ANDRA PETROVAI

THE AI DRIVER

Visual Environment Perception with Deep Learning in Autonomous Vehicles

U.T.PRESS
Cluj-Napoca, 2025
ISBN 978-606-737-767-5

Andra Petrovai

The AI Driver: Visual Environment Perception with Deep Learning in Autonomous Vehicles

U.T. PRESS
Cluj-Napoca, 2025
ISBN 978-606-737-767-5

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. Serghi Nedevschi
Conf.dr. ing. Raluca Brehar

Pregătire format electronic on-line: Gabriela Groza

ISBN 978-606-737-767-5

Preface

The advancement of autonomous vehicles marks one of the most transformative shifts in modern technology, with visual perception playing a pivotal role in enabling these systems to navigate and interact with their surroundings. Autonomous driving relies on precise, real-time understanding of the environment, achieved through a combination of advanced sensors, artificial intelligence, and state-of-the-art algorithms.

This book delves deeply into the field of visual perception for autonomous vehicles, presenting both a comprehensive overview of existing methodologies and novel contributions developed through the author's research. Key focus areas include semantic segmentation, instance segmentation, panoptic segmentation, and depth estimation—all critical tasks that enable vehicles to build a detailed understanding of their environment. Each method presented in this book is analyzed in terms of computational efficiency, accuracy, and real-world applicability, with special attention to challenges like hardware limitations. The theoretical foundations and pragmatic solutions discussed are primarily derived from the author's extensive research and contributions to the field, including methods from the author's Ph.D. thesis titled "Deep Learning-based Visual Perception for Autonomous Driving" (2022).

This book is intended for senior undergraduate and graduate students, as well as early-career researchers seeking to deepen their understanding of the computational and algorithmic foundations of visual perception in autonomous driving. A foundational knowledge of image processing, deep learning, and computer vision will enable readers to engage with the advanced concepts discussed throughout.

By bridging theory and practice, this book aims not only to provide knowledge but also to inspire the development of safer, more efficient, and intelligent transportation systems.

Contents

Chapter 1

Introduction

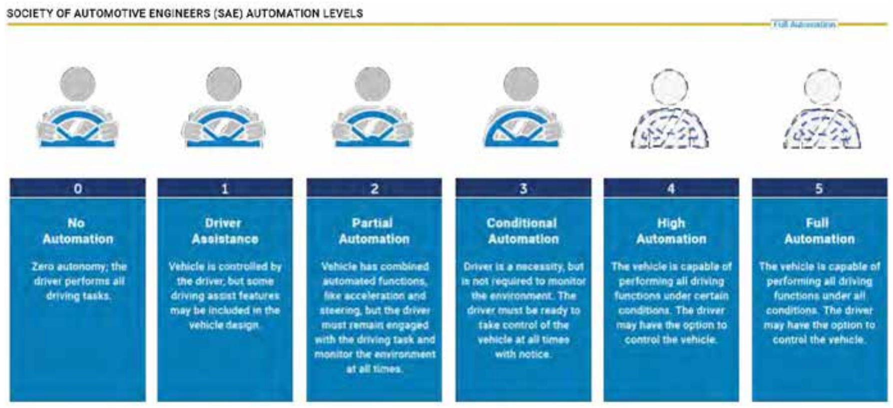
The evolution of the automotive industry is marked by advancements in driving assistance technologies, beginning with innovations like anti-lock braking systems (ABS), adaptive cruise control, and lane departure warnings. These technologies transitioned vehicles from purely passive safety features to active driver assistance systems, creating a foundation for increasingly intelligent capabilities. Today, the industry is moving beyond assistance systems toward the development of fully autonomous vehicles. This shift is driven by the potential to drastically reduce traffic accidents, enhance mobility, and transform transportation through the integration of advanced sensors, artificial intelligence, and state-of-the-art software. Autonomous vehicles have emerged as one of the most transformative research fields of the past decade, fueled by rapid advancements in hardware and software technologies.

Every year, road traffic accidents claim the lives of over 1.3 million people, with children and young adults aged 5 to 29 years being the most affected demographic [10]. Beyond the tragic loss of life, these accidents also result in millions of injuries and significant economic costs, including medical expenses, lost productivity, and infrastructure damage. A staggering 90% of these accidents are attributed to human error, such as speeding, fatigue, distracted driving, and delayed reactions, underscoring the limitations of human performance in dynamic traffic scenarios. This critical statistic has catalyzed efforts within both academia and industry to conceptualize a future of safer roads, underpinned by autonomous vehicles capable of faster, more precise reactions in critical situations and a reduced susceptibility to human mistakes. By leveraging advanced sensor technology, machine learning, and real-time data processing, autonomous systems promise to address the root causes of human error, potentially preventing accidents before they occur. As these efforts converge, the vision of a safer, more reliable transportation ecosystem is steadily becoming a reality.

As urbanization continues at an unprecedented pace, the United Nations projects that by 2050, 70% of the global population will reside in cities [7]. This rapid shift places immense pressure on urban mobility systems, with increasing vehicle numbers straining road infrastructures and public transportation networks. The resulting traffic congestion is not merely an inconvenience but a pervasive issue with profound ecological, economic, and societal implications. Daily traffic congestions in major cities leads to wasted fuel, lost productivity, and deteriorating air quality. In 2020, the estimated economic loss due to traffic congestion, primarily from wasted fuel and delays, exceeded \$100 billion [2], highlighting the urgent need for innovative solutions. Autonomous electric vehicles, complemented by services such as robotaxis, offer a transformative approach to addressing these urbanization-driven challenges. By optimizing traffic flow, reducing idle times, and enabling coordinated vehicle movements, these technologies can significantly decrease congestion and its associated costs. Beyond improving efficiency, autonomous vehicles contribute to combating climate change by reducing greenhouse gas emissions through electrification and smarter route management. For example, dynamic ride-sharing models powered by artificial intelligence can match passengers with similar routes, minimizing vehicle usage and maximizing efficiency.

Furthermore, autonomous mobility solutions promise to enhance inclusivity in transportation. Car-sharing and ride-hailing services integrated with autonomous technology are poised to become increasingly attractive, offering affordable and accessible options for individuals who cannot or choose not to drive. Aging and disabled populations, who often face mobility challenges, stand to benefit significantly from this shift, gaining independence and improved quality of life.

The implementation of autonomous vehicles also presents an opportunity to rethink urban planning. With fewer privately owned cars, cities could repurpose parking spaces and road infrastructure for green spaces, pedestrian zones, and bike lanes, fostering more sustainable and livable urban environments. Collectively, these advancements are not just about solving the problems of today but about shaping the cities of tomorrow—smarter, cleaner, and more connected.


SOCIETY OF AUTOMOTIVE ENGINEERS (SAE) AUTOMATION LEVELS

0 No Automation. Zero autonomy; the driver performs all driving tasks.

1 Driver Assistance. Vehicle is controlled by the driver, but some driving assist features may be included in the vehicle design.

2 Partial Automation. Vehicle has combined automated functions, like acceleration and steering, but the driver must remain engaged with the driving task and monitor the environment at all times.

3 Conditional Automation. Driver is a necessity, but is not required to monitor the environment. The driver must be ready to take control of the vehicle at all times with notice.

4 High Automation. The vehicle is capable of performing all driving functions under certain conditions. The driver may have the option to control the vehicle.

5 Full Automation. The vehicle is capable of performing all driving functions under all conditions. The driver may have the option to control the vehicle.

Figure 1.1: The six levels of autonomy.[1]

The Society of Automotive Engineers (SAE) [3] defines 6 levels of vehicle autonomy, ranging from level 0 (no automation) to level 5 (full automation) as seen in Figure 1.1. The significant technological progress in this domain from the last decade enabled the automotive industry to reach level 2 automation and the latest models of vehicles on the market are often equipped with driving assistance systems such as adaptive cruise control, lane keeping assistance, automatic braking and parking assistance. Companies such as Waymo, Zoox, Baidu and Tesla have already tested level 4 automated cars in limited areas of some cities. However, there are still unsolved challenges and lack of technological maturity, which delays the large-scale adoption of self-driving cars to the next 10 years. Research in this domain for higher levels of automation is very active, with topics ranging from key base technologies, to operation in long-tail scenarios, scaling to new locations and new hardware, adaptation to adverse weather and night conditions, and automatic labeling.

Autonomous vehicles need to see and understand the environment before navigating the world, just as a human driver would do. The autonomy stack, as seen in Figure 1.2, enables fully automated vehicles to safely drive to the desired destination. The stack is usually made of sensors, High-Definition (HD) maps and advanced software for perception, prediction, planning and control.

Vehicles rely on sensors such as cameras, LiDARs and radars to see the environment. Usually, they are mounted all around the vehicle to

[1]https://www.nhtsa.gov/

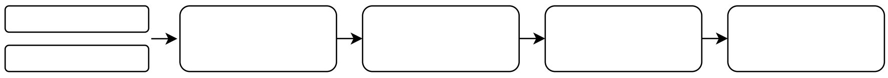


Figure 1.2: The autonomy stack.

enable 360° perception. Color cameras can be employed for daytime vision and provide rich appearance information which is useful for semantic and depth perception. During night, infrared cameras, which measure the apparent surface temperature of objects, can be used instead. Camera lenses are of several types and determine the field-of-view: standard lenses offer a horizontal field-of-view of 60° and a natural perspective, while wide-angle lenses, such as fisheye, can offer a horizontal field-of-view of 180° and are useful for surround perception. LiDARs (Light Detection and Ranging) sensors are able to determine the distance towards a target by illuminating the target with a laser light and measuring the return time. LiDARs provide a highly accurate 3D model of the environment by generating 3D point clouds. Disadvantages of LiDARs are the high cost compared to cameras or radars and the sparsity of the 3D point clouds. The radar uses radio waves to determine the distance, size and speed of objects. The depth from radar is not as accurate as from the other sensors, however it is robust in adverse weather conditions. A robust autonomy system should offer redundancy at the sensorial and algorithmic level to ensure safe operation of the vehicle on roads. This can be achieved by having parallel independent processing pipelines for each sensor (cameras, LiDARs, radars). A multi-modal pipeline which processes the combined outputs of different sensors could be also added to the system for increased robustness.

The first software module in the autonomy stack is the perception, which collects, interprets and understands the sensory information, with associated tasks such as 3D object detection, classification and tracking. Next in the stack is the prediction module, which receives the output from perception, predicts how the actors in a scene will move in the next few seconds and generates multiple plausible versions of the future. The planner takes the HD map data, the perception and prediction outputs and determines a safe trajectory and maneuvers for the autonomous vehicle. And finally, the control module is the software responsible for controlling the vehicle's acceleration, braking and steering.

This book explores a series of methods for environment perception using monocular cameras, including several novel approaches pioneered in our research. In autonomous navigation, vehicles must accurately perceive both the semantics and depth of their surroundings to make informed decisions. Monocular cameras, despite their simplicity, can achieve both: the rich visual information captured by images enables semantic understanding, while depth can be inferred either through stereo vision, multiple views, or motion from video sequences. For safe and reliable operation, an effective environment perception system must deliver high accuracy and real-time performance (10–30 frames per second) on low-power hardware to ensure timely decision-making and control. In the chapters that follow, we present a series of solutions addressing visual environment perception, highlighting both established techniques and novel approaches that tackle the unique challenges of autonomous driving. Particular emphasis is placed on methods that achieve both high accuracy and low processing times, making them suitable for integration into the perception software stack of automated vehicles, which demand real-time operation for safe and efficient navigation.

Chapter 2

Visual Environment Perception

Visual environment perception is a fundamental element of autonomous driving, enabling vehicles to understand and interact with their surroundings. This chapter provides an overview of the key tasks involved in visual perception, including semantic segmentation, instance segmentation, panoptic segmentation, video panoptic segmentation, and depth-aware video panoptic segmentation (DVPS). Together, these tasks contribute to constructing a semantic 4D representation of the environment, where spatial and temporal information is integrated into a cohesive framework for autonomous navigation.

Beyond describing these tasks, the discussion extends to the evaluation metrics commonly employed to assess the performance of perception algorithms, such as accuracy, precision, recall, and computational efficiency. These metrics offer a standardized way to compare methods and determine their suitability for real-world applications. The role of datasets in the development and benchmarking of visual perception algorithms is also emphasized, with high-quality labeled datasets being essential for training deep learning models. Some of the most influential datasets in the domain are examined, highlighting their characteristics and challenges.

The motivation for adopting deep learning is also explored, focusing on its transformative impact on computer vision and its ability to handle the complexity of autonomous driving tasks. Key challenges in visual environment perception are reviewed, including dynamic and diverse driving scenarios, the computational demands of deep models, and the reliance on large, high-quality datasets. Addressing these obstacles is crucial for designing efficient, real-time perception systems capable of meeting the demands of autonomous vehicles.

2.1 Tasks

One of the key tasks of visual environment perception is depth-aware video panoptic segmentation (DVPS), which reconstructs 4D panoptic point clouds from video sequences. These 4D panoptic point clouds augment 3D spatial points with semantic class labels and temporally consistent instance identifiers, offering a holistic, fine-grained representation of the environment. An example of DVPS output is illustrated in Figure 2.1. This representation provides all the essential information about road infrastructure elements and classified, tracked 3D objects such as vehicles and pedestrians. In DVPS, the environment is segmented into two main categories: *things* and *stuff*. *Things* are countable, dynamic objects in the scene, such as vehicles, pedestrians, and cyclists. Each thing is assigned a unique instance identifier that remains consistent over time, enabling object tracking across video frames. The 3D representation of a thing includes all spatial points belonging to the same semantic class and instance. *Stuff* are uncountable, static elements of the scene, such as road surfaces, sidewalks, vegetation, and sky. *Stuff* is represented by its semantic class, without the need for instance-level identification. In a traffic scenario, static infrastructure elements like the road, sidewalks, or traffic signs are categorized as *stuff*, providing a semantic understanding of the surrounding environment. The environment representation resulted from the DVPS task encapsulates all relevant information about the environment, including: the spatial structure of the scene in 3D as a 3D point cloud, semantic classifications for both *things* and *stuff* pixels and temporally consistent instance IDs for dynamic objects.

Depth-aware video panoptic segmentation encompasses two visual recognition sub-tasks: 1. **monocular depth estimation**, which estimates the corresponding 3D point for each pixel in the image 2. **video panoptic segmentation**, which aims at extending image-level panoptic segmentation to the video domain, by tracking instances across frames. Image **panoptic segmentation** unifies semantic and instance segmentation into a coherent output, where each pixel in the image is uniquely assigned a semantic class and an instance identifier. We present in Figure

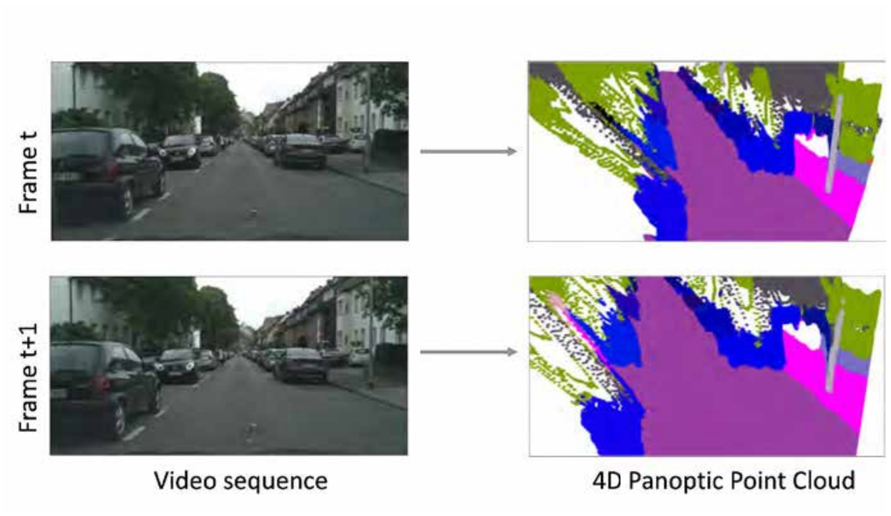


Figure 2.1: Depth-aware video panoptic segmentation is the task of restoring 4D panoptic point clouds from video sequences. On the right, each vehicle is depicted with a different shade of blue which is consistent across time.

