classification confidence provided by the pedestrian detector is also integrated into the aggregated appearance cost. Empirically, the difference between the classification probabilities associated with the same pedestrian across consecutive frames was found to be small. By applying the similarity function F from (3.112) to the classification probability of the current detection and the probability stored in the track from the most recent successful association, the classification similarity term $c_s(i,j)$ is obtained. This term takes values in the interval $[0, 1]$, with values closer to zero indicating higher similarity. The inclusion of this feature is motivated by the observation that pedestrians located at larger distances may exhibit similar classification scores despite corresponding to different physical entities.

The final component of the appearance-based cost function is a size-and-location similarity term. This term combines spatial proximity and scale consistency between a track and a detection. Specifically, the size-based cost $o_s(i,j)$ incorporates both the localization of the detection relative to the predicted position of the track and the similarity between their bounding-box areas, as defined in (3.116). Here, A_i denotes the area of the bounding box associated with track i, A_j denotes the area of the detection j, and $A_\cap$ represents the two-dimensional intersection area between the two bounding boxes.

$$os(i,j) = \frac{|A_j - A_i|}{A_\cap} \tag{3.116}$$

3.3.3.3 Motion Score

In certain scenarios, appearance-based cues alone may be insufficient to reliably discriminate between visually similar pedestrians in thermal imagery. In such cases, motion characteristics provide an additional and often decisive source of information. For this reason, motion consistency is incorporated into the overall data-association similarity measure. The analytical expression of the motion-based similarity term $m(i,j)$, defined for a track i and a detection j, is given in (3.117). The weighting coefficients appearing in this formulation were determined empirically, following the same experimental procedure adopted for the appearance-based terms. In the present implementation, the values

$w_{\mathrm{dst}} = 85$ and $w_{\sigma_m} = 20$ were found to provide a suitable balance between spatial displacement and motion-uncertainty contributions.

$$m(i,j) = w_{dst}dst(i,j) + fc(i,j) + w_{\sigma m}(\sigma m(i,j)_x + \sigma m(i,j)_y) \quad (3.117)$$

Object motion provides essential cues about the expected position of a target in subsequent frames. Consequently, the Euclidean distance between the centers of the two-dimensional bounding boxes of track $i$and detection j, expressed in image coordinates, is used as the first component of the motion-based similarity measure.

A second component is derived from optical flow, which captures the relative motion between a track and a detection. Although dense optical flow methods are computationally expensive and often yield imperfect results, and sparse optical flow approaches—while faster—tend to be unreliable in complex scenes, aggregated motion information over a region of interest can still provide valuable cues about an object's movement pattern. Even if pixel-level trajectories are noisy, their collective behavior within a region often remains consistent for the same object across consecutive frames.

To exploit this property, the optical flow method described in [219] is applied, after which several processing steps are performed to extract representative motion attributes from the region of interest. First, the flow vectors are quantized into 36 angular bins, with each vector contributing a vote to the bin corresponding to its direction. Next, the mean magnitude and mean direction are computed for each bin. The bin receiving the highest number of votes is then selected, and its corresponding mean magnitude and direction are taken as the characteristic flow attributes of the region.

Empirical observations across multiple sequences indicate that real-world objects rarely undergo abrupt changes in their motion patterns between consecutive frames. This temporal coherence motivates the definition of the flow-based component of the motion similarity term, which is formalized in equation (3.118).

$$fc(i,j) = F(\theta_i, \theta_j)w_\theta + F(\vartheta_i, \vartheta_j)\, w_\vartheta \quad (3.118)$$

The flow direction is denoted by θ, while the flow magnitude is represented by ϑ. The function $F(\cdot)$ is defined in (3.111), and the

weighting coefficients $w_\theta = 40$ and $w_\vartheta = 15$ were determined empirically.

The final component of the motion similarity score, σ_m, quantifies the deviation of the candidate detection from the current motion pattern of the tracked object along the x and y directions. This term is introduced in order to penalize large departures from the expected trajectory of the object. The underlying assumption is that, when the recent motion history of a pedestrian over the last five frames is considered, the subsequent displacement is likely to follow a similar trend, with only minor deviations for a correct association. Accordingly, the last five positions of each tracked object are stored and used to model its short-term motion behavior.

The analytical expression of this variation-based distance is given in equation (3.119), and it is applied independently to the horizontal and vertical components of the two-dimensional motion. In this formulation, X_i denotes a detection, while $Z_j(k)$ represents the k-th stored past position of track j.

$$\sigma m(i,j) = |X_i - \sqrt{\frac{1}{5}\sum_{k=0}^{4}\left(Z_j(k) - Z_j(k+1)\right)^2}| \qquad (3.119)$$

It should be noted that the weighting coefficients selected for the appearance- and motion-based similarity terms are not unique. Alternative parameterizations are possible; however, the values adopted here were found to yield the best empirical performance for the considered application.

3.3.3.4 Track Selection, Update and Refinement

After the motion- and appearance-based similarity terms have been computed for a given detection i and track j, they are combined to form the final association cost $c(i,j)$, as defined in (3.108). This cost is evaluated for all tracks maintained in memory against all detections in the current frame that fall within the corresponding covariance ellipse of each track. The resulting affinity values are organized into a cost matrix, which serves as input to the Hungarian algorithm [142] for determining the optimal assignment between detections and tracks in the current frame. If the association cost corresponding to a given track–detection pair exceeds a predefined threshold, that association is rejected and the detection is instead used to initialize a new track.

After all feasible correspondences between tracks and measurements have been established, three possible outcomes can be identified: a track may be matched to a detection, a detection may remain unassigned, or a track may remain unmatched. When a valid track–detection association is obtained, the track state and all its associated parameters are updated using the information provided by the new measurement. Unassigned detections are used to initialize new tracks.

Newly initialized tracks are first maintained in an unstable state and are not immediately displayed. A track is promoted to a stable state only after it has been successfully associated with detections over a predefined number of consecutive frames (five frames in the present implementation), at which point it is considered reliable and becomes visible in the tracking output. A fundamental requirement of any tracking system is the ability to maintain object identities even in the absence of detections over several consecutive frames, which may occur due to detector failures, occlusions, or adverse imaging conditions. To address this, each track is associated with a history counter that records the number of frames during which no valid measurement has been assigned to that track.

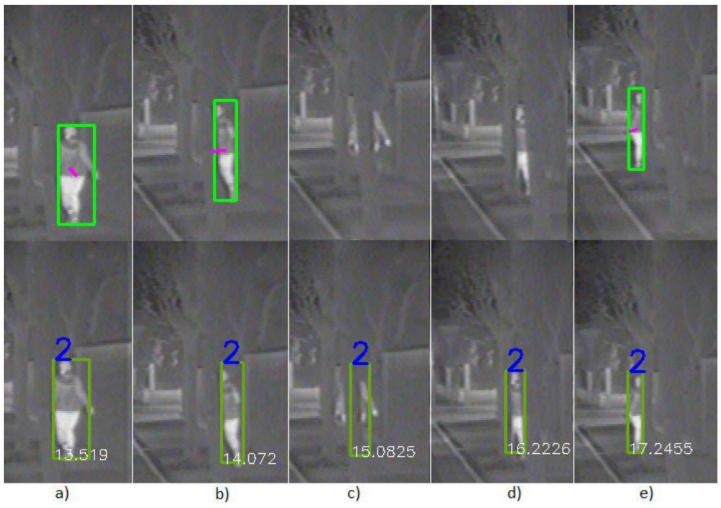

Figure 3.13 a) Pedestrian fully detected and tracked b) pedestrian begins to get occluded; c) pedestrian is fully occluded but continues to be tracked; d) pedestrian is occluded we can observe parts of him between the trees, the detector is unable to detect him, but due to the tracking algorithm his identity is maintained; e) Pedestrian reappears and is detected and tracked.

When a track becomes temporarily unmatched, its state in subsequent frames is propagated using the prediction step of the Kalman filter, based on the motion pattern estimated so far. If the track remains unassociated for several frames, it enters a drifting state, in which it is no longer displayed but is retained in memory in anticipation of a possible future re-association. Only after a prolonged period without successful association is the track permanently removed. Tracks that leave the region of interest are explicitly flagged for termination and subsequently deleted.

In the proposed implementation, the values of the history and drifting counters are adapted to the operating conditions, particularly the camera frame rate and illumination. For night-time scenarios, the history and drifting thresholds are set to 20 and 15 frames, respectively, whereas for daytime operation they are set to 25 and 15 frames. These thresholds apply only to tracks that have already been classified as stable. Tracks that have not yet reached stability are removed more aggressively, namely after 5 consecutive unassociated frames at night or 7 frames during daytime operation.

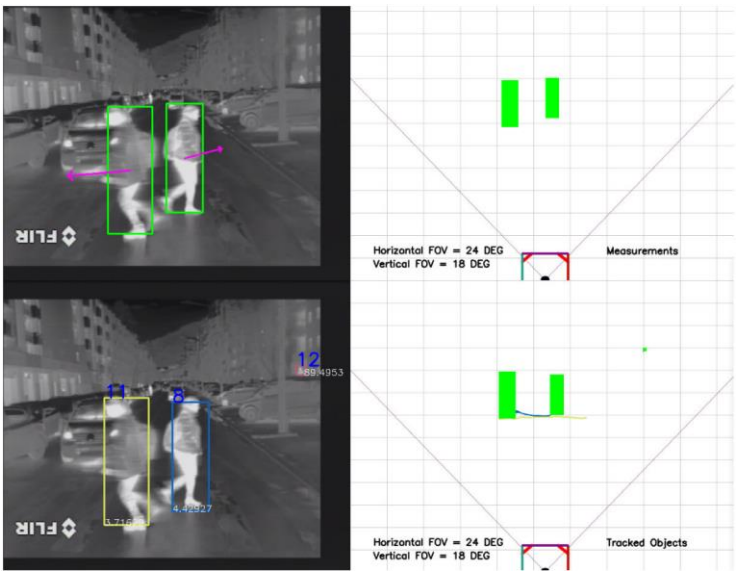

Figure 3.14. In the top left, object detections are shown, with their corresponding motion vectors. In the top right, the detections are projected in a grid. In the bottom left each track is represented with a unique id and color and in the bottom right the tracked objects are depicted as well as the motion trail corresponding to the path of each pedestrian.

Figure 3.13 illustrates a representative scenario in which a pedestrian becomes temporarily occluded by trees; nevertheless, the proposed method preserves the object's identity until it reappears and is again detected by the object detector. Figure 3.14 depicts another challenging case in which two pedestrians cross paths. The proposed approach successfully maintains the correct identity of each individual during and after the overlap, avoiding identity switches. In the bottom-right panel of Figure 3.14, the trajectory history of each pedestrian is visualized as a short trail, rendered in the same color as the corresponding bounding box to facilitate visual association. Supplementary visual results illustrating the performance of the proposed method are provided in an online demonstration[2]

3.3.3.5 Evaluation of Feature Engineered Tracking Approach

The YOLO-based pedestrian detector was initially trained on the FLIR-ADAS dataset [214] and subsequently fine-tuned using the CROSSIR dataset [209]. Starting from the pretrained weights reported in [217], the network was further trained on the pedestrian annotations of the present dataset. The fine-tuning procedure was carried out for 2000 training iterations.

The model achieving the highest mean average precision was selected for evaluation. During training, only pedestrian samples with a minimum bounding-box width of 30 pixels and without occlusions were considered. To assess the performance of the proposed far-infrared tracking approach relative to state-of-the-art methods, experiments were conducted on the PTB-TIR benchmark dataset [230], which consists of multiple thermal image sequences with manually annotated ground truth. The center location error (CLE) is defined as the average Euclidean distance between the estimated position of a tracked object and its corresponding ground-truth location. A track is considered successful at a given frame if the CLE falls below a predefined threshold, which is set to 20 pixels in the PTB-TIR benchmark. The precision score is then computed as the percentage of frames in which the tracker meets this success criterion. In addition to the dataset itself, the benchmark provides evaluation results for a wide range of tracking algorithms on the same sequences, enabling a comparative analysis of the respective strengths and weaknesses of different approaches.

[2] https://youtu.be/fwTrUwWSXDA

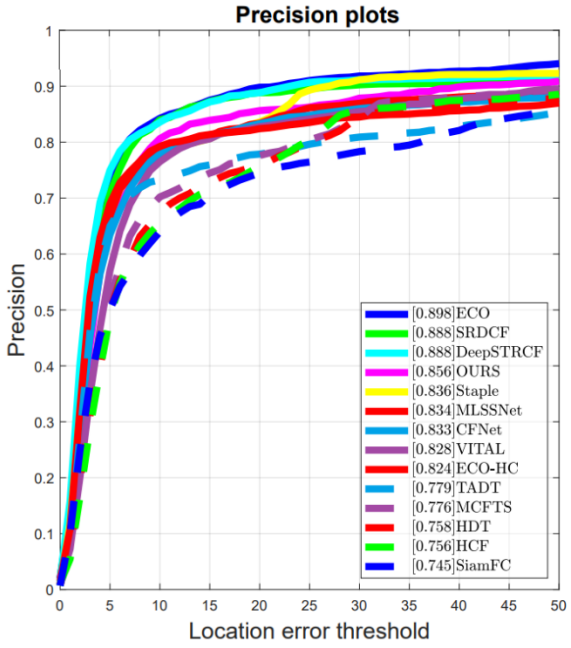

Figure 3.15. Plot of precision results on the PTB-TIR benchmark

For the evaluation of the proposed method on PTB-TIR, only those sequences depicting pedestrians observed from a vehicle-mounted thermal camera were selected, since the tracked has been designed for such use case.

Table 3.6 Comparative evaluation for various tracking solutions with respect to the Precision metric

Method	Tracking Precision Score
ECO [231]	89.8%
SRDCF [232]	88.8%
DeepSTRCF [233]	88.8%
OURS	**85.6%**
Staple [234]	83.6%
MLSSNet [235]	83.4%
CFNet [236]	83.3%
VITAL [237]	82.8%
ECO-HC [238]	82.4%
TADT [239]	77.9%
MCFTS [240]	77.6%

The performance of the presented method on the benchmark is reported in Figure 3.15 under the label *OURS*, alongside several representative approaches from the literature. It is noteworthy that the proposed solution does not rely on hardware acceleration and employs a feature-engineered data association strategy. This design choice enables the contribution of each feature to be analyzed individually, while maintaining full transparency over the feature extraction and association stages. The numerical results corresponding to the most relevant methods shown in Figure 3.15 are also summarized in Table 3.6. The method presented in this section can be found in [209]

3.3.4 Robust Data Association Using Fusion of Data-Driven and Engineered Features

The methods presented in the previous section serve as the baseline for the approach introduced in this section. Building upon this foundation, the proposed methodology extends and refines the earlier solution with the objective of improving its robustness and overall reliability.

3.3.4.1 Introducing a New Engineered Texture Descriptor

It is first observed that the texture characteristics of an object exhibit limited variation across consecutive frames; consequently, texture constitutes a reliable cue for measuring the correspondence between tracks and measurements. To more effectively capture the structural texture properties of each object, a composite appearance descriptor is introduced, which integrates the Histogram of Oriented Gradients (HOG) descriptor with the Uniform Local Binary Pattern (ULBP) descriptor. The use of such engineered features enhances the adaptability of the proposed tracking framework to previously unseen scenarios, while maintaining interpretability and control over the contribution of each component.

For the construction of this descriptor, the gradient magnitude Gand orientation θare first computed from the image derivatives $I_X$and I_Y, as defined in (3.120). The image is subsequently partitioned into non-overlapping cells of size 10×10 pixels. For each cell, a histogram with nine orientation bins is generated. Each pixel within the cell contributes

a weighted vote to the corresponding histogram bin, where the vote is determined by the gradient orientation and weighted by the gradient magnitude. This procedure enables the descriptor to capture both local edge orientation distributions and their relative strengths, thereby providing a compact yet discriminative representation of object texture.

$$|G| = \sqrt{I_X^2 + I_Y^2};\ \ \theta = \arctan(\frac{I_X}{I_Y}) \tag{3.120}$$

Subsequently, each cell is processed independently, and the dominant gradient orientation is assigned to the cell by selecting the histogram bin with the highest accumulated value. This operation yields a coarse orientation map that summarizes the prevailing edge directions within each local region. The resulting representation is then further processed using the Local Binary Pattern (LBP) operator, as defined in (3.121), in order to encode local texture variations and enhance robustness to illumination and contrast changes.

$$LBP_{P,R} = \sum_{p=0}^{P-1} s(g_p - g_c)2^p \tag{3.121}$$

The number of neighbouring pixels considered on a circular neighbourhood of radius R is denoted by P, while the thresholding function $s(\cdot)$ is defined as $s(x) = 0$ if $x > 0$, and $s(x) = 1$ otherwise.

In the present formulation, g_p represents the gradient orientation of the neighbouring pixel p, whereas g_c denotes the gradient orientation of the central pixel.

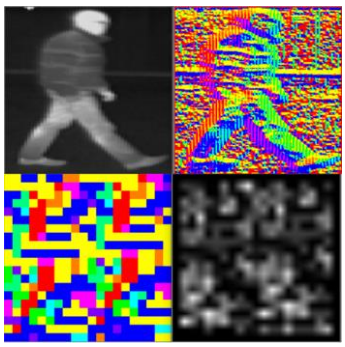

Figure 3.16 Construction of the proposed texture descriptor. Top-left: original image. Top-right: pixel-wise gradient orientation map. Bottom-left: dominant orientation per 10×10 cell. Bottom-right: LBP encoding of the orientation map, used to generate the uniform LBP histogram.

In the proposed approach, a 3×3 neighbourhood is employed, which theoretically allows the representation of all local texture configurations within the region of interest using a 256-bin histogram. However, in order to reduce memory requirements, improve computational efficiency, and increase robustness to noise, a uniform Local Binary Pattern (LBP) histogram is adopted, following the formulation introduced in [218]. For the considered neighbourhood configuration, 256 distinct binary patterns are possible, of which only 58 satisfy the uniformity criterion. Consequently, the descriptor can be compactly represented using a histogram with 59 bins.

To further improve runtime performance, the voting process for each LBP code is implemented using a precomputed lookup table, thereby avoiding repetitive bitwise evaluations. The resulting uniform LBP histogram is denoted by θ_{LBP}. An intuitive illustration of the main processing stages involved in the computation of this descriptor is provided in Figure 3.16.

It is worth noting that the image resizing operation may introduce minor artefacts, such as vertical stripe patterns. However, empirical observations indicate that these artefacts do not significantly affect the robustness or overall performance of the proposed algorithm.

Finally, the similarity between the input image associated with track $i$and the measurement j, with respect to the proposed texture descriptor, is computed by applying a root mean square (RMS) distance metric to the corresponding uniform LBP histograms θ_{LBP} of the track and detection, as formalized in equation (3.122).

$$\tau(i,j) = \sqrt{\frac{1}{59}\sum_{k=1}^{59}\left(\theta LBP(k)_i - \theta LBP(k)_j\right)^2} \qquad (3.122)$$

3.3.4.2 The Data-Driven Object Association Score

Designing a data association function for object tracking can be effectively addressed through similarity learning techniques. In this context, a similarity function $\gamma(i,j)$ is defined to compare a thermal image corresponding to a measurement $j$with a candidate image of identical spatial resolution associated with track i. The function is designed to produce low similarity scores for images originating from the same object instance and high scores otherwise.

This section focuses on the implementation of the similarity function $\gamma(i,j)$ using deep convolutional neural networks. In the domain of convolutional neural networks, similarity learning is most commonly

realized through Siamese network architectures. Such architectures process pairs of input images by applying an identical transformation $\phi(\cdot)$ to each input and subsequently combining the resulting representations using a comparison function $g(\cdot)$, as formalized in equation (3.123).

When the function $g(\cdot)$ is interpreted as a distance or similarity metric, the transformation $\phi(\cdot)$ can be regarded as an embedding function that maps the input images into a feature space in which similarity relationships are preserved.

$$\alpha(i,j) = g(\varphi(x), \varphi(y)) \tag{3.123}$$

To improve the effectiveness of appearance-based similarity estimation for thermal image tracking, a family of Siamese network architectures is introduced. The term family is used to emphasize the structural similarity shared by the proposed networks. In contrast to conventional Siamese-based approaches, which typically compute similarity using

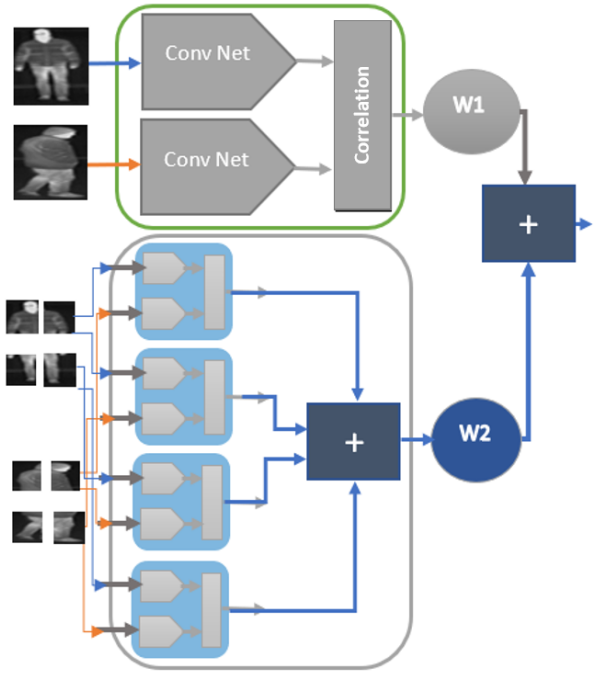

Figure 3.17. Architecture of the proposed data driven cost based on a family of Siamese Networks. These networks work on the whole image and on parts of it improving the robustness of the TIR tracker.

the entire detected object region, the proposed framework evaluates similarity through multiple Siamese networks trained on both the full object detection and on selected object subregions. This design choice, illustrated in Figure 3.17, enhances robustness in the presence of partial occlusions, as it enables reliable similarity estimation even when only portions of the object remain visible.

To this end, two complementary network architectures are considered. The first operates on the full object detection region, while the second evaluates selected subregions of the same detection. These network-based appearance descriptors are applied exclusively to detections whose spatial resolution exceeds a minimum size of 19×50 pixels (width × height). For detections below this threshold, data association relies solely on the feature-engineered association score. As a preprocessing step, each detected image patch is normalized to a fixed resolution of 200×200 pixels.

Although thermal infrared emission is independent of external illumination, variations in clothing materials and their emissivity properties result in distinctive thermal textures and structural patterns for individual pedestrians. The apparent temperature of a target is influenced by environmental conditions, as the measured radiance depends on target emissivity, reflectivity, and ambient thermal context. Nevertheless, this variability does not significantly impact tracking performance, as the sensor operates at a sufficiently high frame rate and the appearance descriptors associated with each track are updated at every time step. Consequently, the appearance characteristics of a given target remain relatively stable across consecutive frames, enabling reliable computation of association scores.

To capture both texture structure and appearance variability, a convolutional neural network is employed to generate an embedding

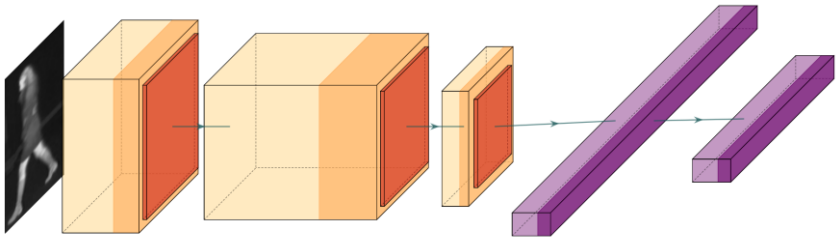

Figure 3.18. Graphical depiction of the embedding function φ used for generating the features for the input image.

function ϕ, which serves as an appearance descriptor for pedestrian tracking. The network architecture comprises eight layers, as illustrated in Figure 3.18. The first stage consists of a convolutional layer with 96 filters of size 3×3, followed by a ReLU activation, max pooling, and a dropout rate of 25%.

This is followed by a second convolutional layer with 128 filters of size 3×3, again followed by ReLU activation, max pooling, and 25% dropout. A third convolutional layer applies 12 filters of size 3×3, followed by ReLU activation, max pooling, and dropout with the same rate. The convolutional layers are succeeded by two fully connected layers: the first comprises 128 neurons with ReLU activation and a dropout rate of 10%, while the second consists of 50 neurons with ReLU activation.

The appearance similarity between two detections is quantified by computing the Euclidean distance between their corresponding embedding vectors. Network training is carried out using a contrastive loss function, defined in (3.124). In this formulation, the label tensor Yindicates whether an image pair corresponds to the same object ($Y = 0$) or to different objects ($Y = 1$); D denotes the Euclidean distance between the associated embeddings; and margin is a constant enforcing a minimum separation between dissimilar samples, set to 1in the current configuration

$$Loss = \frac{YD^2 + (1-Y)\max(margin - D, 0)^2}{2} \tag{3.124}$$

To construct the training data for the part-based appearance model, each input image is partitioned into four non-overlapping regions of equal size, corresponding to the top-left, top-right, bottom-left, and bottom-right quadrants. Appearance similarity in this formulation is evaluated at the level of image parts rather than the full detection, enabling improved robustness in scenarios involving partial occlusions or local appearance variations.

For each of the four regions, a dedicated Siamese network is employed to estimate the similarity between the corresponding part of the track image and that of the measurement image. Let $p$denote one of the four image partitions. The part-based similarity function $h(x_p, y_p)$, defined in (3.125), evaluates the correspondence between part $p$of images x and y. This formulation follows the same general structure as the similarity function introduced in (3.123); however, it relies on a

part-specific embedding function ϕ_p, which is optimized to capture localized texture and structural cues.

The final appearance-based association score is obtained by aggregating the similarity contributions from all four image parts. Specifically, the overall similarity is computed as the sum of the individual part-level similarity scores, as expressed in equation (3.126). This aggregation strategy allows the association mechanism to remain effective even when only a subset of the object regions is reliably observable

$$h(x_p, y_p) = g(\varphi_p(x_p), \varphi_p(y_p)) \quad (3.125)$$

$$\beta(i,j) = \sum_{p=1}^{4} h(x_p, y_p) \quad (3.126)$$

The part-based appearance models are trained using the same contrastive loss formulation as the full-detection model and follow an analogous architectural principle. Variations are introduced in the dimensionality of the embedding space and in the configuration of the convolutional layers. In particular, the final fully connected layer of each part-based network comprises 30 neurons. All convolutional layers employ a kernel size of 3×3, while the number of filters is set to 112 in the first layer, 96 in the second, and 12 in the third.

Each part-based network produces an appearance similarity score corresponding to a specific subregion of the detection. These scores are aggregated to form a part-based appearance similarity component, which complements the similarity score obtained from the full-detection model.

The data-driven appearance similarity score is computed as a weighted combination of the full-detection similarity score and the aggregated part-based similarity score, as defined in equation (3.127). The weighting coefficients $w_1 = 100$ and $w_2 = 25$ were determined empirically through validation experiments and reflect the relative contribution of global and part-based appearance information to the final association cost.

$$\gamma(i,j) = \alpha(i,j)w_1 + \beta(i,j)w_2 \quad (3.127)$$

3.3.4.3 Combining the Data Driven and Feature Engineered Scores

The appearance similarity score between track $i$and measurement j, obtained by jointly combining the feature-engineered descriptors and the data-driven appearance embeddings, is defined in equation (3.128). The weighting coefficient $w_3$is set to 300 and was selected empirically based on extensive experimental evaluation across multiple operating scenarios. The term $\vartheta(i,j)$ was introduced previously in equation (3.109).

It is worth emphasizing that all weighting coefficients employed in the proposed association formulation exhibit stable behavior with respect to the evaluated datasets. In particular, no parameter retuning was required when transitioning between different scenarios or when processing thermal imagery acquired with the same sensor configuration.

$$\mu(i,j) = \vartheta(i,j) + w_3\,\tau(i,j) + \gamma(i,j) \tag{3.128}$$

An optimal assignment algorithm [142] is employed to determine the most consistent correspondences between the set of active tracks and the measurements available in the current frame. Following the assignment step, three possible outcomes are identified: (i) a track is successfully associated with a measurement, (ii) a track remains unmatched, or (iii) a measurement remains unmatched. In the case of a successful track–measurement association, the state of the track and all associated attributes are updated using the newly acquired measurement information. Conversely, an unmatched measurement initiates the creation of a new track, which is initially marked as unstable. Such a track is promoted to a stable state and becomes eligible for visualization only after being consistently associated over five consecutive frames.

A fundamental requirement of a robust tracking system is the ability to maintain object identities despite temporary detection failures caused by occlusions, sensor noise, or missed detections. To this end, each track is augmented with a history counter that records the number of consecutive frames in which no association has occurred. During these intervals, the future state of the object is predicted based on its previously estimated motion pattern. If a track remains unmatched beyond a predefined number of frames, it transitions into a drifting

state, in which it is no longer displayed but is retained in memory to allow potential re-identification. Tracks that remain unassociated throughout the drifting stage are eventually removed from the active track set.

Accordingly, stable tracks are not discarded immediately upon losing associations. Instead, their states continue to be propagated using a Kalman filter operating under a constant-velocity motion model. In the proposed configuration, the history counter threshold is set to 20 frames, while the drifting counter threshold is set to 15 frames. Tracks that fail to obtain any association within the first five frames after initialization are removed immediately, as they are likely to correspond to spurious detections.

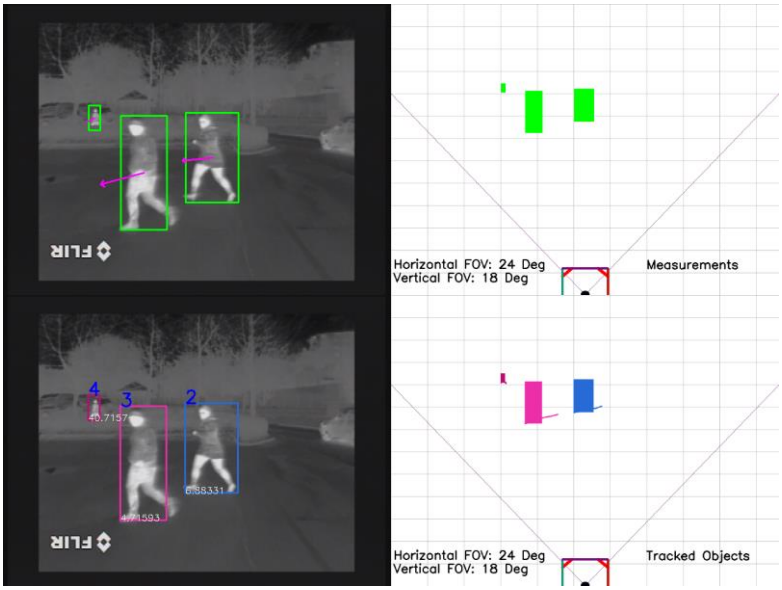

Figure 3.19. Pedestrians in a parking lot. The proposed tracking solution can track pedestrians of different sizes, even when they are partly occluded.

An illustrative example of the tracking behavior is provided in Figure 3.19. The figure depicts a scenario in which two pedestrians are tracked while approaching a vehicle, and a third pedestrian in the background is successfully maintained despite partial occlusion by vegetation. The bottom-right panel illustrates the trajectory history of each pedestrian, while the bottom-left panel displays the tracked objects along with their unique identifiers. The top-left panel shows the raw detections generated by the object detector, and the top-right panel presents the

corresponding detections projected onto a virtual image plane. In the right-hand panels, both detections and tracks are visualized within a two-dimensional grid that also depicts the camera field of view. Supplementary visual results are provided online[3]

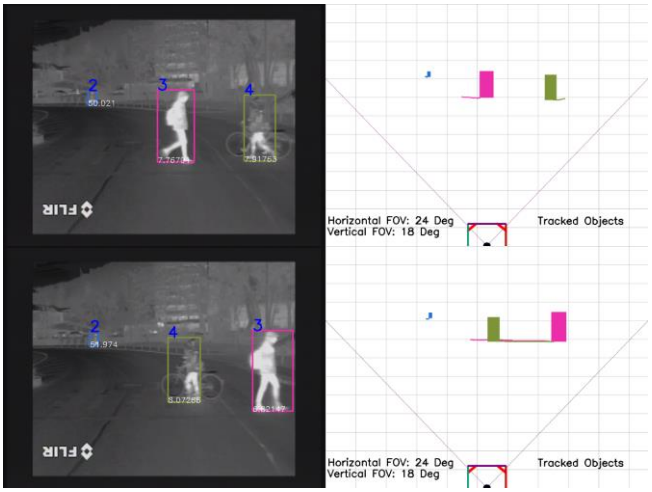

Figure 3.20. Pedestrians overlap as they are crossing the street. The tracker can maintain the correct object ID.

The bounding box of each object has a unique color to highlight its unique identity.

Figure 3.21. Multiple pedestrians are tracked. No ID switch appears among the tracked objects.

[3] https://youtu.be/j9PtbbBCgog

Figure 3.20 illustrates a representative scenario in which two pedestrians intersect their trajectories. Despite the temporary overlap between the individuals, the tracking framework preserves the correct identity of each pedestrian and avoids erroneous identity switches.
Figure 3.21 presents a more complex scene involving multiple pedestrians walking along a sidewalk. Although the individuals are in close proximity to one another and gradually appear smaller as their distance from the ego vehicle increases, no identity switches are observed among the tracked objects. The four visualizations shown in Figure 3.21 follow the same convention as those presented in Figure 3.19. Specifically, the top-left panel displays the detection bounding boxes along with the corresponding flow vectors, while the top-right panel depicts the detections within the camera field of view. The bottom-left panel shows the tracked pedestrian bounding boxes annotated with their unique identifiers, and the bottom-right panel illustrates the tracked bounding boxes projected onto a virtual image, including the trajectory history represented as motion trails behind each tracked object.

3.3.4.4 Evaluation of Feature Engineered and Data Driven Fusion Tracking Approach

The tracking framework was implemented using a combination of C++ and Python. All experimental evaluations reported in this section were conducted on a workstation equipped with an Intel i7-4770K CPU operating at 3.5 GHz, 8 GB of RAM, and an NVIDIA GeForce GTX 1080 Ti GPU. The developed tracker achieved an average processing time of 25 ms per frame when executed on the combined CPU–GPU configuration, excluding the object detection stage. The data-driven association component was executed on the GPU, whereas the feature-engineered association module was implemented on the CPU.

For training the neural network models, the dataset introduced in Section 3.3.6.1 was employed. In addition, the original dataset was augmented using several data augmentation strategies, including horizontal flipping, salt-and-pepper noise injection, motion blur simulation, Gaussian noise addition, image sharpening, and contrast normalization. The resulting dataset was partitioned into training, validation, and test subsets, comprising 70%, 10%, and 20% of the data, respectively. Each model was trained for 40 epochs using a learning

rate of 0.0005, with root mean square propagation (RMSProp) employed as the optimization algorithm.

The classification accuracy obtained on the test set was 98.34% for the model operating on the full detection image. For the part-based similarity models, the achieved accuracies were 96.82% for the top-left region, 96.61% for the top-right region, 95.92% for the bottom-left region, and 96.01% for the bottom-right region.

The pedestrian detection component integrated into the tracking pipeline was based on a YOLO architecture [244], which was initially trained on the FLIR-ADAS dataset [214] and subsequently fine-tuned on the CrossIR dataset [209], acquired using a PathFindIR thermal camera. The CrossIR dataset contains samples captured under diverse illumination conditions (daytime and nighttime), weather scenarios (sunny, rainy, and foggy), and temperature regimes (cold and warm), thereby providing a representative range of real-world operating conditions.

To quantitatively assess tracking performance, the proposed framework was evaluated using the PTB-TIR benchmark dataset [230], which consists of multiple thermal image sequences annotated with frame-level ground truth information. One of the primary evaluation metrics provided by this benchmark is the Center Location Error (CLE), defined as the average Euclidean distance between the estimated target position and the corresponding ground truth location. A tracking result is considered successful at a given frame if the CLE remains below a threshold of 20 pixels, as specified by the benchmark protocol.

In addition, the PTB-TIR benchmark provides comparative results for a variety of state-of-the-art tracking approaches, enabling a systematic performance analysis across different methods. For the evaluation presented in this work, only sequences acquired using vehicle-mounted thermal cameras were considered, in order to maintain consistency with the intended application domain of intelligent transportation systems.

The performance of the proposed tracking framework with respect to the Center Location Error (CLE) metric, evaluated on all automotive-related sequences from the PTB-TIR benchmark, is illustrated by the precision plot shown in Figure 3.22. The numerical values and graphical results presented in Figures 3.22 and 3.23 were obtained using the official PTB-TIR Evaluation Toolkit, which is described in detail in

[230]. For improved readability and quantitative comparison, the values depicted in Figure 3.22 are additionally reported in Table 3.7.

Table 3.7 Evaluation with respect to the precision metric

Method	Tracking Precision Score
DeepSTRCF [233]	88.8%
Proposed	**86.2%**
MLSSNet [235]	83.4%
CFNet [236]	83.3%
VITAL [237]	82.8%
TADT [239]	77.9%
MCFTS [240]	77.6%
HDT [245]	75.8%
HCF [246]	75.6%
SiamFC [160]	74.5%

In addition to the CLE-based precision metric, the PTB-TIR benchmark also provides an overlap-based performance measure, which quantifies the spatial intersection ratio between the predicted bounding box and the corresponding ground-truth annotation. A tracking result is considered successful at a given frame if the overlap ratio exceeds a predefined threshold. The success plot aggregates these measurements over varying overlap thresholds in the interval [0, 1] and

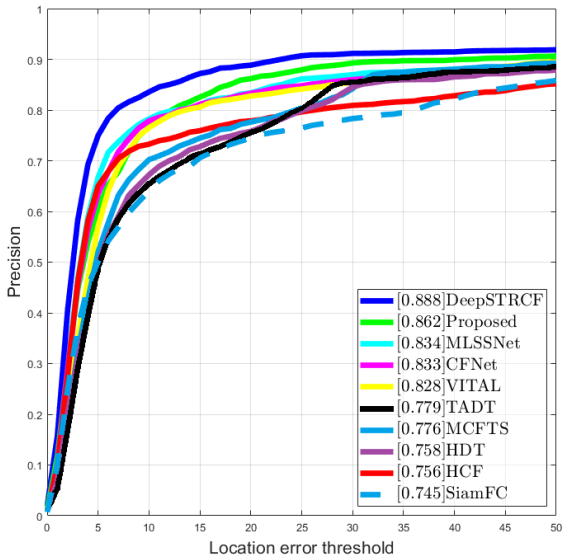

Figure 3.22. Position precision plot on the PTB-TIR benchmark

is commonly used to rank tracking methods based on localization accuracy. The success plot corresponding to the evaluated sequences is presented in Figure 3.23.

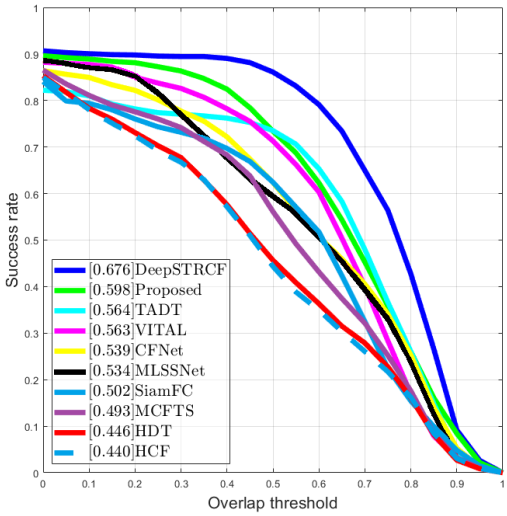

Figure 3.23 Plot that measures the overlapping score between the tracked object and ground truth

Compared to the highest-ranked methods reported in the benchmark, the proposed tracking approach was explicitly developed under the practical constraints imposed by automotive applications. In particular, the framework demonstrates robustness in scenarios involving partial occlusions and maintains tracking performance in previously unseen environmental conditions that were not represented in the training dataset. Furthermore, unlike several benchmark methods that focus exclusively on single-target tracking, the proposed solution supports full multi-object tracking capabilities, thereby addressing the requirements of real-world intelligent transportation systems.

Furthermore, the proposed solution exhibits a relatively low implementation complexity, does not require large-scale training datasets, and can be readily extended through the integration of additional feature representations. For improved clarity and quantitative comparison, the numerical values corresponding to Figure 3.23 are also reported in Table 3.8.

In addition to the evaluation metrics discussed above, the proposed tracking framework was further assessed using the standard MOTA and MOTP metrics.