2.2 an example of panoptic segmentation with the associated sub-tasks.

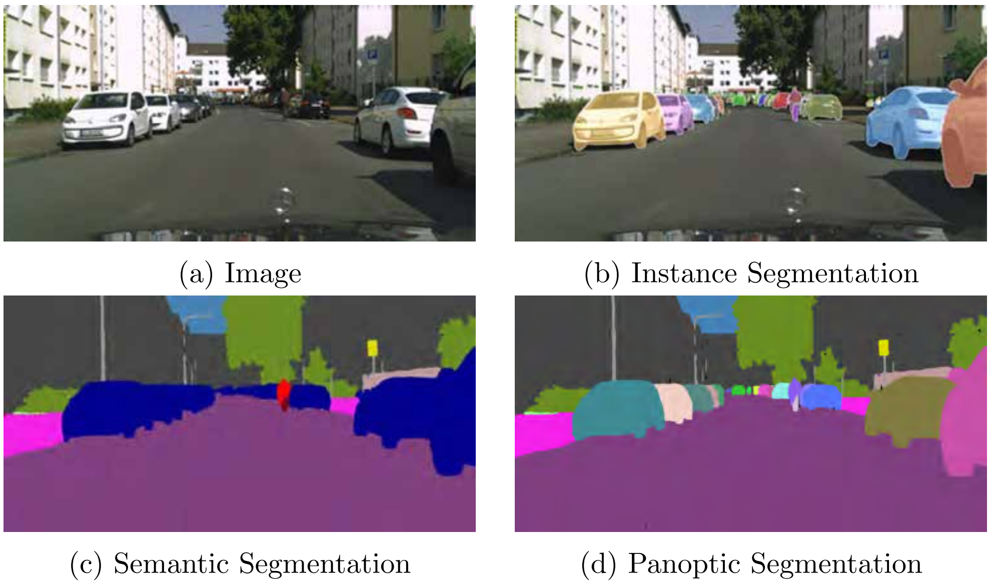

(a) Image
(b) Instance Segmentation
(c) Semantic Segmentation
(d) Panoptic Segmentation

Figure 2.2: Panoptic segmentation unifies the tasks of semantic and instance segmentation.

2.2 Deep Learning

To achieve the level of accuracy and efficiency required for autonomous navigation, traditional computer vision methods have increasingly given way to more advanced approaches. Among these, deep learning has emerged as a transformative paradigm, offering unparalleled capabilities for extracting and understanding complex visual information. The evolution of deep learning, particularly in computer vision, has been pivotal in addressing the challenges of environment perception, enabling robust semantic and spatial understanding from monocular camera data. By leveraging its ability to process large-scale data and adapt to diverse scenarios, deep learning has become the cornerstone of modern autonomous driving systems, forming the foundation for many of the methods discussed in this book.

Deep learning is a class of machine learning algorithms that uses neural networks for representation learning. Neural networks have been inspired by the human brain and are a collection of connected nodes called neurons which extract hierarchical features from input data. In computer vision, convolutional neural networks have been the dominant approach in the last decade due to their superior performance compared to the previous algorithms based on hand-crafted features. The success of deep learning has been attributed to: 1. availability of large datasets of consistent and high-quality labeled images 2. high-performance GPUs with parallel architectures that enable training of multi-layer (deep) neural networks in a short amount of time. Moreover, real-time performance is supported by the new generation of GPU devices and advanced studies in neural network optimization, allowing high inference speed on the new low-power GPUs.

2.3 Challenges

While deep learning has demonstrated exceptional capabilities in solving complex perception tasks, applying these methods to autonomous driving introduces significant challenges. The driving environment, particularly in urban areas, is characterized by its dynamic and diverse nature. Traffic scenes often include a wide range of road users, such as cars, buses, trucks, pedestrians, and cyclists, interacting unpredictably. Effective segmentation algorithms must not only manage this complexity but also address cases of occlusion, where objects are partially visible, and still provide accurate predictions for occluded road users.

Although traffic environments exhibit some structured elements, such as roads, sidewalks, and traffic signs, they also present considerable variability. Pedestrians, for example, differ greatly in appearance due to factors like clothing, posture, and body traits. Additionally, the scale of traffic participants can vary significantly within an image, depending on their distance from the camera and the perspective of the scene. To accurately segment large objects, deep learning models require a broad receptive field, often achieved through downsampling. However, preserving fine details is equally important for segmenting smaller objects, requiring a careful balance between scale and precision.

Deep learning-based segmentation methods also rely heavily on large, high-quality datasets for training. The accuracy of these algorithms depends on the quantity, consistency, and completeness of the labeled data. However, manually annotating images—particularly at the pixel level in video sequences—is labor-intensive and costly. To address this limitation, advancements in automatic and semi-automatic annotation, as well as semi-supervised and self-supervised learning techniques, must be explored.

Another critical challenge lies in achieving real-time performance on low-power hardware. While state-of-the-art segmentation algorithms excel on public benchmarks, their high computational cost makes them unsuitable for real-time applications. The complexity of deep network architectures becomes even more demanding when processing high-resolution images, large video datasets, or 360-degree surround views. In the context of autonomous driving, it is vital to develop efficient perception solutions that maintain a balance between computational efficiency and high performance, ensuring practical integration into real-world systems.

2.4 Datasets

Cityscapes. Cityscapes [34] is a dataset of 5000 high resolution 1024×2048 images with urban driving scenes, captured in several cities across Germany. The dataset provides depth maps from stereo [66] and semantic labels and instance-level annotations for 19 classes, of which 11 *stuff* and 8 *things* classes. The training set consists of 2975 images, the validation set has 500 images, while the test set has 1525 images.

COCO. COCO [97] is a large-scale dataset with common objects having panoptic labels for 80 *things* and 53 *stuff* classes. The dataset has 118K training images, 5K and 20K for validation and testing.

KITTI. KITTI [50] is a driving dataset captured in urban, rural and highway areas. KITTI is employed for training and evaluating depth estimation networks. The depth ground truth provided by KITTI is obtained by projecting raw LiDAR scans onto the image. The Eigen splits [41] with the pre-processing of Zhou *et al.* [171] remove static frames. The training set consists of 39810 image triplets, while the validation set has 4424 images. The reported results for depth are evaluated on 697 test images using Garg's crop [49]. There is also the improved KITTI ground truth [145], where the depth ground truth map is densified by considering 5 consecutive frames and improved by handling occlusions and object motion.

Cityscapes-DVPS. Cityscapes-DVPS [74] extends Cityscapes [34] to panoptic video by providing panoptic and depth annotations to every 5th frame from the 30-frame sequence in the original validation set. The annotations in a video snippet are temporally aligned with consistent instance IDs across frames. The dataset provides 3000, 300 and 300, training, validation and test images.

SemKITTI-DVPS. SemKITTI-DVPS [127] is based on the odometry split of the KITTI dataset [52, 16] and provides annotations for every frame in a long video sequence. The sparse panoptic annotations are obtained by projecting 3D point clouds acquired by LiDAR and augmented with semantic and temporally-consistent instance information to the image plane. The dataset contains 19020 training, 4071 validation and 4342 test images. Annotations are provided for 8 *things* classes and 11 *stuff* classes.

UP-Drive Dataset. This dataset is employed for training and evaluating deep neural networks for semantic and instance segmentation on 360° fisheye images and frontal narrow field-of-view images. The UP-Drive dataset is large and has been designed to capture a wide variety of outdoor weather and lighting conditions. The data was recorded by driving the car in several cities in northern Germany but also on highways and country roads. Recordings were performed in daytime and account for different lighting conditions, from morning to afternoon. Sequences were acquired in a time span of several months in three seasons: spring, summer and autumn. The data was recorded in diverse weather conditions such as sunny or cloudy weather but also in heavy rain. Lens flare, but also lens distortions from rain drops have been captured. From all the recordings, 19562 non-sequential fisheye images were selected for semantic and instance-level annotation. Images cover the surrounding view of the vehicle: front, left, right and rear. There are 5111 front view images, 4684 left view images, 4800 right view images, 4967 rear view images. The UP-Drive dataset was labeled using similar methodology with the Cityscapes dataset [34]. Pixel-level semantic segmentation annotation is provided for all images into 23 classes, and also instance-level labels for a subset of 6 classes that represent traffic participants. The dataset has 15782 images in the training set and 3780 images in the test set. The image dataset of narrow field-of-view images is relatively less diverse and smaller than the fisheye dataset. It contains 1869 images which are labeled with pixel-level instance masks for 6 classes. The training set has 1495 images and the number of test images is 374.

The raw fisheye images are not used in practice since objects in the image are highly distorted due to the wide-angle lenses. Therefore, image undistortion and unwarping is applied in order to obtain a more suitable representation of the scene in the image. Unwarping is the process of backprojecting the fisheye image onto a virtual projection surface. In this case, a cylindrical unwarping process [146] is adopted which generates images with large horizontal field-of-view (HFV) and small distortions, while also preserving the orientation of vertical lines. In the image unwarping process, the horizontal field-of-view is reduced from 185° to 160°. A visual comparison of fisheye and unwarped fisheye images is provided in Figure 2.3.

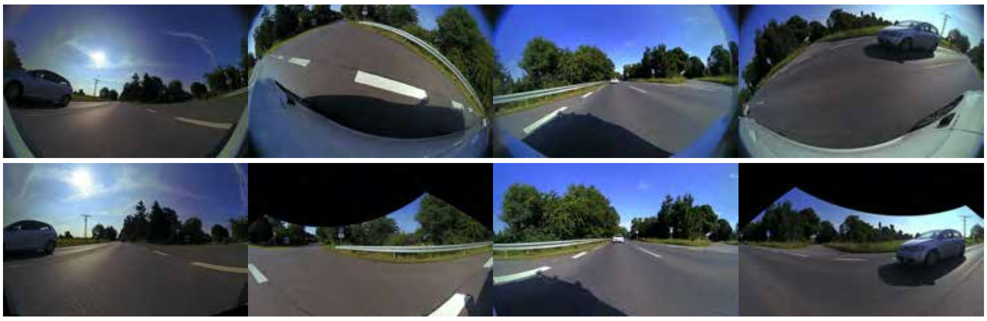


Figure 2.3: Results of the fisheye images unwarping process. First row: fisheye images, second row: cylindrical projection of the fisheye images. From left to right: front view, right view, rear view and left view.

2.5 Metrics

Semantic Segmentation Metrics. The performance of semantic segmentation models is typically evaluated using metrics such as Intersection over Union (IoU) and Mean Intersection over Union (mIoU). IoU is calculated for each class as the ratio between the area of overlap between the predicted and ground truth segments and the area of their union:

$$IoU = \frac{|Prediction \cap GroundTruth|}{|Prediction \cup GroundTruth|} \tag{2.1}$$

The mIoU metric averages the IoU values across all classes, providing a single scalar measure of segmentation performance.

Instance Segmentation. The standard metric used for evaluating instance segmentation is the Mean Average Precision (mAP) [97]. This metric is computed as the average of the Average Precision (AP) values across all classes and 10 IoU thresholds, ranging from 0.5 to 0.95 with a step size of 0.05.

To calculate mAP, AP is first computed for each class and IoU threshold. The process begins by matching instance mask predictions to ground truth masks, with each prediction categorized as a true positive, false positive, or false negative. Predictions are sorted by confidence, and starting from the highest confidence score, they are matched to the ground truth. A prediction is considered a true positive if its IoU with a ground truth mask exceeds the threshold; otherwise, it is classified as a false positive. Precision and recall are then computed as follows:

$$Precision = \frac{TP}{TP + FP} \tag{2.2}$$

$$Recall = \frac{TP}{TP + FN} \tag{2.3}$$

Here, TP + FP represents the total number of predicted objects, and TP + FN represents the total number of ground truth objects. The AP is defined as the area under the Precision-Recall curve and is computed as the mean precision at 101 equally spaced recall levels [97]:

$$AP = \frac{1}{101} \sum_{r \in 0, 0.01, \ldots, 1} p_{interp}(r) \tag{2.4}$$

The precision at each recall level is interpolated as the maximum precision for any recall greater than or equal to r:

$$p_{interp}(r) = \max_{\tilde{r} \geq r} p(\tilde{r}) \tag{2.5}$$

Panoptic Segmentation. Panoptic Quality (PQ) [76] is the standard metric for evaluating panoptic segmentation performance. PQ is calculated for each class and then averaged across all classes. The evaluation process begins with a segment matching step, where each predicted segment is uniquely matched to a ground truth segment with the highest IoU, provided the IoU exceeds 0.5. Pixels from matched predicted segments are classified as true positives, while pixels from unmatched predicted segments are considered false positives. Unmatched ground truth segments are categorized as false negatives.

Unknown pixels may be labeled as void in the ground truth or the predictions. Segments matched to void-labeled ground truth segments are excluded from the PQ calculation, and unmatched predicted segments containing void pixels are not counted as false positives.

The PQ metric is defined as:

$$PQ = \frac{\sum_{p,g \in TP} IoU(p,g)}{|TP| + \frac{1}{2}|FP| + \frac{1}{2}|FN|} \tag{2.6}$$

PQ can also be expressed as the product of Segmentation Quality (SQ) and Recognition Quality (RQ), where:

$$PQ = SQ \times RQ \tag{2.7}$$

The components are defined as follows:

$$SQ = \frac{\sum_{p,g \in TP} IoU(p,g)}{|TP|} \tag{2.8}$$

$$RQ = \frac{|TP|}{|TP| + \frac{1}{2}|FP| + \frac{1}{2}|FN|} \tag{2.9}$$

In this formulation, RQ is equivalent to the F1 score commonly used in classification tasks, while SQ represents the average IoU for matched segments. Panoptic segmentation performance is also evaluated separately for *things* (countable objects) and *stuff* (amorphous regions) using the PQTh and PQSt metrics.

Video Panoptic Segmentation. VPQ (Video Panoptic Quality) [74] is a metric used to evaluate the accuracy of video panoptic segmentation. Given a video sequence with T frames, a temporal window of k consecutive frames is selected. For a k-frame video snippet $I^{t:t+k} = \{I^t, I^{t+1}, ..., I^{t+k}\}$, a tube prediction is defined as the track of its frame-level segments $\hat{u}(c, z) = \{p^t, ..., p^{t+k}\}$, where c represents the semantic class and z is the instance identifier.

Predicted tubes and ground truth tubes are matched, and the IoU between them is computed. Each frame-level predicted segment in a tube is matched to the corresponding frame-level segment in the ground truth tube. Pixels from a matched frame-level predicted segment are considered true positives (TP) if they have the same semantic class and the IoU is greater than 0.5. Pixels from unmatched predicted segments are considered false positives (FP), while those from unmatched ground truth segments are treated as false negatives (FN).

The IoU, TP, FP, and FN metrics are computed for each k-frame window within the video snippet, sampled using a stride s. The stride s is particularly useful when the ground truth is annotated at intervals of s frames. The temporal sliding window is applied over the video sequence $0 : T - k : s$, aggregating the IoU, TP, FP, and FN metrics across all windows.