The MOTA metric, previously defined in (3.102), provides an aggregate measure of tracking performance by accounting for all major error sources, including false positives, missed detections, identity switches, and mismatches across all frames. The maximum attainable MOTA value is 1, which corresponds to an ideal tracker with zero errors.

Table 3.8 Evaluation with respect to the success score

Method	**Tracking Success Score**
DeepSTRCF [233]	67.6%
Proposed	**59.8%**
TADT [239]	56.4%
VITAL [237]	56.3%
CFNet [236]	53.9%
MLSSNet [235]	53.4%
SiamFC [160]	50.2%
MCFTS [240]	49.3%
HDT [245]	44.6%
HCF [246]	44%

The MOTP metric, defined in (3.103), quantifies the spatial precision of the tracker by measuring the alignment accuracy between the estimated bounding boxes and the corresponding ground-truth annotations.

For the computation of MOTP, a distance measure is evaluated between all matched track hypotheses and the corresponding ground-truth bounding boxes and normalized by the number of successfully matched objects to obtain an average localization error. These values are subsequently accumulated over all frames of the evaluation sequence in order to derive the final MOTP score.

The fundamental distinction between the two metrics lies in their evaluation focus: MOTP quantifies the spatial accuracy of bounding box localization for correctly associated targets over time, whereas MOTA summarizes overall tracking performance by accounting for association errors, missed detections, false positives, and identity inconsistencies, including unmatched tracks.

An identity switch (IDSW) event occurs when a tracked object is incorrectly reassigned a new identity after temporary loss, or when two object identities are erroneously swapped as a result of an incorrect association between tracks and detections. Table 3.9 reports the evaluation results of the proposed tracking framework in terms of

MOTA, MOTP, and IDSW for the multi-pedestrian tracking scenario on the CrossIR dataset [209].

Table 3.9 Evaluation with respect to different metrics

Method	MOTA	MOTP	IDSW
Proposed	**86.14%**	**88.63%**	**134**
Base Solution	81.36%	83.17%	143
TADT	80.3%	81.7%	121
MLSSNet	79.8%	82.3%	269
SiamFC	76.4%	82.1%	343

The proposed tracking framework demonstrates reliable association between detections and track hypotheses and enables robust multi-pedestrian tracking in thermal imagery under diverse environmental conditions, including adverse weather and partial occlusions. By integrating data-driven similarity measures with feature-engineered association scores, the system attains increased adaptability to previously unseen traffic scenarios, thereby enhancing overall robustness and generalization capability.

To quantitatively assess the contribution of the tracking component to detection performance, several evaluation metrics are introduced. An object is considered correctly identified if the positional deviation from the corresponding ground-truth annotation does not exceed 10 pixels along either the horizontal or vertical image axis. Precision is defined as the ratio between the number of correctly identified objects and the total number of ground-truth objects in a given frame. Recall is computed as the ratio between the number of correctly identified objects and the total number of detected objects in that frame. Accuracy is defined as the proportion of correctly identified objects relative to the total number of ground-truth objects.

The detector and tracker were evaluated on more than 100 real-world traffic sequences containing multiple pedestrians and exhibiting a wide range of illumination and weather conditions. The evaluation results reported in Table 3.9 were obtained on the CrossIR dataset introduced in [209]. This evaluation was conducted in order to analyze the performance of the proposed tracking framework in complex multi-object scenarios and under challenging environmental conditions.

It is well known that object detectors may fail to provide consistent detections in the presence of occlusions, clutter, or reduced image quality. The objective of this evaluation was therefore to demonstrate the ability of the tracking module to compensate for intermittent

detection failures by maintaining object identities across frames, even when the detector temporarily fails to identify pedestrians reliably.
Comparative results are summarized in Table 3.10. The proposed approach builds upon the baseline tracking solution introduced in the previous section, which relies exclusively on feature-engineered association scores.

Table 3.10 Ablation study with respect to several metrics

	Average Precision	**Average Accuracy**	**Average Recall**
Object Detector	75.98%	66.47%	98.67%
Base Solution	80.01%	76.4%	95.8%
Base with only Data Driven Score	86.15%	79.22%	93.21%
Base with only Engineered Score	83.25%	78.43%	93.71%
Base with all Fused Scores (proposed)	88.61%	80.02%	94.8%

An ablation study is presented to quantify the individual contribution of each proposed component. In particular, the results highlight that the fusion of data-driven similarity measures with feature-engineered association costs, when integrated into the baseline framework, yields consistent improvements across all reported evaluation metrics.

As can be seen, the proposed solution improved the performance of the object detector, leading to better overall results. Furthermore, it is worth noting that the feature engineered score can also be applied to other object classes, such as vehicles; but, illustrating this was out of the scope of the paper. The method from this section can be found at [352].

3.3.5 Multi-Object Tracking Segmentation and Validation (MOTSV)

3.3.5.1 Pipeline and Validation Scheme

This section introduces a unified framework for panoptic perception in thermal imagery, capable of performing both semantic and instance segmentation, as well as multi-object tracking based on bounding boxes and instance-level masks. The proposed pipeline incorporates an additional validation stage aimed at verifying both detection reliability and class-label correctness for the object categories of interest. This

validation mechanism is motivated by the fact that classification outputs may be affected by noise and uncertainty, while object detectors are inherently susceptible to errors such as false positives and missed detections.

Furthermore, the tracking component is configured to operate only on a predefined subset of semantic classes rather than on all objects present in the scene. Consequently, a dedicated validation module plays a critical role in filtering detections and ensuring that only relevant object instances are propagated to subsequent processing stages.

The framework integrates three data-driven modules for validation-related tasks and a model-based approach for data association and temporal tracking. The outputs of the learning-based components and the model-driven tracking module are subsequently fused to produce the final panoptic representation at both bounding-box and pixel-mask levels. An overview of the proposed processing pipeline is illustrated in Figure 3.24.

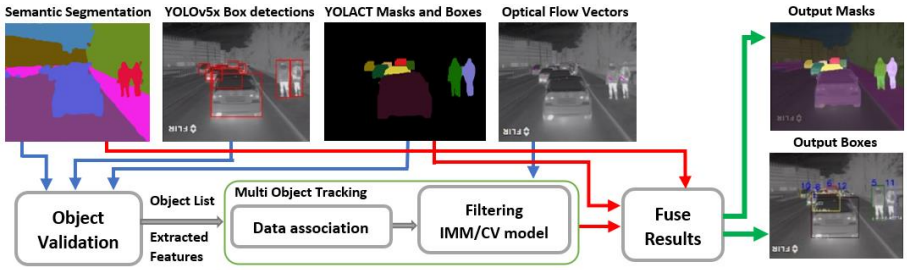

Figure 3.24. The proposed multi-object tracking and segmentation for thermal images pipeline

It is important to emphasize that the primary focus of this work lies on the integration and fusion of segmentation, validation, and tracking components rather than on the design of the object detection architecture itself. As a result, the detection module can be readily replaced with alternative state-of-the-art detectors without affecting the overall structure of the proposed pipeline.

In the presented implementation, semantic segmentation is performed using ERFNet [220], which achieves real-time performance with an average inference time of approximately 8 ms on GPU for input images of resolution 640×480 pixels. The network was trained to recognize 25 semantic classes relevant to the considered driving scenarios.

A dual-detector configuration based on YOLOv5 [221] and YOLACT [222] is employed for instance-level object detection. The use of two independent detection models is motivated by the requirements of the validation stage, whose primary objective is to mitigate classification and localization errors introduced by individual detectors. To enable reliable validation, each object instance is required to be independently analyzed by at least three distinct processing components. In the proposed framework, these components consist of the detections generated by YOLOv5, the instance masks produced by YOLACT, and the semantic labels obtained from the ERFNet-based segmentation module.

The YOLOv5x architecture is used as the primary detector and was trained on the FLIR ADAS dataset, which was adapted to include only the object categories relevant to the targeted application domain. Specifically, the training dataset was restricted to the following classes of interest: car, bus, truck, person, cyclist, motorcycle, bicycle, traffic sign, and traffic light (semaphore).

A known limitation of the YOLACT model is related to its sensitivity to the confidence threshold. When the detection confidence threshold is set above 50%, a significant portion of relevant object instances may be missed. Conversely, reducing this threshold below 50% substantially increases the number of false positives. To address this trade-off, the YOLACT detection threshold is deliberately set to a low value of 10%, thereby maximizing recall at the cost of introducing additional candidate detections.

Subsequently, an intersection-over-union (IoU) based filtering procedure is applied to associate YOLACT detections with those generated by YOLOv5. For each object detected by YOLOv5, the two YOLACT instances exhibiting the highest IoU values are retained as candidate matches. To enable efficient retrieval, the indices of the associated detections are stored in a lookup table. Only associations exceeding a predefined IoU threshold, set to 0.8 in the presented implementation, are considered valid and propagated to the subsequent validation and fusion stages.

During the validation stage, semantic consistency between the YOLOv5 detections and the candidate instances produced by YOLACT is evaluated. For each YOLOv5 detection, the semantic label is first compared with the class of the YOLACT instance exhibiting the highest Intersection over Union (IoU) score. If no correspondence is found, the

comparison is repeated using the second-best YOLACT candidate. When a semantic match is identified, the object hypothesis is considered valid.

In cases where both YOLACT candidates satisfy the semantic consistency criterion, the instance associated with the larger IoU value is selected for subsequent processing.

If the semantic label cannot be validated using the YOLACT outputs, an additional verification step is performed using the semantic segmentation results. Specifically, the class predicted by YOLOv5 is compared with the dominant semantic category extracted from the segmentation map within the region of interest (ROI) defined by the YOLOv5 bounding box.

To efficiently extract this dominant class, a GPU-accelerated histogram computation is implemented using the CUDA framework. The semantic category corresponding to the histogram bin with the highest vote count is selected as the segmentation-based label for the object hypothesis.

An object detection is considered semantically valid if at least two of the three independent classification sources—YOLOv5, YOLACT, and semantic segmentation—agree on the assigned class label. Conversely, when all three predictions differ, the semantic category of the object is marked as unknown.

In the final stage of the validation procedure, remaining YOLACT detections with confidence scores exceeding 50% that have not been previously associated with any YOLOv5 detections are examined. For these instances, semantic consistency is assessed by comparing the YOLACT-predicted class label with the dominant semantic category extracted from the segmentation map within the corresponding region of interest (ROI).

In a similar manner, YOLOv5 detections that have not been matched during the earlier validation steps are also processed. Their semantic labels are verified using the semantic segmentation output by analyzing the dominant class distribution within the bounding box ROI.

At the conclusion of the validation pipeline, a consolidated set of object bounding boxes with verified semantic labels is obtained. For detections associated with YOLACT instances, the corresponding instance segmentation masks are directly adopted. In cases where no instance mask is available, object masks are generated by extracting pixels belonging to the validated semantic class within the bounding box region from the semantic segmentation map.

To ensure real-time performance, all processing stages within the validation and fusion pipeline are parallelized across CPU and GPU resources.

3.3.5.2 Mask Refinement

Instance segmentation masks may occasionally exhibit imperfections, including spurious artifacts or incomplete coverage of the target object. To address these limitations, a lightweight and computationally efficient mask refinement procedure is employed.

The instance segmentation module produces, for each detected object, a binary mask image with the same spatial resolution as the input frame, where foreground pixels are assigned a value of 1 and background pixels a value of 0. Given that the corresponding object bounding box is available, the refinement process is restricted to the associated region of interest (ROI).

In the first stage, a morphological opening operation with a 7×7 structuring element is applied in order to separate small artifacts that may be erroneously attached to the object mask. Subsequently, a two-pass connected-component labeling algorithm based on equivalence classes [223] is performed within the ROI. During this step, all connected foreground components are identified and their respective areas are computed. The two-pass labeling strategy is selected due to its favorable computational efficiency compared to alternative clustering approaches. The connected component exhibiting the largest area within the ROI is retained as the refined object mask.

Following this operation, the mask is further expanded using

Figure 3.25. In the left-hand side the erroneous instance mask. In the right-hand side the corrected instance masks are displayed

information from the semantic segmentation output. Additional pixels are incorporated into the instance mask only if two conditions are simultaneously satisfied: (i) the pixel belongs to the dominant semantic class associated with the object within the ROI, and (ii) the pixel is spatially adjacent to an existing foreground pixel of the refined instance mask. This constrained expansion strategy improves mask completeness while preserving class consistency and spatial coherence.

Mask expansion is constrained to the region of interest defined by the corresponding detection bounding box. Each tracked object instance is therefore associated with an individual refined mask. After the tracking stage, all instance masks are aggregated into a single composite image, where each object is encoded using a unique color corresponding to its assigned track identifier.

Prior to mask fusion, the tracked objects are sorted in ascending order according to their distance from the camera. The distance estimation is performed using the method described in [209]. This ordering strategy ensures correct rendering of overlapping objects by prioritizing closer instances in the visualization process.

An example of the refinement procedure is illustrated in Figure 3.20, where the object detector produces an inaccurate instance mask accompanied by multiple spurious artifacts for the object highlighted in pink. After applying the proposed refinement pipeline, the undesired artifacts are effectively removed, and the object mask is corrected using guidance from the semantic segmentation output.

Although increasing the amount of annotated training data can reduce the occurrence of such erroneous predictions, it is not feasible to exhaustively cover all environmental conditions encountered in real-world driving scenarios. Consequently, the inclusion of a dedicated mask refinement stage contributes to improved robustness and reliability of the overall perception pipeline. Furthermore, while such failure cases occur infrequently, the availability of a mitigation mechanism is essential to prevent the propagation of segmentation errors into downstream tracking and decision-making modules.

3.3.5.3 Data Association

The tracking framework considered in this work follows a tracking-by-detection paradigm, in which data association between tracks and detections is performed using a similarity function that combines appearance-based and motion-based cues. The tracking architecture

has been explicitly optimized for computational efficiency, enabling deployment on resource-constrained embedded platforms.

The input to the tracking module consists of a set of object detections augmented with extracted feature descriptors and instance masks, which are subsequently exploited during the association stage. The output of the tracking module is a collection of object tracks, each associated with a unique identifier and a temporally filtered trajectory representation. Track outputs are maintained at both bounding box level and instance mask level, thereby supporting downstream panoptic perception tasks.

The overall processing pipeline comprises several core components, including clutter suppression, similarity cost computation, track-to-detection association, state estimation and update, and mask refinement. To further reduce computational complexity, a validation gating mechanism is applied around the predicted track positions. Only detections falling within the corresponding validation region are considered during the association stage, significantly limiting the number of candidate correspondences and improving runtime performance.

A feature-engineered data association strategy is employed to compute the similarity cost between tracks and detections. The similarity function is formulated as a weighted combination of multiple complementary descriptors, allowing the integration of heterogeneous information sources. Compared to purely data-driven approaches, this model-based formulation provides greater interpretability and enables explicit control over the contribution of individual features. Moreover, this design choice facilitates systematic performance analysis and targeted optimization in cases where association errors occur, while maintaining low computational overhead suitable for real-time operation.

The appearance-based similarity score enables the tracker to discriminate among objects by exploiting visual descriptors. In addition, the appearance representation associated with each track is continuously updated, allowing the model to adapt to variations caused by illumination changes, object deformation, and viewpoint shifts.

The appearance similarity between track i and detection j is computed as a weighted combination of multiple visual features, as defined in Equation (3.129). The resulting similarity measure, denoted

by α(i, j), attains minimal values when the two entities correspond to the same physical object.

$$\alpha(i,j) = \vartheta(i,j) + w_2\gamma(i,j) + w_3\tau(i,j) + w_4\rho(i,j) + w_5\mu(i,j) + w_6\sigma(i,j) + w_7\delta(i,j)\omega(i,j) + (1 - miou(i,j))w_8 + w_9\theta \quad (3.129)$$

The individual terms of the appearance similarity function have the following interpretations: ϑ denotes the part-based orientation similarity cost, γ corresponds to the semantic segmentation consistency term, τ represents the spatial overlap cost, ρ denotes the uniform local binary pattern (ULBP) texture similarity, μ and σ correspond to the mean intensity and variance similarity terms, θ encodes the object dimension consistency, δ and ω represent the classification confidence similarity, and mIoU denotes the mask-based intersection-over-union metric.

The weighting coefficients associated with these terms were determined empirically through extensive experimental evaluation. The similarity components γ, τ, μ, σ, and θ are computed using the L2 norm between the corresponding feature values stored in the track representation and those extracted from the current detection. The mIoU term is computed directly as the intersection-over-union between the instance segmentation masks associated with the track and the detection.

Texture-based and intensity-related descriptors, including the ULBP histogram, mean intensity, and variance, are extracted following the methodology introduced in [209] and described in the previous section. However, in the present framework, these features are computed exclusively within the spatial support defined by the instance segmentation mask rather than over the entire bounding box region, thereby improving robustness by restricting the analysis to object-specific pixels.

The classification confidence component of the appearance similarity is computed using the L2 norm between the classification probability stored in the track representation (corresponding to the most recent associated detection) and the classification probability of the current detection. The weighting factor ω is set to 1 when the semantic class labels of the track and detection coincide and is assigned a large penalty value otherwise (2000 in the present implementation), thereby

strongly discouraging associations between objects belonging to different semantic categories.

To further incorporate contextual information, a semantic class histogram is extracted within the region of interest defined by the object bounding box. The underlying assumption is that the local semantic context surrounding a given object instance remains relatively stable across consecutive frames. Consequently, the histogram difference between the track and detection representations is expected to be minimal for correct associations. The semantic context similarity between a track and a detection is computed according to (3.130), where γ(x) denotes the value of the histogram bin corresponding to semantic class x.

$$\gamma(i,j) = \sum_{k=0}^{25} |\gamma_i(k) - \gamma_j(k)| \tag{3.130}$$

Although thermal radiation is independent of external illumination, the interaction between human clothing, material properties, and emitted infrared energy gives rise to distinctive textural and structural patterns that can be effectively exploited for data association. Furthermore, while the surrounding environment influences the apparent radiance of a target, the relatively high frame rate of the thermal sensor and the continuous update of track representations ensure that the overall multi-object tracking performance remains stable. Empirical observations indicate that object texture exhibits limited variation across consecutive frames, making it a reliable cue for measuring the similarity between tracks and detections.

To robustly capture object texture, including in scenarios involving partial occlusions, a grid-based matching strategy combined with gradient orientation information is employed. Specifically, multiple histogram-based comparisons are performed between the appearance representations of track $i$and detection j. In the first stage, the gradient magnitude G and orientation θ are computed as defined in (3.120), using the horizontal and vertical image derivatives $I_x$and I_y obtained via the Sobel operator.

The resulting orientation values are subsequently quantized by a factor of 18, yielding a maximum of 20 discrete orientation bins. The bounding box associated with the object of interest is then partitioned into a 3×3 spatial grid. For each grid cell, an orientation histogram with 20 bins is constructed. Only pixels belonging to the object instance

mask (i.e., mask value equal to 1) contribute to the histogram. Each contributing pixel casts a weighted vote proportional to its gradient magnitude.

The similarity measure for an individual grid cell is computed according to (3.131) and (3.132), where $\phi_x(i,j)$ denotes the similarity between the x-th grid cell of track $i$and detection j. The parameter εrepresents an experimentally determined threshold, set to 0.15 in the present implementation. The aggregated grid-based orientation similarity score $\vartheta(i,j)$ is obtained by combining the cell-level similarities. This score attains minimal values when the track and detection correspond to the same physical object instance

$$\varphi(i,j)_x = \begin{cases} 1, \left(\sum_{k=0}^{20} \frac{|H_i(k)-H_j(k)|}{20}\right) \leq \varepsilon \\ 0, otherwise \end{cases} \quad (3.131)$$

$$\vartheta(i,j) = \frac{w_1}{\sum_{x=0}^{8} \varphi(i,j)_x} \quad (3.132)$$

The motion similarity component, denoted by $m(i,j)$, captures the spatial consistency between a track $i$and a detection j. It is computed as the Euclidean distance between the predicted center position of the tracked object and the center position of the detected bounding box, both expressed in the two-dimensional image coordinate frame.

The overall association score between a track and a detection is obtained by combining the appearance-based similarity score and the motion-based similarity score through a weighted summation. Once the pairwise association scores have been computed for all valid track–detection pairs, an optimal assignment procedure is applied using the Hungarian algorithm [142]. This step yields a globally consistent matching between the set of active tracks and the current detections.

Following the association stage, each matched track is updated using the newly assigned detection. A recursive filtering scheme is subsequently applied to refine the state estimate and to smooth the resulting object trajectory over time, thereby reducing the impact of measurement noise and short-term localization fluctuations.

3.3.5.4 Optical Flow Based Model Selector

Objects observed in real-world traffic scenarios exhibit a wide range of motion patterns. Consequently, the use of a single motion model is often insufficient for accurately capturing the dynamics of heterogeneous road users. To address this limitation, this section introduces a multi-model tracking strategy that combines complementary motion models through parallel filtering and adaptive state selection.

The core principle of the proposed approach consists of executing two filtering pipelines in parallel and dynamically selecting the most appropriate state estimate for each tracked object based on motion cues derived from optical flow. The first pipeline relies on a standard Kalman Filter (KF) with a constant velocity (CV) motion model, which is well suited for static targets and objects exhibiting low-speed or near-linear motion. The second pipeline employs an Interacting Multiple Model (IMM) filter that integrates both constant velocity (CV) and constant acceleration (CA) motion models. This configuration is specifically designed to capture the behaviour of manoeuvring targets and objects undergoing abrupt changes in velocity.

In the IMM framework [225], a finite set of motion modes is defined to represent the possible dynamic states of a target. It is assumed that, at any given time instant, the object operates in exactly one of these modes. While increasing the number of modes can improve modelling fidelity, it also leads to a substantial increase in computational complexity. In order to satisfy real-time performance constraints, the proposed implementation employs two motion hypotheses, corresponding to the CV and CA models. Upon track initialization, both modes are assigned equal prior probabilities.

Transitions between motion modes are modelled using a discrete-time Markov process, represented by the mode transition probability matrix Π, which governs the likelihood of switching between the CV and CA models. The values of this transition matrix were determined empirically and are provided in (3.133).

$$\Pi = \begin{bmatrix} 0.97 & 0.03 \\ 0.06 & 0.94 \end{bmatrix} \tag{3.133}$$

Using the mode transition probability matrix, the updated mode probabilities are computed at each recursion step according to (3.134), where m denotes the number of motion models, μ represents the prior mode probabilities, and $\bar{c}$denotes the updated mode probability vector.

$$\overline{c_j} = \sum_{i=0}^{m-1} \mu_i \Pi_{i,j}, j = \overline{0, m-1} \tag{3.134}$$

Since the state vectors associated with the two motion models have different dimensionalities (four state variables for the CV model and six for the CA model), only the four common state components are retained for the fusion process. These shared components correspond to the two-dimensional position and velocity states. Consequently, both the state update and covariance propagation are performed in the reduced four-dimensional state space. The mixing probabilities, also referred to as mixing weights, are subsequently computed according to (3.135).

$$w_{i,j} = \frac{\Pi_{i,j}\mu_i}{\overline{c_j}}, i, j = \overline{0, m-1} \tag{3.135}$$

After the update step of each Kalman filter in the model bank, individual state estimates and covariance matrices are obtained. Subsequently, a mixed state estimate and covariance are computed for each filter by forming weighted combinations of the means and covariances from all filters, using the corresponding mixing probabilities. In this process, filters with lower likelihood receive stronger corrections from the more probable models, whereas highly probable filters are only weakly influenced by less likely ones. This interaction mechanism is formalized in (3.136) and (3.137).

$$\overline{x_j} = \sum_{i=0}^{m-1} w_{i,j} x_i \tag{3.136}$$

$$P_j = \sum_{i=0}^{m-1} w_{i,j} [\left(x_i - \overline{x_j}\right)\left(x_i - \overline{x_j}\right)^T + P_i] \tag{3.137}$$

The mode probability at time step k is updated for each dynamic model by combining the mode probabilities from the previous time step with the corresponding measurement likelihoods of the individual filters, as expressed in (3.138).

$$\mu_k^i = \frac{\mu_{k|k-1}^i L_k^i}{\sum_j \mu_{k|k-1}^j L_k^j}, i, j = \overline{0, m-1} \tag{3.138}$$

The final state estimate of the IMM framework is obtained by forming a weighted combination of the individual Kalman filter estimates, as shown in (3.139) and (3.140), where $\bar{x}_k^i$ and $\bar{P}_k^i$ denote the predicted state mean and covariance of the i-th filter in the model bank at time step k.

$$\widetilde{x_k} = \sum_{i=0}^{m-1} \mu_k^i \overline{x_k^i} \tag{3.139}$$

$$P = \sum_{i=0}^{m-1} \mu_k^i [\left(\overline{x_k^i} - \widetilde{x_k}\right)\left(\overline{x_k^i} - \widetilde{x_k}\right)^T + \overline{P_k^i}] \tag{3.140}$$

The measurement covariance matrices employed by the IMM filter model bank are class-dependent. Specifically, four distinct measurement covariance matrices are used, corresponding to the object categories: car, pedestrian, rider, and other. In addition to the IMM filter, which integrates two motion models, a standalone Kalman filter based on a constant velocity (CV) motion model is also incorporated into the tracking framework. This CV-based Kalman filter operates with a single measurement covariance matrix and employs comparatively low process noise covariance values.

Although the IMM filter provides increased flexibility for modelling manoeuvring targets, there exist scenarios in which its performance may be inferior to that of a simpler CV-based Kalman filter. Such situations include low-dynamic environments (for instance, parking areas), urban driving scenarios characterized by approximately linear vehicle motion relative to the ego vehicle, as well as conditions with elevated measurement noise, such as heavy fog or sensor degradation.

To address these limitations, an adaptive filter selection strategy has been introduced, in which optical flow information is exploited to determine, at each time step, the most suitable filtering approach for state estimation. Sparse optical flow methods may yield unreliable results from a qualitative perspective, whereas dense optical flow techniques are computationally expensive and can produce inaccurate estimates on unstructured or low-texture surfaces. Nevertheless, although individual pixel-level motion trajectories may be noisy or erroneous, the aggregation of multiple flow vectors within a region of interest (ROI) can provide robust cues regarding the dominant motion pattern of an object across consecutive frames.

In the proposed framework, the optical flow algorithm described in [219] is employed, followed by a post-processing stage aimed at

estimating the representative displacement magnitude and direction for each tracked object. Specifically, the angular domain is discretized into 36 bins, and each optical flow vector contributes a vote to the corresponding bin according to its orientation. After the voting procedure, the mean magnitude and mean orientation are computed independently for each bin. The bin accumulating the highest number of votes is then selected, and its associated mean magnitude and orientation are retained as the dominant flow parameters for the ROI.
To estimate the displacement associated with the aggregated flow parameters, two reference points are defined. The first point, $P_1(x_1, y_1)$, corresponds to the center of the object region. The second point, $P_2(x_2, y_2)$, is subsequently computed using the estimated flow magnitude and orientation, as expressed in Equations (3.141) and (3.142).

$$x_2 = x_1 + length \times \cos(angle \times \frac{\pi}{180}) \quad (3.141)$$

$$y_2 = y_1 + length \times \sin(angle \times \frac{\pi}{180}) \quad (3.142)$$

The ratio between the Euclidean distance of points $P_1$and $P_2$and the object width provides an estimate of the relative displacement of the target between consecutive frames. When this ratio is below unity, the object is considered to exhibit limited motion, and the Kalman filter based on a constant velocity model is selected for state prediction. Conversely, when the ratio exceeds one, indicating more pronounced motion dynamics, the interacting multiple model (IMM) filter is employed.

Following motion estimation and state update, the tracked objects are ordered according to their distance from the camera. The depth information is obtained using the approach described in [209]. Subsequently, all active tracks, including both bounding boxes and instance masks, are projected onto the image plane to generate the final visualization results, as illustrated in Figures 3.26 and 3.27.

Track lifecycle management is applied to handle object appearance and disappearance within the field of view, as well as the removal of stale tracks that have not been updated for extended periods. Since comprehensive track management strategies have been extensively discussed in the literature [141, 209] and were also addressed in previous sections of this monograph, this aspect is not further detailed here.

Figure 3.26. Multiple tracked pedestrians and vehicles in a city.

Figure 3.27. Tracked objects inside a parking lot.

3.3.5.5 Evaluation of Multi Object Tracking and Segmentation

The proposed framework was implemented in C++, while the neural network models were trained using the PyTorch framework and subsequently deployed in C++ through the LibTorch interface to ensure native integration. OpenCV was employed for visualization, result rendering, and fusion of intermediate outputs into unified representations. To achieve real-time performance, hardware acceleration techniques based on CUDA and parallel processing using OpenMP were incorporated.

The development and experimental evaluation were conducted on a computing platform equipped with an Intel Core i7-11370H processor operating at 3.3 GHz and an NVIDIA GeForce RTX 3070 GPU. All datasets used for training the neural network components were partitioned into three subsets, following a standard protocol: 70% for training, 10% for validation, and 20% for testing.

The ERFNet model employed for semantic segmentation was trained on a thermal dataset containing more than 3,500 annotated images and 25 semantic classes. Data augmentation techniques, including geometric translations, horizontal flipping, Gaussian noise injection, Gaussian blurring, and gamma contrast adjustment, were applied to improve generalization. The training process was carried out for a total of 200 epochs, comprising 100 epochs for the encoder and 100 epochs for the decoder, using a learning rate of 5×10^{-4} and a batch size of 4. The resulting Intersection over Union (IoU) score on the test set reached 62.03%.

An alternative semantic segmentation architecture, FastSCNN [247], was also evaluated due to its computational efficiency; however, it was not adopted in the final pipeline owing to its substantially lower segmentation accuracy, achieving an IoU of only 26.4%.

The YOLOv5 detector was trained on the FLIR ADAS thermal dataset and subsequently fine-tuned on the CrossIR dataset [209], resulting in a mean Average Precision (mAP) of 55.04%.

For instance segmentation, YOLACT equipped with a ResNet101-FPN backbone was initially trained on the COCO dataset and further fine-tuned using 2,000 thermal images annotated with instance-level masks. The same augmentation strategies applied during semantic segmentation training were also employed for this stage. The integration of the proposed mask refinement procedure led to a measurable improvement in instance segmentation performance: on a validation set of 2,000 thermal images, the refined YOLACT configuration achieved a mAP of 34.6%, compared to 31.2% obtained without refinement.

The tracking performance was evaluated using the PTB-TIR benchmark dataset [230], as no publicly available dataset currently provides joint annotations for tracking and instance segmentation in the thermal domain. Since the proposed framework outputs both bounding boxes and instance masks, quantitative tracking evaluation could be performed using the bounding box annotations provided by PTB-TIR. This benchmark comprises multiple thermal video sequences with manually annotated ground truth and enables standardized performance comparisons.

Two evaluation metrics provided by the benchmark were employed. The first metric, Center Location Error (CLE), measures the average Euclidean distance between the predicted object center and the corresponding ground truth position. A tracking result is considered

successful at a given frame when the CLE remains below a threshold of 20 pixels. The second metric, the overlap score, quantifies the spatial overlap ratio between the predicted bounding box and the ground truth annotation.

A tracking hypothesis is considered successful at a given frame when the overlap score exceeds a predefined threshold. To enable a comprehensive comparison among different tracking approaches, the benchmark evaluates performance over a continuous range of overlap thresholds varying from 0 to 1. The resulting precision and success plots obtained on the PTB-TIR benchmark are illustrated in Figure 3.28.

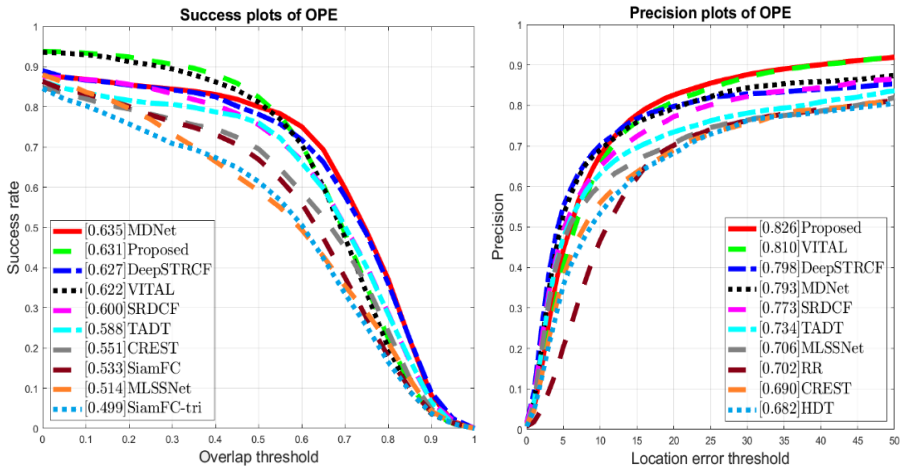

Figure 3.28. Graphical evaluation on the PTB-TIR dataset.

The evaluation was conducted exclusively on sequences acquired using vehicle-mounted thermal cameras, as the proposed framework is primarily intended for intelligent transportation and automotive perception applications. In addition, only the ten most representative tracking methods were included in the comparative plots in order to preserve visual clarity and interpretability. For convenience, the most relevant numerical values extracted from the precision and success curves are also summarized in Table 3.11.

The performance of the prediction stage was further analyzed by independently evaluating two motion modeling configurations: a standard Kalman Filter employing a constant velocity (CV) model and an Interacting Multiple Model (IMM) filter combining constant velocity and constant acceleration (CV/CA) dynamics. Under identical data

association settings, the CV-based Kalman Filter achieved a precision of 80.5%, while the IMM-based approach yielded a precision of 78.7%.

Table 3.11 Results on the PTB-TIR dataset

Method	**Tracking Success Score**	**Tracking Precision Score.**
MDNet [241]	63.5%	79.3%
Proposed	**63.1%**	**82.6%**
DeepSTRCF [233]	62.7%	79.8%
VITAL [237]	62.2%	81.0%
TADT [239]	58.8%	73.4%
CREST [242]	55.1%	69%
MLSSNet [235]	51.4%	70.6%
HSSNet [243]	47.2%	66.5%

In contrast to several existing approaches included in the PTB-TIR benchmark, which primarily focus on single-object or pedestrian-only tracking, the proposed framework supports multi-class and multi-object tracking. This design choice enables broader applicability in complex traffic environments, where multiple object categories coexist and must be tracked simultaneously.

The running time of the tracking approach is 8 ms on the CPU. In Table 3.12 the running time of the proposed method on the intel core I7 on which it was developed and on a Nvidia Jetson Tegra TX2 are presented.

Table 3.12 Running time of the solution on different platforms.

Platform	**Overall Average Running Time**
Intel Core i7-11370H processor having a 3.3 Ghz frequency and an Nvidia GForce RTX 3070 GPU	100 ms
Jetson Tegra TX2	290 ms

The tracking approach presented in this section can be found at [353]. Supplementary video with results available at[4]

[4] https://youtu.be/Bit3DkKeyK4?si=rI7JnVAPDLr4-i0F

3.3.6 Created Datasets

3.3.6.1 Pedestrian Dataset from Thermal Images

Siamese neural architectures extract discriminative representations from pairs of input samples and generate embedding vectors that can be compared using a similarity or energy-based function in order to assess the correspondence between the input elements. To enable the training of Siamese models for pedestrian discrimination and re-identification in thermal imagery, a dedicated dataset was constructed, comprising more than 200 unique pedestrian identities extracted from thermal video sequences.

The resulting dataset contains over 26,000 cropped pedestrian images acquired under diverse environmental and illumination conditions. The recording scenarios are representative of real-world driving environments and include daytime and nighttime operation, as well as varying weather conditions such as rain, fog, clear skies, and seasonal variations including spring and winter.

The dataset construction process was carried out in three successive stages. In the first stage, pedestrian instances were automatically extracted from all frames of each recorded sequence, and the resulting cropped samples were organized according to the corresponding sequence identifiers. In the second stage, pedestrian samples belonging to the same physical object within each sequence were grouped into instance-specific folders. Since identical pedestrian identities may appear across multiple sequences, a third validation stage was introduced, in which cross-sequence matching was performed in order to consolidate samples corresponding to the same individual into a unified identity group. After the consolidation procedure, all pedestrian identity folders were aggregated into a single dataset repository and assigned unique numerical identifiers. Representative samples corresponding to the same pedestrian identity are illustrated in Figure 3.29.

Figure 3.29. Samples from the dataset of the same object instance

The resulting dataset is suitable for training data-driven appearance embedding models and similarity learning frameworks, and can be directly applied to pedestrian re-identification and data association tasks in multi-object tracking pipelines.

The main characteristics and statistical properties of the dataset are summarized in Table 3.13. It is worth noting that some cropped samples exhibit lower spatial resolution compared to datasets acquired with higher-resolution thermal sensors [226]. This effect is primarily caused by the large distance between pedestrians and the vehicle-mounted thermal camera at the time of acquisition.

The dataset has been made publicly available to facilitate reproducibility and further research, and can be accessed at [5]

Table 3.13: The characteristics of the dataset

Attribute	***Value***
Video Sequences Used	160
Total Extracted Image Samples	26153
Pedestrian Instances	207
Camera Position	Vehicle Mounted
Pixel Resolution	8bpp
Original Image Resolution	640x480

3.3.6.2 Semantic Segmentation Dataset used for MOTSV

Semantic segmentation represents a fundamental component of perception systems for autonomous driving, as it enables a structured understanding of the surrounding environment and supports higher-level tasks such as object detection and multi-object tracking. In particular, semantic information extracted from thermal imagery can significantly enhance robustness under adverse illumination and weather conditions.

The thermal semantic segmentation dataset introduced in this work was acquired using three different long-wave infrared camera models: FLIR PathFindIR, FLIR Tau 2, and FLIR Vue Pro. The dataset consists of approximately 3,500 real-world thermal images with a native resolution of 640×480 pixels. All images were manually annotated over multiple years, following a rigorous quality control procedure to ensure labelling accuracy and consistency.

[5] https://users.utcluj.ro/~mmp/DatasetPaper/

Data acquisition was conducted across all seasons and under a wide range of environmental conditions relevant to driving scenarios, including daytime and nighttime operation, clear weather, rain, fog, as well as cold and warm temperature regimes. Furthermore, the dataset was collected in three different urban environments, capturing diverse scene layouts, viewpoints, traffic densities, and partial occlusions. All images correspond exclusively to outdoor driving scenarios. Each annotation was carefully verified and refined whenever inconsistencies or labelling artifacts were identified.

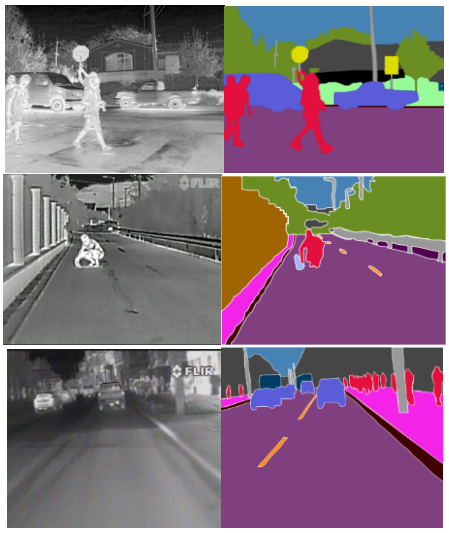

Figure 3.30. Sample images from the proposed dataset

The dataset includes pixel-level annotations for the following semantic classes, together with their corresponding RGB encoding scheme: road (128, 64, 128), sidewalk (244, 35, 232), building (70, 70, 70), live fence (190, 153, 153), fence (160, 100, 0), pole (153, 153, 153), traffic light (250, 170, 30), traffic sign (220, 220, 0), vegetation (107, 142, 35), sky (70, 130, 180), person (227, 15, 60), cyclist (255, 0, 0), animal (160, 180, 75), car (90, 92, 216), truck (0, 0, 70), bus (0, 60, 100), motorcycle (0, 0, 230), bicycle (119, 11, 32), lane markings (255, 140, 0), curb (64, 0, 0), other object (0, 74, 111), unknown (11, 11, 11), wall (0, 0, 0), parking space (250, 170, 160), and terrain/field (152, 251, 152). In addition to the class definitions listed above, the annotation process followed a set of consistent labeling guidelines to ensure inter-annotator consistency across the dataset. This procedure contributed to minimizing class ambiguities and maintaining homogeneous annotation quality across acquisition sessions and annotators.

Representative samples from the thermal semantic segmentation dataset are illustrated in Figure 3.30.

It should be noted that, out of the total 3,500 images included in the proposed dataset, 200 samples were additionally annotated from the FLIR ADAS thermal dataset [214], as these sequences contained scene configurations and environmental conditions that were not sufficiently represented in the remaining data. The main characteristics of the resulting dataset are summarized in Table 3.14.

Table 3.14. Attributes of the proposed semantic segmentation dataset

Attribute	Value
Video Sequences Used	200
Total Annotated Images	3500
Number of Semantic Classes	25
Camera Position	Vehicle Mounted
Pixel Resolution	8 bpp
Image Resolution	640 X 480
Time of the day	Day/Night
Seasons Covered	4 (Spring, Summer, Autumn, Winter)

Table 3.15 summarizes a comparative evaluation of the previously described thermal semantic segmentation dataset in relation to existing publicly available datasets reported in the literature. All considered datasets provide annotated data acquired in automotive driving scenarios, enabling a consistent and fair comparison.

Table 3.15 Comparative Evaluation of Semantic Segmentation Datasets

Dataset	Nr of Images	Semantic Classes	Nr of cameras used	Image Resolution	Seasons
Proposed	**3500**	**25**	**3**	**640x480**	**4**
SODA [208]	2168	20	1	640x480	Not Specified
CrosIR [209]	376	1	1	640X480	1
MFNet [213]	1569	8	1	640X480	Not Specified
SCUT-Seg[212]	2010	10	1	576 X 720	2

Chapter 4. Sensor Fusion

4.1 Introduction

4.1.1 Generalities

Perception of dynamic environments constitutes a fundamental and highly demanding component of autonomous driving and advanced driver assistance systems. The reliable detection and interpretation of all traffic participants, with particular emphasis on vulnerable road users, represent critical prerequisites for the effective operation of core subsystems such as collision avoidance, localization, and trajectory planning. The inherent complexity of real-world environments, combined with the diverse and often unpredictable behavior of road participants, significantly increases the difficulty of designing robust and dependable perception frameworks. Autonomous systems are therefore required to operate under a wide range of challenging scenarios, including densely populated urban environments characterized by the simultaneous presence of multiple static and dynamic entities, such as vehicles, pedestrians, cyclists, and other objects.

To address these challenges, modern perception architectures rely on heterogeneous sensing technologies and measurement modalities, which provide complementary and redundant representations of the surrounding environment. In the literature, the term *modality* is commonly employed to denote the distinct acquisition frameworks associated with different sensor types. However, data acquired from a single sensing modality is generally insufficient to provide a complete and reliable representation of the environment, particularly in the presence of adverse weather conditions, partial occlusions, or sensor-specific limitations.

Sensor fusion aims to overcome these limitations by integrating incomplete and imperfect information obtained from multiple complementary sensors in order to derive a more comprehensive and consistent interpretation of the scene. While the use of multiple sensing sources improves robustness and reliability, it also introduces a series of additional challenges that extend beyond the processing of individual sensor streams. These challenges include, but are not limited to, cross-sensor data association, selection of appropriate fusion architectures, management of false alarms and missed detections, accurate spatial and

temporal calibration, and the efficient combination of heterogeneous information under varying environmental conditions.

Although the concept of sensor fusion has been extensively studied, recent advances in sensing technologies, computational hardware, and data processing methodologies have enabled real-time multi-sensor fusion to become a practical and scalable solution for autonomous systems. As discussed in the previous chapter, sensor fusion can also be formulated as a state estimation and filtering problem, in which measurements from multiple sensors are jointly exploited to estimate the system state over time. Consequently, several principles and algorithmic strategies commonly employed in multi-object tracking are directly applicable to sensor fusion frameworks, enabling consistent and temporally coherent perception of dynamic environments.

The general pipeline of a sensor fusion system contains the following steps:

- Data Acquisition
- Registration (spatial and temporal alignment)
- Gating
- Filtering (data association and tracking)
- Track management (logical analysis of results)
- Temporal and spatial integration

The fusion of data originating from multiple sensing sources offers several significant advantages when compared to the use of a single sensing modality. First, when multiple homogeneous sensors are deployed (for instance, multiple radar units), the optimal combination of their observations enables a more accurate and reliable estimation of object attributes, such as position and velocity. Although similar improvements can be achieved by integrating measurements from a single sensor over time, the use of multiple redundant sensors accelerates the convergence toward accurate estimates and additionally provides fault tolerance in the event of sensor degradation or failure.

A second major advantage arises from the fusion of heterogeneous sensors that measure the same physical quantities using different sensing principles. For example, three-dimensional object position can be estimated using both radar and LiDAR sensors; however, by jointly processing the measurements from these complementary modalities, a more precise and robust position estimate can be obtained. Such redundancy is particularly beneficial under adverse environmental

conditions, where the performance of individual sensors may degrade differently.

The third advantage relates to enhanced observability achieved through the combination of complementary sensing capabilities. By extending the set of observable properties beyond those provided by individual sensors, multi-sensor fusion enables a more comprehensive representation of the environment. A representative example is the joint use of radar and forward-looking infrared (FLIR) imaging sensors for aerial target monitoring [249]. While the imaging sensor provides accurate angular information, the radar system supplies reliable range measurements. The integration of these complementary observations yields a more precise and stable localization of the target than would be possible using either sensor independently.

Furthermore, the effective fusion of complementary sensor data facilitates improved object identification and tracking performance. By incorporating sensor-specific features into data association and tracking frameworks, the system can achieve more robust identity preservation and reduce ambiguities in complex multi-object scenarios.

4.1.2 Common Sensors used in Autonomous Systems and Their Properties

In sensor fusion applications, multiple complementary sensing modalities are employed to perceive the surrounding environment. The use of heterogeneous sensors provides redundancy, which increases system reliability by offering fallback capabilities in the event of sensor malfunction or degradation. At the same time, sensor complementarity enables robust operation across diverse environmental conditions and traffic scenarios. Moreover, the availability of redundant measurements originating from different sensing principles contributes to reducing uncertainty in estimated quantities, such as object position, velocity, or orientation.

Among the most commonly deployed sensors in autonomous and advanced driver assistance systems are cameras, LiDARs, RADARs, ultrasonic sensors, Global Positioning System (GPS) receivers, and Inertial Measurement Units (IMUs). Each of these sensing technologies provides distinct types of information and exhibits specific operational characteristics, advantages, and limitations.

This section provides a concise overview of the primary sensor modalities used in self-driving vehicles, together with a discussion of their respective strengths and weaknesses. Throughout this chapter, the term modality is used to denote the data acquisition framework associated with a particular sensing technology. A comparative analysis of the advantages and disadvantages of commonly employed automotive sensors is presented in [250].

Cameras

Autonomous systems rely extensively on camera-based sensing modalities for visual perception of the surrounding environment. Vision-based approaches have attracted considerable attention from the research community due to the rich semantic and structural information that camera sensors can provide. In addition to object appearance and class information, depth cues can also be extracted from camera data using dedicated reconstruction or learning-based techniques. Among the most commonly employed camera types in autonomous driving applications are RGB cameras [251], thermal cameras [252], and event-based cameras [253].

One of the primary advantages of camera sensors lies in their high spatial resolution and detailed visual representation of the scene. However, certain camera modalities, such as conventional RGB cameras, exhibit limited robustness under adverse environmental conditions, including low illumination, fog, rain, or glare. Furthermore, although monocular vision-based depth estimation approaches offer a cost-effective alternative to hardware-based depth sensors, they often fail to accurately recover geometric properties of objects. In particular, monocular reconstruction methods may produce unreliable distance estimates in non-planar road scenarios or may struggle to generalize to object categories and scene configurations that were not sufficiently represented in the training data [254].

Multi-camera configurations, such as stereo vision systems, provide improved depth estimation accuracy compared to monocular approaches by exploiting geometric triangulation. Nevertheless, these systems introduce additional complexity in terms of calibration requirements and synchronization. Moreover, stereo-based reconstruction may degrade in challenging visual conditions, including low-texture regions, strong illumination variations, solar glare, perspective distortions, or insufficient feature correspondence [12].

Camera-based perception systems are also susceptible to noise introduced by environmental factors or internal sensor processes, such as electronic gain amplification applied during automatic exposure adjustments. Finally, the effective field of view of camera sensors can be adapted through appropriate lens selection, allowing system designers to tailor visual coverage according to the specific requirements of the target application.

Thermal Cameras

Thermal cameras [252] are non-contact sensing devices capable of capturing the thermal radiation emitted by objects and converting the measured infrared energy into image representations. From a structural perspective, thermal imaging systems share common architectural components with visible-spectrum cameras, including an optical lens, a dedicated sensor array (infrared detector), signal processing electronics, and a protective housing designed to ensure reliable operation under outdoor environmental conditions.

In the context of autonomous systems, thermal cameras are particularly attractive due to their robustness under adverse weather and illumination conditions. Unlike conventional RGB cameras, thermal sensors are able to operate independently of external light sources and remain functional in scenarios involving low visibility, such as night-time driving, fog, rain, or snowfall. Furthermore, thermal cameras are less susceptible to saturation effects caused by strong light sources, such as headlights from oncoming vehicles. Another important advantage is their ability to reliably detect living beings, including pedestrians and animals, at relatively large distances, thereby increasing the available reaction time for both human drivers and automated decision-making systems.

Despite these advantages, thermal imaging systems also exhibit several inherent limitations. One major drawback is the absence of color information, which significantly reduces the amount of discriminative visual cues available for object classification and scene interpretation. In addition, thermal cameras typically provide lower spatial resolution compared to RGB sensors. This limitation arises from the longer wavelength of infrared radiation, which requires physically larger detector elements to capture sufficient energy, thereby constraining the achievable pixel density for sensors of comparable physical size.

The combined effects of reduced spatial resolution and the lack of chromatic information pose additional challenges for downstream perception tasks. From an algorithmic standpoint, designing robust solutions for object detection, data association, classification, and multi-object tracking becomes considerably more complex when relying exclusively on thermal imagery, compared to visible-spectrum camera data.

LiDAR

Light Detection and Ranging (LiDAR) systems [255] represent active three-dimensional sensing technologies widely used for acquiring accurate spatial information about the surrounding environment. LiDAR sensors operate by emitting laser pulses and measuring the time-of-flight required for the reflected signals to return to the receiver. Based on this temporal delay and the known speed of light, the distance between the sensor and the observed surfaces is estimated. By continuously scanning the scene and aggregating successive measurements, LiDAR systems generate dense point clouds, in which each point encodes the three-dimensional coordinates of the corresponding surface reflection.

Most automotive LiDAR sensors operate in the near-infrared spectrum, typically around a wavelength of 900 nm. Alternative wavelength configurations have also been explored in order to improve performance under adverse atmospheric conditions such as rain or fog. Scene scanning is commonly achieved through mechanical or solid-state beam steering mechanisms, with rotating mirrors being among the most frequently employed solutions in commercially available systems.

A wide range of LiDAR configurations exists on the market, each optimized for specific application requirements and operational constraints. Low-channel-count sensors, such as four-layer (4L) LiDAR systems, are often used in applications that prioritize long-range measurements. However, these sensors provide relatively sparse point clouds and typically exhibit a limited horizontal field of view, as exemplified by devices such as the SICK LD-MRS sensor, which offers an approximate horizontal coverage of 85 degrees [256]. In contrast, higher-resolution LiDAR systems, including 16-, 32-, and 64-layer configurations, are capable of generating significantly denser point clouds and frequently provide full 360-degree horizontal coverage [257]. The primary limitation of such high-density sensors is the

reduced effective detection range, which may be constrained by hardware and power considerations.
Compared to RADAR sensors, LiDAR systems offer superior spatial resolution, owing to the narrower beam divergence, the increased number of vertical scanning layers, and the high sampling density along each scan line. This characteristic makes LiDAR particularly well suited for precise geometric reconstruction, obstacle detection, and three-dimensional scene understanding.

Despite these advantages, LiDAR technology also presents several limitations. Performance degradation may occur under adverse weather conditions, such as heavy rain, fog, or snow, where atmospheric particles can scatter or attenuate the emitted laser signals. Furthermore, unlike RADAR systems, conventional LiDAR sensors do not directly provide velocity measurements, requiring additional processing or sensor fusion strategies to estimate object motion. Operational reliability is also influenced by environmental factors, as the optical components must be kept clean to maintain measurement accuracy. On the other hand, LiDAR systems tend to perform favorably during night-time operation, as they are unaffected by ambient illumination and solar interference.

RADAR

Radio Detection and Ranging (RADAR) systems [258] represent a core sensing modality in automotive perception, being widely employed in applications such as collision avoidance, adaptive cruise control, blind-spot monitoring, and object tracking. These sensors operate by emitting radio-frequency electromagnetic waves and analyzing the signals reflected by surrounding objects, thereby enabling the estimation of target properties including range, relative velocity, and angular position.

Unlike vision- or LiDAR-based systems, RADAR directly measures target velocity through the Doppler effect. This phenomenon arises from frequency shifts in the reflected signal caused by relative motion between the sensor and the observed object. The availability of direct velocity measurements constitutes a significant advantage for state estimation and sensor fusion frameworks, as it accelerates filter convergence and improves motion-state observability.
RADAR technology offers several advantages in automotive environments. Due to the propagation characteristics of radio waves, RADAR sensors are capable of detecting objects beyond direct line-of-

sight, including partially occluded targets and reflective structures located behind other vehicles. Furthermore, automotive RADAR systems typically provide a wide horizontal field of view, often exceeding 150 degrees, and long operational ranges surpassing 200 meters. Their robustness under adverse weather conditions, such as rain, fog, and snowfall, makes RADAR sensors particularly valuable for safety-critical applications. In addition, their compact form factor facilitates integration into vehicle body panels, and their operation is not significantly affected by surface contamination, unlike optical sensors.

Despite these advantages, RADAR sensors also exhibit notable limitations. Their ability to detect objects composed of low-reflectivity materials, such as wood or porous plastics, is reduced. Moreover, internal filtering mechanisms designed to suppress clutter and false detections may inadvertently remove relevant static objects, which can pose risks in autonomous driving scenarios. The relatively limited vertical angular resolution of RADAR systems may also result in ambiguous detections of small or low-profile structures, such as road debris or manhole covers.

Depending on the intended application and sensing range, automotive RADAR systems are commonly categorized into short-range and long-range configurations, each optimized for specific operational requirements.

Sonar

Sound Navigation and Ranging (SONAR) sensors constitute another sensing modality commonly employed in automotive systems, particularly for short-range object detection and proximity-based navigation tasks. Depending on their operating principle, SONAR devices are generally categorized into active and passive sensors. Active SONAR systems emit ultrasonic pulses and analyze the reflected signals returned from nearby objects or surfaces, whereas passive configurations rely solely on received acoustic signals originating from external sources.

Similarly to LiDAR-based ranging principles, distance estimation in SONAR systems is performed by measuring the time-of-flight of the emitted ultrasonic wave. However, unlike optical sensors, SONAR relies on acoustic wave propagation and therefore uses the speed of sound rather than the speed of light. The ultrasonic frequencies employed in

automotive applications typically exceed 20 kHz, placing them beyond the range of human auditory perception. These acoustic waves are capable of propagating through various media and may penetrate certain materials that are opaque to visible light.
When interacting with moving targets, ultrasonic waves may also exhibit Doppler frequency shifts, enabling limited motion-related measurements. Nevertheless, the comparatively low propagation speed of sound imposes inherent latency constraints on SONAR systems, making them less suitable for long-range or high-speed sensing tasks. In addition, sensor performance is sensitive to environmental conditions, as contaminants such as dust, ice, or moisture on the transducer surface can significantly degrade signal quality.

Signal attenuation further limits the effective operational range of ultrasonic sensors. As acoustic waves propagate, their intensity decreases with distance, resulting in weaker reflected signals and reduced detection reliability. Consequently, SONAR sensors are primarily deployed in short-range applications, such as parking assistance and low-speed maneuvering. Surface characteristics of target objects also influence reflection behavior. Highly smooth or wet surfaces may cause specular reflections, redirecting acoustic energy away from the receiver and leading to weak or ambiguous echo responses that can be difficult to distinguish from background noise [259].

GPS

The Global Positioning System (GPS) [260], also known as NAVSTAR (Navigation System with Timing and Ranging), was initially developed in 1973 through a joint military initiative with the objective of enabling precise navigation for maritime and defense applications. At present, the GPS constellation comprises approximately 31 operational satellites deployed in medium Earth orbit, with orbital planes inclined at approximately 55 degrees relative to the equatorial plane.

GPS is part of the broader class of Global Navigation Satellite Systems (GNSS), which also includes GLONASS (Russia), BeiDou (China), and Galileo (Europe). These satellite constellations are configured to ensure that, at any given time and geographical location, a minimum of four satellites are visible to the receiver. Three satellites are required to determine the three-dimensional position of the receiver, while the fourth enables precise synchronization of the receiver clock through signal time-of-flight estimation.

Each satellite periodically transmits navigation messages at a data rate of approximately 50 bits per second. These messages consist of three main components. The first component contains satellite health status and time information. The second component, referred to as the ephemeris, provides high-precision orbital parameters for the transmitting satellite. The third component, known as the almanac, contains coarse orbital information for the entire satellite constellation and is transmitted in segmented form. A complete almanac download requires approximately 12.3 minutes.

Based on the availability of navigation data, GPS receivers operate under different initialization modes. During a cold start, the receiver must acquire the full almanac and ephemeris information, resulting in acquisition times that may exceed 10–15 minutes. A warm start occurs when almanac data are already available and only ephemeris updates are required, leading to significantly reduced acquisition times. In the case of a hot start, both ephemeris and almanac information are up to date, allowing position estimates to be obtained within a few seconds. Assisted GPS (A-GPS) further reduces initialization time by retrieving navigation data through cellular networks. Position information is commonly transmitted using the NMEA 0183 communication protocol.
Positioning accuracy can be enhanced through the use of augmentation systems. The Wide Area Augmentation System (WAAS) employs a network of ground reference stations that estimate correction parameters, which are subsequently transmitted via geostationary satellites to end users. Differential GPS (DGPS), in contrast, directly broadcasts correction data from nearby reference stations to local receivers using terrestrial communication channels such as radio links or cellular networks. Both approaches significantly improve positioning accuracy by compensating for atmospheric delays, clock errors, and orbital uncertainties.

In automotive applications, GPS serves as a fundamental component for global localization, navigation, and mapping, and is frequently employed for generating ground truth reference trajectories in perception and tracking experiments. However, GPS performance is inherently limited in environments with restricted satellite visibility, such as tunnels, urban canyons, or dense foliage. In such scenarios, inertial measurement units (IMUs) and dead-reckoning techniques are commonly integrated to maintain continuous vehicle localization.

IMU

Global positioning systems provide absolute localization information in the form of latitude, longitude, and altitude estimates. However, in high-dynamic scenarios, such as those encountered by autonomous vehicles operating at elevated speeds or requiring high-rate state updates, GPS measurements alone are insufficient to meet both accuracy and temporal resolution requirements. Furthermore, GPS performance degrades significantly in environments with limited satellite visibility, such as urban canyons, tunnels, underground parking structures, or areas with dense surrounding infrastructure. In such conditions, signal blockage, multipath effects, and reduced satellite geometry can lead to severe positioning errors or complete signal loss.

To address these limitations, inertial sensing is commonly integrated through the use of inertial measurement units (IMUs) [261], which enable short-term navigation continuity even in the absence of reliable satellite signals. An IMU is an electronic sensing module that typically integrates multiple motion sensors, including accelerometers, gyroscopes, and, in some configurations, magnetometers. These sensors measure linear acceleration, angular velocity, and magnetic field orientation relative to an inertial reference frame. For reliable estimation of three-dimensional motion and orientation, each sensing modality is generally implemented with at least three degrees of freedom.

Compared to GPS, IMUs provide measurements at substantially higher sampling rates, enabling rapid state propagation and short-term motion estimation. This property makes inertial sensing particularly valuable for bridging localization gaps during GPS outages and for supporting smooth vehicle motion estimation in challenging environments.

Despite their high temporal resolution, IMU measurements are affected by sensor noise, bias drift, and scale factor errors. Consequently, standalone inertial navigation leads to accumulating position and orientation errors over time. To mitigate these effects, IMU biases and error parameters are typically incorporated into the system state vector within sensor fusion frameworks and are continuously corrected through probabilistic filtering techniques [262].

A comprehensive understanding of the surrounding environment is achieved by fusing complementary and redundant information from multiple sensing modalities. As highlighted in the preceding sensor

overview, each individual sensor exhibits inherent limitations and failure modes. By combining heterogeneous data sources, autonomous systems can construct a more robust representation of the environment, thereby improving perception accuracy, localization stability, and overall operational safety.

4.1.3 The JDL Model

The advancement of modern sensor fusion methodologies has been strongly influenced by the rapid progress in high-performance computing platforms, the availability of high-frequency processors, and large-scale defense research initiatives. A significant historical milestone in this context was the Strategic Defense Initiative (SDI), announced on March 23, 1983 by the United States administration, which stimulated extensive research activities in multisensor information processing and real-time data integration.

As the field of data fusion matured, the Joint Directors of Laboratories (JDL) Data Fusion Working Group was established with the objective of facilitating collaboration among defense researchers and system developers, as well as standardizing terminology and conceptual frameworks related to multisensor data fusion [263]. One of the key outcomes of this initiative was the development of the JDL process model, which provides a structured representation of the functional components, processing stages, and information flows involved in data fusion systems.

The JDL process model has since been adopted across a wide range of

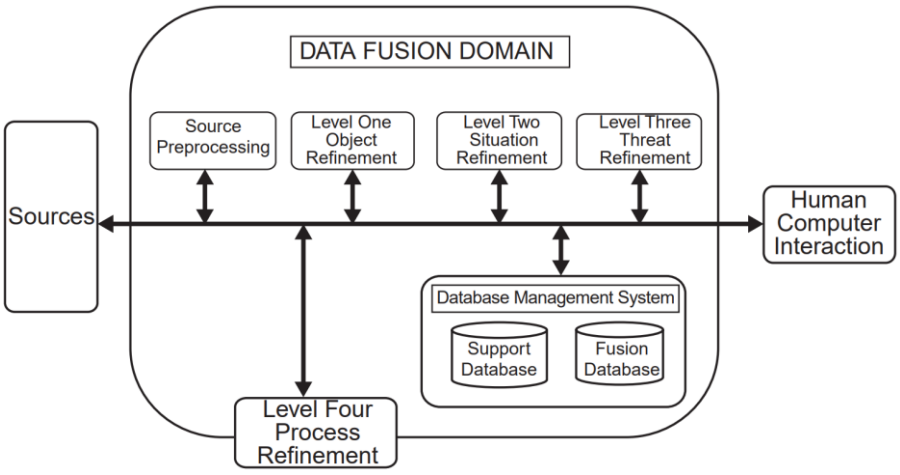

Figure 4.1. The main components of the JDL process model [263]

application domains and has undergone multiple revisions to accommodate evolving technological requirements and application scenarios. The version illustrated in Figure 4.1 corresponds to the updated formulation reported in [263].

The main processing levels defined within the JDL data fusion process model can be summarized as follows:

- **Level 0 (Signal and Data Pre-processing):** This level addresses the initial processing of raw sensor data, independent of sensor modality. It typically includes operations such as noise reduction, signal conditioning, feature extraction, and data normalization, which prepare the measurements for subsequent fusion stages.
- **Level 1 (Object Refinement):** This level focuses on the integration of multisensor information in order to estimate and refine the properties of individual objects. Typical outputs include object state attributes such as position, velocity, identity, size, and other measurable or inferred characteristics derived from sensor observations.
- **Level 2 (Situation Assessment):** At this stage, relationships among objects are analyzed within the context of the surrounding environment. This includes understanding spatial configurations, interactions among entities, and the overall situational context in which the objects operate.
- **Level 3 (Impact Assessment and Threat Projection):** This level is responsible for predicting the future states of objects and assessing potential risks or hazardous situations. It supports decision-making by evaluating whether planned actions can be executed safely and by identifying scenarios that require preventive or corrective measures.
- **Level 4 (Process Refinement and Resource Management):** The highest level supervises and optimizes the overall fusion process. It addresses adaptive resource allocation, performance monitoring, and system-level optimization in order to improve computational efficiency, robustness, and real-time performance.

The database component provides persistent storage and retrieval mechanisms for auxiliary information that supports the fusion pipeline.

Typical examples include access to digital maps, prior environmental knowledge, or historical data, which can be leveraged by downstream modules such as localization and path planning. The human interaction component is commonly implemented through a human–machine interface (HMI), such as a display or touch panel, which enables the visualization of relevant system states and informative feedback for the end user.

With the rapid development of autonomous driving technologies, the authors of [264] proposed an adaptation of the JDL data fusion framework tailored to automotive applications. In this revised formulation, each processing level of the JDL model is mapped onto a corresponding functional layer of the autonomous vehicle architecture. Figure 4.2, adapted from [264], illustrates the conceptual correspondence between the JDL fusion levels and the principal perception, decision-making, and control layers of a self-driving system.

The Perception Layer comprises all processing modules responsible for constructing a virtual representation of the physical environment based on sensor data. The Decision and Application Layer includes the decision-making and trajectory planning components required to guide the autonomous system from an initial state toward a target goal. Finally, the Action and HMI Layer encompasses the control mechanisms that interface with the vehicle actuators, such as steering, braking, and throttle control, as well as the human–machine interaction components. The JDL framework does not prescribe a specific implementation strategy for data fusion, nor does it define explicit mechanisms for information exchange between processing layers. Consequently, multiple system architectures have been proposed in the literature,

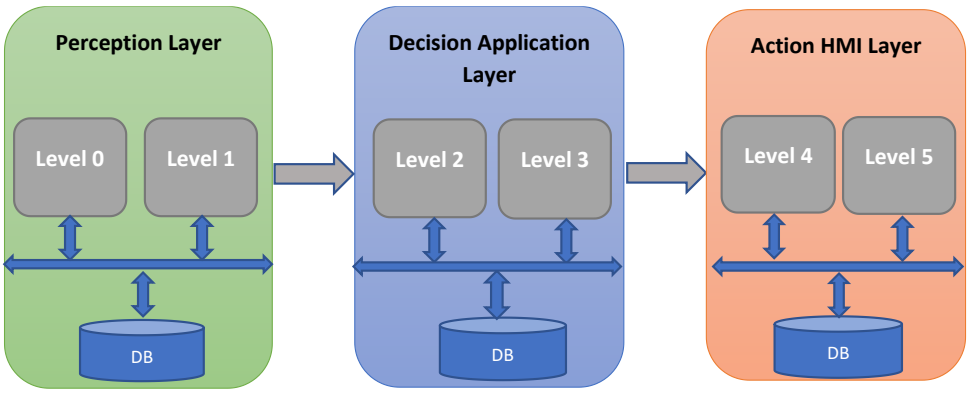

Figure 4.2. Adaptation of JDL model for automotive applications [264]

each tailored to the available computational resources, real-time constraints, and the chosen fusion strategy across the perception and decision-making pipeline.

4.1.4 Challenges of Sensor Fusion for Autonomous Systems

Sensor fusion integrates information acquired from multiple sensing modalities with the objective of obtaining more accurate and reliable estimates of the parameters contained in the system state vector. As previously discussed, sensor fusion and multi-object tracking share several fundamental processing stages, including measurement validation (gating), data association, and state estimation through filtering techniques. Consequently, many of the challenges encountered in tracking pipelines are also inherent to the design and implementation of sensor fusion systems. In addition to the aspects already addressed in Chapter 2, this section introduces a set of further challenges specific to multi-sensor fusion architectures. Several of these issues have been systematically analyzed in the works of Hall and Steinberg [265] and in the JDL data fusion framework [263].

- No fusion strategy can compensate for the absence of reliable measurements of the physical quantities of interest. Sensor fusion is meaningful only when the relevant properties can be directly measured or robustly inferred from the available sensing modalities; otherwise, the fusion process becomes ineffective.
- Advanced fusion algorithms and pattern recognition techniques cannot substitute inadequate preprocessing or poor-quality sensor data. Once erroneous or degraded information enters the fusion pipeline, subsequent fusion stages cannot fully mitigate these deficiencies. For instance, in high-level fusion architectures, object detections generated by individual sensors must exhibit high accuracy in order to enable reliable fusion outcomes. Consequently, careful design and validation of each stage in the perception pipeline are essential.
- Accurate sensor modeling constitutes a fundamental prerequisite for successful fusion. Sensor parameters must be derived from validated technical specifications and empirical calibration data. The use of ad hoc assumptions or overly optimistic noise covariance estimates may lead to biased state estimates and can ultimately degrade fusion performance below that of individual sensors.

- Despite substantial advances in sensor fusion research and the availability of numerous sophisticated algorithms, no universal solution exists that can address all operational scenarios. Fusion methods must be adapted to the specific application context and sensor configuration. In practice, many real-world scenarios violate common fusion assumptions, such as statistical independence between measurements or the availability of reliable prior distributions, which further complicates algorithm selection and deployment.
- Data-driven fusion methods based on neural networks are constrained by the availability and diversity of training data. Since exhaustive coverage of real-world scenarios is impractical, purely learning-based approaches may suffer from limited generalization. Hybrid fusion strategies that integrate model-based and data-driven components typically offer enhanced robustness.
- Quantifying the overall effectiveness of a fusion system at the mission or system level remains a challenging open problem [347].
- Sensor fusion operates as a dynamic and iterative process that continuously refines state estimates. Therefore, computational efficiency is critical to ensure timely integration of incoming measurements. Excessive processing latency can result in outdated estimates and reduced situational awareness.
- Even with state-of-the-art registration and calibration techniques, achieving perfect spatial and temporal alignment across heterogeneous sensors remains difficult, particularly in dynamic environments.
- Multi-sensor systems typically involve significantly higher deployment and maintenance costs compared to single-sensor configurations, which must be considered in practical system design.
- Ideally, measurement errors originating from different sensors should exhibit distinct statistical characteristics and minimal correlation, as correlated error sources can reduce the effectiveness of fusion and lead to biased estimates.

Additional limitations and challenges associated with sensor data fusion, categorized according to the JDL processing levels at which they may arise, are summarized in Chapter 21 (“Multisensor Data Fusion”) of [266], specifically in Table 21.2.

4.2 Review of Sensor Fusion Methods

4.2.1 Fusion Architectures

Depending on the available computational platforms, three distinct architectures can be implemented: centralized, distributed, and hybrid. Each of these fusion architectures exhibits specific advantages and disadvantages.

Centralized architectures have a relatively simple structure, do not require the use of distributed computing techniques, and achieve the highest performance when all sensors measure the same physical properties. In this type of architecture, raw measurements are collected within a central processing block. At the moment when the fusion algorithm is initiated, it has access to the complete set of data originating from all available sensors. The main disadvantage of this fusion solution lies in the increased complexity of the fusion algorithm, compared to the algorithms employed in other types of architectures (for example, hybrid architectures), since it must interpret and process large volumes of data at very high sampling frequencies. Figure 4.3, which was taken from [264] and slightly reconditioned in order to improve visual quality, illustrates the main computational blocks of the centralized fusion architecture.

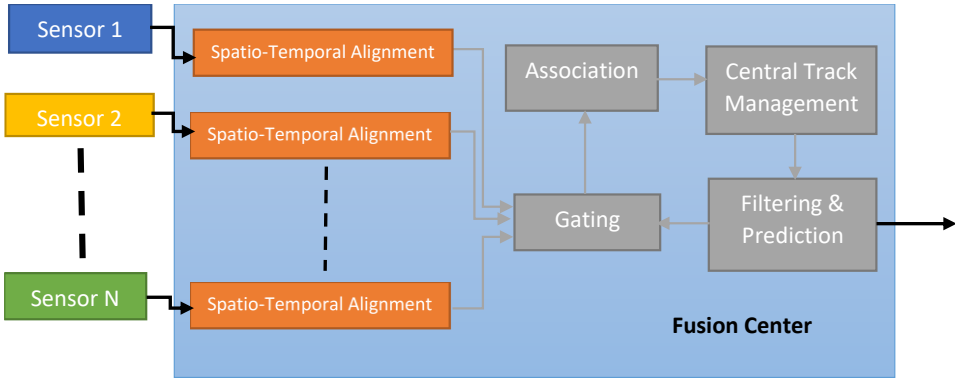

Figure 4.3. Architecture of a system with centralized fusion [264]

Figure 4.3 presents the general architecture of a centralized fusion system. It is worth emphasizing the similarities between the fusion system and the pipeline structure of a multi-object tracking application. The sensors are denoted as Sensor 1, Sensor 2, and so forth, while the spatio-temporal alignment refers to the processing steps required to bring all information into a common reference frame by applying

various motion correction techniques. The terms gating, association and filtering, as well as prediction, have the same meaning as in the case of object tracking applications. The central track management module provides the appropriate measurement covariance matrices to the subsequent functional block or is responsible for selecting the required type of prediction or update. For instance, some sensors may be characterized by non-linear measurement models, whereas others may rely on linear measurement models; in this context, the central track management is responsible for selecting the appropriate functions according to the measurement models of the sensors.

Figure 4.4 illustrates the distributed fusion architecture, taken from [264] and reconditioned. One of the main advantages of this type of architecture is its reduced sensitivity to the correct alignment of sensor data. Data are pre-processed on each computing module on which they are deployed, and the results are subsequently aggregated within a central processing unit, thereby reducing the computational burden that characterizes centralized fusion architectures. The modular and scalable nature of this architecture represents another important advantage, as it enables the mitigation of data congestion and provides increased robustness to sensor failures compared to the previously presented architecture.

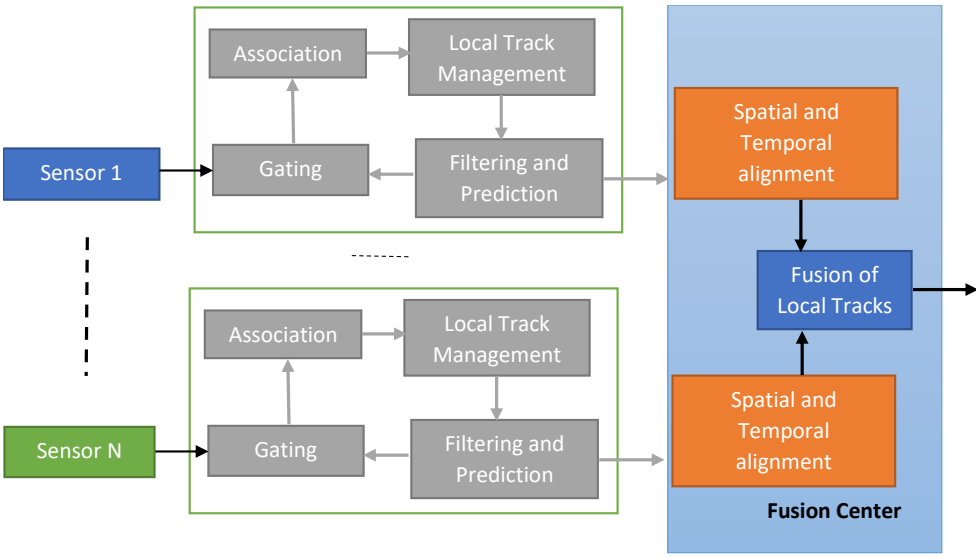

Figure 4.4. Architecture of a distributed fusion system [264]

Each sensor can be processed using different methods, and the final input to the central processor consists of the processed data, together

with all attributes resulting from the pre-processing stage. The main drawbacks of this architecture are related to the assumption that sensors operate independently from one another (an assumption that is not always valid) and to the increased costs associated with the addition of multiple computing platforms to interface with each sensor (for example, in scenarios where each sensor is connected to a different computer and the final results are aggregated on another device). Furthermore, each computing module that must be interfaced with the central processing unit may represent a potential point of failure; consequently, a larger number of modules implies a larger number of possible failure points. The meaning of the terms presented in Figure 4.4 remains identical to that described for Figure 4.3.

The hybrid fusion architecture combines centralized and distributed fusion paradigms. It generally extends centralized fusion by applying dedicated processing algorithms to each sensor. For example, temporal and spatial alignment and tracking can be performed independently for each sensor, while the fusion center carries out an additional association stage across sensors, followed by filtering based on the temporally aligned and associated data.

Each type of measurement acquired from different sensors can be regarded as a form of redundant or backup measurement: the overall system performance is improved when such measurements are available, yet the system is also capable of operating in the absence of data from certain sensors. This type of architecture preserves the advantages of centralized fusion architectures while simultaneously providing the robustness characteristic of distributed architectures. The main disadvantages of this approach are related to its increased structural and algorithmic complexity, the necessity of introducing an additional module responsible for correlating tracked objects across multiple sensors, and the higher data transfer requirements. An example of this type of architecture is presented in [264].

4.2.2 Fusion Levels

Data acquired from homogeneous sensors can be fused directly at the measurement level. In contrast, when heterogeneous sensing modalities are employed, direct fusion is generally not feasible, and higher-level integration strategies, such as feature-level or decision-level fusion, must be adopted.

Data-Level Fusion

Data-level fusion, ilustrated intuitively in Figure 4.5, has been developed on the basis of the fundamental assumption that raw sensor detections, communication mechanisms, and measurement pre-processing stages are reliable, and that, as a consequence, the main focus can be shifted toward the design and implementation of fusion algorithms, with the objective of obtaining output data that are more accurate and more informative than the original information supplied individually by each sensor [267]. The sensors that are employed within this type of architectural framework are regarded as being homogeneous in nature. This fusion approach is based on the use of a centralized processing architecture, and sensor arrays are generally employed in order to achieve an increased level of robustness. A variety of pre-processing techniques can be applied to the raw input data with the purpose of performing noise reduction. Research interest associated with this type of fusion is also oriented toward data association techniques, feature extraction methods, classification approaches, and data compression strategies [268].

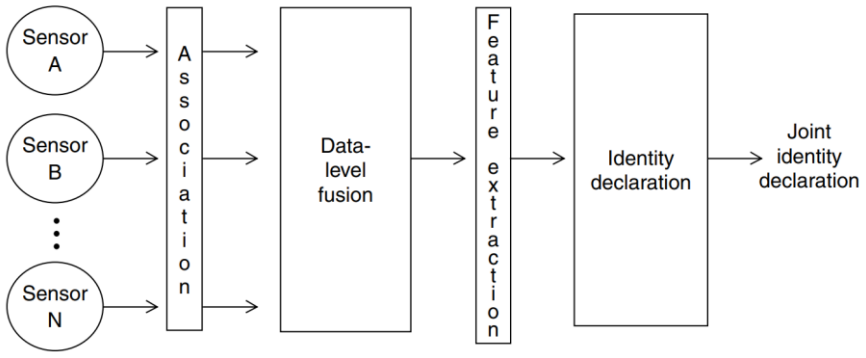

Figure 4.5. Schematic of the data-level fusion architecture [263].

Feature-Level Fusion

The features that are extracted from multiple heterogeneous sensors can be fused together in order to create a hyper-dimensional feature vector that is capable of describing, in a highly adequate and detailed manner, the objects present in the environment. Such features can be utilized in classification processes, in data association and tracking

tasks, or can be employed to support other system modules (such as the path planning module) in making more informed and more reliable decisions [269]. Feature-level fusion is also referred to as tight coupling. In Figure 4.6, the schematic representation of feature-level fusion is presented, which has been taken from reference [263].
The architectures that can be employed in conjunction with feature-level fusion include centralized fusion architectures and distributed architectures (in which each node of the distributed architecture is responsible for extracting information from a single sensor). In the case of centralized fusion, the fusion node receives raw sensor information, performs the extraction of relevant features, and subsequently carries out the fusion process. In the distributed fusion scenario, the fusion node receives the extracted features from each individual sensor node and is required only to execute the fusion algorithm.

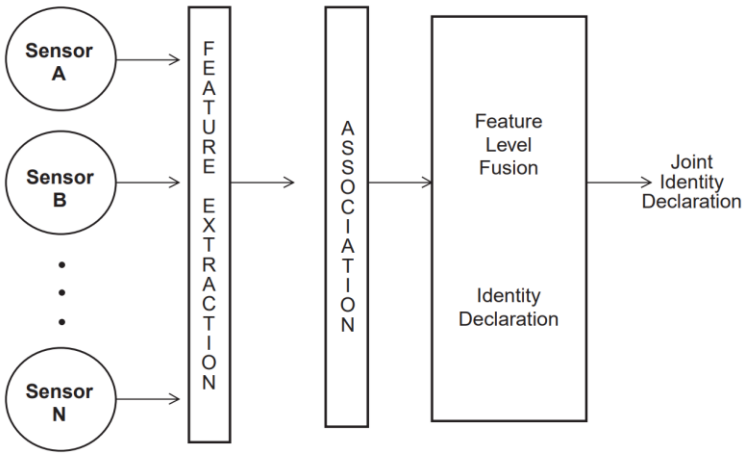

Figure 4.6. Feature level Fusion Diagram [263]

Decision-Level Fusion

Decision-level fusion generates a new unique identity as a result of the fusion process for objects originating from different sensors that have already been assigned an identity (which may correspond either to a shallow identity or to an identity that has been assigned through the tracking of objects by a specific sensor). The new identity is provided after the completion of the fusion process, and the newly

created object contains the information originating from all sensors that were taken into consideration during the fusion stage [270, 271]. This type of fusion makes use of information from sensors that can be either homogeneous or heterogeneous, and the information provided by these sensors has already been processed to a certain extent, such that higher-level decisions can be performed. One of the main advantages of this type of fusion is that sensor information originating from heterogeneous sensors can be processed by using different algorithms [272]. Other advantages include an improvement in decision accuracy and a reduction in communication bandwidth requirements (since it is no longer necessary to transmit all raw sensor data, and the size of the processed information is significantly reduced). In Figure 4.7, which has been taken from reference [263], the conceptual architecture of decision-level fusion is presented.