The VPQ for a k-frame video snippet and C classes is calculated as:

$$VPQ^k = \frac{1}{C} \sum_c \frac{\sum_{(u,\hat{u}) \in TP_c} IoU(u, \hat{u})}{|TP_c| + \frac{1}{2}|FP_c| + \frac{1}{2}|FN_c|} \tag{2.10}$$

The final VPQ for the entire video sequence is computed as the average over all k-frame windows:

$$VPQ = \frac{1}{K} \sum_k VPQ^k \tag{2.11}$$

Depth-aware Video Panoptic Segmentation. DVPQ is the primary metric for the task. It represents an extension of VPQ, but also considers the absolute relative depth errors λ. DVPQ$^k_\lambda$ is equal to VPQ^k computed for all pixels that have the absolute relative depth error smaller than λ. We consider n the number of λ values, which is usually equal to 3. The final DVPQ metric is computed as:

$$DVPQ = \frac{1}{K}\frac{1}{n}\sum_{k}\sum_{\lambda} DVPQ_{\lambda}^{k} \tag{2.12}$$

Monocular Depth Evaluation. The performance of monocular depth estimation is evaluated using a combination of standard metrics, including Absolute Relative Error (absRel), Squared Relative Error (sqRel), Root Mean Square Error (RMS), Root Mean Square Log Error (RMSlog), and Depth Inlier Metrics ($\delta < 1.25, \delta < 1.25^2, \delta < 1.25^3$). These metrics collectively assess the accuracy and reliability of the predicted depth values compared to the ground truth. In this context, $\hat{d}$ represents the predicted depth, d is the ground truth depth, and N is the total number of depth values in the dataset.

The metrics are defined as follows:

- **Absolute Relative Error (absRel)** measures the relative difference between predicted and ground truth depths, normalized by the ground truth depth:

$$absRel = \frac{1}{N}\sum \frac{|d - \hat{d}|}{d} \tag{2.13}$$

- **Squared Relative Error (sqRel)** evaluates the squared difference between the predicted and ground truth depths, normalized by the ground truth depth:

$$sqRel = \frac{1}{N}\sum \frac{(d - \hat{d})^2}{d} \tag{2.14}$$

- **Root Mean Square Error (RMS)** calculates the root of the mean squared difference between predicted and ground truth depths:

$$RMS = \sqrt{\frac{1}{N}\sum (d - \hat{d})^2} \tag{2.15}$$

- **Root Mean Square Log Error (RMSlog)** measures the root of the mean squared logarithmic difference between predicted and ground truth depths, emphasizing relative differences:

$$RMSlog = \sqrt{\frac{1}{N}\sum (\log(d) - \log(\hat{d}))^2} \tag{2.16}$$

- **Depth Inlier Metrics (δ_t)** assess the percentage of predicted depths that fall within a certain threshold t of the ground truth. These thresholds (1.25, 1.25^2, and 1.25^3) reflect increasing tolerances for prediction errors:

  $$\delta_t = \frac{1}{N} \sum \mathbb{I}(\max(\frac{d}{\hat{d}}, \frac{\hat{d}}{d}) < t), \quad t \in \{1.25, 1.25^2, 1.25^3\} \tag{2.17}$$

  Here, $\mathbb{I}(\cdot)$ is the indicator function, which evaluates to 1 if the condition is true and 0 otherwise.

Chapter 3

Semantic Segmentation

This chapter examines key research directions and significant contributions in semantic segmentation, a foundational deep learning-based pixel-level recognition task. In addition to providing an overview of the field, it highlights a selection of influential works for detailed analysis, focusing on advancements in network architectures, challenges in handling complex visual environments, and strategies to enhance accuracy and computational efficiency for real-world applications.

3.1 Overview

Semantic segmentation partitions an image into meaningful segments, which share a common representation. Each pixel in the image receives a semantic class that belongs to either *stuff* or *things* categories. *Stuff* classes represent amorphous and uncountable elements in the scene that usually have repetitive texture, but not a fixed shape or size. In the driving environment, examples of *stuff* classes include road, sidewalk, building, nature. On the other hand, *things* classes define objects that can be counted and have a specific shape. Road users belong to this category: vehicles, pedestrians and cyclists. State-of-the-art semantic segmentation methods use deep learning for dense pixel prediction, which we review next.

Fully Convolutional Networks. Convolutional neural networks (CNN) have been extensively used for the classification task and Long *et al.*[100] adapted them for semantic segmentation by introducing the Fully Convolutional Neural Network (FCN) as seen in Figure 3.1. One of the major benefits of the FCN is that it removes the fixed input

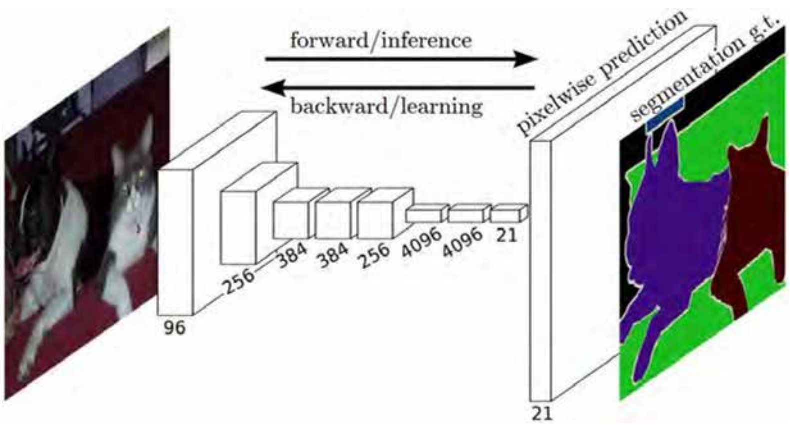


Figure 3.1: The fully convolutional neural network (FCN) [100].

size precondition by replacing the fully connected layers with convolutional layers. FCN [100] uses skip connections by fusing low-level feature maps having fine appearance information with high-level feature maps having coarser information from the deep layers of the network.

Networks with Context Modules. Coarse grained representations provide better localization and stronger context information, while high resolution features provide details of finer scales such as shape and boundaries. As both aspects are equally important for semantic segmentation, several mechanisms have been proposed to achieve good localization and to maintain image details. In order to incorporate context information, the convolutional layers from the deeper levels of the network have been replaced with atrous (dilated) convolutions [21, 22] which apply the convolution on sparsified sampling locations. These convolutions allow control of the output resolution and enlarge the field of view of filters by capturing multi-scale context without decimating the resolution. Therefore, in an atrous (dilated) FCN, the output resolution is typically $8\times$ or $16\times$ lower than the input size, compared to FCNs where the output resolution is $32\times$. Bilinear upsampling is applied on top of the last layer to obtain the final segmentation map. An alternative to atrous convolutions are the scale-adaptive convolutions [166]. The authors overcome the problem of fixed-size receptive fields by introducing adaptive convolutions which are capable of learning dilation rates. Deformable convolutions [36] generalize the atrous convolution by learning 2D offsets to the regular grid of sampling locations. Another line of work proposes context modules such as Pyramid Pooling Module

[168] or Atrous Spatial Pyramid Pooling (ASPP) [21, 22] on top of the dilated FCNs. The Pyramid Pooling Module applies four parallel average pooling operations, which results in feature maps of sizes: $[1 \times 1]$, $[2 \times 2]$, $[3 \times 3]$ and $[6 \times 6]$. Next, a $[1 \times 1]$ convolution operation and upsampling to 1/8 follow at each pyramid level. The input of the module and its output are concatenated and the multi-scale features are fused with a pointwise convolution. The Atrous Spatial Pyramid Pooling (ASPP) as seen in Figure 3.2 [21, 22] captures multi-scale representations with parallel atrous convolutions with different rates. The resulting feature maps at different scales are concatenated and then bilinearly upsampled to the original resolution.

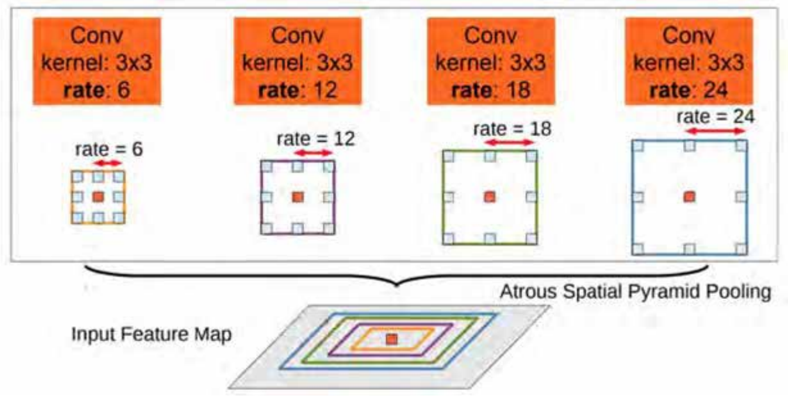


Figure 3.2: Atrous Spatial Pyramid Pooling (ASPP) [21].

Another approach to capture multi-scale information is to resize the input samples at different resolutions and use a shared feature extractor [44]. The resulting feature maps at different scales are aggregated with concatenation [94] or attention models [23].

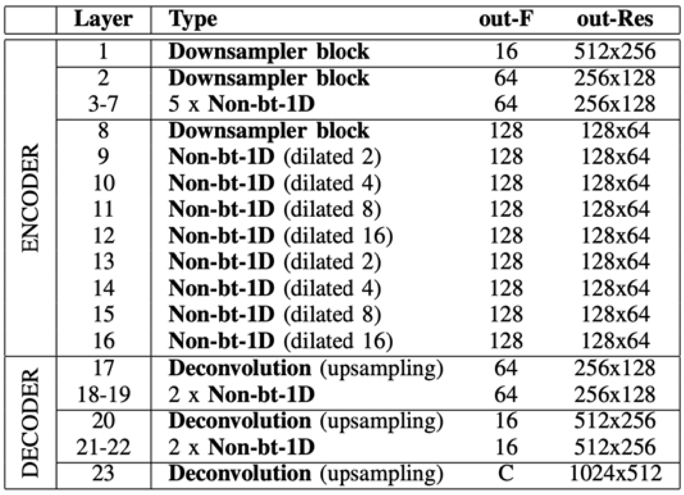
Encoder-Decoder. Networks using atrous convolutions and Spatial Pyramids have a large memory footprint due to the fact that feature maps are generated at higher resolutions. Therefore, a simple and fast bilinear interpolation operation is used to recover the original resolution. On the other hand, encoder-decoder networks usually use a deeper and narrower CNN for feature extraction and a more complex decoder replaces bilinear interpolation. The encoder network learns hierarchical feature representations, while the decoder network upsamples feature maps to the input image resolution and recovers spatial information. Encoder-decoder models achieve outstanding performance by learning the upsampling layers in the decoder. The ENet [112] model has a lightweight encoder and deconvolution layers are used to learn the upsampling of low resolution features. The network runs in real time at the cost of reduced performance. ERFNet [131] achieves a better trade-off between accuracy and efficiency by proposing the factorized convolution. SegNet [14] has a symmetric encoder-decoder architecture and introduces the unpooling layer for upsampling, which transfers maxpooling indices from the encoder module to the decoder. The U-net model [132] uses shortcut connections from the encoder to decoder to help recover object details and spatial information. RefineNet [93, 149] exploits a multi-path refinement network with long-range residual connections.

Transformer-based networks. Transformer-based semantic segmentation networks have emerged as a transformative advancement in computer vision, leveraging the self-attention mechanism of transformers to model long-range dependencies and global context in images. Unlike traditional convolutional neural networks (CNNs), which rely on local receptive fields, transformer architectures excel at capturing relationships across the entire image, making them highly effective for dense prediction tasks like semantic segmentation. Models such as MaskFormer [30], Mask2Former [29], SegFormer [158] have demonstrated state-of-the-art performance by combining the strengths of transformers with hierarchical and adaptive feature extraction.

Next, we discuss a few influential works for semantic segmentation.

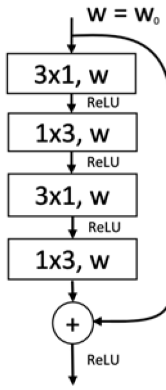
3.2 ERFNet

Method. ERFNet [131] is a fast and accurate network with an encoder-decoder architecture. The building block of ERFNet is the factorized residual layer as seen in Figure 3.3b. This layer represents a 1D non-bottleneck residual module that decomposes a 2D kernel into a linear combination of 1D kernels. In this design, each 3×3 convolution is transformed into 3×1 and 1×3 convolutions. The number of parameters is reduced with 33% when using a kernel size of 3. At the same time, the network is much more memory efficient and faster while having an increased capacity which results in a high segmentation accuracy similar to more complex models. The feature extractor encodes features at three scales: 1/2, 1/4, 1/8 from the original input resolution by stacking residual 1D non-bottleneck blocks with dilated convolutions. The lightweight decoder is formed of 1D non-bottleneck blocks and recovers spatial and semantic information from the last layer of the encoder. The detailed architecture is presented in Figure 3.3a.



	Layer	Type	out-F	out-Res
ENCODER	1	**Downsampler block**	16	512x256
	2	**Downsampler block**	64	256x128
	3-7	5 x **Non-bt-1D**	64	256x128
	8	**Downsampler block**	128	128x64
	9	**Non-bt-1D** (dilated 2)	128	128x64
	10	**Non-bt-1D** (dilated 4)	128	128x64
	11	**Non-bt-1D** (dilated 8)	128	128x64
	12	**Non-bt-1D** (dilated 16)	128	128x64
	13	**Non-bt-1D** (dilated 2)	128	128x64
	14	**Non-bt-1D** (dilated 4)	128	128x64
	15	**Non-bt-1D** (dilated 8)	128	128x64
	16	**Non-bt-1D** (dilated 16)	128	128x64
DECODER	17	**Deconvolution** (upsampling)	64	256x128
	18-19	2 x **Non-bt-1D**	64	256x128
	20	**Deconvolution** (upsampling)	16	512x256
	21-22	2 x **Non-bt-1D**	16	512x256
	23	**Deconvolution** (upsampling)	C	1024x512

(a) ERFNet architecture.



(b) The non-bottleneck block of ERFNet.

Figure 3.3: The architecture and non-bottleneck block of ERFNet [131].

The cross-entropy loss is commonly used for training semantic segmentation networks. This loss is designed for classification tasks and measures the difference between the predicted probability distribution and the ground truth distribution. For semantic segmentation, the loss is computed over all pixels in the image, treating each pixel as an independent classification problem.

$$\mathcal{L}_{ce}(p, y) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} y_{i,c}\log(p_{i,c}) \tag{3.1}$$

where N is the total number of pixels in the image, C is the number of classes, $y_{i,c}$ is a one-hot encoded target indicating whether pixel i belongs to class c and $p_{i,c}$ is the predicted probability of pixel i belonging to class c as output by the segmentation network.

Experiments. The network is trained on all four unwarped fisheye images of the UP-Drive dataset. The model is developed in the PyTorch [113] framework. The network is trained for 150 epochs with a batch size of 12 images per GPU and a polynomial learning rate decay starting from 0.0025. The cross entropy loss function is optimized with the Adam optimizer. The images are cropped from 800×1280 to 640×1280 and augmentations are applied such as random horizontal flipping and random left-right translation during training. The network is initialized with pretrained weights on the Cityscapes dataset [34].

The ERFNet semantic segmentation network was integrated into a demo autonomous vehicle [120, 45] and deployed within a C++ ADTF project to support real-time perception tasks. This integration required optimizing the network for efficient inference while maintaining sufficient accuracy to meet the demands of the autonomous system. The execution time is measured on a NVIDIA GTX 1080 GPU with a batch size of 1.

Initially, the lack of integration options between PyTorch models and C++ projects necessitated the development of a custom cuDNN-based framework, leveraging the NVIDIA CUDA Deep Neural Network library [4]. This framework enabled the seamless deployment of the network and provided equivalent inference performance to the native PyTorch framework. The integration process focused on balancing accuracy and efficiency to meet the constraints of the autonomous system. Inference time was measured using this custom framework and found to be equivalent to the PyTorch framework. The semantic segmentation network was trained using unwarped fisheye images from all four views. Employing the same model for all images enables accelerated inference by processing the batch of four images simultaneously. Using a resolution of 256×512, the model achieved 64.52 mIoU with an inference time of 15 ms per image and 60 ms for all four views. While full resolution provided an accuracy improvement of over 3%, it was computationally expensive and unsuitable for the system. The system integration utilized the model trained on 256×512 images within the custom cuDNN-based framework, balancing accuracy and processing speed by lowering the resolution.

The release of the TensorRT [6] library enabled network optimization and reduced inference time while processing full-resolution images of 640×1280. TensorRT also offers routines for calibration for lower precision (INT8). After network optimization and using FP32 precision, the network achieves high accuracy of 67.87% with a reduced inference time of 20 ms per image. Switching to INT8 precision reduces accuracy by 2.7% but significantly improves performance, allowing all four unwarped fisheye images to be semantically segmented in 36 ms. Ultimately, 8-bit inference with network quantization was adopted, calibrating the network on 300 images, and the INT8-optimized model was integrated into the perception software. In Figure 3.4 we present a few visual results on

Resolution	mIoU	Time (ms)
Custom CUDNN-based framework		
640 × 1280	67.87	38
256 × 512	64.52	15
TensorRT optimization		
640 × 1280 - FP32	67.87	20
640 × 1280 - INT8	**65.10**	**9**

Table 3.1: Evaluation of the semantic segmentation network on fisheye images corresponding to front, left, back, right views on the UPDrive dataset. 640 × 1280 - INT8 is integrated into the demo autonomous vehicle. Time is measured on a NVIDIA GTX 1080 GPU.

the UP-Drive dataset.

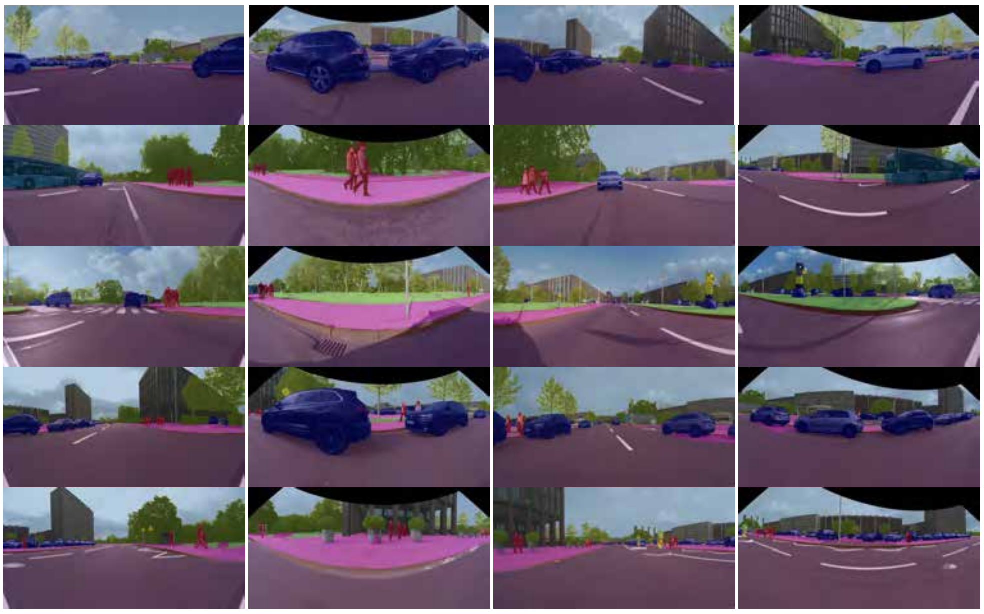


Figure 3.4: Semantic segmentation of unwarped fisheye images. The system processes four images from the fisheye 160° horizontal field-of-view cameras which provide 360° coverage around the vehicle. Each camera views a different direction around the vehicle: front, right, rear and left.

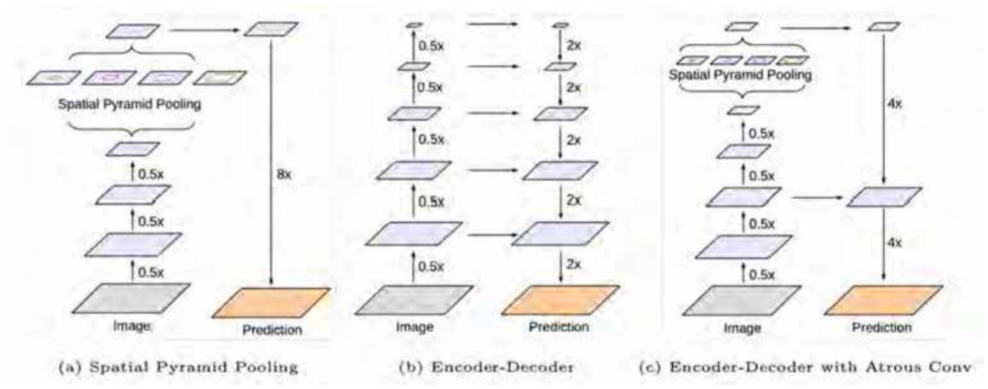


Figure 3.5: DeepLabV3+ [24] improves over DeepLabv3 [22], seen in (a), which employs spatial pyramid pooling module, by using a decoder similar to architecture in (b). Image from [24].

3.3 DeepLabV3

Method. DeepLabv3+ [24] is an advanced semantic segmentation model that builds upon its predecessor, DeepLabv3 [22], by introducing an encoder-decoder architecture designed to enhance the capture of multi-scale contextual information and refine object boundaries, as seen in Figure 3.5. This improvement addresses the challenges of achieving accurate segmentation results while maintaining computational efficiency.

The encoder in DeepLabv3+ utilizes an Atrous Spatial Pyramid Pooling (ASPP) module to extract multi-scale contextual information from input features. The ASPP employs atrous convolution, also known as dilated convolution, with varying rates to probe the feature maps at multiple scales. This approach enables the model to capture both local and global context effectively. Additionally, image-level features are incorporated within the ASPP module to further enhance the global representation, making the encoder robust in handling complex scenes.

The decoder module in DeepLabv3+ plays a critical role in refining the segmentation outputs, particularly along object boundaries. The decoder upsamples the output of the encoder and concatenates it with low-level features extracted from earlier layers of the backbone network. This concatenation helps recover spatial details that may have been lost during the encoding process. The combined features are then refined through a series of 3×3 convolutions before being upsampled again to match the resolution of the input image. The architecture of this model

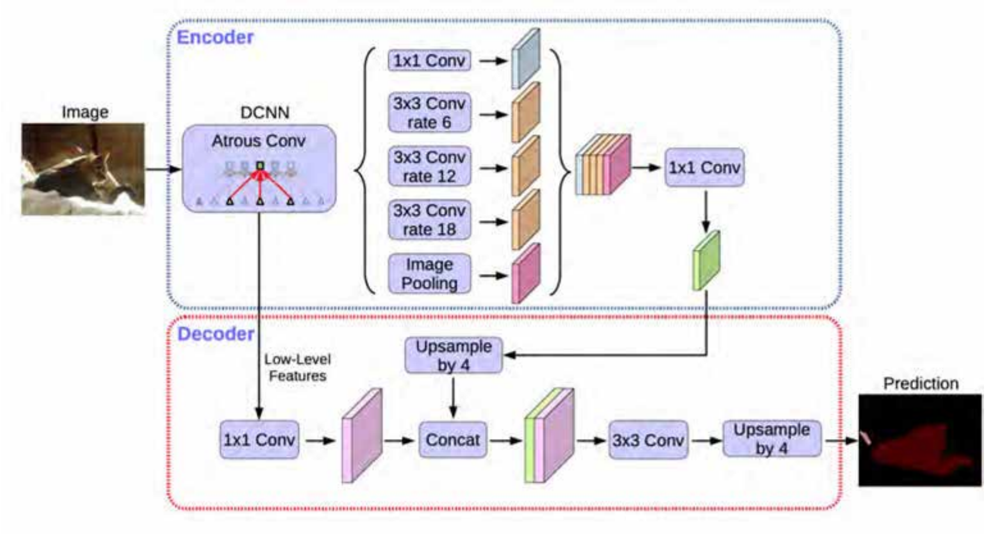


Figure 3.6: The architecture of DeepLabV3+. Image from [24].

can be seen in Figure 3.6.

To improve computational efficiency, DeepLabv3+ integrates depthwise separable convolutions in both the ASPP and decoder modules, as seen in Figure 3.7. This strategy significantly reduces the computational cost and memory requirements of the model without compromising segmentation accuracy. The architecture also supports flexible backbones, such as ResNet-101 [64] and the more advanced Xception [33] model. The Xception backbone is adapted to include atrous separable convolution, enabling high-resolution feature extraction while maintaining computational efficiency.

DeepLabv3+ excels in its ability to recover fine object boundaries and capture multi-scale features, which are essential for semantic segmentation tasks. By combining the ASPP module and the decoder, the model achieves improved accuracy and robustness in segmenting complex scenes. Its computational efficiency, achieved through the use of depthwise separable convolutions, makes it a practical choice for real-world applications where resource constraints are a concern.

Experiments. DeepLabv3 is trained on the Cityscapes dataset using a crop size of 513×513 pixels, which balances computational efficiency and segmentation performance. The training process employs an initial learning rate of 0.007, adjusted using a polynomial decay policy where the learning rate scales as $(1 - \frac{\text{iter}}{\text{max\_iter}})^{0.9}$ over 90k iterations.

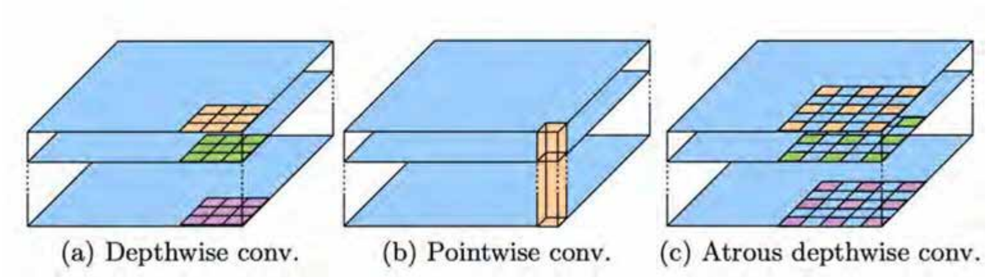


Figure 3.7: A 3 × 3 depthwise separable convolution splits a standard convolution operation into two steps: (a) a depthwise convolution, which applies a single filter to each input channel independently, and (b) a pointwise convolution, which combines the outputs of the depthwise convolution across all channels. In this approach, atrous separable convolution is explored, where atrous convolution is incorporated into the depthwise convolution step, as illustrated with a dilation rate of 2. Image from [24].

Backbone	**Decoder**	**ASPP**	**Image-Level**	**mIoU**
X-65	✓		✓	77.33
X-65	✓	✓	✓	78.79
X-65		✓	✓	79.14
X-71	✓	✓	✓	79.55

Table 3.2: Validation set results for DeepLabv3+ [24] on Cityscapes.

The model is optimized using Stochastic Gradient Descent (SGD) with a momentum of 0.9 and a weight decay of 0.0001. To enhance generalization, standard data augmentation techniques are applied, including random scaling within a range of [0.5, 2.0], random cropping to the target size, and random horizontal flipping. The backbone network, typically ResNet-101 or Xception, incorporates atrous convolutions to capture multi-scale contextual information effectively. During evaluation, the model processes the full image resolution of 1024 × 2048 pixels and often employs multi-scale testing and left-right flipping to further improve accuracy. These settings enable DeepLabv3+ to achieve state-of-the-art performance on semantic segmentation tasks while maintaining computational feasibility. The results on the Cityscapes dataset are displayed in Tables 3.2 and 3.3.

Method	Coarse	mIoU
ResNet-38 [157]	✓	80.6
PSPNet [168]	✓	81.2
Mapillary [133]	✓	82.0
DeepLabv3 [22]	✓	81.3
DeepLabv3+ [24]	✓	82.1

Table 3.3: Test set results for Cityscapes. Coarse indicates the use of the train_extra set from Cityscapes, with coarsely labeled images.

3.4 MaskFormer

Method. MaskFormer [30] is a powerful and versatile transformer-based framework designed to unify multiple segmentation tasks, including semantic segmentation, instance segmentation, and panoptic segmentation. Unlike traditional segmentation methods, which often rely on per-pixel classification, MaskFormer introduces a novel mask classification paradigm. This approach shifts the focus from predicting pixel-level labels to predicting a set of binary masks, each associated with a specific class label, as seen in Figure 3.8. By decoupling segmentation and classification, MaskFormer provides a unified and highly efficient solution for diverse segmentation tasks.

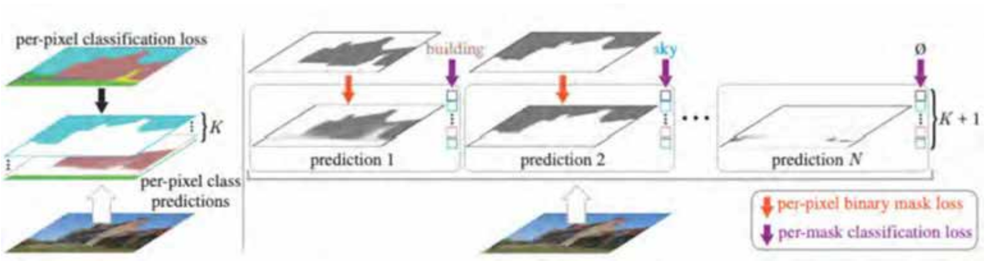

per-pixel classification loss
K
per-pixel class predictions
building
sky
Ø
prediction 1
prediction 2
prediction N
K + 1
per-pixel binary mask loss
per-mask classification loss

Figure 3.8: Semantic segmentation using per-pixel classification applies the same classification loss independently to each pixel in the image. In contrast, mask classification predicts a set of binary masks, each assigned a single class label. This approach combines a per-pixel binary mask loss with a classification loss to supervise the predictions. Matching between the predicted masks and ground truth segments can be performed in two ways: either through bipartite matching or fixed matching when the number of predictions and classes are equal. Image from [30].

At the core of MaskFormer's architecture is its ability to handle all three segmentation tasks without the need for task-specific modifications. This generalization is achieved through a single model architecture, loss function, and training pipeline, making MaskFormer adaptable and efficient for different applications. The mask classification paradigm allows the framework to predict a set of mask-class pairs, where each mask corresponds to a specific region in the image, and a single class label is assigned to the entire region. This method eliminates the need for post-processing steps typically required for task-specific outputs, simplifying the segmentation process and improving overall efficiency. Specifically, the mask classification paradigm clusters pixels into N binary masks and associates a distribution over K semantic classes to each mask $\{(m_i, p_i) | m_i \in [0,1]^{H \times W}, p_i \in \mathbb{R}^K\}_{i=1}^N$.

The MaskFormer architecture consists of three main components: a pixel-level module, a transformer module, and a segmentation module. The pixel-level module begins with a backbone network, such as ResNet [64], to extract rich, multi-scale feature maps from the input image. These features are refined and upsampled by a lightweight pixel decoder to generate per-pixel embeddings, which serve as the foundation for mask prediction.

The transformer module processes these features using a set of learnable queries. These queries interact with the image features through cross-attention layers in a transformer decoder, where each query attends to specific regions of the image. This process enables the model to capture complex spatial relationships and global context within the image. The embeddings generated by the transformer module are then passed to the segmentation module, where they are used to produce the final mask and class predictions. Each query generates a binary mask by performing a dot product with the per-pixel embeddings and assigns a class label to the mask, completing the segmentation process. An overview of the architecture is presented in Figure 3.9.