Due to the fact that the desired information has already been segmented from the raw sensor data at the moment of fusion (for example, in the automotive domain, objects are extracted from raw point cloud data), this type of fusion can also be regarded as a loosely coupled sensor fusion approach.

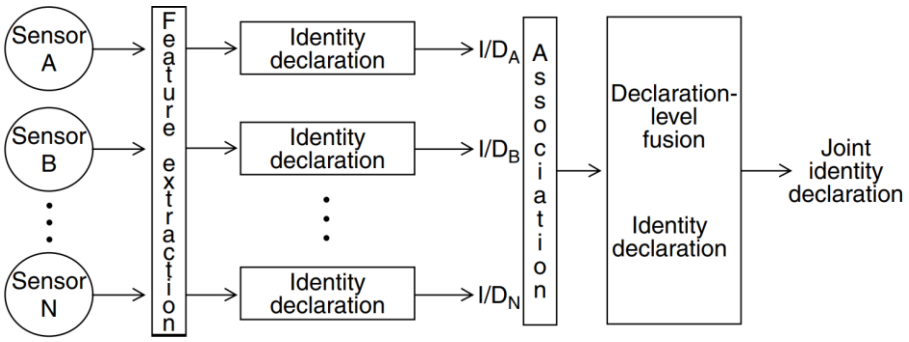

Figure 4.7. Decision-Level fusion architecture [263]

4.2.3 Data Registration

In sensor fusion, multiple sensor measurements are combined with the objective of obtaining a more robust and reliable representation of information from redundant data sources. The inputs of a fusion system may vary and can include raw sensor data, feature vectors, or high-level detections. Typically, within the fusion process, the input data are

compared by means of a dedicated algorithm, after which a specific fusion strategy is applied. In order to enable an efficient comparison of input data originating from independent physical sensors, all information must be transformed to a common time reference and a common coordinate system. This procedure is known in the scientific literature under several denominations, the most frequently used being data registration and motion correction [273]. Data registration represents a pre-processing operation and can be applied to multi-dimensional input data. Nevertheless, in most practical implementations in which registration is employed, the data are limited to two-dimensional or three-dimensional representations. Additional difficulties may arise in situations in which objects are occluded in the data acquired by certain sensors, while remaining visible in others. The registration techniques discussed in this section are applied to both image data and range data.

In the context of image registration, the primary objective is to determine the correct mapping between one image and another image. Ideally, all features extracted from one image should be mapped onto the corresponding features in the second image. However, due to the presence of sensor noise, it is uncommon that for all detected features in the observed image a corresponding feature can be found in the reference image. For this reason, a wide range of techniques have been investigated in order to determine optimal mappings even under challenging conditions [274]. Image stitching, which is used for the generation of panoramic images, relies on image registration in order to merge overlapping regions of multiple images [275]. In the medical domain [276], image registration is employed to achieve a more comprehensive characterization of patient-related conditions, with the prerequisite that the combined images originate from the same patient. Furthermore, reference [276] also discusses the challenges encountered when fusing and combining multiple medical images. Such difficulties may be caused by variations in contrast, sharpness, and resolution, as well as by differences in image dimensionality. In [277], the authors propose a rectification-based approach for transforming two images, namely a thermal image and a grayscale image, into a common stereo configuration with parallel optical axes. Features are extracted in the frequency domain and subsequently correlated between the two images. Wavelet-based features, such as Haar, Symlet,

and Daubechies wavelets, are employed in order to establish correspondences between images. A combination of wavelet-based feature extraction, normalized cross-correlation, and image-based matching techniques is used for image registration, since these approaches provide a sufficient number of matching points to enable successful image alignment [278].

The problem of point cloud registration has been extensively studied for more than three decades. For instance, in [279], the authors propose a solution for point cloud alignment under the assumption that point correspondences are known a priori. For a given set of corresponding points, the authors introduce a solution based on a least-squares transformation model, involving rotation and translation, combined with the singular value decomposition of a 3×3 matrix. Although this method provides a valid solution to the point cloud registration problem, it relies on the assumption that point correspondences are directly available after point cloud acquisition, which is generally not the case in practical scenarios. Consequently, in order to successfully register two acquired point clouds, it is first necessary to identify correspondences between the points belonging to the two three-dimensional scans. From a general methodological perspective, registration approaches can be classified into the following main categories: iterative methods, probabilistic methods, and other alternative approaches.

Iterative methods represent some of the most widely used approaches for point cloud registration and have attracted continuous research interest and development. In general, Iterative Closest Point (ICP)-based solutions consist of two main stages: correspondence estimation and transformation estimation. Besl and McKay were the first to introduce an iterative approach [280] aimed at minimizing the Euclidean distance between points belonging to two scans and at estimating the relative transformation between them. Their method is commonly referred to as the Iterative Closest Point (ICP) algorithm and remains one of the most extensively applied techniques for point cloud alignment. Since its initial introduction, numerous variants of the algorithm have been proposed in the literature, each aiming to improve computational efficiency or robustness to noise [281, 282, 283, 284]. Certain extensions of the ICP framework incorporate point-to-plane

distance metrics [285] for estimating transformation parameters. Fundamentally, the operational principle of these approaches is similar to that of point-to-point methods, with the difference that the error is minimized along the surface normal direction. More specifically, these methods minimize the orthogonal distance between points in one point cloud and the plane defined by the corresponding surface in the second point cloud. Another variation of the ICP algorithm involves the use of plane-to-plane distance metrics [286]. In this case, surface normals of a two-dimensional manifold representation of the point cloud are considered, and correspondences are established based on point-to-plane distance criteria.

An additional approach presented in [287] focuses on improving the performance of ICP-based registration by incorporating information related to the velocity of the vehicle on which the sensors are mounted. In many practical scenarios, point cloud processing algorithms assume that all points within a scan are acquired simultaneously. However, when the sensing platform is in motion, this assumption may lead to the accumulation of temporal errors as multiple point clouds are fused over time. Two of the main factors contributing to the widespread adoption of ICP-based solutions are their relative simplicity and their high computational efficiency.

Probabilistic registration approaches have emerged in response to the limitations of previously discussed methods, which do not explicitly account for the fact that two consecutive scans rarely contain exact point-to-point correspondences. Furthermore, since point clouds are often only partially overlapping, it becomes difficult to establish reliable correspondences by applying simple Euclidean distance thresholds. In [288], the authors introduce a maximum likelihood estimation framework for determining the transformation matrices that align two point clouds, referred to as Expectation Maximization Iterative Closest Point (EM-ICP). This method is capable of handling noisy input point clouds and demonstrates that, when the point distribution follows a Gaussian model, the algorithm exhibits behavior similar to point-to-point ICP approaches, with the Mahalanobis distance used as the similarity metric. Additional probabilistic solutions for point cloud registration involve the use of Gaussian Mixture Models (GMM). In such cases, the registration problem is formulated as a likelihood

maximization task. Both the transformation matrix between point clouds and the parameters of the GMM are estimated through optimization. In [289], the authors propose a deep learning-based approach for learning correspondences between point sets and GMM components, and for estimating both transformation matrices and GMM parameters through a forward inference process. A key advantage of these probabilistic methods is their robustness to noise and outliers.

Other registration strategies include feature learning-based approaches, as well as methods developed within end-to-end learning frameworks. Feature-based registration techniques employ deep learning models to extract discriminative features and to learn feature correspondences [290, 291]. The transformation matrix is subsequently estimated using a RANSAC-based approach. Features can be extracted from diverse data sources, ranging from RGB-D images [290] to handcrafted descriptors derived from point clouds and subsequently processed by deep neural networks [291]. These registration approaches present several advantages as well as notable limitations. One major advantage is that deep learning-based methods can establish reliable correspondences between input data, which can then be utilized within a RANSAC framework to achieve accurate registration. Another advantage is the relative simplicity of implementation. Furthermore, real-time processing capabilities can be achieved when using GPU acceleration. However, these methods also present disadvantages, including the requirement for large training datasets and the difficulty in maintaining high performance when processing scenes that differ significantly from those included in the training data. In addition, two separate neural networks typically need to be trained: one for feature extraction and another for feature correspondence estimation.

Another class of registration methods that has gained increasing attention consists of end-to-end learning-based approaches, which rely exclusively on neural networks without incorporating external optimization techniques for estimating transformation matrices. In these approaches, the input consists of two point clouds, while the output directly represents the transformation matrix used to align the point clouds. The registration problem is formulated as a regression task within a deep learning framework. Representative solutions of this

type can be found in [92, 93]. The approach presented in [94] combines deep learning-based feature extraction with Lukas–Kanade optimization for feature registration. The primary advantage of end-to-end learning approaches lies in the fact that neural networks jointly optimize the registration process by integrating principles from classical mathematical modeling and modern deep learning techniques. However, a significant limitation is that the registration procedure effectively becomes a black-box process that directly outputs transformation matrices. In addition, the employed distance metrics may exhibit sensitivity to noise.

In [95], the authors provide a comprehensive survey of point cloud registration techniques and classify existing approaches according to two main criteria: registration of point clouds originating from the same source and registration of point clouds obtained from different sources. Each category is further subdivided into optimization-based methods, feature-based methods, and end-to-end learning-based approaches.

For the registration of heterogeneous data sources, multiple methods have been developed and reported in the literature [296, 297, 298]. Certain approaches rely on calibration parameters and aim to identify correspondences between points belonging to multiple planar structures [296], while other methods employ neural network-based mutual information estimation in order to achieve targetless LiDAR–camera registration [298].

4.2.4 Data Fusion

This section provides a comprehensive review of state-of-the-art algorithms employed in sensor data fusion across multiple sensing modalities. Depending on the degree of visibility and control that a researcher has over the internal processing mechanisms of the sensor fusion module, a classification framework can be established by dividing existing approaches into white-box and black-box sensor fusion solutions. White-box approaches are based on classical engineering methodologies for feature extraction and on explicit mathematical models for information fusion. In contrast, black-box methods are typically founded on deep learning techniques, and the algorithms belonging to this category are further divided into weak

fusion methods and strong fusion methods. Weak fusion-based approaches employ predefined sets of rules in order to guide the processing of information from one modality by leveraging information originating from another modality. Strong fusion methods, on the other hand, perform a direct merging of information with the objective of obtaining results of higher quality. According to [299], strong fusion approaches can be further categorized based on the stage within the algorithmic processing pipeline of the different modalities at which the fusion operation is performed. The types of fusion algorithms identified under this classification include early fusion, deep fusion, asymmetry fusion, and late fusion. In addition to the fusion strategies described above, there also exist solutions that cannot be strictly assigned to a single category, as they combine multiple fusion approaches in order to obtain the final output.

4.2.4.1 Model-Based Data Fusion

Existing algorithms that have exerted a significant influence on the evolution of the sensor fusion domain and that have contributed to major advancements within the automotive industry include solutions dedicated to advanced tracking techniques [300], optimal filtering methods [301], and multi-sensor fusion frameworks [318]. Contemporary state-of-the-art approaches [302, 303] typically perform an initial detection of object instances within the sensor domain, followed by tracking and fusion of the resulting detections using Kalman filtering techniques [302] or combinations based on the Dempster–Shafer theory of evidence [304]. Alternative solutions adopt LiDAR-based methodologies for generating object hypotheses and subsequently employ camera sensors for hypothesis verification [305, 306, 307]. These methods rely on geometric feature extraction in order to identify three-dimensional region candidates within the LiDAR domain [308]. The image patch corresponding to the identified three-dimensional region, characterized using the Histogram of Oriented Gradients descriptor [309], is then evaluated by multiple classifiers.
Multi-sensor approaches that integrate LiDAR and camera data generally outperform algorithms based on single-sensor configurations; however, their performance remains limited under adverse weather conditions. In the work presented by Chen et al. [310], object detection

proposals are generated from a top-down LiDAR representation and subsequently projected into the frontal LiDAR view. All extracted features are fused, and oriented three-dimensional cuboids are generated. The proposed solution relies on a single LiDAR sensor configuration and assumes that all relevant objects can be localized from the top-down point cloud representation and that these objects are positioned on the same spatial plane.

The authors of [311] introduce a pedestrian tracking system based on the Kalman filter that fuses RADAR and camera information for indoor static application scenarios. The proposed system is specifically designed to handle lower-body occlusions. In this context, the authors train a detector exclusively for the upper part of the pedestrian body, although occlusions are not explicitly addressed within the tracking framework. In [312], a fusion mechanism integrating RADAR and LiDAR data is proposed. The authors generate a binary occlusion map of the environment by detecting occluded objects using LiDAR measurements. Based on this occlusion map, a logical decision-making mechanism is developed to determine whether individual sensors or all available sensors should be utilized. Another fusion-based solution is presented in [313], in which LiDAR, RADAR, and camera data are combined for the detection and classification of moving objects. The sensors are mounted on an intelligent vehicle platform and employ an occupancy grid representation for object identification. Within this framework, LiDAR data are used for detecting moving objects, whereas RADAR and camera sensors are primarily employed for classification purposes.

In [314], a vehicle tracking and behavior analysis algorithm is presented, capable of estimating both vehicle state and driver intentions by means of a multiple-model filtering approach. The authors utilize multiple LiDAR sensors with configurations of four and eight scanning beams, together with a camera, wheel pulse sensor, GPS RTK system, and an inertial measurement unit, in order to obtain accurate road and positioning information. An interacting multiple motion models filter is employed, considering four motion patterns defined by the authors: constant velocity lane keeping (CVLK), constant acceleration lane keeping (CALK), constant velocity lane changing (CVLC), and constant acceleration lane changing (CALC). A standard Kalman filter is applied for system state tracking, as it offers improved

computational efficiency compared to nonlinear alternatives such as the Extended Kalman Filter, Unscented Kalman Filter, or particle filtering techniques. In addition, road constraints are incorporated in order to enhance both state estimation accuracy and behavior classification performance. In [315], a scalable asynchronous multi-sensor fusion framework based on the track-to-track fusion paradigm is introduced. This approach adopts a distributed fusion architecture and is founded on the assumption that the fusion process operates asynchronously, given that sensors transmit data to the fusion center at arbitrary rates. The fusion mechanism is based on a maximum likelihood criterion for identifying correspondences between targets and sensor outputs. When compared to centralized Kalman filtering approaches or Naive Fusion strategies [316], this distributed method demonstrates a reduction in mean square error.

In [317], camera and LiDAR data are combined in order to improve pedestrian tracking performance in automated driving scenarios. To achieve this objective, a probabilistic motion model capable of capturing both lateral and longitudinal acceleration is proposed. Furthermore, the motion model is conditioned on vehicle speed. Although object detections from both camera and LiDAR sensors are obtained using convolutional neural network architectures, the tracking of camera-based and LiDAR-based objects is performed independently using particle filters. In order to enhance pedestrian re-identification performance, the two-dimensional object tracker employs features derived from the HSV and YCbCr color spaces. The tracking results obtained from two-dimensional and three-dimensional domains are not fully integrated into a single unified representation due to uncertainties associated with LiDAR–camera calibration. Finally, a track management strategy is introduced to handle track initialization, termination, and merging events.

In [318], a fusion strategy aimed at improving tracking performance through the integration of camera and RADAR data is presented. The proposed framework is based on a distributed fusion architecture and incorporates two complementary approaches: track-to-track fusion and heuristic fusion with adaptive gating. The heuristic fusion process consists of two primary stages, namely data association and data combination. During the data association stage, the objective is to

determine whether tracks originating from different sensors correspond to the same physical target. Adaptive gating is employed within this process, with gate dimensions determined experimentally. In the track-to-track fusion approach, covariance information is utilized to determine gating thresholds for data association. The authors demonstrate that track-to-track fusion methods that exploit covariance-based gating provide superior performance compared to alternative fusion strategies.

In [319], a real-time sensor fusion framework for combining millimeter-wave RADAR data with camera information is introduced. The proposed decision-level fusion system is based on the Extended Kalman Filter and employs a constant acceleration motion model. A REDBSCAN clustering algorithm [320] is applied in order to extract bounding boxes and associated velocity estimates from RADAR point cloud data. Camera-based bounding boxes are obtained from a bird's-eye view representation. The extracted bounding boxes are transmitted to the fusion center together with their corresponding measurement covariance matrices and time stamps, enabling more accurate estimation of object positions and identities.

4.2.4.2 Data Driven Data Fusion

Artificial Neural Networks (ANNs) provide an alternative paradigm for performing sensor data fusion tasks [321]. Within this framework, image features or data descriptors intended for fusion are first extracted and subsequently normalized. Through its internal architecture, an ANN is capable of approximating arbitrary nonlinear functions when trained on a representative dataset. Once the network training phase has been completed, the trained model can be employed to infer information from previously unseen data samples that were not included during training.

The studies presented in [322] and [323] apply feature-level fusion between point cloud information and semantic data obtained by applying a semantic segmentation network to camera images. This fusion strategy enables improved object detection performance. In [323], a neural network architecture is employed that performs point-wise continuous convolution directly on three-dimensional point cloud data and further incorporates point-pooling and attentive aggregation

operations in order to enhance the performance of the fusion process. A similar feature-level fusion task between LiDAR data and pixel-based representations is addressed in [324]. In this work, the authors transform the three-dimensional point cloud into a two-dimensional image representation and subsequently fuse this representation with semantic information using advanced convolutional neural network techniques in order to obtain high-quality results. In [325], a specialized loss function is proposed for efficiently fusing LiDAR data and image information, while simultaneously emphasizing the role of image-derived features in reducing noise effects. This category of fusion is also commonly referred to as early fusion.

Deep fusion methods integrate feature-level information from one sensing modality with either data-level or feature-level information originating from another modality. For instance, LiDAR point clouds can be voxelized, and the resulting three-dimensional voxel representation can be fused with two-dimensional features extracted by neural network models. Deep fusion approaches typically combine information in a cascading manner, exploiting both raw sensor data and high-level feature representations derived from different modalities. An illustrative example is presented in [326], where one neural network is employed to extract embedding representations from LiDAR point cloud data, while a separate neural network instance is used to extract embeddings from camera images. The features obtained from the two sensing modalities are subsequently fused through a series of cascading modules. Another contribution described in [327] introduces two deep fusion strategies, namely PointFusion and VoxelFusion, which merge LiDAR and RGB information in order to improve object detection performance. In the PointFusion approach, LiDAR points are projected onto the two-dimensional image plane using a calibration matrix, after which features are extracted using a pre-trained two-dimensional convolutional neural network and concatenated at the point level. In contrast, the VoxelFusion method projects three-dimensional voxels onto the image plane, extracts features from two-dimensional regions of interest, and fuses the information at the feature level by combining image-derived descriptors at the voxel level.

Late fusion techniques operate at the object level, meaning that the outputs of object detectors associated with individual sensors are

combined. For example, in [328], the outputs of object detection modules corresponding to LiDAR and camera branches are fused in order to increase the reliability of final object predictions. The authors propose a neural network architecture that exploits geometric consistency constraints between two-dimensional and three-dimensional detection results. The proposed approach learns probabilistic dependencies between sensing modalities from training data and achieves real-time performance while maintaining a low memory footprint. Another late fusion strategy is presented in [329], where road plane segmentation obtained from LiDAR point cloud data is combined with semantic segmentation results generated by applying a convolutional neural network to camera images.

In [330], a comprehensive study is conducted on pedestrian classification using both deep learning-based and traditional non-learning approaches, employing data acquired from a monocular RGB camera and a three-dimensional LiDAR sensor. The study analyzes classification performance obtained from individual sensors as well as from combined sensor data using both late fusion and early fusion strategies. Asymmetry fusion represents a class of fusion algorithms in which information originating from different modalities is combined using heterogeneous fusion mechanisms. For example, data from certain sensors may be fused at the object level, while information from other sensors may be integrated at the feature level. In contrast to the previously described fusion paradigms, asymmetry fusion applies different processing strategies across fusion branches, while still relying on a dominant fusion technique. Within this framework, it is possible that object proposals are generated exclusively from one sensing modality, whereas additional sensors provide supplementary features that contribute to performance improvement.

A representative method employing asymmetry fusion is presented in [331], where the authors propose a fusion architecture that is invariant to variations in point cloud density and spatial distribution. The network is capable of recalculating fusion weights by learning correlations among specific features. The proposed approach utilizes two-dimensional image detections to extract frustums from raw LiDAR point clouds, together with associated RGB image information, and integrates these data along with image-derived feature vectors in order

to estimate three-dimensional bounding box coordinates. In [332], an asymmetric fusion strategy between LiDAR and camera data is introduced, addressing the challenge of extracting fine-grained features from both modalities through a novel processing pipeline. This pipeline employs virtual camera viewpoints generated from a LiDAR point cloud that has undergone depth completion and RGB colorization. The resulting information is further refined using a convolutional neural network that processes cuboid proposals obtained in earlier stages together with depth-completed and colorized LiDAR data in order to generate higher-quality oriented cuboids. In [333], a method focused on the detection of small objects using a combined LiDAR–camera framework is proposed. A confidence map derived from LiDAR point clouds, together with RGB information, is used to enhance detection performance.

Weak fusion-based algorithms rely on rule-based mechanisms to guide the interaction between different sensor fusion branches using information originating from a specific modality. An example of such an approach is presented in [334], where a two-dimensional object proposal generated by a convolutional neural network detector is used to define a frustum within the LiDAR point cloud. Within the selected subset of raw point cloud data, object instances are segmented by performing binary classification at the point level. The resulting data are subsequently provided to a bounding box estimation network in order to determine the final object bounding box. In [335], a real-time fusion solution is introduced in which a reinforcement learning policy is employed to select among multiple detections originating from different sensing modalities, with the objective of maximizing average detection precision.

Fusion approaches that do not fall within the previously described categories can be considered as alternative data-driven sensor fusion methods. For instance, Thomas et al. [336] propose an efficient pixel-level fusion technique based on a fully connected artificial neural network for combining low-light television (LLTV) camera images and far-infrared (FLIR) imagery, with the objective of preserving salient information from both sensors. Among the normalized features provided as network inputs are straight edges, curved edges, anisotropy measures, and contrast descriptors extracted from each

image. In [337], a novel deep learning-based LiDAR–image fusion network, referred to as PMNet, is introduced for extracting meaningful information from aerial imagery and three-dimensional point clouds. The fusion process exploits spatial correspondences through point-wise feature-level integration and demonstrates improved performance while maintaining low memory consumption and reduced computational complexity.

Another example of integrating two-dimensional images with three-dimensional point clouds is presented in [338], where the authors propose a network architecture for accurate three-dimensional object detection by leveraging multiple related tasks, including two-dimensional and three-dimensional object detection, depth completion, and ground plane estimation. In this framework, fusion is initially performed at the point and feature levels and is subsequently refined using outputs generated by depth completion and ground plane estimation modules. Caltagirone et al. introduce a novel deep neural network-based fusion approach for integrating LiDAR point clouds and camera images in order to detect road surfaces [339]. The proposed cross-fusion fully convolutional network (FCN) demonstrates superior performance when compared to single-modality methods and alternative fusion strategies designed for road surface detection. A wide range of artificial neural network models have also been developed for sensor fusion applications, including backpropagation neural networks (BPNN) [340], which utilize fully connected architectures trained through backpropagation and Bayesian inference; self-organizing feature maps (SOFM) [341], which employ competitive learning to construct hierarchical network structures and perform data fusion in wireless sensor networks; and Adaptive Resonance Theory Maps (ARTMAP) [342], which implement neural network models capable of generating one-to-many and many-to-one mappings between input vectors and output classes for the purpose of terrain classification and object recognition in complex and conflicting data environments.

4.3 Architectural and Methodological Enhancements to the Sensor Fusion Pipeline

This section presents the proposed enhancements to the sensor fusion pipeline. For clarity and continuity, selected equations are reintroduced, although they have been previously defined in earlier sections. The methods described in this section have been partially disseminated in peer-reviewed publications, while additional aspects, including the hardware architecture of the multi-sensor platform, are presented here for the first time.

4.3.1 General Overview

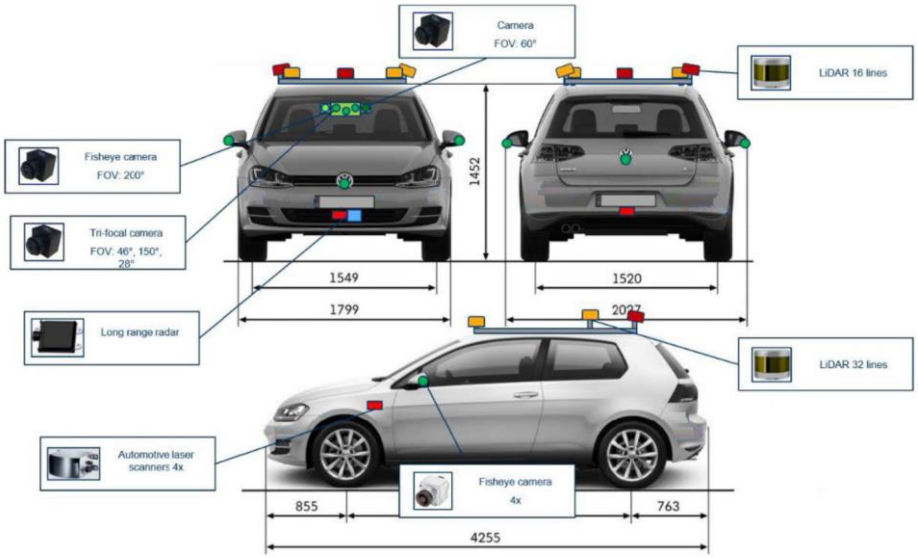

Figure 4.8. Position of sensors on the ego vehicle.

In the present book, redundant information acquired from multiple types of sensors, including a trifocal camera, a fisheye camera, a four-layer LiDAR, a sixteen-layer LiDAR, and a long-range RADAR sensor, is fused. The sensor configuration and the corresponding mounting arrangement on the reference vehicle (also referred to as the ego vehicle) are illustrated in Figure 4.8. An additional thirty-two-layer LiDAR sensor was installed on the vehicle; however, it was not utilized within the scope of the presented experiments and analyses.

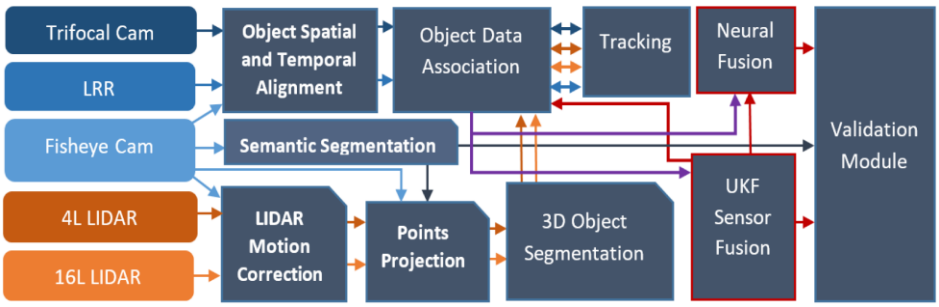

Figure 4.9. Processing pipeline used in the self-driving car solution. The flow of data from each sensor to the modules is depicted using the sensor or module color assigned

The main processing pipeline modules are illustrated in Figure 4.9. Each sensor is represented using a distinct color, and, in order to intuitively emphasize the data flow originating from specific sensors, the arrows connecting the different processing modules are depicted using the same color as the corresponding sensor. The five complementary sensors employed within the proposed system are the trifocal camera (Trifocal Cam), the long-range RADAR (LRR), the fisheye camera (Fisheye Cam), and the four-layer and sixteen-layer LiDAR sensors. Although individual processing modules may receive input data from multiple sensors, the algorithms applied to each sensor type may differ; however, for reasons of clarity and conciseness, the modules are assigned generic designations. Within the presented processing pipeline, the Object Spatial and Temporal Alignment module and the LiDAR Motion Correction module are responsible for performing spatio-temporal alignment of raw sensor data to a common reference timestamp, which is defined by the front-facing fisheye camera. This alignment procedure is carried out at the object level for data originating from the long-range RADAR and trifocal camera sensors, and at the point cloud level within the LiDAR Motion Correction module, as described in Section 4.3.2.

The motion-corrected point clouds are subsequently projected, by means of the Points Projection module, onto the intensity image and onto a semantic segmentation image generated by the Semantic Segmentation module, as described in [220]. This process results in the creation of an enhanced point cloud representation in which each three-dimensional point is augmented with semantic labels and color information. The enhanced point clouds are then provided as input to

the 3D Object Segmentation module, where cuboidal representations corresponding to real-world objects are extracted independently for each of the two LiDAR sensors. The objects obtained from all sensors are subsequently forwarded to the Object Data Association and Tracking modules, where sensor-specific processing algorithms are applied depending on the type of input data. Within the scope of this book, the improvements related to the data association component are discussed in Section 4.3.3.2.

Section 4.3.3.3 introduces an enhanced object-level sensor fusion framework for integrating complementary sensory information. While object detections from the majority of sensors in the presented fusion pipeline are provided directly as input, no dedicated object detector was available for the four-layer LiDAR data. Since detections from the four-layer LiDAR sensor are required by the fusion algorithm described in Section 4.3.3.1, an original object detection method specifically designed for four-layer LiDAR data is proposed. The object-level sensor fusion approach presented in Section 4.3.3.3 and is structured into two main components. The UKF Sensor Fusion module is responsible for fusing information originating from the long-range RADAR and LiDAR sensors, while maintaining the semantic class information provided by the trifocal camera as an associated attribute, without directly fusing the spatial positions of trifocal camera detections with the super-sensor object states. The Neural Fusion module subsequently integrates trifocal camera detections with the hypotheses generated by the UKF Sensor Fusion component, producing a final set of fused object hypotheses.

The sensor fusion process is executed in two successive stages. In the first stage, when no fused object hypotheses are yet available, objects originating from different sensors are associated within the Object Data Association module, and the resulting correspondences are combined to generate initial fused objects. In the second stage, newly acquired data from complementary sensors are associated with the fused objects obtained during the first stage. Subsequently, additional fused object hypotheses are generated for sensor measurements that cannot be associated with any existing hypothesis. The lifecycle management strategy applied to fused objects follows a similar approach to that described in Chapter 3. The primary objective of the sensor fusion module is to stabilize object state parameters, such as position and velocity, while simultaneously assigning a unique identifier to each object present within the scene. In order to validate semantic class

information, a neural network-based fusion mechanism is implemented, with the purpose of generating an additional validation hypothesis for each object fused using the Unscented Kalman Filter (UKF). The bidirectional connections between the Data Association and Tracking modules indicate that raw sensor data are initially processed by the tracking component, and that the resulting tracked object hypotheses are subsequently employed during the data association stage of the fusion process. A detailed description of the core functions of the fusion modules is provided in Section 4.3.1.

Following the sensor fusion stage, a novel Validation Module is introduced, with the purpose of verifying both the class labels and spatial positions of the generated super-sensor objects by exploiting the semantic segmentation image produced by the Semantic Segmentation module. The Validation Module is described in detail in Section 4.3.3.3.2.

4.3.2 Spatio-Temporal Data Alignment

4.3.2.1 Point Cloud Motion Correction and Temporal Fusion for 4-Layer LiDAR

4.3.2.1.1 Motion Correction

The four-layer LiDAR (4L LiDAR) sensor measures the surrounding environment by emitting and receiving laser beams. The complete spatial profile of the environment is constructed through the continuous rotation of the mirror assembly coupled with the laser emission system.

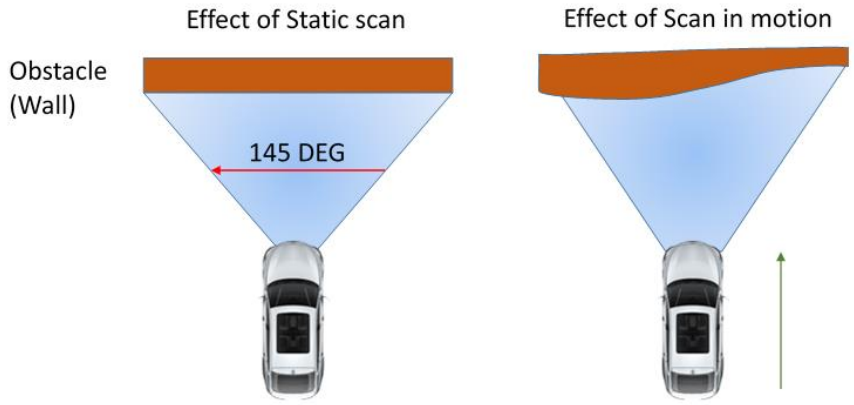

Figure 4.10 Measurement errors caused by motion

A challenging scenario arises when environmental scanning is performed from a mobile sensing platform. Due to the motion of the vehicle, the points acquired within each individual scan are affected by displacement-induced distortions. An intuitive illustration of this phenomenon is presented in Figure 4.10.

Four-layer LiDAR sensors typically provide time stamp information corresponding to one of three possible acquisition instants: the beginning of the scan, the end of the scan, or the midpoint of the acquisition interval. In the present implementation, the time stamp associated with each 4L LiDAR frame corresponds to the beginning of the acquisition process. Based on this assumption, the time stamp of each individual laser point is computed by exploiting knowledge of the angular resolution of the scan, the total scan duration, and the channel identifier associated with each point. All of this information is available either in the sensor datasheet or in published studies that evaluate the performance characteristics of the sensing device [344]. A representative visualization of the data generated by the employed sensor is illustrated in Figure 4.11.

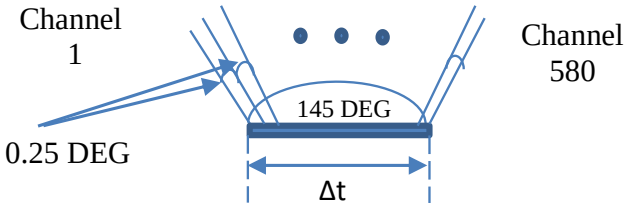

Figure 4.11. Depiction of sparse LIDAR capabilities

By exploiting the aforementioned information, the time stamp associated with each individual point can be computed using Equation (4.1), which is presented below

$$TS_i = Channel_i * \frac{\Delta t}{NumberOfChannels} \tag{4.1}$$

In this equation, TS_i denotes the time stamp corresponding to point i, $Channel_i$represents the index of the i-th channel within the acquired data, $NumberOfChannels$ denotes the total number of channels available for the LiDAR sensor, and Δt refers to the time interval required to complete a full scan. The adopted motion correction approach computes geometric transformations at the level of individual points, as described in [345], while simultaneously relying on ego-

motion information provided by the inertial measurement unit (IMU) installed on the vehicle.

The adopted motion correction approach computes geometric transformations at the level of individual points, as described in [345], while simultaneously relying on ego-motion information provided by the inertial measurement unit (IMU) installed on the vehicle. When computing the point cloud transformation matrix, translational displacements along the x, y, and z axes, as well as rotational components corresponding to pitch, yaw, and roll, are taken into account. Equations (4.2) and (4.3) present the mathematical formulation used to compute the correction transformation applied to each individual point. In these equations, TT denotes the target time stamp, TP represents the time stamp associated with the first acquired point, $TransformationMatrix$ refers to the transformation matrix obtained from ego-motion information and translational displacement values, T_{p_i} denotes the time stamp corresponding to point i, and C_i represents the correction transformation applied to point i.

$$\Delta Cloud = TT - TP \tag{4.2}$$

$$C_i = e^{\left(-\frac{\Delta Tp_i}{\Delta Cloud} log(TransformationMatrix)\right)} \tag{4.3}$$

Considering the i-th point acquired at time stamp t, characterized by spatial coordinates along the x, y, and z axes, the correction applied to the entire point cloud is expressed by Equation (4.4). In this formulation, N denotes the total number of points in the point cloud, while the summation operator $\sum$ represents the iterative aggregation process performed over all points belonging to the point cloud.

Figure 4.12. Exemplification of the motion correction step for a traffic scene.

$$CorrectedCloud(\text{targetTime}) = \sum_{i=0}^{N} C_i * p_i^t \quad (4.4)$$

Figure 4.12 illustrates the effect of the point cloud motion correction process. The uncorrected points are represented in white, whereas the corrected points are depicted in red. Points located in the lower region of the image undergo the largest correction, while those situated in the upper region of the image are subject to smaller correction values. It should also be noted that the temporal difference between the two considered time stamps is relatively small; consequently, the corrected points (shown in red) exhibit only moderate deviations with respect to the original uncorrected points (shown in white). The three-dimensional scene displayed on the right-hand side of the figure is presented in a magnified view in order to facilitate a clearer visualization of the differences between the original and the corrected three-dimensional point representations.

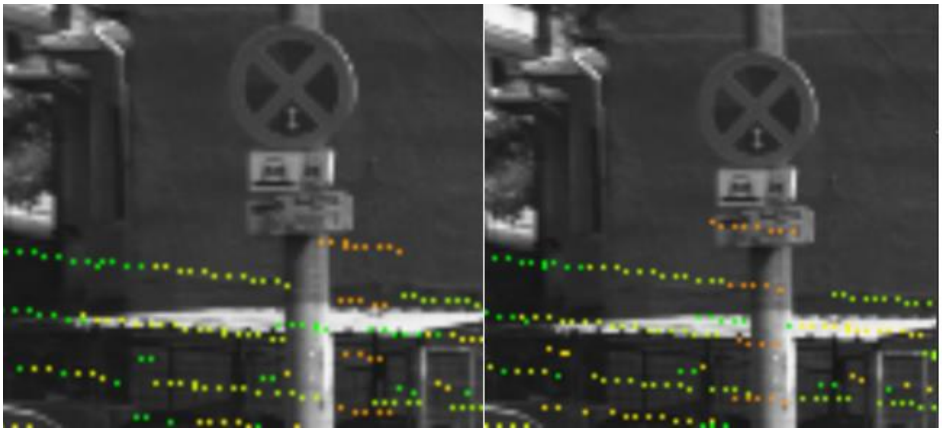

Figure 4.13. Projection onto the image of the original and motion corrected point clouds.

The results obtained after applying the point cloud motion correction procedure are illustrated in Figure 4.13. In this scenario, the three-dimensional point cloud is projected onto a grayscale image. In the use case presented in Figure 4.13, it can be observed that, following the motion correction stage, the three-dimensional points are accurately aligned with the traffic sign pole, falling onto its correct spatial position. In the left-hand side figure, the original 3D points are projected onto the image, and it is visible that they do not fall on the corect objects. On the other hand in left-hand side figure, the motion-corrected points are projected onto the grayscale image and their position is more precise.

4.3.2.1.2 Temporal Fusion

In order to achieve an increased data density, multiple point clouds are fused together. The underlying rationale for this approach is that informative elements present in a given frame, such as road surface points, can be propagated into subsequent frames, thereby facilitating and improving the performance of downstream processing tasks within the pipeline. In the considered experimental setup, six consecutive LiDAR frames are fused, as graphically illustrated in Figure 4.14.

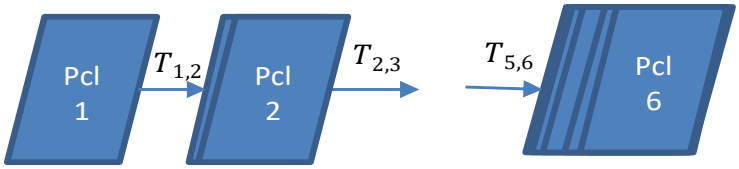

Figure 4.14. Graphical depiction of the fusion process across frames

The transformation function between consecutive point clouds is computed using the motion correction module described previously. The notation $T_{(i,i+1)}$ is used to denote the motion correction transformation applied between two consecutive point clouds. The mathematical formulation employed for fusing multiple point clouds is provided in Equation (4.5).