Mask2Former [29] builds upon the foundation laid by MaskFormer, introducing several significant improvements that enhance its performance, training efficiency, and flexibility across various segmentation tasks, including panoptic, instance, and semantic segmentation. One of the most notable advancements is the replacement of the standard cross-attention mechanism in the transformer decoder with a masked attention mechanism. Unlike traditional attention, which considers the entire image, masked attention restricts the focus to regions within the

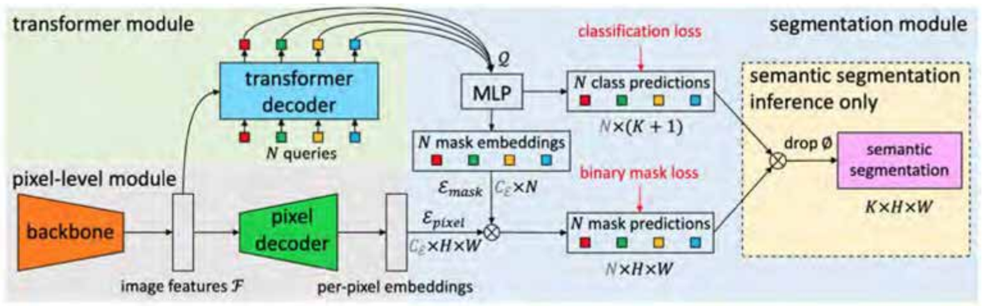


Figure 3.9: The architecture of MaskFormer with the three main components: pixel-level module, transformer module, segmentation module. Image from [30].

predicted mask. This refinement not only speeds up convergence by reducing the attention search space but also improves segmentation accuracy, particularly in complex scenes with overlapping objects or cluttered backgrounds.

Another improvement in Mask2Former is the introduction of a more efficient multi-scale processing strategy. While MaskFormer utilizes multi-scale features from the image, Mask2Former processes features at different resolutions sequentially, in a round-robin manner, rather than feeding them all simultaneously into the decoder. This enhancement enables the model to effectively handle objects of varying sizes without incurring excessive computational overhead, further contributing to its versatility.

Several architectural refinements have been incorporated into Mask2Former's transformer decoder to improve its overall performance. One key change is the reordering of the self-attention and cross-attention layers, where cross-attention is applied before self-attention. This modification allows the queries to incorporate signals from the image earlier in the processing pipeline, enriching their representation and leading to better mask proposals. Additionally, the decoder employs learnable query features that are directly supervised, which improves the precision of the predicted masks. Mask2Former also eliminates the use of dropout in the decoder, as experiments showed it reduced performance without significantly improving robustness.

Memory efficiency during training is another critical improvement in Mask2Former. The model calculates mask losses on a randomly sampled subset of points within the mask instead of using all pixels. This approach significantly reduces GPU memory consumption—by approximately three times compared to MaskFormer—making Mask2Former more accessible for training on hardware with limited memory resources.

Another standout feature of Mask2Former is its faster training process. The model achieves competitive or superior results within 50 training epochs, compared to the 300 epochs required by MaskFormer for comparable performance. This efficiency makes it an appealing choice for applications where training time is a critical factor. An overview of the Mask2Former architecture is presented in Figure 3.10.

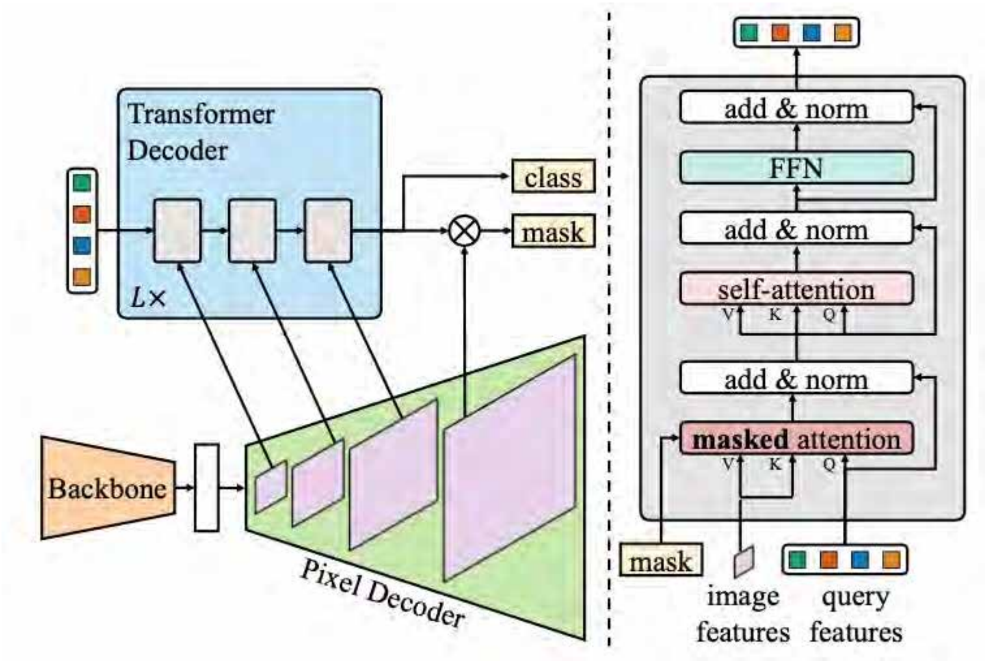


Figure 3.10: Mask2Former [29] retains the same meta-architecture as MaskFormer [30], consisting of a backbone, a pixel decoder, and a Transformer decoder. However, it introduces a novel Transformer decoder that incorporates masked attention in place of the standard cross-attention mechanism. To effectively address small objects, Mask2Former employs an efficient approach to utilizing high-resolution features from the pixel decoder by sequentially feeding one scale of the multi-scale feature maps to each Transformer decoder layer. Additionally, it reorders the self-attention and cross-attention layers (masked attention), makes query features learnable, and eliminates dropout, enhancing computational efficiency and performance. Image from [29].

Mask2Former prioritizes achieving strong performance across various segmentation tasks. For this reason, the more advanced multi-scale deformable attention Transformer (MSDeformAttn) [172] is used as the default pixel decoder. This module comprises six MSDeformAttn layers applied to feature maps with resolutions of 1/8, 1/16, and 1/32. Additionally, a simple upsampling layer with lateral connections is applied to the final 1/8 feature map, generating a per-pixel embedding with a resolution of 1/4.

In Mask2Former, the Transformer decoder includes L = 3 Transformer decoder layers, resulting in a total of 9 layers. By default, the decoder uses 100 queries. Auxiliary losses are applied to each intermediate Transformer decoder layer as well as to the learnable query features prior to the Transformer decoder.

During training, the final loss function is formulated as:

$$\mathcal{L} = \sum_{l=1}^{L} \sum_{s=1}^{S} \lambda_{\text{ce}} \mathcal{L}_{\text{ce}} + \lambda_{\text{dice}} \mathcal{L}_{\text{dice}} + \lambda_{\text{cls}} \mathcal{L}_{\text{cls}} \tag{3.2}$$

Here, $L = 3$ represents the number of Transformer decoders, $S = 3$ denotes the number of layers within each Transformer decoder, and the loss weights are set to $\lambda_{\text{ce}} = 5.0$, $\lambda_{\text{dice}} = 5.0$, and $\lambda_{\text{cls}} = 2.0$. For learning binary panoptic masks, the binary cross-entropy loss $\mathcal{L}_{\text{ce}}$ and dice loss $\mathcal{L}_{\text{dice}}$ [105] are utilized. The classification component of the model is supervised using the cross-entropy loss $\mathcal{L}_{\text{cls}}$.

This formulation ensures that the contributions of segmentation and classification tasks are appropriately weighted during training to optimize overall performance.

Experiments. Training is performed using a crop size of 512×1024, a batch size of 16, and a total of 90k iterations. During inference, the models are applied to the entire image with a resolution of 1024×2048. The AdamW [101] optimizer is used. An initial learning rate of 0.0001 and a weight decay of 0.05 are employed for all backbones. The backbone learning rate is scaled by a multiplier of 0.1, and the learning rate is reduced by a factor of 10 at 90% and 95% of the total training steps. Experiments on the Cityscapes dataset, as seen in Table 3.4 showed that Mask2Former achives better results than other state-of-the-art segmentation methods.

In terms of runtime, the network trades speed for accuracy. It is expected to reach 9.7 frames per second on a resolution of 800×1333 on

Method	**Backbone**	**mIoU (s.s.)**	**mIoU (m.s.)**
Panoptic-DeepLab [28]	R50	80.5	-
Segmenter [139]	ViT-L†	-	81.3
SETR [170]	ViT-L†	-	82.2
SegFormer [158]	MiT-B5	-	84.0
MaskFormer [30]	R101	78.5	-
Mask2Former [29]	R50	79.4	82.2
	R101	80.1	81.9
	Swin-T	82.1	83.0
	Swin-S	82.6	83.6
	Swin-B†	**83.3**	**84.5**
	Swin-L†	**83.3**	84.3

Table 3.4: Comparison of semantic segmentation performance (mIoU) across different methods and backbones on the Cityscapes validation set using single-scale (s.s.) and multi-scale (m.s.) images.

a powerful Tesla V100 GPU. Using techniques such as quantization and pruning and specialized libraries such as TensorRT, the runtime can be decreased, increasing the potential to be used on a autonomous vehicle. At the current speed, the network can be used in an offline manner, to generate for example, semantic pseudo-labels, in order to extend existing datasets or create new ones.

Chapter 4

Instance Segmentation

Instance segmentation aims to detect and segment each object in an image with pixel-level precision, assigning unique masks to individual instances. This chapter provides an overview of key methods and then focuses on detailing two significant works.

4.1 Overview

Instance segmentation predicts a semantic mask and an instance identifier for each object with a *thing* class. Classification is performed at instance-level, which means that instance masks could overlap. All pixels from an instance mask have the same semantic class and the same instance identifier. Instance segmentation approaches follow in general two directions: top-down methods perform segmentation of candidate regions given by object detectors, while bottom-up methods perform semantic segmentation and cluster pixels belonging to the same instance based on similarity measures.

Top-down methods. These methods are using one-stage or two-stage object detectors. The most representative solution is Mask-RCNN [63] based on the two-stage Faster R-CNN object detector [130], demonstrating outstanding performance on public benchmarks [34, 97]. Cascade R-CNN [18], Non-local Networks [150], PANet [99] bring improvements to Mask R-CNN [63] at different stages of the pipeline and provide accurate masks, but at high computational costs. Mask R-CNN addresses the foreground-background imbalance problem with a two-stage design, where the first stage (Region Proposal Network) filters out a large portion of the negatives, while in the second stage a fixed foreground-background batch sampling ratio solves the issue.

With the introduction of specialized losses such as Focal Loss in RetinaNet [96] which addresses the foreground-background imbalance problem by including hard examples in training or by using multi-scale predictions and anchors in YOLO [129], one-stage object detectors manage to obtain on-par accuracy with two-stage detectors while being significantly faster. Box2Pix [144] is built on a fully-convolutional one stage detector with a semantic segmentation head and regresses offsets for every segmented pixel to the bounding box center. This method runs in real-time at the cost of reduced accuracy. Yolact [17] is a real-time network that learns to localize instances on its own by generating a dictionary of prototype masks and predicting a set of linear combination coefficients. A simple and efficient solution for instance segmentation is proposed in CenterMask [85] which extends the fast one-stage object detector FCOS [142] with a Mask-RCNN type of mask achieving improved performance. CenterMask proposes also the efficient VoVNet2 backbone. The building blocks of the VoVNet2 backbone are the One-Shot Aggregation (OSA) modules, which consist of several convolutions followed by the concatenation of their resulted feature maps, this way different receptive fields are captured. An identity mapping is added to the OSA modules in order to boost the performance and facilitate the flow of the gradient through the network. A channel attention module named effective Squeeze-Excitation (eSE) is plugged in the OSA modules and learns a channel specific descriptor with a global average pooling, one convolution operation followed by the sigmoid activation.

Bottom-up methods. Proposal-free methods perform semantic segmentation and then cluster similar pixels into instances. Kendall *et al.*[73] and Neven *et al.*[109] propose learning an offset vector for each pixel that points to the instance centroid. PersonLab [111], CornerNet [83] introduce keypoint guided instance segmentation, while Deep Polygon Transformer [92] and DeepSnake [115] formulate instance segmentation as the problem of fitting a polygon around the object. TensorMask [26] employs 4D Tensors and demonstrates advantages over 3D Tensors with increased computational costs. DWT [15] models the energy of the watershed transform with CNNs, but cannot handle objects separated into multiple parts. Instance segmentation can be also viewed as a graph partition problem in which the total score of edges connecting different components is maximized, but these types of methods [48, 77] are currently time-consuming due to the complexity of the graph space.

4.2 Mask R-CNN

Method. Mask R-CNN [63] is a two-stage instance segmentation network which extends the two-stage object detector Faster R-CNN [54] with a mask prediction head. The network performs instance segmentation inside detected 2D bounding boxes. In the first stage of Mask R-CNN, a Region Proposal Network (RPN) proposes object candidates. In the second stage, the RoIAlign samples a fixed number of features inside each candidate box, and thus achieves scale invariance. RoIAlign extracts a small feature map of size 7×7 from each Region of Interest (RoI) by sampling values from the input feature map at four regularly spaced locations using bilinear interpolation. The Mask R-CNN network employs a shared convolutional network backbone based on ResNet [64] and Feature Pyramid Network (FPN) [95] for feature extraction. The FPN encodes multi-scale representations at output stride from $32\times$ to $4\times$ and is built in a top-down manner by upsampling low resolution features and merging them with higher level features via lateral connections. Instance mask predictions are performed at a fixed low resolution of 28×28, which might introduce errors especially for large objects.

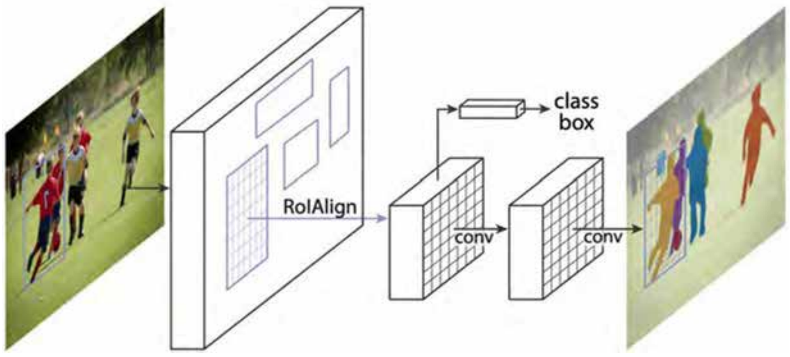


Figure 4.1: Instance segmentation with the Mask R-CNN framework [63].

The network is trained end-to-end and the final loss is defined as the weighted sum of five losses:

$$\begin{aligned}\mathcal{L}_{mrcnn}(p, y) = &\gamma_{rpncls}\mathcal{L}_{rpncls} + \gamma_{rpnbox}\mathcal{L}_{rpnbox} + \gamma_{rcnncls}\mathcal{L}_{rcnncls} \\ &+ \gamma_{rcnnbox}\mathcal{L}_{rcnnbox} + \gamma_{rcnnmask}\mathcal{L}_{rcnnmask}\end{aligned} \tag{4.1}$$

where $\mathcal{L}_{rpncls}$ and $\mathcal{L}_{rpnbox}$ are the classification and regression losses for the RPN, $\mathcal{L}_{rcnncls}$ and $\mathcal{L}_{rcnnbox}$ are the classification and regression losses for the second stage and $\mathcal{L}_{rcnnmask}$ is the mask loss.

During training, Mask R-CNN uses anchor boxes with a predefined size. The network regresses bounding box predictions with respect to these anchors. Each location in the 5-level FPN is assigned 3 anchors with different aspect ratios: 1:1, 1:2, 2:1. For the largest scale of the FPN features (1/4), anchors in the original image are small and sampled with a stride of 4 (e.g. 32×32 for 1:1 ratio and image of size 1024×2048). For the smallest feature map in the FPN (1/64) the largest anchors are used with a sampling stride of 64 (e.g. in the original image of resolution 1024×2048 anchors have the size 512×512 at ratio 1:1). Lower levels in the FPN are responsible for detecting large objects while higher levels are responsible for detecting small objects.

The objective of the first stage RPN is to regress object bounding boxes and classify them as objects/non-objects. To train the RPN, the anchors are first classified as positive or negative. Positive anchors are represented by: anchors with the highest Intersection-over Union (IoU) overlap with a ground-truth box and anchors that have the IoU > 0.7 with any ground truth box. A bounding box can assign a positive label to multiple anchors. Negative anchors have the IoU < 0.3 for all ground truth boxes. Only the positive and negative anchors contribute to the loss. The regression targets for the RPN are computed as transformation deltas for the anchor box relative to the ground truth box:

$$d_x = \frac{g_x - a_x}{a_w} \tag{4.2}$$

$$d_y = \frac{g_y - a_y}{a_h} \tag{4.3}$$

$$d_w = \log(\frac{g_w}{a_w}) \tag{4.4}$$

$$d_h = \log(\frac{g_h}{a_h}) \tag{4.5}$$

where (d_x, d_y, d_w, d_h) are the RPN regression targets, (g_x, g_y, g_w, g_h) is the ground truth box represented by its center point and width and height, while (a_x, a_y, a_w, a_h) are the anchor center point and size.

In order to balance the positive-negative samples for the RPN training, a minibatch of 256 samples per image is constructed, where foreground samples are at most half (e.g. 128). All the samples from the minibatch will be used to compute the classification loss, while for the regression loss only the positive samples will be used. The classification loss is the binary cross entropy loss:

$$\mathcal{L}_{rpncls}(p, y) = -\frac{1}{N} \sum_{i=1}^{N} y_i \cdot \log(p_i) + (1 - y_i) \cdot \log(1 - p_i) \tag{4.6}$$

where p is the predicted probability and y is the label (with a value of 1 for object and 0 for non-object).

The regression loss is the smooth L1 loss, which is a combination of the L1 and L2 losses:

$$\mathcal{L}_{rpnbox}(p, y) = SmoothL1(p - y) \tag{4.7}$$

$$SmoothL1(x) = \begin{cases} 0.5 \cdot \frac{x^2}{\beta}, & \text{if } |x| < \beta, \\ |x| - 0.5 \cdot \beta, & \text{otherwise.} \end{cases} \tag{4.8}$$

where β is an adjustable hyperparameter, which marks the point of transition between L2 and L1 loss.

After the Region of Interest (RoI) predictions are obtained at each location of the 5-levels FPN by applying the predicted regression transformations to the anchors, the RoIs are collected (concatenated), NMS (Non-Maxima Suppression) is applied and top N RoIs based on the objectness score (from classification) are kept. Next, a minibatch for the second stage of the network is constructed. In Mask R-CNN, a minibatch of 256 RoIs is used, where at most 25% are positive. Positive RoIs are the ones that overlap the ground truth boxes with intersection over union IoU > 0.5, while the negative RoIs have an IoU with the ground truth boxes lower than 0.5. The classification loss for the second stage RCNN is the multi-class cross entropy loss:

$$\mathcal{L}_{rcnncls}(p, y) = -\frac{1}{N} \sum_{i=1}^{N} \sum_{c=1}^{C} y_{i,c} \log(p_{i,c}) \tag{4.9}$$

where p_c is the predicted probability for class c and y_c is the ground truth for class c.

The regression loss $\mathcal{L}_{rcnnbox}$ is the smooth L1 loss, the same as $\mathcal{L}_{rpnbox}$ as seen in equation 4.8. The last part of the framework is the mask head training. Mask targets are prepared by associating one ground truth mask to each positive RoI from the RCNN minibatch. The association is done based on the maximum bounding box overlap. The mask loss is the binary cross entropy loss, the same as $\mathcal{L}_{rpncls}$ as seen in equation 4.6.

Experiments. The training process on the Cityscapes dataset involves randomly sampling the image scale (shorter side) from the range [800, 1024] to mitigate overfitting. During inference, a single scale of 1024 pixels is used. The model is trained with a mini-batch size of 1 image per GPU, resulting in a total of 8 images on 8 GPUs. Training is conducted for 24k iterations, starting with a learning rate of 0.01, which is reduced to 0.001 after 18k iterations. The results on the Cityscapes dataset are presented in Table 4.1.

Method	**Training Data**	**AP [val]**	**AP**	**AP50**	**Person**	**Rider**	**Car**	**Truck**	**Bus**	**Train**	**MCycle**	**Bicycle**
InstanceCut [77]	fine + coarse	15.8	13.0	27.9	10.0	8.0	23.7	14.0	19.5	15.2	9.3	4.7
DWT [15]	fine	19.8	15.6	30.0	15.1	11.7	32.9	17.1	20.4	15.0	7.9	4.9
DIN [11]	fine + coarse	-	20.0	38.8	16.5	16.7	25.7	20.6	30.0	23.4	17.1	10.1
SGN [98]	fine + coarse	29.2	25.0	44.9	21.8	20.1	39.4	24.8	33.2	30.8	17.7	12.4
Mask R-CNN	fine	31.5	26.2	49.9	30.5	23.7	46.9	22.8	32.2	18.6	19.1	16.0
Mask R-CNN	fine + COCO	**36.4**	**32.0**	**58.1**	**34.8**	**27.0**	**49.1**	**30.1**	**40.9**	**30.9**	**24.1**	**18.7**

Table 4.1: Instance segmentation results using Mask R-CNN on Cityscapes validation dataset.

4.3 RetinaMask

Method. RetinaMask [46] integrates a mask prediction head on top of the one-stage object detector RetinaNet [96]. It also improves the ground truth-anchor matching policy and introduces the self-adjusting smooth L1 loss that increases robustness during training.

One of the problems that object detectors need to address is foreground-background imbalance. The issue is even more pronounced for one-stage object detectors, which do not use the Region Proposal Network (RPN) to narrow down the candidate object locations to a smaller number (e.g. 1000 or 2000). One-stage detectors process a larger number of candidate object locations (e.g 100k), densely sampled across the image and covering different aspect ratios and scales. RetinaNet proposes the focal loss to address the foreground-background class imbalance.

The focal loss applies a dynamic scaling factor to the cross entropy loss, which puts more focus on the hard examples. The focal loss is defined as:

$$\mathcal{L}_{focal} = \frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} -\alpha_c(1-p_{i,c})^{\lambda} y_{i,c}\log(p_{i,c}) \tag{4.10}$$

where λ is a parameter that down-weights the contribution of well-classified examples if it has a positive value, and increases the weight for the misclassified ones if it has a negative value. The α is a class specific parameter that is usually used to address class imbalance within foreground samples and is computed as the inverse class frequency.

The final loss in RetinaNet is:

$$\mathcal{L}_{retinanet} = \mathcal{L}_{focal} + \mathcal{L}_{box} \tag{4.11}$$

where $\mathcal{L}_{box}$ is the smooth L1 loss defined in equation 4.8.

The RetinaMask network has a shared feature extraction backbone, a Feature Pyramid Network (FPN) which allows multi-scale object detection by encoding multi-resolution representations from 1/4 to 1/64. The FPN follows the original implementation [95] with 256 feature maps and 5 anchor scales. A bounding box regression and classification head with four convolutional layers is appended to each level of the pyramid. The bounding box predictions of RetinaNet are segmented in the next steps by the mask head. First, a post-processing operation takes place, where the bounding box predictions are aggregated, filtered and distributed to layers in the FPN. Next, the ROIAlign [63] operation samples the same number of features (14×14) from each predicted bounding box, which are finally processed by the mask prediction head with four convolutional layers and one transposed convolution. Finally, a $[1 \times 1]$ convolution generates the final class-wise masks of size 28×28. RetinaMask introduces the self-adjusting smooth L1 loss for bounding box regression. A problem with the original smooth L1 loss would be that the choice of the β factor is heuristic. To improve the formulation, the self-adjusting smooth L1 loss computes the running mean and variance of the absolute loss and their difference will be assigned to the β factor. Another improvement over RetinaNet is the anchor-ground truth matching policy. While in RetinaNet, positive anchors are considered those with IoU > 0.5 and negative those with IoU < 0.4, in RetinaMask for each ground truth a best matching anchor will be assigned without considering its overlap. The final RetinaMask loss is computed as:

$$\mathcal{L}_{retinamask} = \mathcal{L}_{focal} + \mathcal{L}_{box} + \mathcal{L}_{mask} \tag{4.12}$$

The mask loss is implemented as the binary cross entropy loss defined in equation 4.6.

Experiments. RetinaMask is trained on the UP-Drive dataset. The deep instance segmentation network is designed to be both accurate and efficient while ensuring compatibility with deep learning inference engines such as TensorRT [6]. ResNet-50 [64] with a 5-level Feature Pyramid Network (FPN) [95] serves as the feature extraction backbone. The network is pretrained on the Microsoft COCO dataset [97] and the Cityscapes dataset [34]. The training process uses a batch size of 16 images for 30,000 iterations, starting with a base learning rate of 0.01, which is reduced by a factor of 10 at 20,000 iterations. Stochastic Gradient Descent (SGD) is used for optimizing the loss function. Fisheye images are cropped to 640×1280 pixels and further scaled during training, with the shorter edge randomly sampled from the range [480, 640]. Narrow field-of-view images are resized to 416×832 ith multi-scale training performed at scales within [320, 416]. Random horizontal flipping is applied as a data augmentation technique.

RetinaMask is trained and evaluated for instance segmentation network on all four unwarped fisheye images (front, left, right, back) and we present the results in Table 4.2. The experiments are done with three different resolutions: 640×1280, 416×832, 320×640. The network is accelerated with the TensorRT library using FP32 precision for increased quality. Compared to the two-stage Mask R-CNN based network, there is a slight decrease in accuracy for 640×1280 resolution from 31.3% mask mAP to 30% mask mAP, but the inference time is reduced 2.5 times to 66 ms. Adopting the largest resolution available of 640×1280 is critical for fisheye images, where the apparent size of objects is small.

Visual results for semantic, instance and panoptic segmentation on the UP-Drive dataset can be seen in Figure 4.2.

Resolution	mAP box	mAP mask	Time (ms)
640×1280 - FP32	36.8	30	66
416×832 - FP32	30.1	24.8	44
320×640 - FP32	26.4	21.6	33

Table 4.2: Evaluation of the instance segmentation network on fisheye images corresponding to front, left, back, right views. Time is measured on a NVIDIA GTX 1080 GPU.

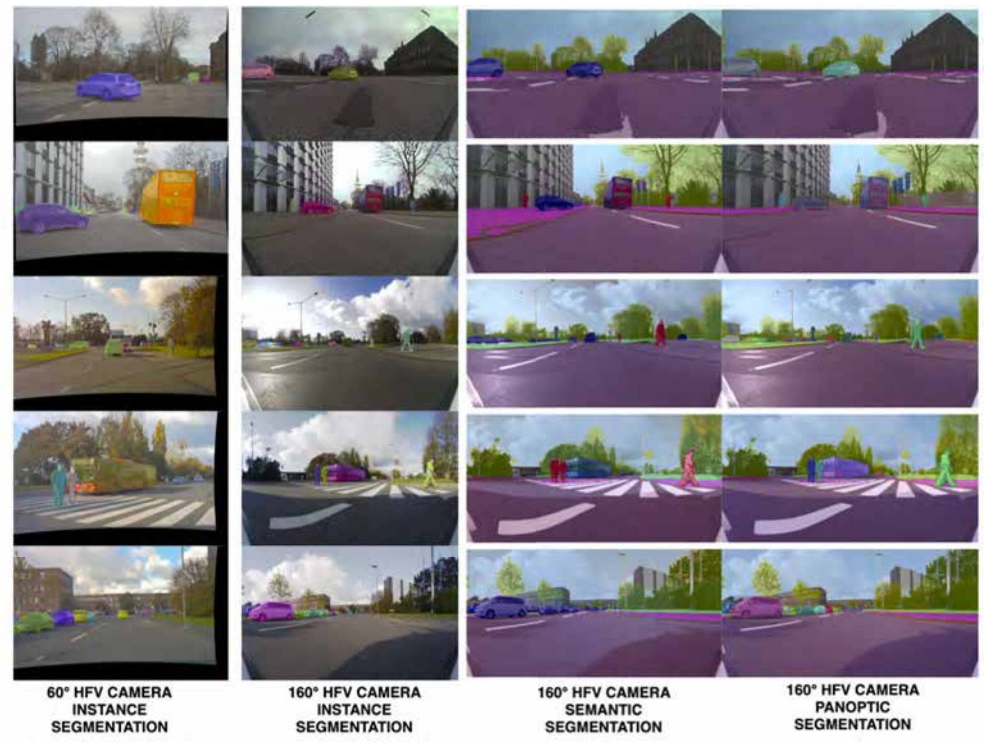

60° HFV CAMERA INSTANCE SEGMENTATION
160° HFV CAMERA INSTANCE SEGMENTATION
160° HFV CAMERA SEMANTIC SEGMENTATION
160° HFV CAMERA PANOPTIC SEGMENTATION

Figure 4.2: The front area of the vehicle is covered by two cameras: a narrow 60° horizontal field-of-view camera which provides instance segmentation at increased depth and a wider 160° horizontal field-of-view camera, which provides instance, semantic and panoptic segmentation for the near-range.

The network is also trained on images from the front camera with a narrow 60° HFV. Results are in Table 4.3. In order to achieve the trade-off between processing speed and quality, the network is optimized with FP32 using TensorRT and the resolution of 416×832 is adopted, thus obtaining 21.4% mask mAP and 44 ms inference time on a NVIDIA GTX 1080 GPU.

As seen in Figure 4.3, pedestrians are visible only in the front unwarped fisheye image up to a distance of 20 meters. Processing these unwarped fisheye images allows for robust instance segmentation of pedestrians up to 25 meters. At this distance, pedestrians are visible and detected in both the unwarped fisheye image and the 60° horizontal field-of-view (HFV) image. Beyond 25 meters, the size of the pedestrian in the unwarped fisheye image becomes too small for detection; however, detec-

Resolution	mAP box	mAP mask	Time (ms)
604 × 960 - FP32	30.2	22.8	58
416 × 832 - FP32	28.7	21.4	44
320 × 640 - FP32	23.4	18.3	33

Table 4.3: Evaluation of the instance segmentation network on narrow field-of-view images corresponding to front view. Time is measured on a NVIDIA GTX 1080 GPU.

tion remains possible in the 60° HFV image. By combining the outputs of both cameras, the pedestrian detection range is effectively extended to 75 meters.

Deployment. To integrate the instance segmentation network into a framework such as Automotive Data and Time-Triggered Framework (ADTF) [1], which is typically used in the automotive industry, the TensorRT library [6] is utilized to generate a high-performance runtime engine that can be seamlessly loaded into a C++/CUDA project. TensorRT provides significant advantages, including network optimization and quantization, which substantially reduce inference time. The library optimizes the layer graph by removing unused layers, fusing operations such as convolution, bias, and ReLU, aggregating and merging operations, and combining concatenation layers. Additionally, quantization reduces precision from 32-bit floating-point to 8-bit integers for network weights, resulting in a significantly higher computational throughput.