Figure 4.15. In the left-hand side is the corrected point cloud and in the right-hand side is the temporal fused cloud

$$FinCl = \sum_{i=1}^{5} T_{i,i+1} Pcl_i + Pcl_6 \quad (4.5)$$

The result of the fusion process is illustrated in Figure 4.15. On the left-hand side of the figure, the motion-corrected point cloud is presented, whereas on the right-hand side, the point cloud that has been both motion-corrected and temporally fused is shown. In addition, a video demonstrating the operation of the motion correction and temporal fusion is available. The video can be viewed by accessing the

footnote link [6].

4.3.2.2 Object Spatial and Temporal Alignment

The first stage of the sensor fusion processing pipeline involves the spatial and temporal alignment of sensor data. All object-level detections originating from the individual sensors must be synchronized with respect to a common reference time. Temporal alignment is performed by selecting, for each sensor, the data frame whose time stamp is closest to a predefined reference time stamp, that is, the frame exhibiting the smallest temporal difference relative to the reference.

In order to perform motion correction, it is first necessary to account for the motion of the reference vehicle, also referred to as the ego vehicle. Since sensors typically provide only the relative velocity components of detected objects, denoted as V_{relx} and V_{rely}, it is required to compute the corresponding absolute velocity components, V_x and V_y, along the longitudinal (x) and lateral (y) directions, respectively, as expressed in Equation (4.6). The ego vehicle velocity components along the $x$and $y$axes are denoted by V_{0x} and V_{0y}.

$$V_x = V_{relx} + V_{0x}$$
$$V_y = V_{rely} + V_{0y} \tag{4.6}$$

After determining the absolute velocity of the target object, it is necessary to compute the position (x, y) of the target vehicle after its displacement over a time interval Δt within the ego vehicle reference frame. This can be achieved by applying the kinematic equations presented in Equation (4.7), where (xt_0, yt_0) denote the initial spatial coordinates of the target vehicle prior to the motion interval:

$$x = xt_0 + V_x \Delta t$$
$$y = yt_0 + V_y \Delta t \tag{4.7}$$

The motion of the ego vehicle during the considered time interval is subsequently taken into account. The ego vehicle is assumed to move

[6] https://youtu.be/Qc19N_Xp82k

from point A to point B over a distance S. During this displacement, the vehicle describes a rotational motion characterized by a swept angle θ, which corresponds to the change in orientation resulting from the vehicle trajectory over the time interval Δt. The angle θ can be expressed as a function of the ego vehicle yaw rate φ, as formulated in Equation (4.8).

$$\theta = \phi \Delta t \tag{4.8}$$

The magnitude of the ego vehicle velocity, denoted by V_0, can be computed using the longitudinal and lateral velocity components along the x and y axes, as expressed in Equation (4.9).

$$V_0 = \sqrt{v_{0x}^2 + v_{0y}^2} \tag{4.9}$$

Finally, the displacement S is computed as illustrated in (4.10).

$$S = V_0 \Delta t \tag{4.10}$$

If the angle θ is smaller than a predefined threshold value, the vehicle motion is approximated as linear. Otherwise, the motion is modeled as a circular trajectory segment, characterized by a radius R and a sector angle θ. Under these assumptions, the traveled distance S can be computed using two alternative formulations. The first approach is expressed in Equation (4.10), while the second method is provided in Equation (4.11).

$$S = \Theta R \tag{4.11}$$

By considering the geometric representation illustrated in Figure 4.16, it can be observed that, within the right triangle ODB, the

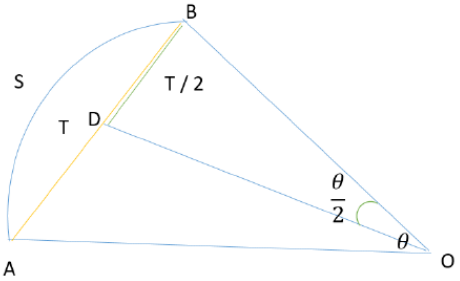

Figure 4.16. Notations (on the figure) depicting ego vehicle movement from point A to point B

expression of $T/2$ can be derived, where the radius R corresponds to the length of the segments OB and OA.
The final expression of T can be identified by combining equations (4.10), (4.11) and (4.12).

$$\frac{T}{2} = Rsin\left(\frac{\Theta}{2}\right) \tag{4.12}$$

$$T = \frac{2(V\Delta t)}{\Theta}\sin\left(\frac{\Theta}{2}\right) \tag{4.13}$$

If the ego vehicle undergoes rectilinear motion, the classical kinematic equations can be applied. Otherwise, the displacement can be computed using Equation (4.13). After determining the distance T traveled by the ego vehicle over the time interval Δt, the corresponding displacement components along the x and y axes, denoted as T_x and T_y, can be computed, as illustrated in Figure 4.17 and expressed analytically in Equation (4.14).

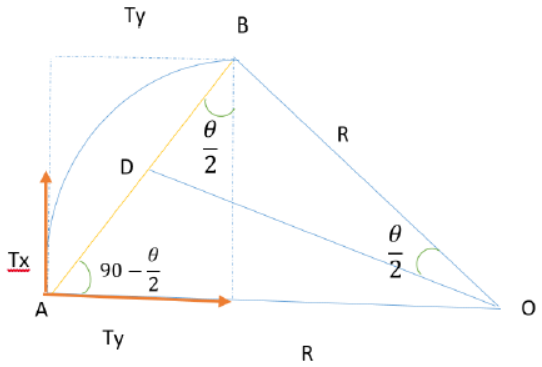

Figure 4.17. Tx and Ty movement components.

$$Tx = Tcos(\frac{\Theta}{2})$$

$$Ty = Tsin(\frac{\Theta}{2}) \tag{4.14}$$

The final stage of the object motion correction procedure consists of determining the position of the target vehicle within the current ego vehicle reference frame, corresponding to point B. The resulting target positions obtained by applying the rotation and translation transformations are expressed in Equations (4.15) and (4.16).

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \cos(\Theta) & -\sin(\Theta) \\ \sin(\Theta) & \cos(\Theta) \end{bmatrix} \begin{bmatrix} y - Ty \\ x - Tx \end{bmatrix} \tag{4.15}$$

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} (y - Ty)\cos(\Theta) - (x - Tx)\sin(\Theta) \\ (y - Ty)\sin(\Theta) + (x - Tx)\cos(\Theta) \end{bmatrix} \tag{4.16}$$

The transformations defined in Equation (4.16) must be applied to all detections originating from all sensors in order to align the data to a common reference time stamp.

4.3.3 Object Level Fusion

4.3.3.1 Object Detection for 4-Layer LiDAR

In order to enable high-level sensor fusion, object detections originating from all available sensors must be provided to the fusion center. Within the scope of the Up-Drive project, for which the proposed fusion framework was implemented, no dedicated object detection algorithm was available for data acquired by the four-layer LiDAR sensor. Such detections are of particular importance, as this type of LiDAR is capable of detecting objects located at relatively large distances, which cannot be reliably perceived by the VLP-16 LiDAR sensor. Prior to presenting the high-level fusion mechanism, the methodology employed for object detection using sparse four-layer LiDAR data is introduced, as this facilitates a more complete and clearer understanding of the overall fusion framework.

One of the primary challenges associated with object detection using four-layer LiDAR data arises from the sparsity of the acquired point cloud, which can lead to suboptimal performance of conventional clustering-based detection algorithms. In order to increase point cloud density, the temporal fusion approach described in the previous section is applied. The three-dimensional object detection process is structured into two main stages. The first stage consists of road point detection and the construction of an elevation grid, while the second stage involves obstacle detection and filtering. Each of these processing steps is discussed in detail in the following sections.

4.3.3.1.1 Road Point Detection and Creation of Elevation Grid

Road surface estimation using four-layer LiDAR data represents a challenging task, primarily due to the limited number of points that intersect the ground surface. The initial step in ground plane detection consists of filtering out points whose height coordinate (z) exceeds a predefined threshold of 0.5 m. The resulting three-dimensional point set is further refined by retaining only those points whose amplitude values are greater than 0.8, as high-amplitude returns are typically associated with lane markings or pedestrian crossings.

The coordinate system is defined such that the origin is located in front of the vehicle at ground level. Under this assumption, the road surface within a range of approximately 10–20 meters can be approximated by a straight line passing through the origin. This line can be characterized by the pitch angle it forms with the horizontal axis passing through the defined origin. Consequently, the number of points falling on candidate lines corresponding to different pitch angles is evaluated in the side-view projection. A histogram is constructed to store the number of points associated with each candidate line. The line corresponding to the road surface is then identified as the one that accumulates the maximum number of points.

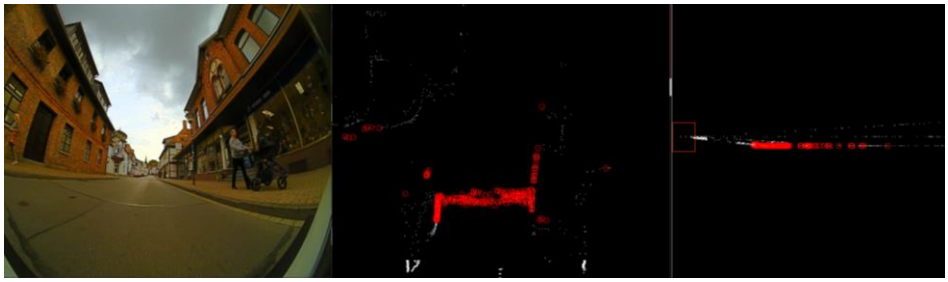

Figure 4.18. The left image represents the scene viewed with a fisheye camera. In the center and right-hand side, the points belonging to the ground surface are highlighted in red.

Once the road line has been determined, the corresponding ground points are removed from the initial three-dimensional point cloud. In order to identify the points associated with each candidate line, the polar coordinates of the points are converted into Cartesian coordinates, as described by Equation (4.17). Subsequently, the equation of the line passing through the origin is computed in Cartesian form, with the radial distance parameter $r$set to a value of 20.

$$x = rcos(\theta); y = rsin(\theta) \tag{4.17}$$

Figure 4.18 highlights the points corresponding to the road surface, which are marked by a red contour. The left-hand side of the figure presents the top-view representation, while the right-hand side illustrates the road points in the side-view projection, corresponding to the distance–height plane

4.3.3.1.2 Obstacle Detection and Filtering

After removing the road surface points, a 2.5D elevation grid is constructed. The three-dimensional space is discretized into cells with dimensions of 40 × 60 cm, and the elevation grid is generated for points located within a range of less than 50 m in depth and less than 20 m in width, corresponding to 10 m on each side of the ego vehicle. The three-dimensional point cloud is projected onto this grid using a top-view representation. For each grid cell, the maximum height value among all points falling within the cell is stored, together with the number of points associated with that cell, which represents the local point density.

A grid cell is assigned an object label if the point density exceeds a predefined threshold and the maximum height value is lower than 4 m. In the considered implementation, the density threshold is set to a value of 4; however, this parameter depends on the number of temporally fused LiDAR frames, such that a larger number of fused frames requires a correspondingly higher density threshold. After labeling all grid cells, a clustering procedure is applied in order to aggregate neighbouring object cells. Two cells are merged into the same object cluster if they are adjacent in any of the eight possible neighbourhood directions, including horizontal, vertical, and diagonal connectivity.

The initially detected object clusters may also correspond to buildings or other large static structures. Therefore, an additional filtering stage is introduced in order to retain only relevant objects of interest. This is achieved by imposing geometric size constraints. Specifically, an object is considered valid only if its width and length are smaller than predefined limits, which are set to 3 m and 12 m, respectively, in the current configuration. Furthermore, clusters consisting of fewer than two grid cells are discarded. The final filtered detection results are illustrated in Figure 4.19. The first image presents the scene as captured by a two-dimensional camera, the central image depicts the elevation grid along with the grouped object clusters, and the color-coded visualization highlights different elements: green lines represent grid boundaries, red regions indicate clustered objects, white regions correspond to road points or filtered object points, and blue contours denote the bounding boxes associated with detected object boundaries.

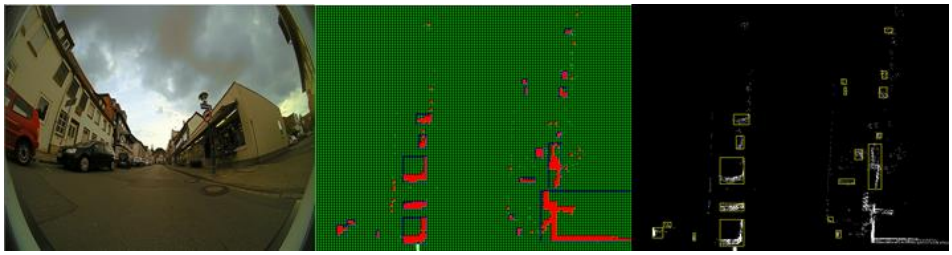

Figure 4.19. Intensity image of the scene (left); Unfiltered labeled objects (middle); Filtered objects in yellow bounding boxes (right)

After the filtering stage, additional detections may still be present, such as poles, longitudinal roadside barriers, or small portions of buildings which, due to the sparsity of the point cloud, may be incorrectly segmented as individual objects. While such structures can be intuitively interpreted by human observers based on contextual visual information, it is not always possible to determine their semantic relevance solely from sparse point cloud data. For this reason, these detections are retained and reported as objects rather than being removed at this stage. Within the subsequent sensor fusion framework, such detections are treated as clutter and are further filtered during the fusion process.

The remaining sensors either directly provide object-level detections, such as the RADAR sensor and the Mobileye trifocal camera, or rely on dedicated object detection modules that were developed for the corresponding sensor data by other members of the Up-Drive project team [346].

4.3.3.1.3 Evaluation of the Object Detection using 4L LIDAR

In this section, an evaluation of the proposed algorithm is presented in terms of both qualitative performance and computational execution time. The object list generated by the proposed approach is compared with the object list provided by a 77 GHz long-range RADAR sensor in a traffic scenario. To this end, two experimental scenarios are considered. In the first scenario, a single vehicle is tracked over a sequence of frames, and the number of instances in which the proposed algorithm fails to detect the vehicle while the RADAR sensor successfully detects it is recorded. In the second scenario, a traffic intersection containing multiple vehicles is analyzed, and, over a predefined number of frames,

the number of vehicles missed by the proposed solution relative to the RADAR-based detections is quantified.

Although the two sensing modalities exhibit different operational characteristics, experimental scenes are selected such that environmental conditions are favorable for both LiDAR and RADAR sensors, thereby ensuring that objects can be reliably detected by both systems. The experimental platform used for performance evaluation is equipped with an Intel i5-2500 CPU operating at a clock frequency of 3 GHz. No hardware acceleration techniques are employed during the execution of the proposed algorithm.

Both LiDAR and RADAR detections are expressed within a common reference coordinate frame, and RADAR measurements are additionally subjected to motion correction in order to align them temporally with the LiDAR data. For simplification purposes, the three-dimensional RADAR detections are projected into a top-view representation within the same virtual image space as the LiDAR detections. Consequently, each three-dimensional world object is represented by an associated two-dimensional virtual object. For each virtual LiDAR-based object, an attempt is made to identify the corresponding RADAR-based object located within a circular search region of 60 pixels radius in the virtual

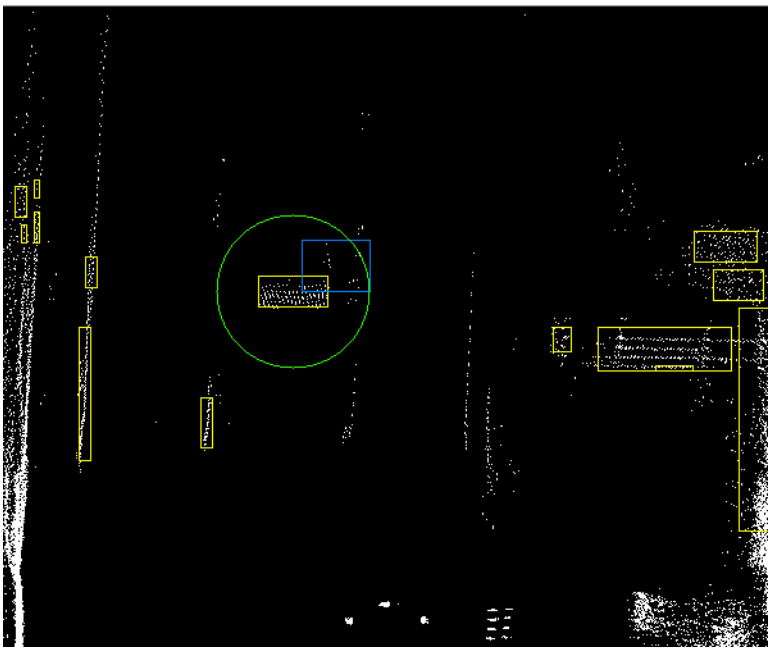

Figure 4.20. LIDAR-RADAR object association

image. In cases where multiple RADAR objects fall within this search region, the object exhibiting the most similar geometric dimensions to the LiDAR-based virtual object is selected as the corresponding match. An example illustrating the association between a LiDAR-based object and a RADAR-based object is presented in Figure 4.20. It should be noted that, in this example, the RADAR object lies within the association search circle centered on the corresponding LiDAR object.

In Figure 4.20, the identified LiDAR-based objects are highlighted in yellow, the RADAR-based object is depicted in blue, the filtered three-dimensional point cloud is shown in white, and the established associations are indicated in green. Figure 4.21 presents an example of RADAR–LiDAR association in a densely populated intersection scenario, where the color coding retains the same semantic meaning as in the previous figure.

Table 4.1 Object detection accuracy

Scenario	Nr. of frames	Accuracy
Single object detection	300	97%
Multiple object detection	1000	93%

Table 4.1 reports the percentage of RADAR objects that remain unassociated with LiDAR detections in the case of a single road object,

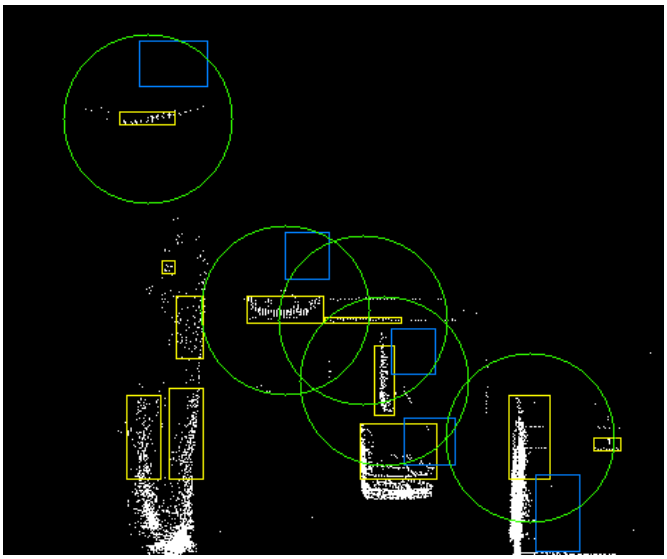

Figure 4.21. Sensor object associations in an intersection

evaluated over a sequence of 300 frames. In the second row of the table, the percentage of unassociated road objects is presented for an extended dataset comprising more than 1000 images.

It should be noted that, in certain scenarios, the LiDAR sensor may fail to capture objects located in front of the ego vehicle due to occlusion effects, whereas the RADAR sensor may still be capable of detecting such objects. This situation was observed during the evaluation of the proposed algorithm in a crowded intersection scenario, in which multiple vehicles were positioned consecutively in close proximity. This phenomenon is primarily attributed to the physical placement and relative mounting positions of the sensors on the vehicle platform.

Figure 4.22 illustrates a scenario in which the leading vehicle is correctly detected and successfully associated with a corresponding RADAR-based object.

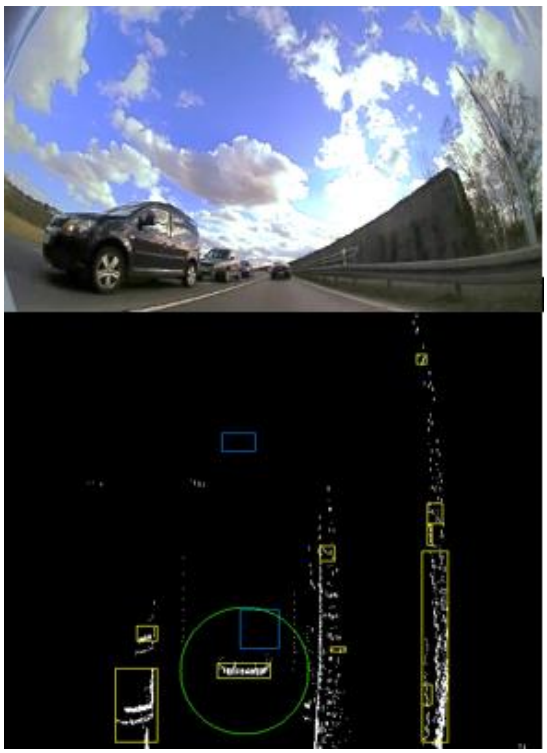

Figure 4.22. Partially occluded object scenario; the top image represents the data from the camera; the bottom image represents the top view image containing LIDAR and RADAR objects.

The second vehicle, relative to the ego vehicle position, is not detected by the LiDAR-based algorithm due to the occlusion of the sensor field of

view caused by the leading vehicle. The processing performance achieved by the proposed algorithm on the specified hardware platform corresponds to a frame rate of approximately 15 frames per second.

4.3.3.2 Data Association for the Trifocal Sensor

Data association represents a fundamental stage within the sensor fusion pipeline. During this stage, measurements originating from multiple sensors are correlated with the objective of enriching the available information associated with detected objects. The association of measurements provided by the trifocal camera with super-sensor objects, or object hypotheses, constitutes a particularly challenging task, primarily due to the fact that the trifocal camera delivers highly inaccurate estimates of object spatial positions. For the trifocal sensor employed in this work, no predefined rules or parameter sets describing the measurement error covariance matrix are available.

In order to address this limitation, an association strategy structured in the form of a decision tree has been implemented to enable efficient matching between trifocal camera measurements and target objects. The proposed approach aims to exploit as many object-level features as possible, while simultaneously avoiding a significant degradation of real-time performance. In order to simplify the association problem, virtual two-dimensional objects are generated from the motion-corrected three-dimensional detections and subsequently projected

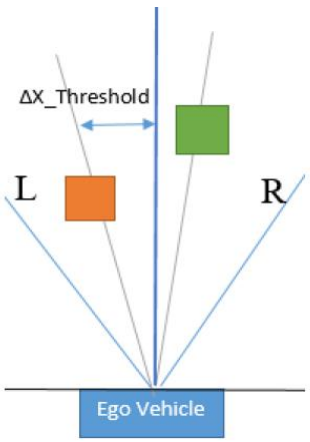

Figure 4.23. Corresponding objects in the left half (orange) and the right half (green) spaces

into a virtual two-dimensional image space. From the complete spatial domain, only objects located within a maximum distance of 50 m in front of the ego vehicle and within 20 m on both the left and right sides are considered.

The resulting search space is further divided into two regions corresponding to the left-hand side and the right-hand side of the ego vehicle. Each detected object is assigned to one of these regions. When the association algorithm searches for a corresponding object on a given side, it also accounts for the possibility that the matching object may be located near the boundary of the opposite side. For instance, in the case of an object situated on the right-hand side, the search process is extended to the left-hand side up to a predefined distance threshold along the longitudinal (x) axis. This is exemplified in Figure 4.23 for better visual understanding.

After identifying measurements that are located in the vicinity of a target object, the corresponding cuboid candidates are further filtered by taking into account their geometric dimensions.

For each object hypothesis, a search is performed over the set of filtered measurements in order to determine the candidate exhibiting the most similar visible façade relative to that of the target object. This problem is addressed by computing the difference between the visible façade of the target object and the visible façade of each potential trifocal camera detection. In this manner, the correspondence that yields the highest similarity, in terms of visible façade characteristics, is identified.

For subsequent processing stages, objects whose dimensional differences fall within a specified tolerance range relative to the best-matching candidate are also retained, since discrepancies may arise due to sensor measurement uncertainties. The tolerance parameter, denoted by ζ, is determined empirically and is set to a value of 5 pixels.

Candidate objects whose dimensional differences do not satisfy the interval defined by the minimum identified difference are discarded according to Equation (4.18). In this equation, the parameter ∂represents the dimensional difference metric, while δdenotes the minimum difference value obtained during the matching process.

.

$$f(\partial)=\begin{cases}1, if\ \partial\ \in [\delta-\zeta, \delta+\zeta]\\ 0, otherwise\end{cases} \quad (4.18)$$

Another physical attribute exploited within the proposed architecture is the object area. In the adopted approach, a trifocal camera detection is considered suitable for further evaluation if the ratio between the area of the target object and that of the corresponding trifocal object is below a predefined threshold, denoted by η. This criterion is introduced in order to eliminate hypotheses that are significantly smaller than the target object.

In addition, similar objects are identified by scanning the complete set of detected objects using a polar ray originating from a fixed position located in the central region of the ego vehicle reference frame. The transformation from Cartesian coordinates to polar coordinates is performed using the expressions provided in Equation (4.19).

$$\rho = \sqrt{x^2 + y^2}$$
$$\Theta = \tan^{-1}\left(\frac{y}{x}\right) \tag{4.19}$$

The association process is illustrated in Figure 4.20, where green squares denote target objects, orange squares represent trifocal camera measurements, and the blue square located at the center indicates the position of the ego vehicle. The polar rays used in the association procedure are depicted in grey. By applying this methodology, it can be observed that corresponding objects are located in close spatial proximity, exhibiting minimal differences in their associated angular and radial components, specifically small values of $\Theta_1 - \Theta_2$ and $\rho_1 - \rho_2$ for matching object pairs.

Based on this observation, potential trifocal camera detections are further filtered by retaining only those candidates whose differences in radial distance ρ and angular position Θ, relative to the target object, fall below predefined threshold values, denoted as Θ_t and ρ_t. An intuitive representation of this filtering mechanism is provided in Figure 4.24. For the remaining candidate detections, the final association step consists of matching the target object with the trifocal measurement that is spatially closest, using the Euclidean distance as the similarity metric. System state variables are updated each time a potential correspondence is identified. The algorithm outputs the best association obtained between object hypotheses and trifocal camera measurements.

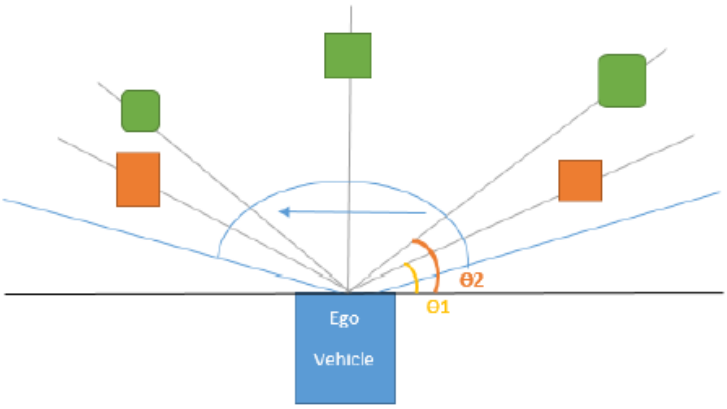

Figure 4.24. The two-dimensional sweeping polar rays employed for object association, which are depicted in light blue. Motion-corrected trifocal detections are orange, while target objects are in green.

For illustration purposes, the results of the association process are presented for a single-vehicle detection example. Although the proposed algorithm is capable of handling multiple object instances simultaneously, a single-object scenario is used here to ensure clarity and facilitate understanding of the association mechanism.
The proposed algorithm is capable of reliably associating trifocal camera measurements with all categories of target objects, including super-sensor objects, LiDAR-based detections, and RADAR-based detections.

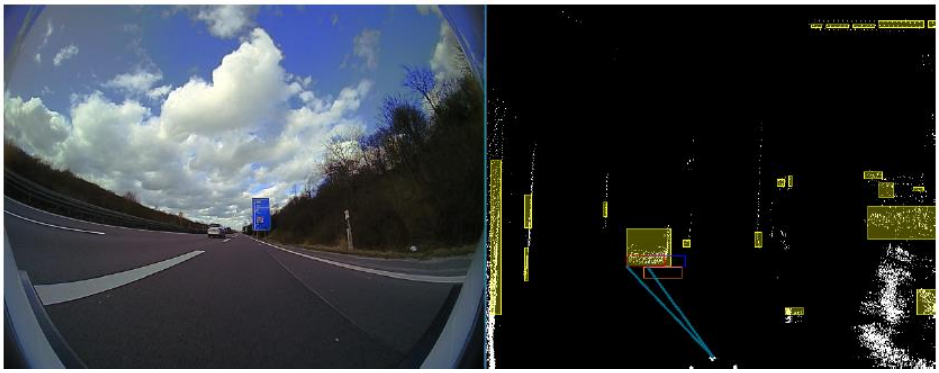

Figure 4.25. Results of LIDAR object to trifocal object association. On the left-hand side, the color image of the recorded scene is displayed. On the right-hand side, the processed sensory data containing trifocal and LIDAR objects are shown.

Figure 4.25 presents, on the right-hand side, the fused three-dimensional point cloud obtained from the four-layer and sixteen-layer LiDAR sensors, which is displayed in white. The filtered LiDAR-based object detections are illustrated using yellow bounding boxes.

The original position of the trifocal camera detection is shown in orange, the motion-corrected trifocal position is depicted in dark blue, and the final associated trifocal object position is represented in red. The intensity image displayed on the left-hand side illustrates the corresponding real-world scene. In this visualization, the polar association rays are rendered in light blue. In situations in which no suitable correspondence is identified for a given object, the association process outputs the motion-corrected trifocal camera position as the final estimate.

In Figure 4.26, the same association algorithm is employed to establish correspondences among objects detected by the LiDAR, RADAR, and trifocal camera sensors. The LiDAR-based object is represented in yellow, the RADAR-based object is displayed in cyan, and the trifocal camera detection is depicted in red. The left-hand side of the figure illustrates the real-world scene as observed by the sensing system. The parameter values used for the trifocal camera data association process were determined experimentally. The selected constants are as follows: $X_{Threshold} = 25$, $\zeta = 5$, $\eta = 1$, $\Theta_t = 5$, and $\rho_t = 5$.

Figure 4.26. The correspondence between trifocal camera detections and measurements obtained from the LiDAR and RADAR sensors. The left-hand side of the figure presents the color image captured by the camera, while the right-hand side depicts the data association results for objects detected by the different sensing modalities.

4.3.3.3 Object Level Sensor Fusion

Within the sensor fusion module, information acquired from multiple sensors is aggregated into a unified object state, effectively treating the combined measurements as if they were provided by a single super-sensor. Measurements originating from each individual sensor are associated with existing fused object hypotheses, after which the corresponding object states are updated using the newly available detection information. From an implementation perspective, two distinct operational scenarios are considered.

In the case of an online implementation, object detection requests are first issued to the detection modules associated with each sensor. These detection modules return either a list of detected objects or an empty list. The returned detections are initially temporally aligned and subsequently forwarded to the fusion center for further processing. In the case of pre-recorded experimental data stored on disk, where each data sample is associated with a recording time stamp, the reference time is defined by the front-facing video camera. For each sensor, the frame whose time stamp exhibits the smallest difference relative to this reference time stamp is selected. If the time stamp difference corresponding to a given sensor exceeds a predefined threshold, the associated frame is discarded, under the assumption that data from that sensor were not available for the current reference frame. This threshold is defined as the time difference between two consecutive reference image frames.

When no fused object hypotheses have yet been initialized, the number of detected objects in each sensor-specific object list is evaluated. The detection list containing the largest number of objects is selected as the reference set, and the detections within this list are used to initialize the initial fused object hypotheses. Subsequently, detections originating from the remaining sensor lists are associated with the existing fused objects, and the corresponding object states are updated using information provided by the complementary sensors. In general, data association for LiDAR- and RADAR-based detections is performed using a distance-based metric, which is described in the subsequent sections, while the trifocal camera data association relies on the methodology presented in the previous section. After each association step involving sensor measurements, an optimal assignment algorithm is executed in order to identify the most suitable correspondences.

Sensor detections that cannot be associated with any existing fused object are used to initialize new fused object hypotheses.
The processes of association, optimal assignment, state update, prediction, and fused object management are executed for all object lists provided by the different sensors. The fused object lifecycle management strategy follows an approach similar to that employed in the tracking framework described in the previous chapter and is therefore not reiterated here. The principal distinction lies in the fact that, in the present fusion framework, detections are received not only across consecutive time frames, as in the tracking case, but also from multiple sensors within the same frame.

For the prediction and update stages, the Controlled Turn Rate and Velocity (CTRV) motion model is employed in conjunction with the Unscented Kalman Filter (UKF). Following the data association phase, measurements originating from each sensor are sequentially incorporated into the UKF in order to obtain filtered estimates of object position and velocity and to accumulate all relevant sensor information within a unified object representation. The state vector corresponding to the adopted motion model is defined in Equation (4.20).

$$\mathrm{X_k} = [\mathrm{px} \quad \mathrm{py} \quad \mathrm{v} \quad \psi \quad \dot{\psi}]^{\mathrm{T}} \tag{4.20}$$

The CTRV motion model is described by two distinct analytical formulations, depending on whether the vehicle undergoes turning motion or follows a rectilinear trajectory. The mathematical representation of the CTRV process model corresponding to the turning motion case is provided in Equation (4.21), as presented below:

$$X_{k+1} = X_k + \begin{bmatrix} \frac{\mathrm{v_k}}{\dot{\Psi}_k}(\sin(\Psi_k + \dot{\Psi}_k \Delta t) - \sin(\Psi_k)) \\ \frac{\mathrm{v_k}}{\dot{\Psi}_k}(-\cos(\Psi_k + \dot{\Psi}_k \Delta t) + \sin(\Psi_k)) \\ 0 \\ \dot{\Psi}_k \Delta t \\ 0 \end{bmatrix} + \begin{bmatrix} \frac{1}{2}(\Delta t)^2 \cos(\Psi_k)\, \gamma_{a,k} \\ \frac{1}{2}(\Delta t)^2 \sin(\Psi_k)\, \gamma_{a,k} \\ \Delta t \gamma_{a,k} \\ \frac{1}{2}(\Delta t)^2 \gamma_{\dot{\Psi},k} \\ \Delta t \gamma_{\dot{\Psi},k} \end{bmatrix} \tag{4.21}$$

When the vehicle exhibits rectilinear motion, the corresponding process model is described by the analytical formulation presented in Equation (4.22).

Within the Unscented Kalman Filter framework, a set of sigma points is generated and subsequently propagated through the nonlinear process function.

$$X_{k+1} = X_k + \begin{bmatrix} v_k \cos(\Psi_k)\Delta t \\ v_k \sin(\Psi_k)\Delta t \\ 0 \\ \dot{\Psi}\Delta t \\ 0 \end{bmatrix} + \begin{bmatrix} \frac{1}{2}(\Delta t)^2 \cos(\Psi_k)\,\gamma_{a,k} \\ \frac{1}{2}(\Delta t)^2 \sin(\Psi_k)\,\gamma_{a,k} \\ \Delta t \gamma_{a,k} \\ \frac{1}{2}(\Delta t)^2 \gamma_{\dot{\Psi},k} \\ \Delta t \gamma_{\dot{\Psi},k} \end{bmatrix} \tag{4.22}$$

Following this transformation, the Gaussian distribution is reconstructed from the transformed sigma points. The first sigma point corresponds to the mean value of the state vector, as defined in Equation (4.23).

$$X_{K|K}^{0} = X_{K|K}^{*} \tag{4.23}$$

The remaining sigma points are generated by applying a scaling operation around the mean state using a spreading factor denoted by λ, as expressed in Equations (4.24) and (4.25).

$$X_{K|K}^{i} = X_{K|K}^{*} + \sqrt{(\lambda + n_x)P_{(k|k)}} \tag{4.24}$$

$$X_{K|K}^{i} = X_{K|K}^{*} - \sqrt{(\lambda + n_x)P_{(k|k)}} \tag{4.25}$$

The resulting probability density function represents an approximation of the underlying Gaussian distribution. Although the Unscented Kalman Filter does not constitute an optimal estimation algorithm in a strict theoretical sense, it is widely adopted in LiDAR-based multi-object tracking applications due to its relatively low computational complexity when compared to the classical Kalman Filter. Subsequently, the state covariance matrix is reconstructed using the propagated sigma points. In order to perform this operation, the dispersion introduced during the sigma point generation process is compensated by applying a corresponding set of weighting coefficients, as defined in Equations (4.26) and (4.27). As can be observed, these weighting coefficients are explicitly dependent on the spreading parameter λ.

$$w_i = \frac{\lambda}{\lambda + n_a}, i = 0 \tag{4.26}$$

$$w_i = \frac{1}{2(\lambda + n_a)}, i = 2, \ldots, n_a \tag{4.27}$$

The predicted state mean and covariance are computed using the formulations presented in Equations (4.28) and (4.29), respectively.

$$X_{k+1|K} = \sum_{i=1}^{n_\sigma} w_i X_{k+1|k,i} \tag{4.28}$$

$$P_{k+1|K} = \sum_{i=1}^{n_\sigma} w_i (X_{k+1|k,i} - x_{k+1|k})\,(X_{k+1|k,i} - x_{k+1|k})^T \tag{4.29}$$

During the update stage, the measurement models are linear; therefore, no additional linearization procedure is required. The Kalman gain is computed according to the formulation provided in Equation (4.30).

$$K = P_k H^T (\, H P_k H^T + R)^{-1} \tag{4.30}$$

In the formulations presented in Equation (4.31), the state vector and the corresponding covariance matrix are updated based on the measurement readings provided by each individual sensor.

$$X_k = X_k + K(z_k - H X_k),\; P_k = (I - KH) P_k \tag{4.31}$$

It is important to note that, for the initial frame, an alternative association strategy has also been evaluated. Within this approach, correspondences are established among all objects originating from the available sensors. The associated object pairs are stored in two lookup tables in order to improve the computational efficiency of the subsequent fusion procedure. For the first frame, the associated objects are directly fused. After the initial frame, the association process is performed with respect to the already fused object hypotheses, such that LiDAR and RADAR measurements are associated with the fused objects.

As the association criterion, a weighted combination of candidate object area and Euclidean distance is employed, as defined by Equations (4.32)–(4.34). The candidate object position in the planar coordinate system is denoted by c_x and c_y, while the fused object position is represented by f_x and f_y. The Euclidean distance between the candidate object and the fused object is denoted by d. Furthermore, the area of the fused object is represented by FA, and the area of the candidate object is denoted by CA.

$$d = \sqrt{(c_x - f_x)^2 + \left(c_y - f_y\right)^2} \tag{4.32}$$

$$dims = \frac{FA}{CA} \tag{4.33}$$

$$rez = \frac{0.6*d+0.4*dims}{dims+d} \tag{4.34}$$

The association of trifocal camera measurements is performed using the methodology described in the previous section.
Each fused object maintains an associated class frequency vector.

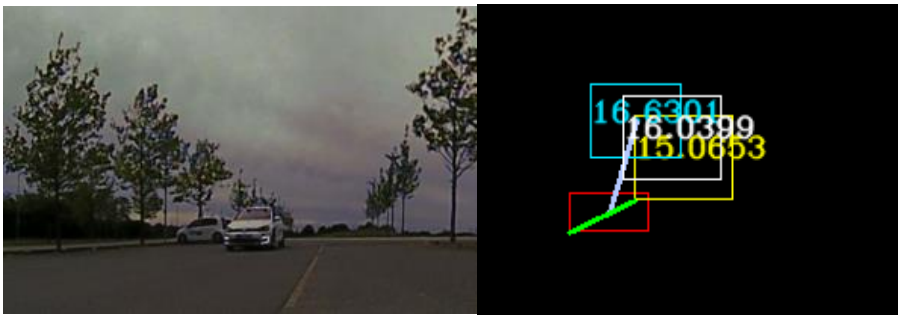

Figure 4.27. Associations between measurements originating from different sensors and their corresponding fusion results are illustrated in this figure. On the left-hand side, the RGB image is presented, while on the right-hand side, the data associations among the sensors are shown together with the resulting super-sensor object, which is depicted in white.

When a trifocal camera detection is associated with a fused object, it contributes a vote to the entry corresponding to the detected semantic class within the class frequency vector. Fused object hypotheses are maintained as active entities as long as new measurements continue to be associated with them. The overall number of fused objects is regulated by adopting a management strategy similar to that employed in the sixteen-layer LiDAR object tracking framework. Specifically, if no measurement associations are observed over a period of ten consecutive frames or for a duration exceeding two seconds, the corresponding fused object is removed. In Figure 4.27, the fused object is represented in white, while the remaining bounding boxes correspond to measurements originating from the other sensors.