RetinaMask cannot be directly optimized with TensorRT due to hand-crafted operations such as ROIAlign, candidate box filtering, and assignment to specific FPN layers. Additionally, Non-Maxima Suppression (NMS) is implemented to eliminate overlapping boxes. To address this, the network is divided into three components: the backbone with object detection heads, the hand-crafted operations (including filtering low-confidence boxes, selecting the top 1000 boxes per FPN layer, applying NMS, selecting the top 50 scoring bounding boxes, and ROIAlign), and the mask prediction head. The backbone, object detection, and mask prediction heads, which consist of operations natively supported in ONNX and TensorRT, are converted to the ONNX format using the ONNX Parser [5]. Hand-crafted operations are implemented as TensorRT Plugin layers using native CUDA. An optimized engine for instance segmentation is then generated with FP32 precision, resulting in nearly twice

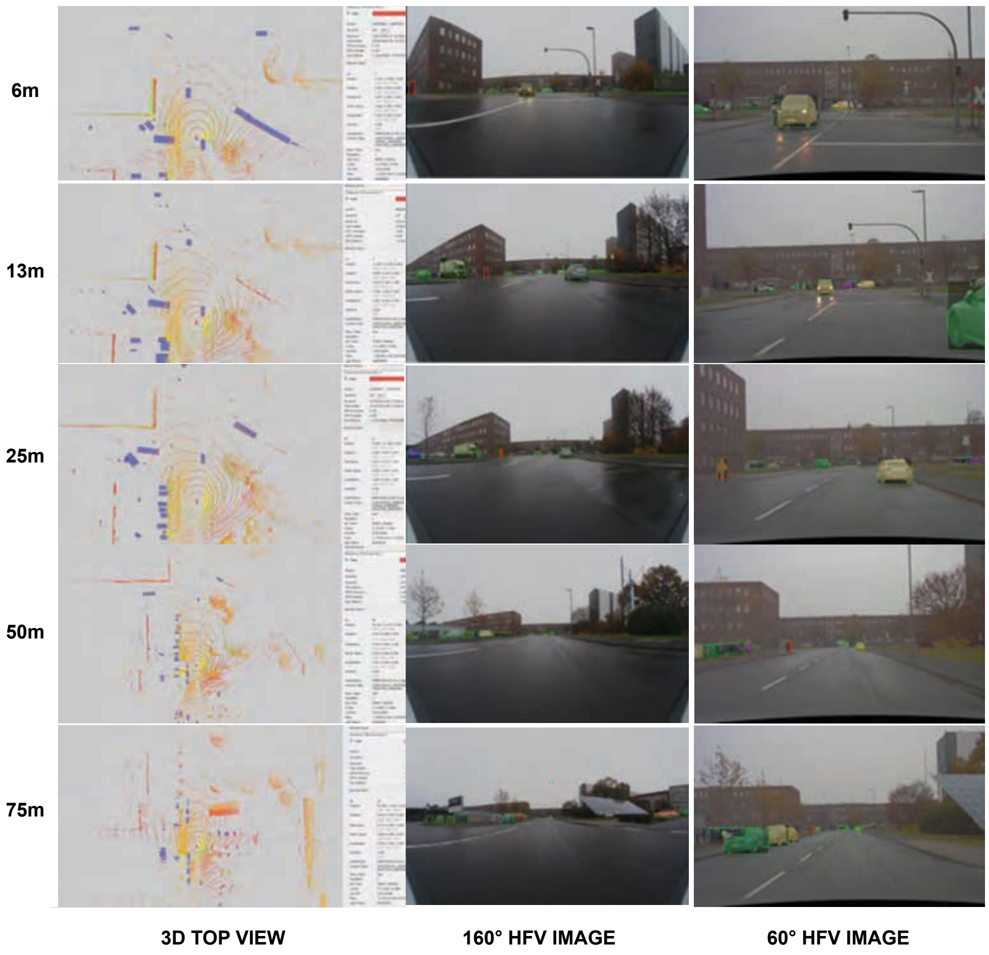


Figure 4.3: Comparison between wide and narrow field-of-view instance segmentation. A pedestrian is marked with a green box in the 3D top view image and a red bounding box in the wide and narrow field-of-view images. In the first column, the bird's eye view of the 3D point cloud with detected objects is depicted. Best viewed in color and zoom.

the inference speed compared to the unoptimized network.

Discussion. From a practical perspective, a setup based only on fisheye cameras mounted on the vehicle in all four directions is suitable only for robust segmentation and detection within the near range around the vehicle. For pedestrian detection, segmentation remains effective up to 25 meters, making it suitable primarily for low-speed driving and parking maneuvers. While near-range detection for the left, right, and rear views offers sufficient information for maneuver prediction and decision-making, far-range detection is particularly crucial for the front view, especially at higher driving speeds.

A solution that combines fisheye cameras and narrow field-of-view cameras for the front view is essential to cover both the near and far ranges. Additionally, increasing the resolution of processed images further extends the detection range, albeit at a higher computational cost. To maintain efficient processing times when handling high-resolution front narrow field-of-view images and four fisheye images, network optimization and quantization techniques can be employed. This approach ensures an effective balance between detection range, accuracy, and computational efficiency.

One-stage networks for instance segmentation bring major benefits in terms of speed compared to two-stage networks. However, optimization of one-stage and two-stage networks is not easily achieved when there are hand-crafted processing steps between the backbone and network heads that need to be separately implemented and integrated with the TensorRT optimized network parts. A fully convolutional network for instance segmentation, free of hand-crafted processing steps, is preferable as it enables seamless end-to-end optimization with TensorRT.

Chapter 5

Panoptic Segmentation

Panoptic segmentation provides a unified semantic and instance representation. It performs dense pixel classification into *things* and *stuff* classes and assigns an instance identifier to every *things* pixel in the image. This chapter provides a review of the panoptic segmentation literature and continues with the in-depth description of a few methods.

5.1 Overview

The panoptic segmentation task has gained popularity since introduced by Kirillov *et al.*in [76]. In general, panoptic segmentation methods follow two directions, they are either proposal-based methods, also named top-down approaches or proposal-free methods, known as bottom-up approaches.

Top-down methods. Panoptic segmentation can be achieved by simultaneously solving two other tasks: semantic and instance segmentation. Top-down methods solve panoptic segmentation by merging instance predictions and semantic segments. They are built on top of an object detector and use object proposals for generating overlapping instance masks. A post-processing step usually follows, which solves the overlaps and conflicts between semantic and instance predictions. Most works [76, 75, 35, 159, 90, 126, 137, 116] employ the two-stage instance segmentation framework Mask R-CNN [63] and extend the shared feature extractor with a lightweight segmentation head. In [76], the authors propose a post-processing algorithm to merge instance and semantic segmentation into a unified panoptic output. To solve instance-wise overlaps, a Non-Maxima Suppression (NMS) step is implemented: predicted segments are sorted by their confidence scores, low-score instances are removed and then starting from the most confident instance segment, the non-overlapping part of the segment will be pasted in the panoptic segmentation output only if it does not significantly overlap the current panoptic segmentation. The next step is to solve the instance-segmentation conflicts, which is resolved in favor of the instance prediction. UPSNet [159] introduces a parameter-free panoptic head that employs the semantic and instance logits and is supervised with an explicit panoptic segmentation loss. In [90], the authors propose AUNet, a unified framework on top of Mask R-CNN, that learns the relations between instance and semantic pixels with an attention module. Seamless segmentation [126] introduces a lightweight DeepLab-inspired segmentation head. AdaptIS [137] takes a point proposal from a semantic segment of a *things* class and generates an instance mask corresponding to that point. All the aforementioned approaches achieve high accuracy at an increased computational cost due to the complexity of the Mask R-CNN two-stage detector. Lately, one-shot called also one-stage object detectors, such as RetinaNet [96] or FCOS [142], have achieved great progress and even surpassed two-stage detectors on public benchmarks. DenseBox [67] builds on top of FCOS and introduces a parameter-free mask prediction head that reuses discarded dense object proposals to generate instance masks.

Bottom-up methods. The second type of approaches have not been so exhaustively studied due to their initial inferior performance compared to proposal-based methods. SSAP [48] models pixel-pair affinities in a hierarchical manner and formulates panoptic segmentation as a graph partition problem. DeeperLab [162] proposes an encoder-decoder network with semantic and instance segmentation heads, where instances have a keypoint-based representation with their four bounding box corners and object centers. Panoptic-DeepLab [28] predicts semantic segmentation, instance center locations and instance center offsets, and has been the first bottom-up method to surpass top-down approaches on public benchmarks.

5.2 Panoptic-DeepLab

Method. Panoptic-DeepLab [28] is a state-of-the-art architecture for panoptic segmentation, which aims to provide a unified solution for segmenting both *stuff* (amorphous regions like sky, road, grass) and *things* (distinct objects like cars, pedestrians) in an image. It is designed to offer simplicity and efficiency compared to previous panoptic segmentation approaches. The network regresses instance offsets and predicts semantic segmentation and instance centers. Class-agnostic instance segmentation is obtained by grouping foreground pixels to their closest center based on the predicted offsets. The final panoptic segmentation is generated by merging the semantic segmentation with the class-agnostic instance segmentation results by using a majority voting principle. The network consists of a shared backbone, dual decoders for semantic and instance segmentation and three heads for semantic, instance center and instance offset regression. The network architecture is illustrated in Figure 5.1. During training, the following loss is minimized:

$$\mathcal{L}_{pan} = \gamma_{seg}\mathcal{L}_{seg} + \gamma_{offset}\mathcal{L}_{offset} + \gamma_{center}\mathcal{L}_{center} \tag{5.1}$$

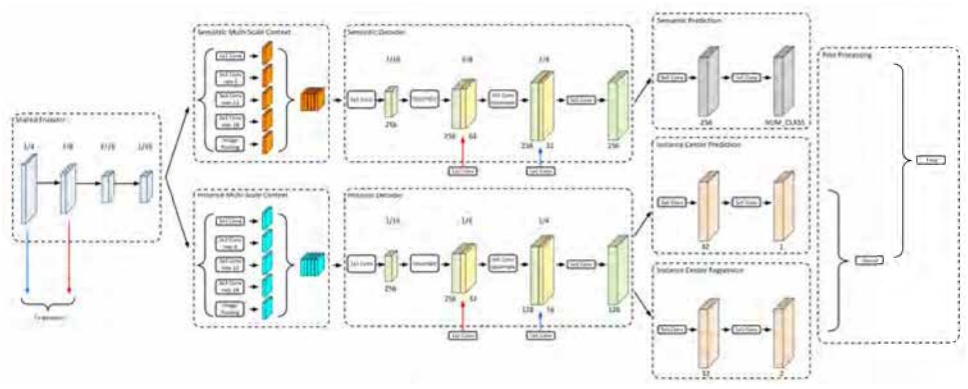


Figure 5.1: Panoptic-DeepLab network architecture [28].

The segmentation loss is the bootstrapped cross entropy loss [162]. Pixels are sorted based on the cross entropy loss and only top K positions will backpropagate the errors. Moreover, the loss of small instances is weighted. With this loss, the network focuses on hard pixels and small instances. The weight is set to 3 for instances that have an area smaller than 64×64 and $K = 0.15 \cdot N$, where N is the total number of pixels.

The offset loss is implemented as the L1 loss only for pixels belonging to instances:

$$\mathcal{L}_{offset}(p, g) = \frac{1}{N_{inst}} \sum_{i=0}^{N_{inst}} |p_i - g_i| \tag{5.2}$$

where p and g are the predicted and ground truth offsets and N_{inst} is the number of instance pixels.

The instance center is encoded in a heatmap as a 2D Gaussian with a standard deviation of 8 pixels. The center loss is the Mean Squared Error (MSE) loss between the predicted and ground truth heatmap:

$$\mathcal{L}_{center}(p, g) = \frac{1}{N} \sum_{i=0}^{N} (p_i - g_i)^2 \tag{5.3}$$

During inference, the center point is obtained by applying non-maximum suppression (NMS) over the 2D Gaussian heatmap, which in practice means applying max pooling and filtering out locations that have a low confidence score.

Experiments. Panoptic-DeepLab demonstrates strong performance on the Cityscapes dataset, achieving state-of-the-art results across key evaluation metrics for panoptic segmentation. Without additional training data, Panoptic-DeepLab achieves a PQ (Panoptic Quality) score of 63.0%, which improves further to 64.1% when multiscale inputs and horizontal flipping are applied during inference, as seen in Table 5.1. The model also delivers competitive AP (Average Precision) and mIoU (mean Intersection over Union) scores, reaching 38.7% and 81.5%, respectively. With the inclusion of extra data, such as Mapillary Vistas (MV), Panoptic-DeepLab achieves even higher performance, with a PQ score of 67.0%, AP of 42.5%, and mIoU of 83.1% when using multiscale inputs and horizontal flipping. These results highlight the effectiveness of Panoptic-DeepLab in producing accurate and robust panoptic segmentation predictions, demonstrating its ability to handle both *stuf* and *thing* classes with high precision and scalability.

5.3 ISS-Fusion

Method. ISSNet (Instance and Semantic Segmentation Network) [35] is a multi-task network designed to perform object detection, instance segmentation, and semantic segmentation simultaneously. The network extends the two-stage Mask R-CNN [63] framework, originally

Table 5.1: Panoptic-DeepLab [28] results on the Cityscapes *val* set. **Flip**: Adding left-right flipped inputs. **MS**: Multiscale inputs. **MV**: Mapillary Vistas.

Method	**Extra Data**	**Flip**	**MS**	**PQ (%)**	**AP (%)**	**mIoU (%)**
			w/o Extra Data			
TASCNet [87]	-	-	-	55.9	-	75.5
Panoptic FPN [75]	-	-	-	58.1	33.0	75.7
AUNet [90]	-	-	-	59.0	33.3	76.3
UPSNet [159]	-	-	-	59.3	33.3	77.0
UPSNet [159]	-	✓	✓	60.3	33.3	77.8
Seamless [126]	-	-	-	60.2	33.6	78.2
AdaptIS [137]	-	-	-	61.0	34.3	78.6
DeeperLab [162]	-	-	-	56.5	-	78.7
	-	✓	✓	61.1	-	81.0
SSAP [48]	-	-	-	61.1	-	78.5
	-	-	-	63.0	35.3	80.5
Panoptic-DeepLab	-	✓	✓	63.4	36.1	80.9
	-	✓	✓	64.1	**38.7**	**81.5**
			w/ Extra Data			
TASCNet [87]	COCO	-	-	59.3	37.1	78.1
TASCNet [87]	COCO	✓	✓	63.1	39.1	78.7
UPSNet [159]	COCO	-	-	61.8	37.8	79.7
UPSNet [159]	COCO	✓	✓	62.7	39.0	80.5
Seamless [48]	COCO	-	-	63.6	37.7	80.3
	MV	-	-	65.3	38.8	82.5
Panoptic-DeepLab	MV	✓	-	65.7	39.4	82.6
	MV	✓	✓	**67.0**	**42.5**	**83.1**

developed for object detection and instance segmentation, by incorporating an additional semantic segmentation head. Given the complexity of the two-stage Mask R-CNN architecture and its relatively high inference time, using a separate model for semantic segmentation would result in significant computational and memory overhead, making it unsuitable for real-time applications. This challenge is addressed through multi-task learning, employing a single network for both tasks. The inference speed is improved by reusing visual features through a shared CNN feature extractor, referred to as the backbone, and multiple task-specific network heads. ISSNet uses the ResNet-50 [64] backbone for feature extraction, enhanced with a 5-level Feature Pyramid Network (FPN) [95] for robust

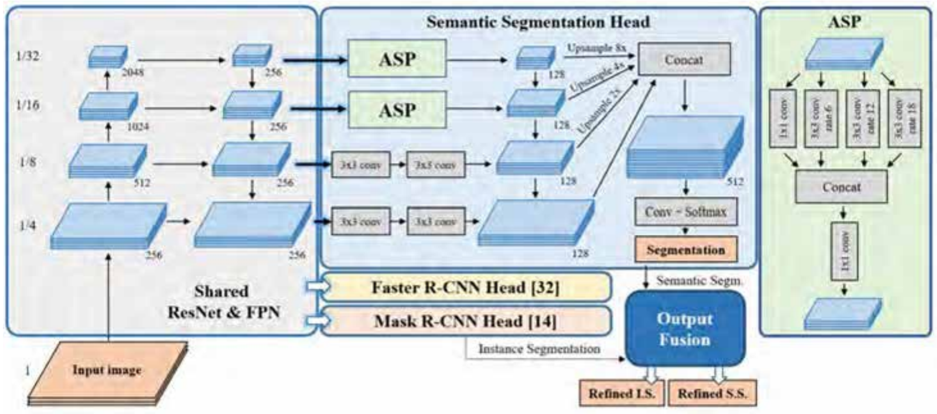


Figure 5.2: A shared ResNet-FPN network is used for three tasks. The Faster-RCNN head performs object detection, Mask-RCNN head performs instance segmentation. The semantic segmentation head is based on Atrous Spatial Pyramid (ASP).

multi-scale feature representation.