4.3.3.3.1 Trifocal Camera Object Sensor Fusion

For the fusion of trifocal camera detections, a neural network–based approach is adopted. This fusion strategy is selected due to the absence of available parameters required to construct a reliable measurement

covariance matrix for trifocal camera objects. In order to generate ground truth data, a reference vehicle equipped with a high-precision GPS system, providing positional accuracy of approximately 2 cm, is employed. Multiple data sequences are recorded in a variety of controlled experimental environments as well as in real-world driving scenarios. Several neural network architectures are evaluated for the fusion task; however, a single-layer perceptron model is found to provide the most suitable performance under the considered experimental conditions. The network is trained for a maximum of 500 epochs or until the training error decreases below a threshold value of 0.1. The learning rate is set to 0.0001, and no momentum term is applied during the training process.

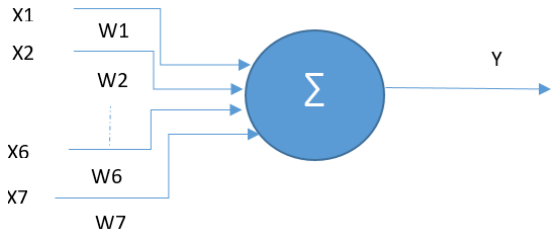

Figure 4.28. Model with 7 inputs 1 processing unit and 1 output

The adopted neural network model comprises seven input variables and a single output, as illustrated in Figure 4.28. The output of the network is computed according to the formulation provided in Equation (4.35).

$$y = f(x) = \sum w_i X_i \tag{4.35}$$

The model was trained using the delta learning rule, as defined in Equation (4.36), with the GPS-derived position of the target vehicle serving as the reference ground truth data.

$$w = \Delta w + w_{old} = w_{old} + \eta\delta x \text{ , where } \delta = y_{target} - y \tag{4.36}$$

The multimodal sensor fusion architecture is illustrated in Figure 4.29. Data acquired from each sensor are first subjected to tracking and filtering procedures prior to being introduced into the sensor fusion module. The object position estimates obtained after the fusion process exhibit increased stability and can therefore be utilized with a higher level of confidence by subsequent processing components.

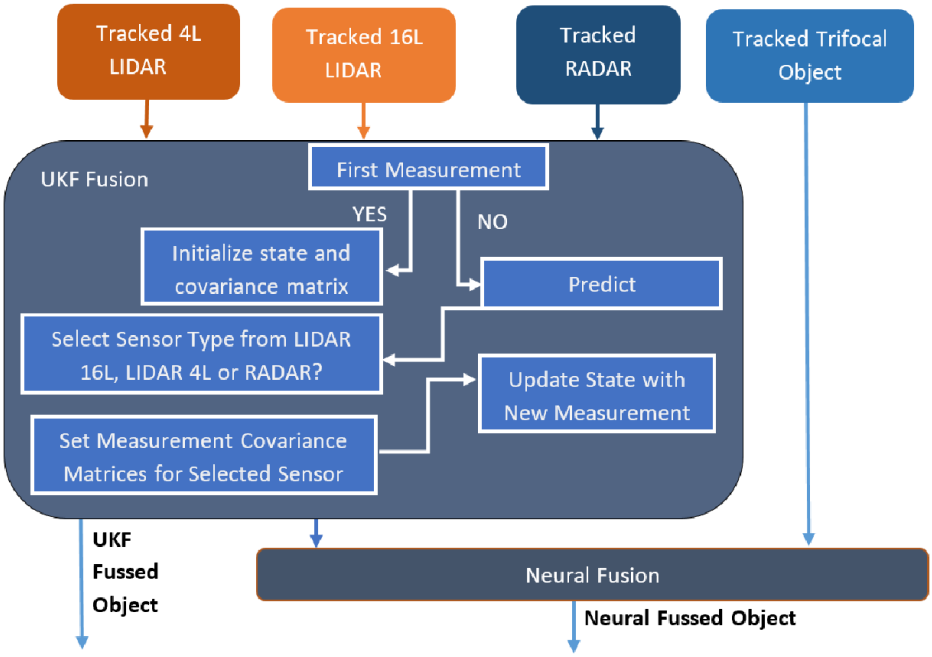

Figure 4.29 UKF and neural fusion architecture.

Figure 4.30 presents an example in which multiple detections corresponding to the same physical object are observed. In this visualization, the red rectangle denotes the trifocal camera detection, the yellow rectangle represents the LiDAR-based detection, and the green rectangle indicates the RADAR-based position estimate. The white bounding box corresponds to the object obtained through LiDAR–

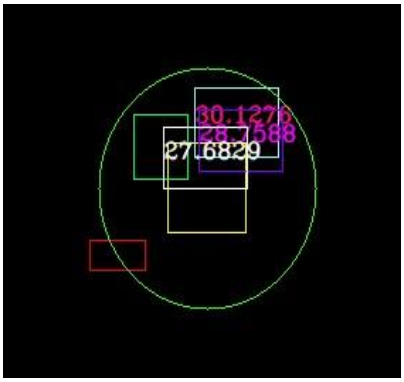

Figure 4.30. The result of the UKF and neural fusion is depicted in white and purple.

RADAR fusion, the cyan square with reddish text denotes the precise target position provided by the GPS reference system, and the purple square with pink text represents the output of the neural network–based fusion model. As can be observed from the figure, the final fused object position obtained through the proposed multimodal fusion approach is closer to the ground truth target position than the estimate produced solely by the LiDAR–RADAR fusion module.

The fused object is estimated to be located at a distance of 27.68 m, while the ground truth target position is measured at 30.12 m. After applying the neural proposed fusion approach, the resulting fused object position is obtained at 28.75 m.

4.3.3.3.2 Validation Procedure

Because object classification results may be affected by errors and uncertainties, it is necessary to validate the semantic class of all fused object hypotheses. In order to ensure a reliable validation procedure, information from at least three sensing modalities is required. The three modalities employed for this purpose are as follows: the classification output provided by the trifocal camera, a second semantic vote obtained by projecting the fused object onto the semantic segmentation map generated by applying the ERF neural network to the undistorted color image, and a third source of information derived from the fused object itself, where the validated class is stored in the form of a histogram embedded within the object representation in order to enable temporal tracking across consecutive frames.

During the initial stages of object existence, namely until an object transitions from the initialization phase to the updated (isUpdated) state, the third class estimate is obtained by projecting the neural network–fused object onto the semantic segmentation image and selecting the dominant semantic class within the corresponding projected region.

Finally, the dominant semantic category is determined by selecting the class corresponding to the index at which the maximum membership value is stored. In an analogous manner, the class associated with the fused object is obtained by identifying the index that contains the maximum value within the corresponding class histogram.

Subsequently, a comparison is performed among the three estimated class labels in order to determine whether a consistent match exists.

Figure 4.31. Example of successful validation. The super-sensor object is projected onto the RGB image in the left image. In the right image, the same object is projected onto the semantic segmentation image. The semantic classes' match is represented using green color.

For evaluation and visualization purposes, a green bounding box is displayed on the semantic segmentation image when at least two of the three class estimates are in agreement, as illustrated in Figure 4.31. Conversely, a red bounding box is rendered when no agreement is observed among the class estimates, as shown in Figure 4.32.

In Figure 4.32, the trifocal camera fails to correctly identify the vehicle class and assigns the label *unknown*.

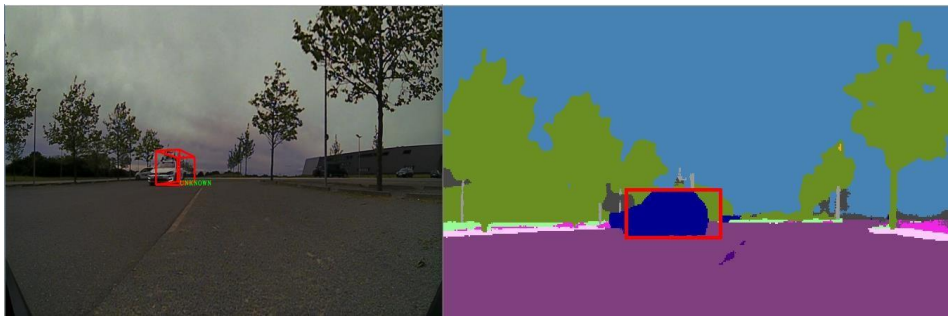

Figure 4.32. Example of mismatch in the validation procedure. On the left-hand side, we observe the projected fused object on the RGB image with the semantic class UNKNOWN. In the right image, the same object is projected onto the semantic class image in a region with the dominant class car. The class mismatch is represented using red color.

In the corresponding semantic segmentation image, the dominant detected region is classified as car. As a result, the class labels do not match and a red bounding box is displayed. Conversely, when the class labels obtained from two sensing modalities are consistent, a green bounding box is rendered over the region of interest, as illustrated in Figure 4.31.

In order to further increase the robustness of the proposed validation framework, a double verification procedure is applied to each fused object hypothesis. Specifically, during the validation stage, both the UKF-based fusion result and the neural network–based fusion result are evaluated. This strategy is motivated by the fact that, under certain conditions and due to sensor-related uncertainties or measurement errors, one of the fusion approaches may produce suboptimal estimates. By jointly considering the outputs of both fusion mechanisms, the final validation decision becomes more reliable and robust. Both fusion hypotheses are therefore taken into account. If, for each hypothesis, the estimated class matches the class obtained from the semantic segmentation image, the result is classified as a successful detection (*hit*) and the object is validated. Otherwise, the outcome is labeled as a missed detection (*miss*), and the object is not validated.

4.3.3.3.3 Evaluation of the object level fusion approach

In this section, we evaluate the results of the proposed object level fusion solution with respect to the position given by a high-precision GPS placed on a tracked target vehicle. The system on which we have tested our method has an Intel i7-4770 K CPU with 3.5 GHz frequency and 8 GB of RAM memory. This section is split into two subsections: Section 4.1 presents the characteristics of the sensors used, and Section 4.2 presents the results of our solutions in different scenarios.

4.3.3.3.3.1 Experimental Setup

The main technical specifications of the GPS system employed to acquire positional information from the tracked vehicle are summarized in Table 4.3. The vehicle on which the GPS receiver is installed is referred to as the target vehicle, whereas the vehicle equipped with the perception sensors is denoted as the reference vehicle or ego vehicle. In order to evaluate the accuracy of the estimated object positions, the nearest neighbor to the target vehicle is selected and used as a reference to assess whether the predicted cuboid

positions are correctly determined. Throughout all experimental evaluations, the virtual image space used for object projection is defined with an inverted horizontal (y) axis. The average processing time of the proposed solution is approximately 90 ms per frame, and the resulting mean positional error is 0.8 m. The main technical characteristics of each sensor mounted on the ego vehicle are detailed in Tables 4.3–4.7.

In addition, for accurate ego vehicle localization, the GPS system provides measurements of the yaw (heading), pitch, and roll angles, which jointly define the complete three-dimensional attitude of the vehicle. These orientation angles are commonly applied as three successive rotations, namely heading, followed by pitch, and subsequently roll, in order to transform measurement vectors between the ego vehicle coordinate frame and the navigation (Earth) coordinate frame. The navigation coordinate frame corresponds to a local Earth-referenced system oriented along the north, east, and down axes. The axis orientations associated with zero values of heading, pitch, and roll are specified in Table 4.2. In this table, the column indices correspond to the following parameters: I. Standard, II. Positioning mode, III. Position accuracy, IV. Velocity accuracy, V. Roll and pitch accuracy (1σ), VI. Heading accuracy (1σ), VII. Track angle accuracy (1σ), VIII. Slip angle accuracy (1σ), IX. Axis designation, X. Earth reference axis (direction), and XI. Ego vehicle axis (direction).

If V_e denotes a vector expressed in the ego vehicle coordinate frame and V_n denotes the corresponding vector expressed in the navigation coordinate frame, the relationship between the two representations is defined by the heading angle ψ, pitch angle θ, and roll angle ϕ, as expressed in Equation (4.37).

$$V_n = \begin{pmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\varphi) & -\sin(\varphi) \\ 0 & \sin(\varphi) & \cos(\varphi) \end{pmatrix} V_e \quad (4.37)$$

The technical specifications of the sixteen-layer LiDAR sensor employed for object detection are summarized in Table 4.3, while the main characteristics of the four-layer LiDAR sensor are presented in Table 4.4. The principal technical specifications of the RADAR sensor employed in

the experimental setup are presented in Table 4.5, while the characteristics and functional capabilities of the trifocal camera are detailed in Table 4.6.

Table 4.2 Characteristics of the GPS system.

I	II	III	IV	V	VI	VII	VIII	IX	X	XI
RT3003	L1, L2	0.01 m	0.05 Km/h	0.03◦	0.1◦	0.07◦	0.15◦	X, Y, Z	North, East	Down Forward, Right, Down

Table 4.3 16L LIDAR CHARACTERISTICS

Feature
Time of flight distance measurement with calibrated reflectivities
16 channels
Measurement range up to 100m
Accuracy +/- 3cm
Dual returns
Field of view (vertical): 30° (+15° to -15°)
Angular resolution (vertical): 2°
Angular resolution (horizontal/azimuth): 0.1° - 0.4°
Rotation rate: 5 - 20 Hz

Table 4.4 4L LIDAR CHARACTERISTICS

Feature
Time of flight distance measurement with calibrated reflectivities
4 channels
Measurement range up to 327m
Accuracy +/- 3cm
Dual returns
Field of view (vertical): 3.2°
Angular resolution (vertical): 4 Layers @ 0.8°
Field of view (horizontal/azimuth): 145°
Angular resolution (horizontal/azimuth): 0.25° Rotation rate: 12.5 Hz

Table 4.5 RADAR CHARACTERISTICS

Feature
Operation Frequency 77 Ghz
Distance Measurement 0.25 – 250m
Accuracy for distance measurement +- 2m
Speed Measurement -400 +200kph
Sensitivity 0.1 kph
Field of view in (horizontal/ azimuth): -9° +9° (Far range scan) -75° +75° (Near range scan)
Field of view in elevation 20°

Table 4.6 TRIFOCAL CAMERA CHARACTERISTICS

Feature
3 Cameras with fields of view: 34, 46, 150 degrees
The sensor provides data for the following functions: • Road infrastructure detection • Object classification (cars, trucks, pedestrians etc.) • 3D terrain perception (curb stones, generic road boundaries) • Free space grid construction
Resolution: 1280x960
Upgrade Rate 33ms
Color Grayscale
Bits per pixel 12

4.3.3.3.3.2 Experiments and Validation

The trifocal camera data association algorithm was evaluated through multiple experimental scenarios involving the ego vehicle, equipped with the previously described sensor suite, and a designated target vehicle. Representative scenarios included the following cases:

- the ego vehicle followed the target vehicle under various road conditions, including straight driving segments and curved trajectories;

- the ego vehicle was followed by the target vehicle, after which the target vehicle performed an overtaking maneuver.
- the target vehicle approached the ego vehicle from the front while traveling in an adjacent lane

Figure 4.33. Speed chart(top) and position chart(bottom) of a trifocal object association scenario

All experimental scenarios were executed using varying velocity profiles for both the ego vehicle and the target vehicle. The plots presented in Figures 4.33 and 4.34 illustrate the position of the trifocal camera detection before and after the association process, relative to the ground truth position of the target vehicle. A nearest-neighbor matching strategy was applied between the target vehicle position and the enhanced trifocal object position in order to quantify the accuracy of the trifocal data association procedure. It should be noted that the achievable association accuracy is inherently constrained by the positional accuracy of the reference object to which the trifocal detection is associated, which, in this case, corresponds to the LiDAR-based object detection.

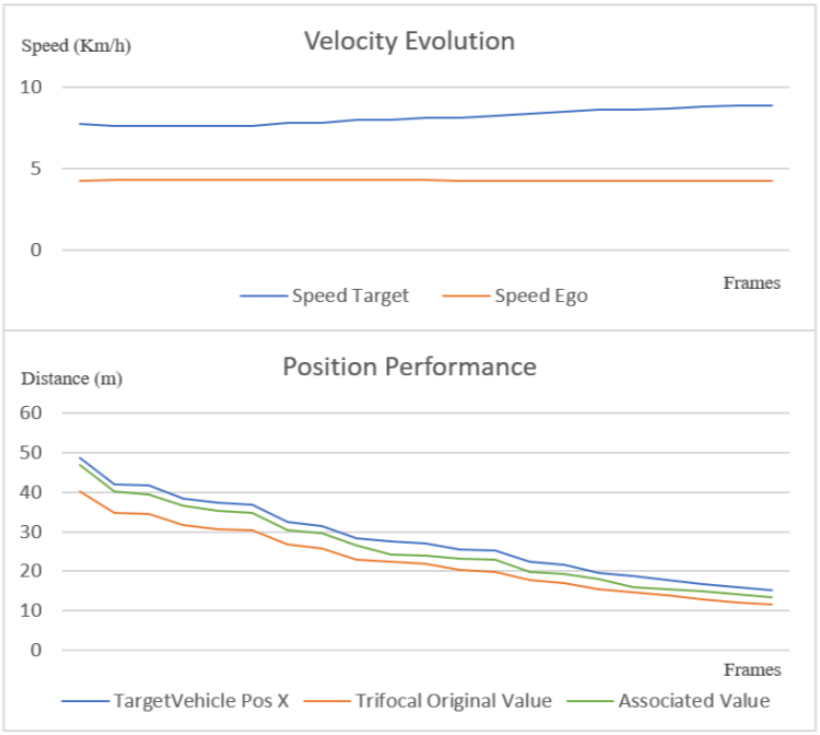

Figure 4.34. Another scenario that demonstrates the performance of the object association

In Figures 4.33–4.35, the horizontal axis represents the number of frames over which the association process is evaluated, while the vertical axis indicates the distance to the target vehicle. In order to provide additional insight into the relative motion of the ego and target vehicles, the upper sections of Figures 4.33 and 4.34 illustrate the velocity profiles recorded over the evaluated frame sequences. In these velocity plots, the horizontal axis corresponds to the frame index, and the vertical axis represents the vehicle speed expressed in kilometers per hour.

In the scenario depicted in Figure 4.34, the target vehicle exhibits an acceleration phase, while the ego vehicle maintains an approximately constant velocity. The green curve represents the final position of the associated trifocal camera detection that is closest to the target vehicle, which is illustrated in blue. In Figure 4.35, a different scenario is presented, in which the target and ego vehicles initially travel at similar speeds, followed by a phase in which the target vehicle increases its velocity. The corresponding position plot indicates that the corrected

trifocal object position is closer to the ground truth reference than the original, uncorrected measurement.

The single-layer perceptron model was trained using a dataset comprising 800 data samples and subsequently evaluated on an independent test set consisting of 465 samples, achieving a classification accuracy of 94%. For the evaluation of the proposed sensor fusion framework, the estimated object positions obtained from the fusion module were compared against the ground truth positions provided by the high-precision GPS system. Representative position estimates extracted from different frames and experimental scenarios, selected at random, are reported in Table 4.7.

Figure 4.35 illustrates a graphical comparison between the fused position estimates and the corresponding ground truth values. The plots presented in Figure 4.35 are generated using a dataset of more than 900 frames, recorded during a scenario in which the ego vehicle follows the target vehicle under varying driving conditions, including straight segments, turning maneuvers, and changes in vehicle speed. As can be observed, both fusion approaches produce position estimates that remain in close proximity to the ground truth target trajectory and accurately follow the motion pattern of the target vehicle.

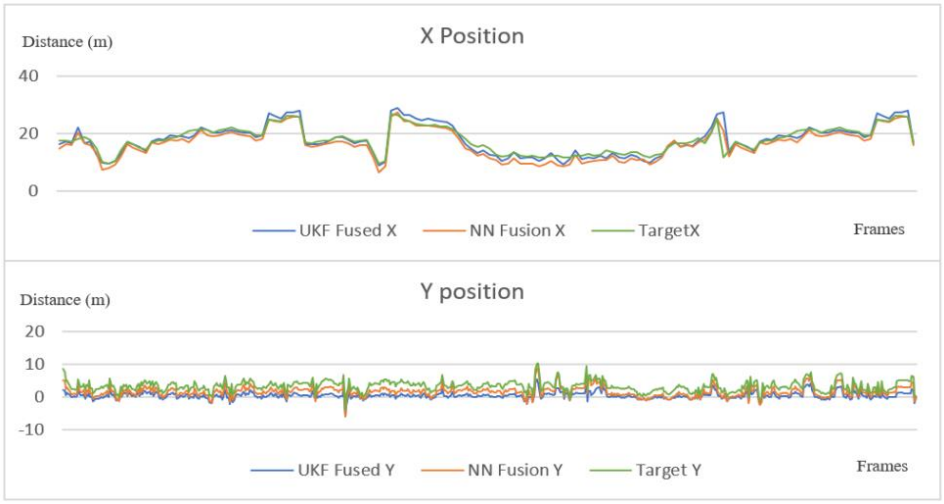

Figure 4.35. Position performance on the x and y axes of the proposed sensor fusion methods and the target vehicle.

Sensor-related measurement uncertainties also contribute to the small discrepancies observed along the $x$and $y$axes between the ground truth positions and the estimated results. The trifocal camera data association mechanism, together with the tracking and sensor fusion strategies, plays a significant role in stabilizing object state parameters, with particular emphasis on improving the robustness and consistency of object position estimates. Validation of the sensor fusion output is performed by comparing the semantic class assigned to each fused object with the dominant semantic class extracted from the region of interest corresponding to the projection of the fused object onto the semantic segmentation image.

Figure 4.36 illustrates two representative scenarios in which a vehicle is tracked in different environmental contexts. In these examples, only fused and validated object hypotheses are projected onto the intensity image. In both cases, the target vehicle is successfully tracked, and redundant as well as complementary information originating from multiple sensors is exploited to estimate key object attributes, including position, velocity, and semantic class, among other relevant parameters.

Table 4.7 Different fusion position samples

Fused Object UKF		**Fused Object NN**		**GPS Ground Truth**	
x	**y**	**x**	**Y**	**x**	**y**
15.46	0.54	13.87	0.8	16.01	1.3
17.04	2.57	16.95	1.87	16.71	1.44
24.37	0.65	23.36	1.62	25.29	1.56
15.16	0.04	15.24	0.6	18.01	1.03
9.79	1.00	8.1	0.71	10.03	1.4

In Figure 4.36a, only the UKF-based fusion module is enabled and visualized. In Figure 4.36b, the same validation and stabilization framework is executed using both fusion strategies in a scenario characterized by significant environmental clutter. In this case, cuboidal detections originating from the trifocal camera, the sixteen-layer LiDAR, and the four-layer LiDAR are observed, many of which correspond to noise or small, irrelevant objects. The same color coding scheme used to represent sensor-specific detections, as introduced in Section 3.6, is employed for consistency.

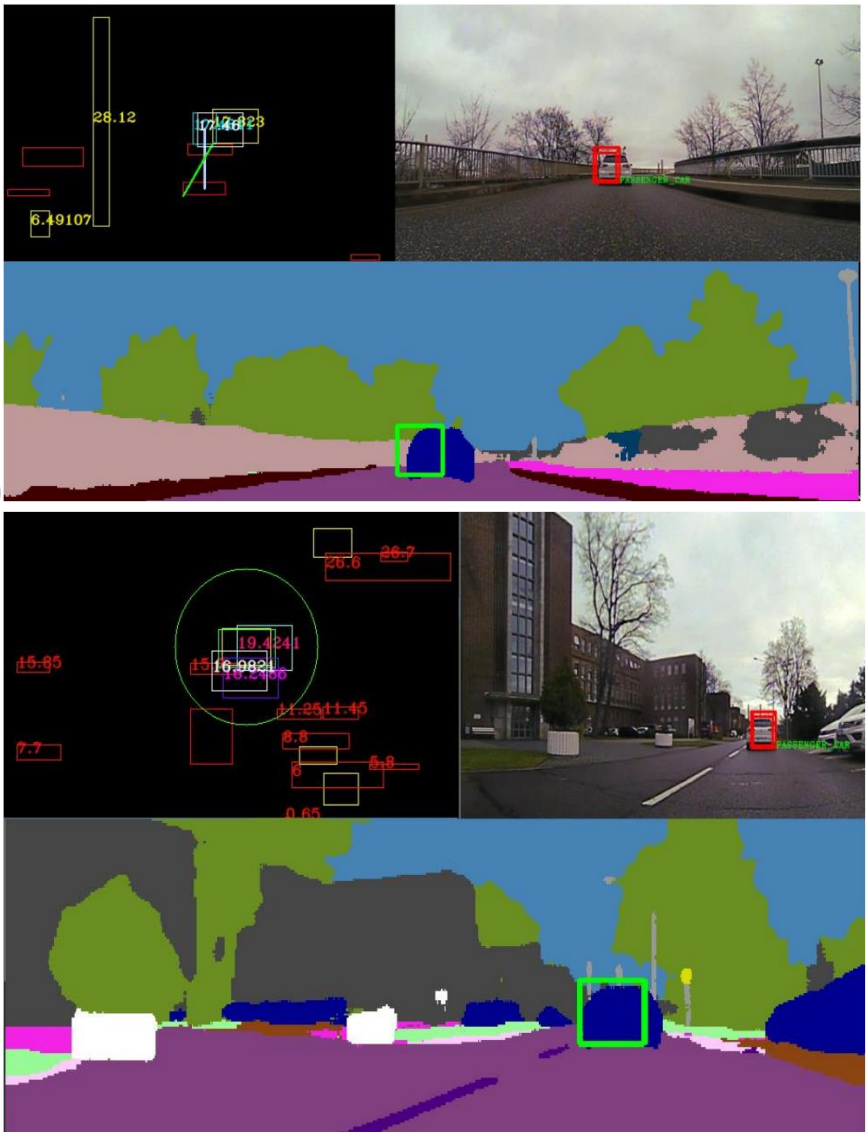

Figure 4.36. (a) Scenario where a vehicle is on a bridge with few surrounding objects (b) Scenarios with a fused object in heavy clutter.

In the upper-left subfigure of Figure 4.36a, the bird's-eye-view virtual representation of the scene is shown, in which all detected objects are projected. The upper-right subfigure presents the RGB image with the validated and stabilized super-sensor object overlaid. The lower

subfigure illustrates the semantic segmentation image, where the validated super-sensor object is highlighted using a green bounding box.

Figure 4.36b depicts the execution of the proposed algorithm under heavy clutter conditions. The upper-left subfigure displays the bird's-eye-view virtual image of the scene, while the upper-right subfigure shows the RGB image with the validated super-sensor object projected onto it. The lower subfigure presents the semantic segmentation image with the validated fused object overlaid.

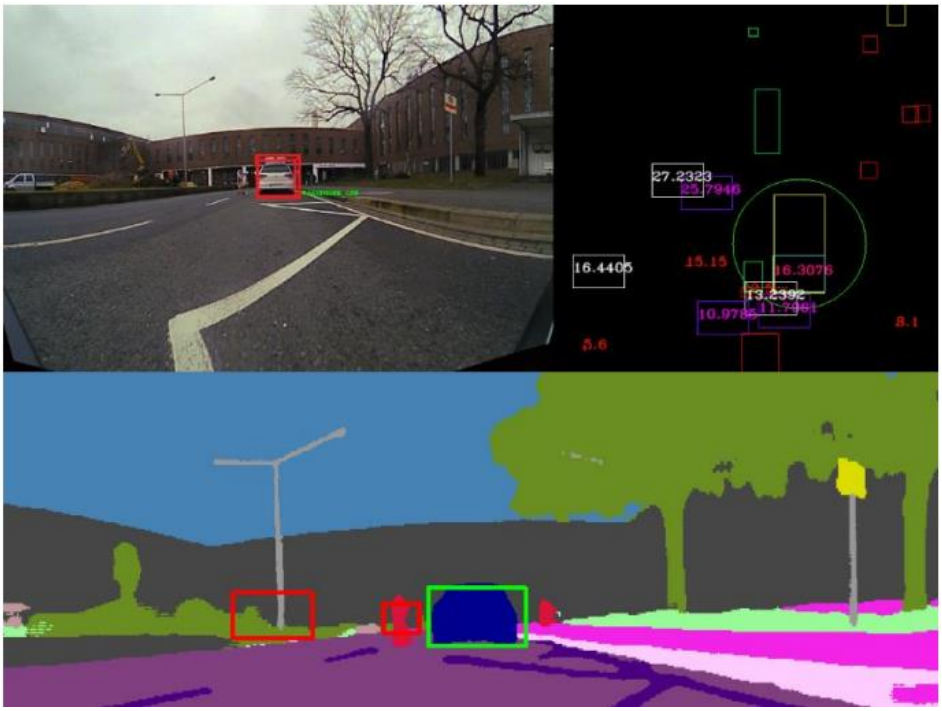

Figure 4.37. Another scene where the fusion method is tested in the presence of multiple traffic objects.

Figure 4.37 further illustrates the validation of multiple fused object hypotheses through projection onto the semantic segmentation image. In this scenario, several fused objects are observed that do not satisfy the validation criteria and are therefore rejected. The target vehicle has made a sudden slight right; even so, the data fused from multiple sensors are able to capture the position of the car and validate its semantic class.

In the top right of Figure 4.37, the birds-eye view image is depicted. The bottom image shows the semantic image with the validated super-sensor objects drawn in green and objects that are not validated drawn in red. The top-left image illustrates the RGB image, where the super-sensor objects are displayed. The Neural fusion is kept as a secondary validation when there is a full mismatching in previous step and the comparison is repeated with the neural fused object. This phenomenon can appear in moving objects due to projection errors. Neural fusion is also used when the semantic class from the histogram of the UKF fused object does not contain any values, that is, the fused object is not yet stable (in this case, the neural fusion object is also projected into the semantic segmentation image to have the third vote). The contents of this chapter can be found in the publications [343] and [351].

4.3.4 Mobile Multi Sensorial Platform for Development and Testing

Primarily as a consequence of the fact that, within the research projects in which the present work was conducted, the international partners involved in the collaborations provided incomplete or insufficient datasets for the effective development and experimental validation of the proposed algorithms, the decision was made to design and implement a dedicated multi-sensor platform equipped with automotive-grade sensors. The developed platform is mobile and has been conceived to enable the testing and evaluation of perception algorithms in scenarios that reproduce, as closely as possible, real-world conditions encountered in autonomous driving applications. The main hardware components of the proposed sensor platform are illustrated in Figure 4.38, while a detailed description of their functionality and role within the system is provided in Table 4.8. It should also be noted that the sensors employed in the platform, as well as all auxiliary hardware components required for its construction, were acquired using personal financial resources. For the purpose of environment perception, multiple sensors are integrated and mounted on the sensor head of the platform. Two RGB cameras configured as a stereo pair are employed in order to obtain dense three-dimensional reconstructions of the surrounding environment. These cameras are synchronized using a dedicated hardware-based synchronization mechanism. In addition, a thermal camera is incorporated into the

system, enabling robust operation under adverse weather conditions as well as during nighttime scenarios, in which conventional RGB cameras may fail to provide reliable environmental perception. For the purpose of environment perception, multiple sensors are integrated and mounted on the sensor head of the platform.

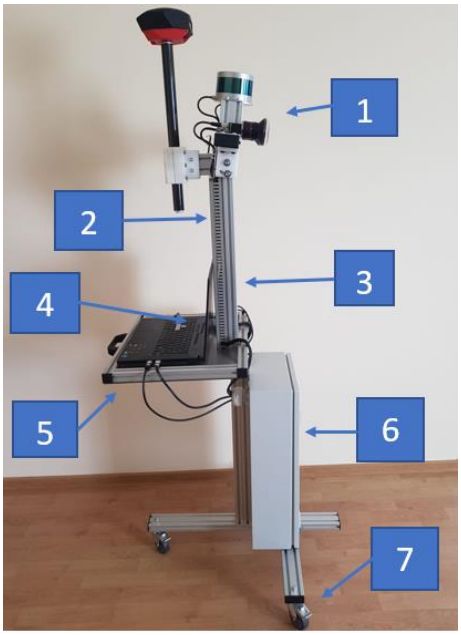

Figure 4.38. Sensorial platform with its main components

Table 4.8 Component description

Component Number	Meaning
1	Sensorial head which contain main sensors used for environment perception.
2	Plastic cable tray for masking the cables which connect the sensors to the processing module.
3	Aluminium profile rig used for mounting the sensor rig.
4	Computing platform.
5	Custom aluminium table with handle designed to hold the computing platform.
6	Electrical box used for holding the electrical equipment, the lidar interface box, external batteries etc.
7	Wheels and custom aluminium chassis.

Two RGB cameras configured as a stereo pair are employed in order to obtain dense three-dimensional reconstructions of the surrounding environment. These cameras are synchronized using a dedicated hardware-based synchronization mechanism. In addition, a thermal camera is incorporated into the system, enabling robust operation under adverse weather conditions as well as during nighttime scenarios, in which conventional RGB cameras may fail to provide reliable environmental perception.

A VLP-16 LiDAR sensor is utilized to acquire three-dimensional information about the environment based on the time-of-flight measurement principle. The LiDAR sensor represents a critical component of the autonomous driving perception pipeline, as it is capable of operating in conditions where vision-based sensors may degrade and provides accurate distance measurements to surrounding objects, along with detailed geometric information.

Furthermore, a Real-Time Kinematic (RTK) GPS system is integrated in order to obtain high-precision localization data, achieving positional accuracy on the order of 14 mm. A visual representation of the developed sensor rig is presented in Figure 4.39.

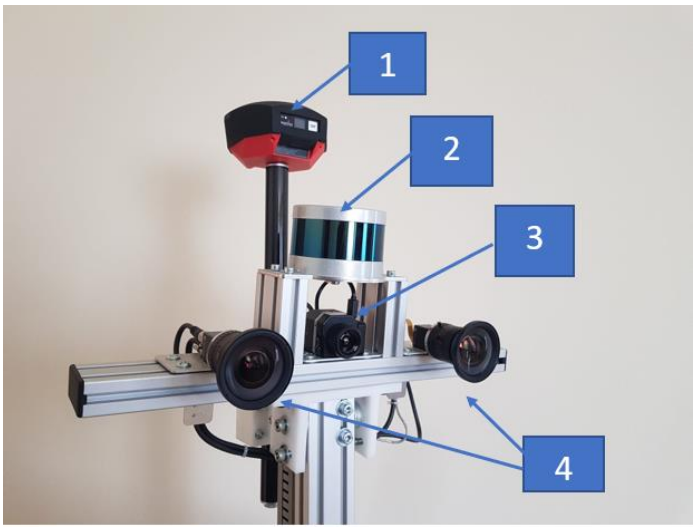

Figure 4.39. Sensor rig. The sensors presented in the image are the following: 1 - GPS RTK, 2 – VLP 16 LiDAR, 3 – Thermal Camera, 4 – Stereo system.

Chapter 5. Conclusions

5.1 Summary of the Presented Work

The present monograph provides both a structured synthesis of representative state-of-the-art approaches in multimodal perception for autonomous systems and a coherent set of original contributions that extend and refine these approaches. The work is organized around three fundamental components of the perception pipeline—depth estimation, multi-object tracking, and multi-sensor fusion—each preceded by a systematic review of the principal algorithmic families and methodological paradigms reported in the literature.

Within each addressed domain, the main algorithmic directions have been examined and comparatively analyzed, including feature-engineered methods, model-based estimation techniques, probabilistic filtering approaches, and contemporary data-driven solutions based on deep learning. The review is structured along the processing stages specific to each task, thereby enabling a clear identification of methodological trends, performance trade-offs, and existing limitations. This analytical positioning of the state of the art provides the conceptual and technical framework necessary for motivating the proposed developments and for precisely situating the original contributions within the broader research landscape.

The first major contribution area concerns stereo-based depth reconstruction. The literature review covers classical local, semi-global, and global stereo matching algorithms, diverse cost computation strategies, aggregation mechanisms, and optimization schemes, as well as recent learning-based approaches for both stereo and monocular depth estimation. Against this background, the proposed methodological enhancements introduce improvements at multiple stages of the stereo pipeline, with the objective of increasing robustness to real-world challenges while preserving low computational complexity and real-time capability. Furthermore, a fault-tolerant reconstruction framework is introduced, integrating stereo vision with monocular depth estimation in order to ensure operational continuity under sensor degradation. These contributions are explicitly positioned as extensions of local stereo paradigms, addressing their known sensitivity to perspective distortions and failure modes, while avoiding the high computational demands characteristic of fully global optimization or deep end-to-end architectures.

The second contribution area addresses multi-object tracking within a tracking-by-detection framework. The state-of-the-art analysis encompasses classical data association methods, Bayesian filtering techniques, multiple motion model strategies, and recent hybrid or deep learning–based tracking approaches. Particular attention is devoted to the challenges specific to three-dimensional LiDAR detections and two-dimensional thermal imagery. Building upon this foundation, novel data association mechanisms and tracking strategies are proposed, combining feature-engineered descriptors with data-driven representations. The integration of multiple motion models and the development of robust association functions are formulated as responses to well-documented limitations in handling motion and origin uncertainty. The proposed methodologies are therefore positioned as hybrid solutions that aim to retain the interpretability and efficiency of model-based tracking while enhancing adaptability through learned components.

The third contribution area focuses on sensor fusion for automotive perception. The review examines established fusion architectures, levels of fusion, registration techniques, and probabilistic fusion frameworks, as well as the JDL model and contemporary multimodal perception systems. Within this context, the original contributions introduce architectural and methodological refinements spanning spatio-temporal alignment, multi-sensor data association, and high-level object fusion. The proposed fusion strategies are explicitly designed to address synchronization inaccuracies, heterogeneous sensing characteristics, and semantic inconsistencies identified in prior work. The resulting architecture is conceived as a modular and fault-tolerant framework, capable of integrating complementary sensing modalities without over-reliance on any single sensor.

In addition to the algorithmic developments, two novel datasets are introduced and documented, and a dedicated multi-sensor hardware platform equipped with an integrated perception system is developed to enable systematic real-time validation. These infrastructural contributions are intended to support reproducibility, facilitate quantitative evaluation against state-of-the-art benchmarks, and ensure that the proposed methods are assessed under realistic operating conditions.

Across all contribution areas, particular emphasis is placed on computational efficiency, memory footprint, and energy consumption. The proposed solutions are explicitly designed for compatibility with

embedded hardware platforms and real-time deployment, thereby addressing a recurring limitation of many high-performance state-of-the-art approaches that rely heavily on GPU-intensive deep learning architectures.

Overall, this monograph establishes a coherent bridge between a structured review of the current state of knowledge and a set of methodologically grounded innovations. By systematically analyzing existing algorithmic paradigms and explicitly positioning the proposed contributions within this context, the work advances robust, scalable, and resource-aware perception methodologies for autonomous systems.

5.2 Closing Words

Although current artificial intelligence methods reported in the literature, as well as those presented in this work, do not yet exhibit true general intelligence, they represent an important technological foundation for future developments in this direction. Intelligence may be interpreted as the capability of an autonomous agent to construct and adapt internal models that support goal-oriented behavior, rather than relying exclusively on externally designed pattern-based predictors, which primarily perform advanced information processing. Recent advances in large-scale learning systems, such as ChatGPT [348, 354], provide early indications of increasingly sophisticated reasoning and generalization capabilities. With continued progress, current limitations related to data availability, computational efficiency, and energy consumption are expected to be progressively reduced.

From a broader societal perspective, contemporary technological trends indicate the onset of a major transformation, in which artificial intelligence systems may exceed the collective perceptual and analytical capacities of human operators. This evolution presents significant opportunities, enabling the development of new classes of applications, including both currently envisioned technologies and solutions that remain yet to be conceptualized. Such advances have the potential to support societal progress by facilitating innovation in transportation systems, healthcare technologies, and large-scale infrastructure optimization, thereby contributing to improved quality of life and expanded access to essential resources.

At the same time, the emergence of artificial general intelligence introduces substantial risks associated with the concentration of

technological power. Even in the absence of malicious intent, uncontrolled deployment of such systems may result in unintended consequences with far-reaching societal implications. These risks may manifest through overt authoritarian mechanisms, reminiscent of dystopian scenarios such as those described in Orwell's 1984, or through more subtle forms of technological dependency and behavioral conditioning, analogous to the societal dynamics portrayed in Huxley's Brave New World, in which autonomy and critical reasoning capacity are progressively diminished.

In this context, the adoption of explainable or hybrid artificial intelligence paradigms that integrate data-driven learning with model-based and physics-informed approaches represents a sustainable development direction. Such architectures offer improved interpretability, enhanced robustness to previously unseen operational conditions, increased fault tolerance, and reduced computational and environmental cost when compared to large-scale purely data-driven solutions. Consequently, the balanced integration of learning-based and model-driven methodologies constitutes a technically sound and socially responsible pathway for the future development of intelligent systems.

LIST OF FIGURES

LIST OF TABLES

REFERENCES

[1] K. Hao, “Training a single AI model can emit as much carbon as five cars in their lifetimes,” MIT Technology Review, 07-Dec-2020. [Online]. Available: https://www.technologyreview.com/2019/06/06/239031/training-a-single-ai-model-can-emit-as-much-carbon-as-five-cars-in-their-lifetimes/. [Accessed: 31-Mar-2023].

[2] Richard Fütterer, Mathias Schellhorn, Gunther Notni, "Implementation of a multiview passive-stereo-imaging system with SoC technology," Proc. SPIE 11144, Photonics and Education in Measurement Science 2019, 111440Q (17 September 2019);

[3] A. Geiger, J. Ziegler and C. Stiller, "StereoScan: Dense 3d reconstruction in real-time," 2011 IEEE Intelligent Vehicles Symposium (IV), Baden-Baden, Germany, 2011, pp. 963-968, doi: 10.1109/IVS.2011.5940405.

[4] R. M. Haralick and L. G. Shapiro. Computer and robot vision. Addison-Wesley Longman Publishing Co., Inc., Boston, 1992.

[5] Hajar Sadeghi, Payman Moallem, and S. Amirhassn Monadjemi. Feature-based dense stereo matching using dynamic programming and color. International Journal of Information and Mathematical Sciences, 4(3):179–186, 2008

[6] Andrea Fusiello, Emanuele Trucco, and Alessandro Verri. A compact algorithm for rectification of stereo pairs. Machine Vision and Applications, 12:16–22, 2000

[7] D. Scharstein and R Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. IJCV, 47(1-3):7–42, 2002.

[8] Roy Shilkrot, David Millan Escriva, Mastering OpenCV 4 - Third Edition, Packt Publishing, December 2018, ISBN: 9781789533576.

[9] Bouguet, J.-Y. (2022). Camera Calibration Toolbox for Matlab (1.0). CaltechDATA. https://doi.org/10.22002/D1.20164.

[10] C. Vancea, S. Nedevschi, “Analysis of different image rectification approaches for binocular stereovision systems”, Proceedings of IEEE 2nd International Conference on Intelligent Computer Communication and Processing, vol. 1, pp. 135-142, September 2006.

[11] C. Vancea and S. Nedevschi, "LUT-based Image Rectification Module Implemented in FPGA," 2007 IEEE International Conference on Intelligent Computer Communication and Processing, Cluj-Napoca, Romania, 2007, pp. 147-154

[12] N. Navab, “Stereo Vision II: Dense Stereo Matching.” Available:http://campar.in.tum.de/twiki/pub/Chair/TeachingWs11Cv2/3D_CV2_WS_2011_Stereo.pdf

[13] S. T. Birchfield, „Chapter 10 Stereo” https://cecas.clemson.edu/~stb/ece847/internal/cvbook/ch13_stereo.pdf

[14] R. Hartley and A. Zisserman, Multiple View Geometry in Computer Vision, 2nd ed. New York, NY, USA: Cambridge University Press, 2003.
[15] U. R. Dhond and J. K. Aggarwal, "Structure from stereo-a review," in IEEE Transactions on Systems, Man, and Cybernetics, vol. 19, no. 6, pp. 1489-1510, Nov.-Dec. 1989, doi: 10.1109/21.44067.
[16] L. Qiuming, Z. Jingli, S. Yu, D. Xiao, Stereo matching and occlusion detection with integrity and illusion sensitivity, Pattern Recognition Letters, Volume 24, Issues 9–10, 2003, Pages 1143-1149, ISSN 0167-8655, https://doi.org/10.1016/S0167-8655(02)00284-2
[17] W. Ende, Z. Yalong, P. Liangyu, L. Yijun and W. Tianyao, "Stereo matching algorithm based on the combination of matching costs," 2017 IEEE 7th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER), 2017, pp. 1001-1004, doi: 10.1109/CYBER.2017.8446444.
[18] Radim Sara. Finding the largest unambiguous component of stereo matching. In European Conference on Computer Vision, volume 2, pages 900–914, 2002.
[19] Ke Zhu, Doktor-Ingenieurs genehmigten Dissertation, Technischen Universität München, Dense Stereo Matching with Robust Cost Functions and Confidence-based Surface Prior
[20] Sanni Siltanen, Theory and applications of marker-based augmented reality
[21] Stephen T Barnard and Martin A Fischler. “Computational stereo”. In: ACM Computing Surveys (CSUR) 14.4 (1982), pp. 553–572
[22] Heiko Hirschmuller and Daniel Scharstein. “Evaluation of cost functions for stereo matching”. In: 2007 IEEE Conference on Computer Vision and Pattern Recognition. IEEE. 2007, pp. 1–8.
[23] H. Baker and T. Binford. “Depth from Edge and Intensity Based Stereo”. In: IJCAI. 1981
[24] Douglas Arnold. “Local context in matching edges for stereo vision”. In: 1978.
[25] Takeo Kanade and Masatoshi Okutomi. “A stereo matching algorithm with an adaptive window: Theory and experiment”. In: Pattern Analysis and Machine Intelligence, IEEE Transactions on 16 (Oct. 1994), pp. 920 –932. doi: 10.1109/34.310690.
[26] Rostam Affendi Hamzah, Rosman Abd Rahim, and Zarina Mohd Noh. “Sum of Absolute Differences algorithm in stereo correspondence problem for stereo matching in computer vision application”. In: 2010 3rd International Conference on Computer Science and Information Technology. Vol. 1. 2010, pp. 652–657.
[27] Maged Marghany, M.R.B. Tahar, and Mazlan Hashim. “3D stereo reconstruction using sum square of difference matching algorithm”. In:

Scientific Research and Essays 6 (Dec. 2011), pp. 6404–6423. DOI: 10.5897/SRE11.1661.
[28] Gaojian Li. “Stereo matching using normalized cross-correlation in LogRGB space”. In: 2012 International Conference on Computer Vision in Remote Sensing. 2012, pp. 19–23.
[29] Ramin Zabih and John Woodβill. “Non-parametric local transforms for computing visual correspondence”. English. In: Computer Vision – ECCV 1994. Ed. by Jan-Olof Eklundh. Vol. 801. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 1994, pp. 151–158 ISBN: 978-3-540-57957-1. DOI: 10. 1007 / BFb0028345. URL: http://dx.doi.org/10.1007/BFb0028345.
[30] Christian Banz et al. “Real-time stereo vision system using semi-global matching disparity estimation: Architecture and FPGA-implementation”. In: International Conference on Embedded Computer Systems (SAMOS). Aug. 2010, pp. 93 –10
[31] Stan Birchfield and Carlo Tomasi. “Depth Discontinuities by Pixel to-Pixel Stereo”. English. In: International Journal of Computer Vision 35.3 (1999), pp. 269–293. ISSN: 0920-5691. DOI: 10.1023/ A: 1008160311296.
[32] R. Spangenberg, T. Langner, S. Adfeldt and R. Rojas, "Large scale Semi-Global Matching on the CPU," 2014 IEEE Intelligent Vehicles Symposium Proceedings, 2014, pp. 195-201, doi: 10.1109/IVS.2014.6856419.
[33] I. Haller and S. Nedevschi, "Design of Interpolation Functions for Subpixel-Accuracy Stereo-Vision Systems," Image Processing, IEEE Transactions on, vol. 21, pp. 889-898, 2012.
[34] M. P. Muresan, M. Negru and S. Nedevschi, "Improving local stereo algorithms using binary shifted windows, fusion and smoothness constraint," 2015 IEEE International Conference on Intelligent Computer Communication and Processing (ICCP), 2015, pp. 179-185, doi: 10.1109/ICCP.2015.7312626.
[35] Martin Humenberger et al. “A Fast Stereo Matching Algorithm Suitable for Embedded Real-time Systems”. In: Comput. Vis. Image Underst. 114.11 (Nov. 2010), pp. 1180–1202. ISSN: 1077-3142. DOI: 10. 1016 / j . cviu . 2010. 03. 012
[36] W. S. Fife and J. K. Archibald. “Improved Census Transforms for Resource-Optimized Stereo Vision”. In: IEEE Transactions on Circuits and Systems for Video Technology 23.1 (Jan. 2013), pp. 60–73. ISSN: 1051-8215. DOI: 10.1109/TCSVT.2012.2203197.
[37] Robert Spangenberg, Tobias Langner, and Raul Rojas. “Weighted Semi-Global Matching and Center-Symmetric Census Transform for Robust Driver Assistance”. English. In: Computer Analysis of Images and Patterns. Ed. by Richard Wilson et al. Vol. 8048. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2013, pp. 34–41. ISBN: 978-3-642-40245-6. DOI: 10.1007/978-3-642-40246-3_5.

[38] E. Rublee, V. Rabaud, K. Konolige and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," 2011 International Conference on Computer Vision, 2011, pp. 2564-2571, doi: 10.1109/ICCV.2011.6126544.

[39] E. Tola, V. Lepetit and P. Fua, "DAISY: An Efficient Dense Descriptor Applied to Wide-Baseline Stereo," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 32, no. 5, pp. 815-830, May 2010, doi: 10.1109/TPAMI.2009.77

[40] Mohd Saad Hamid, NurulFajar Abd Manap, Rostam Affendi Hamzah, Ahmad Fauzan Kadmin, Stereo matching algorithm based on deep learning: A survey, Journal of King Saud University - Computer and Information Sciences, 2020, ISSN 1319-1578, https://doi.org/10.1016/j.jksuci.2020.08.011.

[41] J. Zbontar and Y. LeCun. "Computing the stereo matching cost with a convolutional neural network". In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). June 2015, pp. 1592–1599. ĉĚĎ: 10.1109/CVPR.2015.7298767

[42] S. Zagoruyko and N. Komodakis. Learning to compare image patches via convolutional neural networks. In CVPR, 2015.

[43] C. Guo, D. Chen and Z. Huang, "Learning Efficient Stereo Matching Network With Depth Discontinuity Aware Super-Resolution," in IEEE Access, vol. 7, pp. 159712-159723, 2019, doi: 10.1109/ACCESS.2019.2950924

[44] V. Miclea and S. Nedevschi, "Optimizing Census-based Semi Global Matching by genetic algorithms," 2016 IEEE 12th International Conference on Intelligent Computer Communication and Processing (ICCP), 2016, pp. 193-198, doi: 10.1109/ICCP.2016.7737146

[45] W. S. Fife and J. K. Archibald. "Improved Census Transforms for Resource-Optimized Stereo Vision". In: IEEE Transactions on Circuits and Systems for Video Technology 23.1 (Jan. 2013), pp. 60– 73. ISSN: 1051-8215. DOI: 10.1109/TCSVT.2012.2203197

[46] Xing Mei et al. "On building an accurate stereo matching system on graphics hardware". In: ICCV Workshops. IEEE, 2011, pp. 467– 474. ISBN: 978-1-4673-0062-9.

[47] N. Einecke and J. Eggert, "Block-matching stereo with relaxed fronto-parallel assumption," 2014 IEEE Intelligent Vehicles Symposium Proceedings, 2014, pp. 700-705, doi: 10.1109/IVS.2014.6856414.

[48] N. Einecke and J. Eggert, "A multi-block-matching approach for stereo," 2015 IEEE Intelligent Vehicles Symposium (IV), 2015, pp. 585-592, doi: 10.1109/IVS.2015.7225748.

[49] K.-J. Yoon and I.-S. Kweon. Locally adaptive support-weight approach for visual correspondence search. In IEEE Conference on Computer Vision and Pattern Recognition, 924- 931, 2005

[50] Andrey Kuzmin, Dmitry Mikushin, and Victor S. Lempitsky. “End to-end Learning of Cost-Volume Aggregation for Real-time Dense Stereo”. In: CoRR abs/1611.05689 (2016). arXiv: 1611. 05689.
[51] Vladimir Kolmogorov and Ramin Zabih. Computing Visual Correspondence with Occlusions via Graph Cuts. Tech. rep. Ithaca, NY, USA, 2001.
[52] Y. Boykov, O. Veksler, and R. Zabih. “Fast approximate energy minimization via graph cuts”. In: IEEE Transactions on Pattern Analysis and Machine Intelligence 23.11 (2001), pp. 1222–1239
[53] Jian Sun, Nan-Ning Zheng, and Heung-Yeung Shum. “Stereo Matching Using Belief Propagation”. In: IEEE Trans. Pattern Anal. Mach. Intell. 25.7 (July 2003), pp. 787–800. ISSN: 0162-8828. DOI: 10. 1109 / TPAMI. 2003 . 1206509.
[54] R. Ranftl et al. “Pushing the limits of stereo using variational stereo estimation”. In: Intelligent Vehicles Symposium (IV), 2012 IEEE. June 2012, pp. 401–407. DOI: 10.1109/IVS.2012.6232171.
[55] H. Hirschmuller. “Stereo Processing by Semiglobal Matching and Mutual Information”. In: Pattern Analysis and Machine Intelligence, IEEE Transactions on 30.2 (Feb. 2008), pp. 328–341. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2007.1166.
[56] C. Banz, P. Pirsch, and H. Blume. “Evaluation of Penalty Functions for Semi-Global Matching Cost Aggregation”. In: ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences (July 2012), pp. 1–6. DOI: 10 . 5194 / isprsarchives-XXXIX-B3-1-2012
[57] Akihito Seki and Marc Pollefeys. “SGM-Nets: Semi-Global Matching With Neural Networks”. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). July 2017.
[58] R. Spangenberg et al. “Large scale Semi-Global Matching on the CPU”. In: Intelligent Vehicles Symposium Proceedings, 2014 IEEE. June 2014, pp. 195–201. doi: 10.1109/IVS.2014.6856419.
[59] I. Haller and S. Nedevschi. “GPU optimization of the SGM stereo algorithm”. In: Intelligent Computer Communication and Processing (ICCP), 2010 IEEE International Conference on. Aug. 2010, pp. 197–202. DOI: 10.1109/ICCP.2010.5606438
[60] S. B. Pollard, J. E. W. Mayhew, and J. P. Frisby. Pmf: A stereo correspondence algorithm using a disparity gradient limit. Perception, 14:449–470, 1985.
[61] S.Z. Li. Inexact matching of 3d surfaces. In Technical Report VSSP-TR-3/90. Vision Speech & Signal Processing, Dept. Electronic and Electrical Engineering, University of Surrey, UK, February 1990
[62] T. Huang, G. Yang, and G. Tang. “A fast two-dimensional median filtering algorithm”. In: Acoustics, Speech and Signal Processing, IEEE

Transactions on 27 (1979), pp. 13–18. ISSN: 0096-3518. DOI: 10. 1109/TASSP.1979.1163188.
[63] C. Tomasi and R. Manduchi. “Bilateral Filtering for Gray and Color Images”. In: Proceedings of the Sixth International Conference on Computer Vision. ICCV ’98. Washington, DC, USA: IEEE Computer Society, 1998, pp. 839–. ISBN: 81-7319-221-9.
[64] Kaiming He, Jian Sun, and Xiaoou Tang. “Guided Image Filtering”. In: Proceedings of the 11th European Conference on Computer Vision: Part I. ECCV’10. Heraklion, Crete, Greece: Springer-Verlag, 2010, pp. 1–14. ISBN: 3-642-15548-0, 978-3-642-15548-2.
[65] Sun, X., Mei, X., Jiao, S., Zhou, M., and Wang, H. (2011). Stereo matching with reliable disparity propagation. In 3DIMPVT, pages 132–139
[66] Perona, P. and Malik, J. (1990). Scale-space and edge detection using anisotropic diffusion. IEEE Trans. Pattern Anal. Mach. Intell., 12(7):629–639.
[67] Zhelun Shen, Yuchao Dai, and Zhibo Rao. Msmd-net: Deep stereo matching with multi-scale and multi-dimension cost volume. arXiv preprint arXiv:2006.12797, 2020
[68] Li Xu et al. “Deep Edge-aware Filters”. In: Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37. ICML’15. Lille, France: JMLR.org, 2015, pp. 1669–1678.
[69] J. T. Barron and B. Poole "The fast bilateral solver" Proc. 14th European Conference on Computer Vision (ECCV). Amsterdam NETHERLANDS pp. 617-632 Oct. 2016.
[70] Y. Chen, C. Cai, and K.-K. Ma, “Stereoscopic video error concealment for missing frame recovery using disparity-based frame difference projection,” in 2009 16th IEEE International Conference on Image Processing (ICIP), Nov 2009, pp. 4289–4292.
[71] T. Y. Chung, S. Sull, and C. S. Kim, “Frame loss concealment for stereoscopic video plus depth sequences,” IEEE Transactions on Consumer Electronics, vol. 57, no. 3, pp. 1336–1344, August 2011
[72] V. Miclea, L. Miclea and S. Nedevschi, "Real-time Stereo Reconstruction Failure Detection and Correction using Deep Learning," 2018 21st International Conference on Intelligent Transportation Systems (ITSC), 2018, pp. 1095-1102, doi: 10.1109/ITSC.2018.8569928
[73] B. K. Horn, “Shape from shading: A method for obtaining the shape of a smooth opaque object from one view,” 1970
[74] N. Kong and M. J. Black, “Intrinsic depth: Improving depth transfer with intrinsic images,” in ICCV, 2015, pp. 3514–3522
[75] A. Torralba and A. Oliva, “Depth estimation from image structure,” IEEE TPAMI, vol. 24, no. 9, pp. 1226–1238, 2002
[76] S. H. Raza, O. Javed, A. Das, H. Sawhney, H. Cheng, and I. Essa, “Depth extraction from videos using geometric context and occlusion boundaries,” arXiv preprint arXiv:1510.07317, 2015.

[77] J.-I. Jung and Y.-S. Ho, “Depth map estimation from single-view image using object classification based on bayesian learning,” in 3DTV Conference, 2010, pp. 1–4
[78] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab, “Deeper depth prediction with fully convolutional residual networks,” in 3DV, 2016, pp. 239–248
[79] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in CVPR, 2016, pp. 770–778
[80] J. Hu, L. Shen, and G. Sun, “Squeeze-and-excitation networks,” in CVPR, 2018, pp. 7132–7141
[81] J. H. Lee, M.-K. Han, D. W. Ko, and I. H. Suh, “From big to small: Multi-scale local planar guidance for monocular depth estimation,” arXiv preprint arXiv:1907.10326, 2019.
[82] M. Mancini, G. Costante, P. Valigi, and T. A. Ciarfuglia, “Fast robust monocular depth estimation for obstacle detection with fully convolutional networks,” in IROS, 2016, pp. 4296–4303
[83] I. Alhashim and P. Wonka, “High quality monocular depth estimation 152 via transfer learning,” arXiv preprint arXiv:1812.11941, 2018
[84] M. Yue, G. Fu, M. Wu, and H. Wang, “Semi-supervised monocular depth estimation based on semantic supervision,” JIRS, 2020
[85] X. Dong, M. A. Garratt, S. G. Anavatti and H. A. Abbass, "Towards Real-Time Monocular Depth Estimation for Robotics: A Survey," in IEEE Transactions on Intelligent Transportation Systems, vol. 23, no. 10, pp. 16940-16961, Oct. 2022, doi: 10.1109/TITS.2022.3160741.
[86] C. Godard, O. M. Aodha and G. J. Brostow, "Unsupervised Monocular Depth Estimation with Left-Right Consistency," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 6602-6611
[87] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," Int. J. Rob. Res., vol. 32, pp. 1231-1237, 2013
[88] Spangenberg, R., Langner, T., Rojas, R. (2013). Weighted Semi-Global Matching and Center-Symmetric Census Transform for Robust Driver Assistance. In: Wilson, R., Hancock, E., Bors, A., Smith, W. (eds) Computer Analysis of Images and Patterns. CAIP 2013. Lecture Notes in Computer Science, vol 8048. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-40246-3_5
[89] N. Mayer et al., "A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 2016, pp. 4040-4048, doi: 10.1109/CVPR.2016.438
[90] N. Einecke and J. Eggert, "Stereo image warping for improved depth estimation of road surfaces," 2013 IEEE Intelligent Vehicles Symposium (IV), Gold Coast, QLD, Australia, 2013, pp. 189-194, doi: 10.1109/IVS.2013.6629469.

[91] M. Ferrera, A. Boulch and J. Moras, "Fast Stereo Disparity Maps Refinement By Fusion of Data-Based And Model-Based Estimations," 2019 International Conference on 3D Vision (3DV), Quebec City, QC, Canada, 2019, pp. 9-17, doi: 10.1109/3DV.2019.00011.
[92] M. Menze and A. Geiger, "Object scene flow for autonomous vehicles," 2015 IEEE Confderence on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 3061-3070
[93] A. Chakrabarti, Y. Xiong, S. J. Gortler and T. Zickler, "Low-level vision by consensus in a spatial hierarchy of regions," 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 4009-4017
[94] Yamaguchi K., McAllester D., Urtasun R. (2014) Efficient Joint Segmentation, Occlusion Labeling, Stereo and Flow Estimation. In: Fleet D., Pajdla T., Schiele B., Tuytelaars T. (eds) Computer Vision – ECCV 2014. ECCV 2014. Lecture Notes in Computer Science, vol 8693. Springer, Cham.
[95] Y. Ji, Q. Zhang, K. Sugimoto and S. Kamata, "Disparity refinement with stability-based tree for stereo matching," 2015 IEEE Intelligent Vehicles Symposium (IV), 2015, pp. 469-474
[96] Mahendra Mallick; Vikram Krishnamurthy; Ba-Ngu Vo, "Track-Before-Detect Techniques," in Integrated Tracking, Classification, and Sensor Management: Theory and Applications , IEEE, 2012, pp.311-362, doi: 10.1002/9781118450550.ch8.
[97] Z. Sun, J. Chen, L. Chao, W. Ruan and M. Mukherjee, "A Survey of Multiple Pedestrian Tracking Based on Tracking-by-Detection Framework," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 31, no. 5, pp. 1819-1833, May 2021, doi: 10.1109/TCSVT.2020.3009717.
[98] J. R. B. Del Rosario, A. A. Bandala and E. P. Dadios, "Multi-view multi-object tracking in an intelligent transportation system: A literature review," 2017IEEE 9th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management (HNICEM), Manila, Philippines, 2017, pp. 1-4, doi: 10.1109/HNICEM.2017.8269524.
[99] "Radio Progress during 1946," in Proceedings of the IRE, vol. 35, no. 4, pp. 399-425, April 1947, doi: 10.1109/JRPROC.1947.232611.
[100] Brown, Louis; A Radar History of World War II, Inst. of Physics Publishing, 1999
[101] S. Scheidegger, J. Benjaminsson, E. Rosenberg, A. Krishnan and K. Granström, "Mono-Camera 3D Multi-Object Tracking Using Deep Learning Detections and PMBM Filtering," 2018 IEEE Intelligent Vehicles Symposium (IV), 2018, pp. 433-440, doi: 10.1109/IVS.2018.8500454
[102] F. Teich, S. Yang and M. Baum, "GM-PHD filter for multiple extended object tracking based on the multiplicative error shape model and network flow labeling," 2017 IEEE Intelligent Vehicles Symposium (IV), 2017, pp. 7-12, doi: 10.1109/IVS.2017.7995691

[103] Lyudmila Mihaylova, Avishy Y. Carmi, François Septier, Amadou Gning, Sze Kim Pang, Simon Godsill,Overview of Bayesian sequential Monte Carlo methods for group and extended object tracking, Digital Signal Processing,Volume 25,2014,Pages 1-16,ISSN 1051-2004, https://doi.org/10.1016/j.dsp.2013.11.006.
[104] Huang Y, Shi Y, Song TL. An Efficient Multi-Path Multitarget Tracking Algorithm for Over-The-Horizon Radar. Sensors (Basel). 2019 Mar 20;19(6):1384. doi: 10.3390/s19061384. PMID: 30897805; PMCID: PMC6471521.
[105] R. B. Angle, R. L. Streit and M. Efe, "Multiple Target Tracking With Unresolved Measurements," in IEEE Signal Processing Letters, vol. 28, pp. 319-323, 2021, doi: 10.1109/LSP.2021.3051858.
[106] G. V. Weinberg "Constant false alarm rate detectors for Pareto clutter models" IET Radar Sonar Navigat. vol. 7 no. 2 pp. 153-163 Feb. 2013.
[107] S. Thrun, W. Burgard, and D. Fox, Probabilistic robotics. Cambridge, Mass.: MIT Press, 2005.
[108] S. Challa R. Evans M. Morelande and D. Mušicki Fundamentals of Object Tracking Cambridge United Kingdom:Cambridge University Press 2011.
[109] Kalman, R. (1960) A New Approach to Linear Filtering and Prediction Problems. ASME Journal of Basic Engineering, 82, 35-45. http://dx.doi.org/10.1115/1.3662552
[110] A. Houles and Y. Bar-Shalom, "Multisensor tracking of a maneuvering target in clutter," in IEEE Transactions on Aerospace and Electronic Systems, vol. 25, no. 2, pp. 176-189, March 1989, doi: 10.1109/7.18679.
[111] Xuezhi Wang, S. Challa and R. Evans, "Gating techniques for maneuvering target tracking in clutter," in IEEE Transactions on Aerospace and Electronic Systems, vol. 38, no. 3, pp. 1087-1097, July 2002, doi: 10.1109/TAES.2002.1039426.
[112] Y. Bar-Shalom, F. Daum and J. Huang, "The probabilistic data association filter," in IEEE Control Systems Magazine, vol. 29, no. 6, pp. 82-100, Dec. 2009, doi: 10.1109/MCS.2009.934469.
[113] Y. Bar-Shalom, X.R. Li, Multitarget-multisensor Tracking: Principles and Techniques Yaakov Bar-Shalom, 1995, ISBN 0964831201.
[114] Wang, M.H.; Wan, Q.; You, Z. A gate size estimation algorithm for data association filters. Sci. China Ser. F Inf. Sci. 2008, 51, 425–432.
[115] Li, T.; Sun, S.; Sattar, T.P. High-speed Sigma-gating SMC-PHD filter. Signal Process. 2013, 93, 2586–2593.
[116] Gade, B.; Kloster, M.; Aronsen, M. Non-Elliptical Validation Gate for Maritime Target Tracking. In Proceedings of the IEEE 2018 International Conference on Information Fusion, Cambridge, UK, 10–13 July 2018; pp. 1301–1308

[117] Kwak, N.; Lee, G.; Kwon, J. Robust Measurement Validation for Radar Target Tracking using Prior Information. IET Radar Sonar Navig. 2019, 13, 1842–1849
[118] J. Sun, Z. Wang, and Q. Li, "A New Multiple Hypothesis Tracker Using Validation Gate with Motion Direction Constraint," Sensors, vol. 20, no. 17, p. 4816, Aug. 2020, doi: 10.3390/s20174816.
[119] Xuezhi Wang, S. Challa and R. Evans, "Gating techniques for maneuvering target tracking in clutter," in IEEE Transactions on Aerospace and Electronic Systems, vol. 38, no. 3, pp. 1087-1097, July 2002, doi: 10.1109/TAES.2002.1039426.
[120] G. W. Pulford and R. J. Evans, "A multipath data association tracker for over-the-horizon radar," in IEEE Transactions on Aerospace and Electronic Systems, vol. 34, no. 4, pp. 1165-1183, Oct. 1998, doi: 10.1109/7.722704.
[121] Kirubarajan, T., Bar-Shalom, Y., Blair, W. D., and Watson, G. A. (1998) IMMPDAF for radar management and tracking benchmark with ECN. IEEE Transactions on Aerospace and Electronic Systems, 34, 4 (Oct. 1998), 1115—1133
[122] R. Schubert, C. Adam, M. Obst, N. Mattern, V. Leonhardt and G. Wanielik, "Empirical evaluation of vehicular models for ego motion estimation," 2011 IEEE Intelligent Vehicles Symposium (IV), 2011, pp. 534-539, doi: 10.1109/IVS.2011.5940526.
[123] Bar-Shalom and X.-R. Li,Multitarget-Multisensor tracking: Princi-ples and Techniques. Storrs, CT: YBS Publishing, 1995.
[124] R. Schubert, E. Richter and G. Wanielik, "Comparison and evaluation of advanced motion models for vehicle tracking," 2008 11th International Conference on Information Fusion, Cologne, 2008, pp. 1-6.
[125] "Introduction to discretization - Department of Scientific Computing." Available: https://people.sc.fsu.edu/~jpeterson/IVP.pdf. [Accessed: 02-Apr-2023].
[126] Kendall Atkinson, Weimin Han, David Stewart, NUMERICAL SOLUTION OF ORDINARY DIFFERENTIAL EQUATIONS, University of Iowa, Iowa City, Iowa.
[127] X. Rong Li and V. P. Jilkov, "Survey of maneuvering target tracking. Part I. Dynamic models," in IEEE Transactions on Aerospace and Electronic Systems, vol. 39, no. 4, pp. 1333-1364, Oct. 2003, doi: 10.1109/TAES.2003.1261132.
[128] Lu Tao;Yousuke Watanabe;Shunya Yamada;Hiroaki Takada; (2021). Comparative evaluation of Kalman filters and motion models in vehicular state estimation and path prediction . Journal of Navigation. doi:10.1017/S0373463321000370
[129] D. Svensson, "Derivation of the discrete-time constant turn rate and acceleration motion model," 2019 Sensor Data Fusion: Trends, Solutions, Applications (SDF), 2019, pp. 1-5, doi: 10.1109/SDF.2019.8916654

[130] M. Kolat, O. Törő, and T. Bécsi, "Performance Evaluation of a Maneuver Classification Algorithm Using Different Motion Models in a Multi-Model Framework," Sensors, vol. 22, no. 1, p. 347, Jan. 2022, doi: 10.3390/s22010347
[131] Hazem J. A. J., Motion Estimation of On-Road Vehicles from Moving Platform Using Stereo Vision, May 2018.
[132] Yuan, X.; Lian, F.; Han, C. Models and Algorithms for Tracking Target with Coordinated Turn Motion. Math. Probl. Eng. 2014, 2014, 1–10
[133] X. Rong Li and V. P. Jilkov, "Survey of maneuvering target tracking. Part I. Dynamic models," in IEEE Transactions on Aerospace and Electronic Systems, vol. 39, no. 4, pp. 1333-1364, Oct. 2003, doi: 10.1109/TAES.2003.1261132.
[134] W. Farag "LiDAR and radar fusion for real-time road-objects detection and tracking" Intell. Decis. Technol. vol. 15 no. 2 pp. 291-304 2021.
[135] Patric, J. (2001). Approaches to Mobile Robot Localization in Indoor Environments. Ph.D. dissertation, Department of Signals, Sensors and Systems, Royal Institute of Technology, Stockholm, Sweden.
[136] Everett, H. R. (1995). Sensors for Mobile Robots. A K Peters/CRC Press, Wellesley, Massachusetts
[137] X. Chen, R. Tharmarasa, M. Pelletier, and T. Kirubarajan, "Integrated clutter estimation and target tracking using poisson point processes," IEEE Transactions on Aerospace and Electronic Systems, vol. 48, no. 2, pp. 1210–1235, 2012.
[138] Streit, R., Angle, R. B., & Efe, M. (2021). Analytic combinatorics for multiple object tracking.
[139] Lee, B.; Erdenee, E.; Jin, S.; Nam, M.Y.; Jung, Y.G.; Rhee, P.K. Multi-Class Multi-Object Tracking Using Changing Point Detection; Computer Vision—ECCV Workshops; Hua, G., Jégou, H., Eds.; Springer: Cham, Switzerland, 2016; pp. 68–83.
[140] Gündüz, G.; Acarman, T. A lightweight online multiple object vehicle tracking method. In Proceedings of the IEEE Intelligent Vehicles Symposium (IV), Changshu, China, 26–30 June 2018; pp. 427–432.
[141] Karunasekera, H.; Wang, H.; Zhang, H. Multiple Object Tracking With Attention to Appearance, Structure, Motion and Size. IEEE Access 2019, 7, 104423–104434.
[142] Kuhn, H.W. The Hungarian method for the assignment problem. Nav. Res. Logist. Q. 1955, 2, 83–97.
[143] Choi, W. Near-Online Multi-target Tracking with Aggregated Local Flow Descriptor. In Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 7–13 December 2015; pp. 3029–3037.
[144] Kim, D.; Kwon, D. Pedestrian detection and tracking in thermal images using shape features. In Proceedings of the 2015 12th International

Conference on Ubiquitous Robots and Ambient Intelligence (URAI), Goyangi, Korea, 28–30 October; 2015; pp. 22–25.

[145] Bertozzi, M.; Broggi, A.; Fascioli, A.; Graf, T.; Meinecke, M.M. IR pedestrian detection for advanced driver assistance systems. IEEE Trans. Veh. Technol. 2004, 53, 1666–1678

[146] Munder, S.; Schnorr, C.; Gavrila, D. Pedestrian Detection and Tracking Using a Mixture of View-Based Shape–Texture Models. IEEE Trans. Intell. Transp. Syst. 2008, 9, 333–343.

[147] Kallhammer, J.E.; Eriksson, D.; Granlund, G.; Felsberg, M.; Moe, A.; Johansson, B.; Wiklund, J.; Forssen, P.E. Near Zone Pedestrian Detection using a Low-Resolution FIR Sensor. In Proceedings of the 2007 IEEE Intelligent Vehicles Symposium, Istanbul, Turkey, 13–15 June 2007; pp. 339–345.

[148] Kwak, J.-Y.; Ko, B.C.; Nam, J.Y. Pedestrian Tracking Using Online Boosted Random Ferns Learning in Far-Infrared Imagery for Safe Driving at Night. IEEE Trans. Intell. Transp. Syst. 2017, 18, 69–81.

[149] Yu, X.; Yu, Q.; Shang, Y.; Zhang, H. Dense structural learning for infrared object tracking at 200+ Frames per Second. Pattern Recognit. Lett. 2017, 100, 152–159.

[150] Zhu, G.; Porikli, F.; Li, H. Beyond local search: Tracking objects everywhere with instance-specific proposals. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 943–951.

[151] S. Lee and E. Kim, "Multiple Object Tracking via Feature Pyramid Siamese Networks," in IEEE Access, vol. 7, pp. 8181-8194, 2019, doi: 10.1109/ACCESS.2018.2889442.

[152] X. Dong, J. Shen, Triplet Loss in Siamese Network for Object Tracking, in Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 459-474.

[153] Elezi, I., Vascon, S., Torcinovich, A., Pelillo, M., Leal-Taixe, L.: The group loss for deep metric learning. arXiv preprint arXiv:1912.00385 (2019)

[154] POI: Multiple Object Tracking with High Performance Detection and Appearance Feature, Fengwei Yu, Wenbo Li, Quanquan Li, Yu Liu, Xiaohua Shi, Junjie Yan, 20

[155] Wang, Z., Zheng, L., Liu, Y., Li, Y., Wang, S. (2020). Towards Real-Time Multi-Object Tracking. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, JM. (eds) Computer Vision – ECCV 2020. ECCV 2020. Lecture Notes in Computer Science(), vol 12356. Springer, Cham.

[156] L. Leal-Taixé, C. Canton-Ferrer and K. Schindler, "Learning by Tracking: Siamese CNN for Robust Target Association," 2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2016, pp. 418-425, doi: 10.1109/CVPRW.2016.59

[157] Nam, H.; Baek, M.; Han, B. Modeling and propagating cnns in a tree structure for visual tracking. arXiv 2016, arXiv:1608.07242.
[158] Alahari, K.; Berg, A.; Hager, G.; Ahlberg, J.; Kristan, M.; Matas, J.; Leonardis, A.; Cehovin, L.; Fernandez, G.; Vojir, T.; et al. The thermal infrared visual object tracking vot-tir2015 challenge results. In Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 7–13 December 2015; pp. 639–651.
[159] Danelljan, M.; Hager, G.; Khan, F.S.; Felsberg, M. Learning spatially regularized correlation filters for visual tracking. In Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 7–13 December 2015; pp. 4310–4318.
[160] Bertinetto, L.; Valmadre, J.; Henriques, J.F.; Vedaldi, A.; Torr, P.H. Fully-convolutional siamese networks for object tracking. In Computer Vision—ECCV 2016 Workshops. ECCV 2016. Lecture Notes in Computer Science; Hua, G., Jégou, H., Eds.; Springer: Cham, Switzerland, 2016; Volume 9914, pp. 850–865.
[161] Liu, Q.; Yuan, D.; He, Z. Thermal infrared object tracking via Siamese convolutional neural networks. In Proceedings of the 2017 International Conference on Security, Pattern Analysis, and Cybernetics (SPAC), Shenzhen, China, 7–13 December 2017; pp. 1–6.
[162] Zhang, X.; Chen, R.; Liu, G.; Li, X.; Luo, S.; Fan, X. Thermal Infrared Tracking using Multi-stages Deep Features Fusion. In Proceedings of the 2020 Chinese Control And Decision Conference (CCDC), Hefei, China, 22–24 August 2020; pp. 1883–1888.
[163] Guo, Q.; Feng, W.; Zhou, C.; Huang, R.; Wan, L.; Wang, S. Learning Dynamic Siamese Network for Visual Object Tracking. In Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 December 2017; pp. 1781–1789.
[164] Wang, G.; Luo, C.; Xiong, Z.; Zeng, W. SPM-Tracker: Series-Parallel Matching for Real-Time Visual Object Tracking. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 15–20 June 2019; pp. 3643–3652.
[165] Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet classification with deep convolutional neural networks. Commun. ACM 2017, 60, 84–90.
[166] Zhang, L.; Gonzalez-Garcia, A.; Weijer, J.; Danelljan, M.; Khan, F.S. Synthetic Data Generation for End-to-End Thermal Infrared Tracking. IEEE Trans. Image Process. 2019, 28, 1837–1850.
[167] Valmadre, J.; Bertinetto, L.; Henriques, J.; Vedaldi, A.; Torr, P.H. End-to-end representation learning for correlation filter-based tracking. Computer Vision and Pattern Recognition (CVPR). In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 5000–5008.

[168] Qiang Wang, Li Zhang, Luca Bertinetto, Weiming Hu, Philip H.S. Torr; Fast Online Object Tracking and Segmentation: A Unifying Approach The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 1328-1338
[169] P. Voigtlaender, M. Krause, A. Ousep, J. Luiten, B. Sekar, A. Geiger and B. Leibe: MOTS: Multi-Object Tracking and Segmentation. CVPR 2019.
[170] J. Luiten, T. Fischer and B. Leibe: Track to Reconstruct and Reconstruct to Track. https://arxiv.org/pdf/1910.00130.pdf
[171] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: NeurlPS (2017)
[172] Bahdanau, D.; Kyunghyun, C.; Yoshua, B. Neural machine translation by jointly learning to align and translate. arXiv 2014, arXiv:1409.0473.
[173] Zhu, X.; Jia, Y.; Jian, S.; Gu, L.; Pu, Z. ViTT: Vision Transformer Tracker. Sensors 2021, 21, 5608. https://doi.org/10.3390/ s21165608
[174] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Detect to track and track to detect, 2018
[175] Zhongdao Wang, Liang Zheng, Yixuan Liu, and Shengjin Wang. Towards real-time multi-object tracking. arXiv preprint arXiv:1909.12605, 2019.
[176] P. Sun, J Cao, Y. Jiang, R. Zhang, E. Xie, Z. Yuan, C. Wang, P. Luo, TransTrack: Multiple Object Tracking with Transformer, arXiv : 2012.15460 v2, 4 May 2021, https://arxiv.org/ pdf/ 2012. 15460 .pdf
[177] Cui, Y., Fang, Z., Shan, J., Gu, Z., & Zhou, S. (2021). 3D Object Tracking with Transformer. In 32nd British Machine Vision Conference 2021, BMVC 2021, Online, November 22-25, 2021 (p. 317). BMVA Press.
[178] D. Frossard and R. Urtasun, "End-to-end Learning of Multi-sensor 3D Tracking by Detection," 2018 IEEE International Conference on Robotics and Automation (ICRA), 2018, pp. 635-642, doi: 10.1109/ICRA.2018.8462884
[179] M. M. Zavlanos, L. Spesivtsev and G. J. Pappas, "A distributed auction algorithm for the assignment problem," 2008 47th IEEE Conference on Decision and Control, 2008, pp. 1212-1217, doi: 10.1109/CDC.2008.4739098
[180] R. Jonker, A. Volgenant, A shortest augmenting path algorithm for dense and sparse linear assignment problems, J. Comput. 38 (4) (1987) 325–340
[181] Ford, L. R.; Fulkerson, D. R. (1956). "Maximal flow through a network" (PDF). Canadian Journal of Mathematics. 8: 399–404. doi:10.4153/CJM-1956-045-5. S2CID 16109790.
[182] Murty, K. (1968). An Algorithm for Ranking all the Assignments in Order of Increasing Cost. Operations Research, 16(3), 682-687. Retrieved from http://www.jstor.org/stable/168595
[183] L. M. Wolf and M. Baum, "Deterministic Gibbs Sampling for Data Association in Multi-Object Tracking," 2020 IEEE International Conference on

Multisensor Fusion and Integration for Intelligent Systems (MFI), 2020, pp. 291-296, doi: 10.1109/MFI49285.2020.9235211.

[184] M. Hosseini and S. S. Moghaddam, "Sub-Optimum Radio Resource Allocation in Vehicle-To-Vehicle Communications Based on A Multi-Step Hungarian Algorithm," 2021 IEEE Microwave Theory and Techniques in Wireless Communications (MTTW), 2021, pp. 86-91, doi: 10.1109/MTTW53539.2021.9607143.

[185] M. Motro and J. Ghosh, "Scaling Data Association for Hypothesis-Oriented MHT," 2019 22th International Conference on Information Fusion (FUSION), 2019, pp. 1-8.

[186] A. Trezza, D. J. Bucci and P. K. Varshney, "Multi-Sensor Joint Adaptive Birth Sampler for Labeled Random Finite Set Tracking," in IEEE Transactions on Signal Processing, vol. 70, pp. 1010-1025, 2022, doi: 10.1109/TSP.2022.3151553.

[187] L. Zhang, Y. Li, and R. Nevatia. Global data association for multi-object tracking using network flows. In CVPR, 2008.

[188] P. Lenz, A. Geiger and R. Urtasun, "FollowMe: Efficient Online Min-Cost Flow Tracking with Bounded Memory and Computation," 2015 IEEE International Conference on Computer Vision (ICCV), 2015, pp. 4364-4372, doi: 10.1109/ICCV.2015.496

[189] Kullback, S.; Leibler, R.A. (1951). "On information and sufficiency". Annals of Mathematical Statistics. 22 (1): 79–86. doi:10.1214/aoms/1177729694

[190] Bonnici, V. (2020). "Kullback-Leibler divergence between quantum distributions, and its upper-bound".

[191] D. Reid, "An algorithm for tracking multiple targets," in IEEE Transactions on Automatic Control, vol. 24, no. 6, pp. 843-854, December 1979, doi: 10.1109/TAC.1979.1102177.

[192] T. Fortmann, Y. Bar-Shalom and M. Scheffe, "Sonar tracking of multiple targets using joint probabilistic data association," in IEEE Journal of Oceanic Engineering, vol. 8, no. 3, pp. 173-184, July 1983, doi: 10.1109/JOE.1983.1145560

[193] Fitzgerald, R.J. (1999) Development of practical PDA logic for multitarget tracking by microprocessor. In Multitarget Multisensor tracking: Advanced Application; Artech House: Boston, USA; pp. 1-23.

[194] Y. Liu, W. Zhang and M. Chen, "Near Neighbor Cheap JPDA IMM based on amplitude information," The 2012 International Workshop on Microwave and Millimeter Wave Circuits and System Technology, 2012, pp. 1-4, doi: 10.1109/MMWCST.2012.6238188.

[195] J. A. Roecker and G. L. Phillis, "Suboptimal joint probabilistic data association," in IEEE Transactions on Aerospace and Electronic Systems, vol. 29, no. 2, pp. 510-517, April 1993, doi: 10.1109/7.210087

[196] James L. Fisher and David P. Casasent, "Fast JPDA multitarget tracking algorithm," Appl. Opt. 28, 371-376 (1989)

[197] T. Kropfreiter, F. Meyer, S. Coraluppi, C. Carthel, R. Mendrzik and P. Willett, "Track Coalescence and Repulsion: MHT, JPDA, and BP," 2021 IEEE 24th International Conference on Information Fusion (FUSION), 2021, pp. 1-8, doi: 10.23919/FUSION49465.2021.9626958.

[198] S. Zhao, Y. Wang, P. Wang, T. Ma and K. Guo, "Adaptive Non-Linear Joint Probabilistic Data Association for Vehicle Target Tracking," in IEEE Access, vol. 9, pp. 14138-14147, 2021, doi: 10.1109/ACCESS.2021.3052555.

[199] C. Kim, F. Li, A. Ciptadi and J. M. Rehg, "Multiple Hypothesis Tracking Revisited," 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 2015, pp. 4696-4704, doi: 10.1109/ICCV.2015.533.

[200] Coraluppi, Stefano P. "Fundamentals and Advances in Multiple-Hypothesis Tracking." (2015).

[201] S. S. Blackman, "Multiple hypothesis tracking for multiple target tracking," in IEEE Aerospace and Electronic Systems Magazine, vol. 19, no. 1, pp. 5-18, Jan. 2004, doi: 10.1109/MAES.2004.1263228.

[202] S. Challa, M. R. Morelande, D. Mušicki, and R. J. Evans, Fundamentals of Object Tracking. Cambridge: Cambridge University Press, 2011.

[203] Y. Bar-Shalom and T. E. Fortmann, "Tracking and Data Association, Volume 179 of Mathematics in Science and Engineering," Academic Press Professional, Inc., San Diego, 1987.

[204] S. J. Julier and J. K. Uhlmann, "Unscented filtering and nonlinear estimation," in Proceedings of the IEEE, vol. 92, no. 3, pp. 401-422, March 2004, doi: 10.1109/JPROC.2003.823141.

[205] M. Schreier, V. Willert and J. Adamy, "Compact Representation of Dynamic Driving Environments for ADAS by Parametric Free Space and Dynamic Object Maps," in IEEE Transactions on Intelligent Transportation Systems, vol. 17, no. 2, pp. 367-384, Feb. 2016, doi: 10.1109/TITS.2015.2472965.

[206] M. Allodi, A. Broggi, D. Giaquinto, M. Patander and A. Prioletti, "Machine learning in tracking associations with stereo vision and lidar observations for an autonomous vehicle," 2016 IEEE Intelligent Vehicles Symposium (IV), 2016, pp. 648-653, doi: 10.1109/IVS.2016.7535456.

[207] W. Liu, X. Tang and X. Ren, "Trajectory Smoothing Constraint and Hard Negative Mining for Distractor-Aware Regression Tracking," in IEEE Access, vol. 7, pp. 84658-84667, 2019, doi: 10.1109/ACCESS.2019.2921562.

[208] Chenglong Li, Wei Xia, Yan Yan, Bin Luo, and Jin Tang. Segmenting objects in day and night: Edge-conditioned cnn for thermal image semantic segmentation. IEEE Transactions on Neural Networks and Learning Systems, 2020.

[209] R. D. Brehar, M. P. Muresan, T. Mariţa, C. -C. Vancea, M. Negru and S. Nedevschi, "Pedestrian Street-Cross Action Recognition in Monocular Far Infrared Sequences," in IEEE Access, vol. 9, pp. 74302-74324, 2021, doi: 10.1109/ACCESS.2021.3080822.
[210] Peng Wang and Xiangzhi Bai. Thermal infrared pedestrian segmentation based on conditional gan. IEEE transactions on image processing, 28(12):6007–6021, 2019
[211] Shaohui Chen, Zengzhao Chen, Xiaogang Xu, Ningyu Yang, and Xiuling He. Nv-net: Efficient infrared image segmentation with convolutional neural networks in the low illumination environment. Infrared Physics & Technology, 105:103184, 2020.
[212] Haitao Xiong, Wenjie Cai, and Qiong Liu. Mcnet: Multilevel correction network for thermal image semantic segmentation of nighttime driving scene. Infrared Physics & Technology, 113:103628, 2021.
[213] Qishen Ha, Kohei Watanabe, Takumi Karasawa, Yoshitaka Ushiku, and Tatsuya Harada. Mfnet: Towards real-time semantic segmentation for autonomous vehicles with multispectral scenes. In 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 5108–5115. IEEE, 2017
[214] Flir Thermal Datasets for Algorithm Training, FLIR, Wilsonville, OR, USA, 2018. Available online: https://www.flir.com/oem/adas/adas-dataset-form/
[215] Z. Kütük and G. Algan, "Semantic Segmentation for Thermal Images: A Comparative Survey," 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), New Orleans, LA, USA, 2022, pp. 285-294, doi: 10.1109/CVPRW56347.2022.00043
[216] J. Redmon and A. Farhadi, ‘‘YOLO9000: Better, faster, stronger,’’ in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jul. 2017, pp. 6517–6525.
[217] R. Brehar, F. Vancea, T. Marita, C. Vancea, and S. Nedevschi, ‘‘Object detection in monocular infrared images using classification–regresion deep learning architectures,’’ in Proc. IEEE 15th Int. Conf. Intell. Comput. Commun. Process. (ICCP), Sep. 2019, pp. 207–212.
[218] O. Lahdenoja, J. Poikonen, and M. Laiho, ‘‘Towards understanding the formation of uniform local binary patterns,’’ ISRN Mach. Vis., vol. 2013, pp. 1–20, Jul. 2013.
[219] G. Farnebäck, ‘‘Two-frame motion estimation based on polynomial expansion,’’ in Scandinavian conference on Image analysis, vol. 2749. Berlin, Germany: Springer, Jun. 2003, pp. 363–370
[220] E. Romera, J. M. Álvarez, L. M. Bergasa and R. Arroyo, "ERFNet: Efficient Residual Factorized ConvNet for Real-Time Semantic Segmentation," in IEEE Transactions on Intelligent Transportation Systems, vol. 19, no. 1, pp. 263-272, Jan. 2018, doi: 10.1109/TITS.2017.2750080

[221] G. Jocher; A. Chaurasia; A. Stoken; J. Borovec; NanoCode012; Y Kwon; M. Kalen; TaoXie; J Fang; imyhxy; Lorna; Y. Zeng; C. Wong; Abhiram V; D. Montes; Z Wang; C. Fati; J Nadar; Laughing; UnglvKitDe; V. Sonck; tkianai; yxNONG; P. Skalski; A. Hogan; D. Nair; M. Strobel; M. Jain; ultralytics/yolov5: v7.0 - YOLOv5 SOTA Realtime Instance Segmentation, Nov. 2022

[222] Bolya, D.; Zhou, C.; Xiao, F.; Lee, Y.J. YOLACT: Real-time Instance Segmentation. arXiv 2019, arXiv:1904.02689

[223] Umbaugh, S.E. (1997). Computer Vision and Image Processing: A Practical Approach Using CVIPTools.

[224] M. P. Muresan, S. Nedevschi, and R. Danescu, "A multi patch warping approach for improved stereo block matching," in Proc. Int. Conf. Comput. Vis. Theory App., 2017, pp. 459–466

[225] H. A. P. Bloom, "An efficient filter for abruptly changing systems", in Proceedings of the 23rd IEEE Conference on Decision and Control Las Vegas, NV, Dec. 1984, 656-658.

[226] Lee, F.-F.; Chen, F.; Liu, J. Infrared Thermal Imaging System on a Mobile Phone. Sensors 2015, 15, 10166–10179.

[227] Xiang, Y.; Alahi, A.; Savarese, S. Learning to Track: Online Multi-object Tracking by Decision Making. In Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 7–13 December 2015; pp. 4705–4713

[228] Osep, A.; Mehner, W.; Mathias, M.; Leibe, B. Combined image- and world-space tracking in traffic scenes. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017; pp. 1988–1995

[229] Milan, A.; Schindler, K.; Roth, S. Detection- and Trajectory-Level Exclusion in Multiple Object Tracking. In Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition, Portland, OR, USA, 23–28 June 2013; pp. 3682–3689

[230] Q. Liu, Z. He, X. Li, and Y. Zheng, ''PTB-TIR: A thermal infrared pedestrian tracking benchmark,'' IEEE Trans. Multimedia, vol. 22, no. 3, pp. 666–675, Mar. 2020

[231] M. Danelljan, G. Bhat, F. S. Khan, and M. Felsberg, ''ECO: Efficient convolution operators for tracking,'' in Proc. Conf. Comput. Vis. Pattern Recognit., 2017, pp. 6638–6646

[232] M. Danelljan, G. Hager, F. S. Khan, and M. Felsberg, ''Learning spatially regularized correlation filters for visual tracking,'' in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), Dec. 2015, pp. 4310–4318, doi: 10.1109/ICCV.2015.490

[233] F. Li, C. Tian, W. Zuo, L. Zhang, and M.-H. Yang, ''Learning spatial-temporal regularized correlation filters for visual tracking,'' in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., Jun. 2018, pp. 4904–4913.

[234] L. Bertinetto, J. Valmadre, S. Golodetz, O. Miksik, and P. H. S. Torr, ‘‘Staple: Complementary learners for real-time tracking,’’ in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2016, pp. 1401–1409
[235] Q. Liu, X. Li, Z. He, N. Fan, D. Yuan, and H. Wang, ‘‘Learning deep multi-level similarity for thermal infrared object tracking,’’ IEEE Trans. Multimedia, early access, Jul. 10, 2020, doi: 10.1109/TMM.2020.3008028.
[236] J. Valmadre, L. Bertinetto, J. Henriques, A. Vedaldi, and P. H. S. Torr, ‘‘End-to-end representation learning for correlation filter based tracking,’’ in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jul. 2017, pp. 5000–5008.
[237] Y. Song, C. Ma, X. Wu, L. Gong, L. Bao, W. Zuo, C. Shen, R. W. H. Lau, and M.-H. Yang, ‘‘VITAL: VIsual tracking via adversarial learning,’’ in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., Jun. 2018, pp. 8990–8999
[238] Y. Wang, H. Huang, X. Huang, and Y. Tian, ‘‘ECO-HC based tracking for ground moving target using single UAV,’’ in Proc. 39th Chin. Control Conf. (CCC), Jul. 2020, pp. 6414–6419.
[239] X. Li, C. Ma, B. Wu, Z. He, and M.-H. Yang, ‘‘Target-aware deep tracking,’’ in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2019, pp. 1369–1378.
[240] Q. Liu, X. Lu, Z. He, C. Zhang, and W.-S. Chen, ‘‘Deep convolutional neural networks for thermal infrared object tracking,’’ Knowl.- Based Syst., vol. 134, pp. 189–198, Oct. 2017
[241] H. Nam and B. Han, "Learning Multi-domain Convolutional Neural Networks for Visual Tracking," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 2016, pp. 4293-4302, doi: 10.1109/CVPR.2016.465.
[242] Y. Song, C. Ma, L. Gong, J. Zhang, R. W. H. Lau and M. -H. Yang, "CREST: Convolutional Residual Learning for Visual Tracking," 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 2017, pp. 2574-2583, doi: 10.1109/ICCV.2017.279
[243] X. Li, Q. Liu, N. Fan, Z. He, and H. Wang, ‘‘Hierarchical spatial-aware Siamese network for thermal infrared object tracking,’’ Knowl.-Based Syst., vol. 166, pp. 71–81, Feb. 2019
[244] Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You Only Look Once: Unified, RealTime Object Detection. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788.
[245] Qi, Y.; Zhang, S.; Qin, L.; Yao, H.; Huang, Q.; Lim, J.; Yang, M.-H. Hedged deep tracking. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 4303–4311
[246] Ma, C.; Huang, J.-B.; Yang, X.; Yang, M.-H. Hierarchical convolutional features for visual tracking. In Proceedings of the 2015 IEEE International

Conference on Computer Vision (ICCV), Santiago, Chile, 7–13 December 2015; pp. 3074–3082

[247] Poudel, R. P., Liwicki, S., & Cipolla, R. (2019). Fast-scnn: Fast semantic segmentation network. In: Proc. British Machine Vision Conference.
[248] M. Bertozzi, A. Broggi, A. Fascioli, A. Tibaldi, R. Chapuis, and F. Chausse, ‘‘Pedestrian localization and tracking system with Kalman filtering,’’ in Proc. IEEE Intell. Vehicles Symp., Jun. 2004, pp. 584–589.
[249] J. White, C. Archer, J. G. Haidt, C. L. Britt and R. Neece, "Fusion of airborne radar and FLIR sensors for runway incursion detection," 2009 IEEE/AIAA 28th Digital Avionics Systems Conference, Orlando, FL, 2009, pp. 5.A.2-1-5.A.2-9, doi: 10.1109/DASC.2009.5347475.
[250] Z. Wang, Y. Wu and Q. Niu, "Multi-Sensor Fusion in Automated Driving: A Survey," in IEEE Access, vol. 8, pp. 2847-2868, 2020, doi: 10.1109/ACCESS.2019.2962554.
[251] A. Ceccarelli and F. Secci, "RGB cameras failures and their effects in autonomous driving applications," in IEEE Transactions on Dependable and Secure Computing, doi: 10.1109/TDSC.2022.3156941.
[252] Y. Sun, W. Zuo, P. Yun, H. Wang and M. Liu, "FuseSeg: Semantic Segmentation of Urban Scenes Based on RGB and Thermal Data Fusion," in IEEE Transactions on Automation Science and Engineering, vol. 18, no. 3, pp. 1000-1011, July 2021, doi: 10.1109/TASE.2020.2993143.
[253] A. Tomy, A. Paigwar, K. S. Mann, A. Renzaglia and C. Laugier, "Fusing Event-based and RGB camera for Robust Object Detection in Adverse Conditions," 2022 International Conference on Robotics and Automation (ICRA), Philadelphia, PA, USA, 2022, pp. 933-939, doi: 10.1109/ICRA46639.2022.9812059
[254] X. Dong, M. A. Garratt, S. G. Anavatti and H. A. Abbass, "Towards Real-Time Monocular Depth Estimation for Robotics: A Survey," in IEEE Transactions on Intelligent Transportation Systems, vol. 23, no. 10, pp. 16940-16961, Oct. 2022, doi: 10.1109/TITS.2022.3160741.
[255] Y. Li et al., "Deep Learning for LiDAR Point Clouds in Autonomous Driving: A Review," in IEEE Transactions on Neural Networks and Learning Systems, vol. 32, no. 8, pp. 3412-3432, Aug. 2021, doi: 10.1109/TNNLS.2020.3015992.
[256] “3D lidar sensors LD-Mrs,” SICK. [Online]. Available: https://www.sick.com/at/en/lidar-sensors/3d-lidar-sensors/ld-mrs/c/g91913.
[257] “Puck lidar sensor, high-value surround Lidar,” Velodyne Lidar, 22-Feb-2023. [Online]. Available: https://velodynelidar.com/products/puck/.
[258] A. Manjunath, Y. Liu, B. Henriques and A. Engstle, "Radar Based Object Detection and Tracking for Autonomous Driving," 2018 IEEE MTT-S

International Conference on Microwaves for Intelligent Mobility (ICMIM), Munich, Germany, 2018, pp. 1-4, doi: 10.1109/ICMIM.2018.8443497.
[259] W.H. Strickland, R. H. King, Characteristics of Ultrasonic Ranging Sensors in an Underground Environment, U.S. Department of the Interior, Bureau of Mines, 1993
[260] "Building a GPS system," Building a GPS System - SparkFun Electronics. [Online]. Available: https://www.sparkfun.com/gps. [Accessed: 05-Apr-2023].
[261] D. McGriffy, Make: Drones. Sebastopol: O'Reilly Media, Inc, USA, 2016.
[262] "Pose estimation from asynchronous sensors," Pose Estimation From Asynchronous Sensors - MATLAB & Simulink. [Online]. Available: https://www.mathworks.com/help/fusion/ug/pose-estimation-from-asynchronous-sensors.html?s_eid=PSM_15028.
[263] M.E. Leggins, D.L. Hall, J Llians (eds.), Handbook of Multisensor Data Fusion – Theory and Practice, 2nd edn.(CRC Press, Boca Raton, 2008)
[264] Panagiotis Lytrivis, George Thomaidis Angelos Amditis, Sensor Data Fusion in Automotive Applications,2009.
[265] D.L. Hall and A. Steinberg, Dirty secrets in multisensor data fusion, Proceedings of the National Symposium on Sensor Data Fusion (NSSDF), San Antonio, TX, June 2000.
[266] David Hall, James Llinas, Multisensor Data Fusion (Electrical Engineering & Applied Signal Processing Series) 1st Edition
[267] D. Schuldhaus, H. Leutheuser, B. M. Eskofier, Towards big data for activity recognition: a novel database fusion strategy, in: 9th International Conference on Body Area Networks, 2014, pp. 97–103
[268] X. Lai, Q. Liu, X. Wei, W. Wang, G. Zhou, G. Han, A survey of body sensor networks, Sensors 13 (5) (2013) 5406
[269] C. Chen, R. Jafari, N. Kehtarnavaz, A survey of depth and inertial sensor fusion for human action recognition, Multimedia Tools and Applications (2015) 1–2
[270] A. Bulling, U. Blanke, B. Schiele, A tutorial on human activity recognition using body-worn inertial sensors, ACM Computing Surveys 46 (3) (2014) 1–33
[271] P. Zappi, T. Stiefmeier, E. Farella, D. Roggen, L. Benini, G. Troster, Activity recognition from on-body sensors by classifier fusion: sensor scalability and robustness, in: Intelligent Sensors, Sensor Networks and Information, 2007. ISSNIP 2007. 3rd International Conference on, 2007, pp. 281–286.
[272] C. Chen, R. Jafari, N. Kehtarnavaz, A survey of depth and inertial sensor fusion for human action recognition, Multimedia Tools and Applications (2015) 1–2
[273] L. G. Brown, A Survey of Image Registration Techniques, ACM Computing Surveys, 24(4), 325–376, 1992.

[274] S. Nag, “Image Registration Techniques: A Survey,” 28-Nov-2017. [Online]. Available: osf.io/gp5r6.
[275] Szeliski R, Shum HY (1997) Creating full view panoramic image mosaics and environment maps. In: Proceedings of the 24th annual conference on Computer graphics and interactive techniques. SIGGRAPH ’97, New York. ACM Press/Addison-Wesley Publishing Co, pp 251–258
[276] Walter, Andreas and Mannheim, Julia G and Caruana, Carmel J,May 2021, Imaging Modalities for Biological and Preclinical Research: A Compendium, Volume 2.
[277] J. Thomanek, H. Lietz and G. Wanielik, "Pixel-based data fusion for a better object detection in automotive applications," 2010 IEEE International Conference on Intelligent Computing and Intelligent Systems, 2010, pp. 385-390, doi: 10.1109/ICICISYS.2010.5658327
[278] Gang Hong Yun hang , “Combination of feature-based and areabased image registration technique for high resolution remote sensing image”, IEEE international symposium on Geoscience and remote sensing, IGARSS 2007 (July 2007), 377-380
[279] K. S. Arun, T. S. Huang and S. D. Blostein, "Least-Squares Fitting of Two 3-D Point Sets," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. PAMI-9, no. 5, pp. 698-700, Sept. 1987, doi: 10.1109/TPAMI.1987.4767965.
[280] P. J. Besl and N. D. McKay, "A method for registration of 3-D shapes," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 14, no. 2, pp. 239-256, Feb. 1992, doi: 10.1109/34.121791.
[281] M. Alshawa, “ICL: Iterative closest line a novel point cloud registration algorithm based on linear features,” Ekscentar, vol. 10, pp. 53–59, 2007.
[282] Greenspan, M., and M. Yurick (2003), Approximate k-d tree search for efficient ICP, in Proceedings of Fourth International Conference on 3D Digital Imaging and Modeling, pp. 442–448, Banff, Alberta, Canada.
[283] Fitzgibbon, A. W. (2003), Robust registration of 2D and 3D point sets, Image and Vision Computing, 21(14), 1145–1153.
[284] N. Gelfand, L. Ikemoto, S. Rusinkiewicz and M. Levoy, "Geometrically stable sampling for the ICP algorithm," Fourth International Conference on 3-D Digital Imaging and Modeling, 2003. 3DIM 2003. Proceedings., Banff, AB, Canada, 2003, pp. 260-267, doi: 10.1109/IM.2003.1240258.
[285] Kourosh Khoshelham. Closed-form solutions for estimating a rigid motion from plane correspondences extracted from point clouds. ISPRS Journal of Photogrammetry and Remote Sensing, 114:78–91, 2016
[286] Wolfgang Forstner and Kourosh Khoshelham. Efficient and accurate registration of point clouds with plane to plane correspondences. In Proceedings of the IEEE International Conference on Computer Vision Workshops, pages 2165–2173, 2017

[287] S. Hong, H. Ko and J. Kim, "VICP: Velocity updating iterative closest point algorithm," 2010 IEEE International Conference on Robotics and Automation, Anchorage, AK, 2010, pp. 1893-1898.
[288] Granger, S., and X. Pennec (2002), Multi-scale EM-ICP: A fast and robust approach for surface registration, in Proceedings of the European Conference on Computer Vision, pp. 418–432, London, UK.
[289] Wentao Yuan, Benjamin Eckart, Kihwan Kim, Varun Jampani, Dieter Fox, and Jan Kautz. Deepgmr: Learning latent gaussian mixture models for registration. In European Conference on Computer Vision, pages 733–750. Springer, 2020
[290] Andy Zeng, Shuran Song, Matthias Nießner, Matthew Fisher, Jianxiong Xiao, and Thomas Funkhouser. 3dmatch: Learning local geometric descriptors from rgb-d reconstructions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 1802– 1811, 2017.
[291] Zan Gojcic, Caifa Zhou, Jan D Wegner, and Andreas Wieser. The perfect match: 3d point cloud matching with smoothed densities. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 5545–5554, 2019.
[292] Zhenpei Yang, Jeffrey Z. Pan, Linjie Luo, Xiaowei Zhou, Kristen Grauman, and Qixing Huang. Extreme relative pose estimation for rgbd scans via scene completion. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2019.
[293] Lingjing Wang, Jianchun Chen, Xiang Li, and Yi Fang. Non-rigid point set registration networks. arXiv preprint arXiv:1904.01428, 2019.
[294] Gil Elbaz, Tamar Avraham, and Anath Fischer. 3d point cloud registration for localization using a deep neural network auto-encoder. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 4631–4640, 2017.
[295] Huang, X., Mei, G., Zhang, J., & Abbas, R. (2021). A comprehensive survey on point cloud registration. ArXiv, abs/2103.02690.
[296] Z. Hu, Y. Hu, Y. Li and G. Huang, "Registration of image and 3D LIDAR data from extrinsic calibration," 2015 International Conference on Transportation Information and Safety (ICTIS), 2015, pp. 102-106, doi: 10.1109/ICTIS.2015.7232189.
[297] H. Yu, W. Zhen, W. Yang and S. Scherer, "Line-Based 2-D–3-D Registration and Camera Localization in Structured Environments," in IEEE Transactions on Instrumentation and Measurement, vol. 69, no. 11, pp. 8962-8972, Nov. 2020, doi: 10.1109/TIM.2020.2999137.
[298] M. Hermann, D. Grießer, B. Gundel, D. Dold, G. Umlauf and M. O. Franz, "Targetless Lidar-camera registration using patch-wise mutual information," 2022 25th International Conference on Information Fusion (FUSION), 2022, pp. 1-8, doi: 10.23919/FUSION49751.2022.9841290.

[299] Huang, Keli and Shi, Botian and Li, Xiang and Li, Xin and Huang, Siyuan and Li, Yikang, Multi-modal Sensor Fusion for Auto Driving Perception: A Survey, arXiv, 2022
[300] Bar-Shalom, Y. Multitarget-Multisensor Tracking: Applications and Advances, Volume III; Artech House Publishers: London, UK, 2000
[301] Hall, D.; Llinas, J. An introduction to multisensor data fusion. Proc. IEEE 1997, 85, 6–23
[302] Sasiadek, J.Z.; Hartana, P. Sensor data fusion using Kalman filter. In Proceedings of the Third International Conference on Information Fusion, Paris, France, 10–13 July 2000; Volume 2, pp. WED5/19–WED5/25.
[303] Qi, C.; Uwe, A. Anomaly Detection Using the Dempster-Shafer Method. In Proceedings of the International Conference on Data Mining (DMIN), Nagoya, Japan, 26–29 June 2016; pp. 232–240.
[304] Bahador, K.; Alaa, K.; Fakhri, K.; Saiedeh, R. Multisensor Data Fusion: A Review of the State-of-the-art. Inf. Fusion INFFUS 2013
[305] Asvadi, A.; Garrote, L.; Premebida, C.; Peixoto, P.; Nunes, U.J. Multimodal vehicle detection: Fusing 3D-LIDAR and color camera data. Pattern Recognit. Lett. 2018, 115, 20–29.
[306] Aijazi, A.K.; Checchin, P.; Trassoudaine, L. Multi sensorial data fusion for efficient detection and tracking of road obstacles for inter-distance and anti-colision safety management. In Proceedings of the 2017 3rd International Conference on Control, Automation and Robotics (ICCAR), Nagoya, Japan, 24–26 April 2017; pp. 617–621
[307] Kaempchen, N.; Buehler, M.; Dietmayer, K. Feature-level fusion for free-form object tracking using laserscanner and video. In Proceedings of the Intelligent Vehicles Symposium, Las Vegas, NV, USA, 6–8 June 2005; pp. 453–458.
[308] Premebida, C.; Monteiro, G.; Nunes, U.; Peixoto, P. A lidar and vision-based approach for pedestrian and vehicle detection and tracking. In Proceedings of the Intelligent Transportation Systems Conference, Seattle, WA, USA, 30 September–3 October 2007; pp. 1044–1049
[309] Garcia, F.; Musleh, B.; de la Escalera, A.; Armingol, J. Fusion procedure for pedestrian detection based on laser scanner and computer vision. In Proceedings of the 2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC), Washington, DC, USA, 5–7 October 2011; pp. 1325–1330
[310] Chen, X.; Ma, H.; Wan, J.; Li, B.; Xia, T. Multi-view 3d object detection network for autonomous driving. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 6526–6534
[311] R. Streubel and B. Yang, "Fusion of Stereo Camera and MIMO-FMCW Radar for Pedestrian Tracking in Indoor Environments," Int. Conf. on Information Fusion, pp. 565–572, 2016.

[312] S. K. Kwon, E. Hyun, J.-H. Lee, J.-H. Lee, and S. H. Son, "Detection scheme for a partially occluded pedestrian based on occluded depth in lidar-radar sensor fusion," Optical Engineering, vol. 56, no. 11, p. 113112, 2017.
[313] R. O. Chavez-Garcia and O. Aycard, "Multiple Sensor Fusion and Classification for Moving Object Detection and Tracking," IEEE Transactions on Intelligent Transportation Systems, vol. 17, no. 2, pp. 525–534, 2016
[314] K. Jo, M. Lee, J. Kim and M. Sunwoo, "Tracking and Behavior Reasoning of Moving Vehicles Based on Roadway Geometry Constraints," in IEEE Transactions on Intelligent Transportation Systems, vol. 18, no. 2, pp. 460-476, Feb. 2017, doi: 10.1109/TITS.2016.2605163.
[315] K. Lu, K. C. Chang, and R. Zhou, ''The exact algorithm for multi-sensor asynchronous track-to-track fusion,'' in Proc. 18th Int. Conf. Inf. Fusion, Washington, DC, USA, 2015, pp. 886–892.
[316] K. Chang, C.-Y. Chong, and S. Mori, "On scalable distributed sensor fusion," in Information Fusion, 2008 11th International Conference on, 2008, pp. 1-8.
[317] M. Dimitrievski, P. Veelaert, and W. Philips, "Behavioral pedestrian tracking using a camera and lidar sensors on a moving vehicle," Sensors (Switzerland), vol. 19, no. 2, 2019, DOI: 10.3390/s19020391.
[318] K. -E. Kim, C. -J. Lee, D. -S. Pae and M. -T. Lim, "Sensor fusion for vehicle tracking with camera and radar sensor," 2017 17th International Conference on Control, Automation and Systems (ICCAS), 2017, pp. 1075-1077, doi: 10.23919/ICCAS.2017.8204375.s
[319] R. Zhang and S. Cao, "Extending Reliability of mmWave Radar Tracking and Detection via Fusion With Camera," in IEEE Access, vol. 7, pp. 137065-137079, 2019, doi: 10.1109/ACCESS.2019.2942382.
[320] R. Zhang and S. Cao, ''Robust and adaptive radar elliptical density-based spatial clustering and labelling for mmWave radar point cloud,'' in Proc. Asilomar Conf. Signals, Syst., Comput., Nov. 2019
[321] Dai, X.; Khorram, S. Data fusion using artificial neural networks: A case study on multitemporal change analysis. Comput. Environ. Urban Syst. 1999, 23, 19–31
[322] Sourabh Vora, Alex H Lang, Bassam Helou, and Oscar Beijbom. Pointpainting: Sequential fusion for 3d object detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 4604–4612, 2020.
[323] Liang Xie, Chao Xiang, Zhengxu Yu, Guodong Xu, Zheng Yang, Deng Cai, and Xiaofei He. Pi-rcnn: An efficient multi-sensor 3d object detector with point-based attentive cont-conv fusion module. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 34, pages 12460– 12467, 2020
[324] Gregory P Meyer, Jake Charland, Darshan Hegde, Ankit Laddha, and Carlos Vallespi-Gonzalez. Sensor fusion for joint 3d object detection and

semantic segmentation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, pages 0–0, 2019.
[325] Tengteng Huang, Zhe Liu, Xiwu Chen, and Xiang Bai. Epnet: Enhancing point features with image semantics for 3d object detection. In European Conference on Computer Vision, pages 35–52. Springer, 2020.
[326] Jin Hyeok Yoo, Yecheol Kim, Ji Song Kim, and Jun Won Choi. 3d-cvf: Generating joint camera and lidar features using cross-view spatial feature fusion for 3d object detection. arXiv preprint arXiv:2004.12636, 3, 2020.
[327] Vishwanath A Sindagi, Yin Zhou, and Oncel Tuzel. Mvxnet: Multimodal voxelnet for 3d object detection. In 2019 International Conference on Robotics and Automation (ICRA), pages 7276–7282. IEEE, 2019.
[328] Su Pang, Daniel Morris, and Hayder Radha. Clocs: Camera-lidar object candidates fusion for 3d object detection. arXiv preprint arXiv:2009.00784, 2020.
[329] S. Gu, Y. Zhang, J. Tang, J. Yang, J. M. Alvarez and H. Kong, "Integrating Dense LiDAR-Camera Road Detection Maps by a Multi-Modal CRF Model," in IEEE Transactions on Vehicular Technology, vol. 68, no. 12, pp. 11635-11645, Dec. 2019, doi: 10.1109/TVT.2019.2946100
[330] Gledson Melotti, Cristiano Premebida, Nuno MM da S Gonc¸alves, Urbano JC Nunes, and Diego R Faria. Multimodal cnn pedestrian classification: a study on combining lidar and camera data. In 2018 21st International Conference on Intelligent Transportation Systems (ITSC), pages 3138–3143. IEEE, 2018.
[331] Xin Zhao, Zhe Liu, Ruolan Hu, and Kaiqi Huang. 3d object detection using scale invariant and feature reweighting networks. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 33, pages 9267–9274, 2019
[332] Jason Ku, Alex D Pon, Sean Walsh, and Steven L Waslander. Improving 3d object detection for pedestrians with virtual multi-view synthesis orientation estimation. arXiv preprint arXiv:1907.06777, 2019.
[333] Aasheesh Singh, Aditya Kamireddypalli, Vineet Gandhi, and K Madhava Krishna. Lidar guided small obstacle segmentation. arXiv preprint arXiv:2003.05970, 2020.
[334] Charles R Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J Guibas. Frustum pointnets for 3d object detection from rgb-d data. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 918–927, 2018.
[335] Nicolai Dorka, Johannes Meyer, and Wolfram Burgard. Modality-buffet for real-time object detection. arXiv preprint arXiv:2011.08726, 2020.
[336] Thomas, F.; Grzegorz, G. Optimal fusion of TV and infrared images using artificial neural networks. In Proceedings of the Applications and Science of Artificial Neural Networks, Orlando, FL, USA, 21 April 1995; pp. 919–925.

[337] Vinayaraj, P.; Weimin, W.; Ryosuke, N. A Point-Wise LiDAR and Image Multimodal Fusion Network (PMNet) for Aerial Point Cloud 3D Semantic Segmentation. Remote Sens. 2019, 11, 2961
[338] Liang, M.; Yang, B.; Chen, Y.; Hu, R.; Urtasun, R. Multi-Task Multi-Sensor Fusion for 3D Object Detection. In Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 15–20 June 2019; pp. 7337–7345.
[339] Luca, C.; Bellone, M.; Svensson, L.; Wahde, M. LIDAR-Camera Fusion for Road Detection Using Fully Convolutional Neural Networks. Robot. Auton. Syst. 2018, 111, 125–131
[340] Liandong, L. Multi-sensor Information Fusion Method Based on BP Neural Network. J. Online Biomed. Eng. 2016, 12, 53–57
[341] Chen, Z.; Li, S.; Yue, W. SOFM Neural Network Based Hierarchical Topology Control for Wireless Sensor Networks. J. Sens. 2014, 2014
[342] Carpenter, G.A.; Martens, S.; Ogas, O.J. Self-organizing information fusion and hierarchical knowledge discovery: A new framework using ARTMAP neural networks. Neural Netw. 2005, 18, 287–295
[343] Muresan, M.P.; Nedevschi, S.; Giosan, I. Real-time object detection using a sparse 4-layer LIDAR. In Proceedings of the 2017 13th IEEE International Conference on Intelligent Computer Communication and Processing (ICCP), Cluj-Napoca, Romania, 7–9 September 2017; pp. 317–322.
[344] Zeisler, J., and H. G. Maas. "ANALYSIS OF THE PERFORMANCE OF A LASER SCANNER FOR PREDICTIVE AUTOMOTIVE APPLICATIONS." ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences (2015): 49-56.
[345] W. Xin and J. Pu, "An Improved ICP Algorithm for Point Cloud Registration," 2010 International Conference on Computational and Information Sciences, Chengdu, China, 2010, pp. 565-568, doi: 10.1109/ICCIS.2010.144.
[346] F. Oniga and S. Nedevschi, "A Fast Ransac Based Approach for Computing the Orientation of Obstacles in Traffic Scenes," 2018 IEEE 14th International Conference on Intelligent Computer Communication and Processing (ICCP), 2018, pp. 209-214, doi: 10.1109/ICCP.2018.8516642.
[347] E. Waltz and J. Llinas, Multisensor Data Fusion, Artech House, Inc., Norwood, MA, 1990
[348] OpenAI, "GPT-4 Technical Report," arXiv:2303.08774 [cs.CL], Mar. 2023.
[349] M. P. Muresan, M. Raul, S. Nedevschi and R. Danescu, "Stereo and Mono Depth Estimation Fusion for an Improved and Fault Tolerant 3D Reconstruction," 2021 IEEE 17th International Conference on Intelligent Computer Communication and Processing (ICCP), Cluj-Napoca, Romania, 2021, pp. 233-240.

[350] M. P. Muresan and S. Nedevschi, "Multi-Object Tracking of 3D Cuboids Using Aggregated Features," 2019 IEEE 15th International Conference on Intelligent Computer Communication and Processing (ICCP), Cluj-Napoca, Romania, 2019, pp. 11-18
[351] M. P. Muresan, I. Giosan, and S. Nedevschi, "Stabilization and Validation of 3D Object Position Using Multimodal Sensor Fusion and Semantic Segmentation," Sensors, vol. 20, no. 4, p. 1110, Feb. 2020
[352] M. P. Muresan, S. Nedevschi, and R. Danescu, "Robust Data Association Using Fusion of Data-Driven and Engineered Features for Real-Time Pedestrian Tracking in Thermal Images," Sensors, vol. 21, no. 23, p. 8005, Nov. 2021
[353] M. P. Muresan, R. Danescu and S. Nedevschi, "Multi-Object Tracking, Segmentation and Validation in Thermal Images," 2023 IEEE Intelligent Vehicles Symposium (IV), Anchorage, AK, USA, 2023, pp. 1-8, doi: 10.1109/IV55152.2023.10186655.
[354] Brown T, Mann B, Ryder N, Subbiah M, Kaplan JD, Dhariwal P, Neelakantan A, Shyam P, Sastry G, Askell A, Agarwal S. Language models are few-shot learners. Advances in neural information processing systems. 2020;33:1877-901.

Mircea Paul Mureșan

About the Book

This book presents the foundations and recent advances of multimodal perception in autonomous systems, with a focus on depth perception, multi-object tracking, and multi-sensor fusion. By examining data from complementary sensors such as cameras, LiDAR, and RADAR, the work offers a rigorous framework for understanding and developing perception algorithms capable of processing, tracking, and fusing sensory information from complex real-world environments.

Combining state-of-the-art analysis with original research contributions, the volume emphasizes robust, fault-tolerant, and computationally efficient perception methods suitable for real-time deployment. Through its integration of geometric modelling, computer vision, and machine learning, the book provides a valuable resource for researchers, engineers, and specialists in autonomous vehicles, robotics, and intelligent sensing technologies