A novel semantic segmentation head is introduced, sharing the same feature representation with the instance segmentation and object detection heads, as illustrated in Figure 5.2. Given that the size of objects in a scene varies depending on their distance from the camera, the model is designed to effectively learn multi-scale features. To address this, the architecture incorporates several mechanisms: the Feature Pyramid Network (FPN), atrous convolutions, and multi-scale image processing.

The FPN output consists of 256 feature maps at five different scales: 1/4, 1/8, 1/16, 1/32, and 1/64. The semantic segmentation head utilizes FPN feature maps at four scales, ranging from 1/4 to 1/32, with an individual segmentation head at each scale to capture multi-resolution features. Atrous (dilated) convolutions are employed to extract context and long-range information, enhancing the model's ability to understand global features. The top pyramid levels (1/32 and 1/16), which provide stronger semantics and better localization, are further enhanced using an Atrous Spatial Pyramid (ASP) [22]. The ASP module applies a parallel combination of 1×1 convolution and 3×3 dilated convolutions with dilation rates of 6, 12, and 18, followed by Group Normalization layers [156], which were found to be more effective than Batch Normalization [69] in this setup. Group Normalization, unlike Batch Normalization, normalizes along groups of channels, making it invariant to batch size.

This approach addresses the issue of inaccurate statistics that arise with small batch sizes, as the network is trained with a batch size of 2 images due to GPU memory constraints. Each Group Normalization layer is followed by a ReLU activation. The resulting feature maps from the ASP are concatenated and passed through 128 1×1 filters. To incorporate finer scale features, two 3×3 convolutions are applied to the 1/8 and 1/4 levels, following the approach in [75]. Each segmentation head at the four scales produces 128 feature maps.

For further refinement, the outputs are fused using a refinement pyramid (RP), which aggregates stronger semantics from higher levels into the lower levels of the feature pyramid. Starting with the highest scale layer, the feature maps are upsampled by a factor of two and added to the outputs of the subsequent layer. The fused outputs are finally upsampled and concatenated into 512 feature maps at 1/4 scale. A 1×1 convolution is applied to generate the final class predictions, completing the segmentation process.

To obtain the final segmentation predictions, a per-pixel softmax is applied, and the cross-entropy loss is minimized during training. The model is optimized using a multi-task loss, defined as the weighted sum of the bounding box regression loss, object classification loss, instance segmentation loss, and semantic segmentation loss:

$$\mathcal{L} = \lambda_{box}\mathcal{L}_{box} + \lambda_{cls}\mathcal{L}_{cls} + \lambda_{mask}\mathcal{L}_{mask} + \lambda_{segm}\mathcal{L}_{segm}$$

The weights for the bounding box regression, object classification, and instance segmentation losses follow the settings in Mask R-CNN [63], with the semantic segmentation loss weight set to $\lambda_{segm} = 1$.

An instance segmentation network generates overlapping instances, meaning that a single pixel can be associated with multiple identifiers. Additionally, since the semantic and instance segmentation networks process images independently, their outputs may not align perfectly, resulting in mismatches in pixel-level semantic labels. An example of such conflicts between instance and semantic segmentation is shown in Figure 5.3.

To address this issue, a novel panoptic fusion scheme is introduced to integrate the outputs of instance and semantic segmentation. This fusion ensures that each pixel is assigned a unique semantic and instance label while simultaneously improving overall accuracy. The approach builds on the observation that pixel-level semantic segmentation

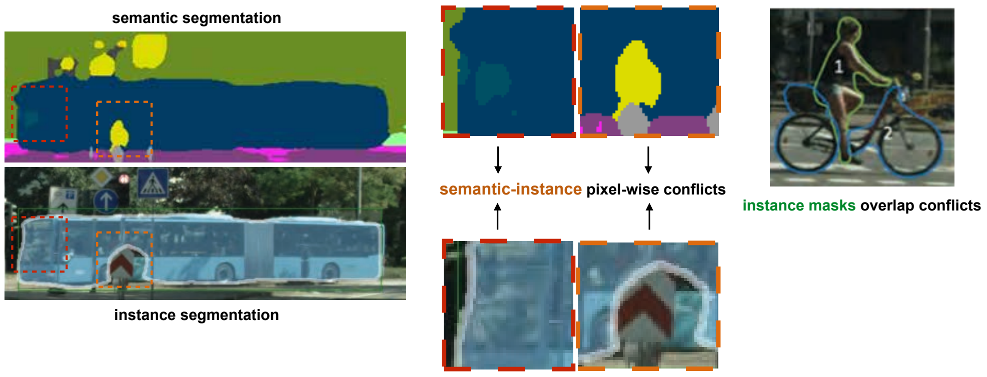

semantic segmentation
instance segmentation
semantic-instance pixel-wise conflicts
1
2
instance masks overlap conflicts

Figure 5.3: Semantic-instance and instance-instance conflicts. A pixel can receive different semantic classes from the semantic and instance outputs, as seen on the truck example. Also, a pixel can belong to two instances, as the instance masks overlap, as seen in the cyclist-bicycle example.

excels at background classification and foreground-background separation, while instance segmentation is more effective at recognizing and classifying objects as a whole. By leveraging the strengths of both methods, the fusion scheme corrects classification errors in semantic segmentation using the semantic class information provided by instance segmentation.

This approach proves particularly beneficial for large objects, where instance segmentation masks, due to their lower resolution (28×28), are inaccurate at boundaries. In contrast, semantic segmentation offers more precise boundary delineation between things and stuff classes (e.g., distinguishing pedestrians from sidewalks) but encounters difficulties differentiating between classes within the same category (e.g., pedestrians and cyclists). By combining the complementary strengths of the two segmentation methods, the fusion scheme enhances both accuracy and consistency.

The fusion approach leverages the strengths of both methods by utilizing semantic pixel-level classification for *stuff* classes and instance-level classification for *things* classes. To accomplish this, pixels are first categorized into *stuff* and *things* classes based on semantic segmentation results. Additionally, the *things* classes, such as bus, car, truck, pedestrian, cyclist, bicycle, and motorcycle, are grouped into broader categories: vehicles, humans, and two-wheeled objects. The detailed algorithm is presented in Algorithm 1 and illustrated in Figure 5.4.

Stuff. Pixels classified as *stuff* are those assigned to a *stuff* class in the semantic segmentation.

Things. For *things* classes, the masks provided by instance segmentation and semantic segmentation may not align perfectly, and their semantic classes can differ. To address this, a new semantic class is assigned to each *things* pixel identified in the semantic segmentation. This process considers the class provided by the object detector, using the instance mask to facilitate pixel-to-pixel matching. A pixel's class label and instance label from the instance segmentation are retained only if they are consistent with the pixel's semantic category in the semantic segmentation. For *things* pixels in the semantic segmentation that are not covered by an instance mask but are connected to previously matched pixels via a semantic path, a breadth-first-search region-growing algorithm is applied. This algorithm propagates the semantic class and instance identifier, enabling more accurate object-level classification compared to pixel-level classification. For *things* pixels that remain without an instance identifier after the region-growing process, typically due to isolation, a new instance identifier is generated, while the semantic class from the semantic segmentation is preserved. If any segments not labeled by the matching and region-growing algorithms are smaller than a predefined threshold, they are assigned an unknown class to minimize the introduction of false-positive segments.

The output fusion scheme can be applied for any semantic segmentation and instance segmentation output without depending on the employed approaches. It can be used as a fast post processing step which provides a unified panoptic output.

Experiments. The multi-task network for instance and semantic segmentation (ISSNet) is evaluated, and the complete solution, which includes ISSNet along with the semantic and instance segmentation fusion scheme, is referred to as ISS-Fusion. The model is initialized using Mask R-CNN [63] weights pretrained on the Microsoft COCO dataset [97] for instance segmentation and object detection. Stochastic Gradient Descent (SGD) with a momentum of 0.9 is employed, alongside a poly learning rate policy starting at 5×10^{-3}. The network converges after 32k iterations on the Cityscapes and UP-Drive datasets. Data augmentation techniques, including horizontal flipping and multi-scale image training (scales ranging from 0.8 to 1), are applied during training.

Table 5.2 presents the results for the ISS-Fusion on the

Algorithm 1 Semantic and Instance Fusion Scheme

```
Input: Semantic segmentation $S$, overlapping instance masks $M$, instance area threshold $T$
Output: Panoptic segmentation $P$
  $I_v \leftarrow$ GenerateInstanceSegmentation($M, cat = vehicle$)
  $I_h \leftarrow$ GenerateInstanceSegmentation($M, cat = human$)
  $I_t \leftarrow$ GenerateInstanceSegmentation($M, cat = two - wheeled$)
  P ← Matching(S, $I_v$, $I_h$, $I_t$)
  P ← Filling(P)
  P ← CreateNewInstances(P, T)

  procedure GenerateInstanceSegmentation($M$, $c$)
      Let $masks$ be an empty instance mask vector;
  // Each mask has a semantic class, instanceID and confidence score
      for mask $m \in M$ do
          if $score(m) > 0.5$ then
              if $class(m) \in c$ category then
                  Add $m$ to masks;
              end if
          end if
      end for
      Sort masks with respect to the scores in descending order;
      Let $I$ be an empty instance segmentation;
      for mask $m \in masks$ do
          n ← no of pixels in $m$, e ← no of empty locations in $I(m)$;
          if $e/n > 0.5$ then
              Paste $m$ in $I$
          end if
      end for
      return $I$
  end procedure
```

```
procedure MATCHING($S, I_v, I_h, I_t$)
    Initialize panoptic segmentation $P$ as a copy of $S$
    Set instanceIDs of every location in P to 0
    for pixel $p \in S$ do
        for $I \in I_v, I_h, I_t$ do
            if $cat(p) == cat(I)$ and $I(p) > 0$ then
                $P(p) \leftarrow I(p)$ // assign class, instanceID and score
            end if
        end for
    end for
    return $P$
end procedure

procedure FILLING($S, I$)
    Let Q be a queue
    for $p \in P$ do
        if $instanceID(p) > 0$ then
            Q.enqueue($p$)
        end if
    end for
    while Q is not empty do
        q = Q.dequeue()
        for each neighbor $n \in N_8(q)$ do
            if $cat(n) == cat(q)$ and $instanceID(n) == 0$ then
                P(n) $\leftarrow$ class(q), instanceID(q), score(q)
                Q.enqueue(n)
            end if
        end for
    end while
    return $P$
end procedure
```

```
procedure CreateNewInstances($P$, T)
    for $p \in P$ do
        if $class(p) \in things$ classes and $instanceID(p) == 0$ then
// Find connected component of pixels with the same semantic class
            blob ← BFS(p)
            if $Area(blob) < T$ then
                class(blob) = unknown // Set the class to unknown
                instanceID(blob ← = 0
            else
                instanceID(blob) ← max(instanceID(P)) + 1
            end if
        end if
    end for
    return $P$
end procedure
```

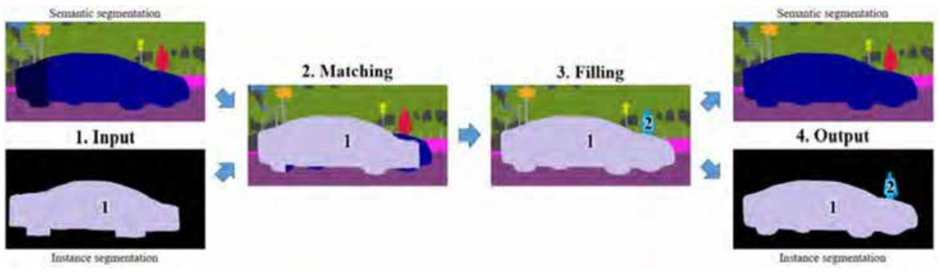


Figure 5.4: Fusion process overview. (1) Input: semantic segmentation (car is partially classified as truck) and instance segmentation (mask for car is slightly misaligned and cropped, and the pedestrian behind the car is not detected); (2) Matching: the pixels of the instance segmentation are matched to the pixels from semantic segmentation only if the object class is compatible with the semantic category from semantic segmentation; (3) Filling: semantic region growing is applied to finalize the object shape and the unmatched object segments receive a new object ID; (4) Output: refined segmentation.

Cityscapes validation set. The unified baseline model demonstrates an improvement over the Mask R-CNN framework for instance segmentation [63]. The unified baseline employs a simplified segmentation head consisting of two 3×3 convolutions after each level of the FPN, followed by upsampling and concatenation. In comparison to the unified baseline, incorporating the Atrous Spatial Pyramid (ASP) and the refinement pyramid (RP) yields an approximate improvement of 1.5% in mIoU for semantic segmentation. Additionally, the fusion scheme further enhances the performance, resulting in a 3% increase in mIoU for semantic segmentation. This improvement is attributed to more robust object-level classification and the assignment of a unique label per instance, which strengthens the semantic segmentation for foreground classes. Furthermore, the instance masks achieve better alignment with object boundaries.

Method	backbone	mAP mask	mIoU	PQ
Mask-RCNN [63]	ResNet-50	36.4	-	-
Unified baseline	ResNet50-FPN	37.0	71.6	-
+ ASP and RP	ResNet50-FPN	37.2	72.9	-
+ fusion	ResNet50-FPN	**37.3**	**76.0**	**56.8**

Table 5.2: Evaluation of ISS-Fusion on the Cityscapes *val* set.

Table 5.3 presents the class-wise semantic segmentation evaluation on the Cityscapes dataset. The results show performance improvements across all things classes when using the semantic and instance fusion scheme. Notably, significant IoU improvements are observed for large-scale semantic classes, such as truck, bus, and train.

Table 5.4 presents a comparison with other approaches on the Cityscapes test set. The solution achieves competitive results but is outperformed by methods that utilize larger backbone networks, such as ResNet-101, which were trained with large batch sizes. However, the use of larger backbones comes at the expense of increased computational costs and memory requirements.

Figure 5.5 shows results for semantic and instance segmentation on Cityscapes validation images, both before and after the fusion process. For semantic segmentation, pixel-level classification can produce erroneous semantic labels for large-scale, challenging objects such as buses, trams, or trucks. These errors are corrected during the fusion process by incorporating object-level classification results for foreground classes. For instance masks, the improvements are particularly noticeable at object boundaries, where better alignment and preservation of details are achieved. Additionally, the fusion module generates panoptic segmentation outputs, as illustrated in Figure 5.6.

Class	ISSNet	ISS-Fusion
road	97.7	97.7
sidewalk	82.3	82.3
building	91.2	91.2
wall	48.6	48.6
fence	51.3	51.3
pole	56.9	56.9
traffic light	66.9	66.9
traffic sign	73.1	73.1
vegetation	91.5	91.5
terrain	61.8	61.8
sky	93.1	93.1
person	80.1	**81.0**
rider	59.8	**65.6**
car	93.1	**94.0**
truck	63.0	**81.6**
bus	77.5	**89.8**
train	64.1	**80.1**
motorcycle	59.7	**61.9**
bicycle	75.3	**75.6**
mIoU	72.9	**76.0**

Table 5.3: Semantic segmentation evaluation for Cityscapes semantic classes before (ISSNet) and after fusion (ISS-Fusion).

The network is also evaluated on the UP-Drive dataset to assess whether the it meets the requirements of the 2D perception system on an autonomous vehicle such as inference speed. The results are presented in Table 5.5 and the processing time is measured on a NVIDIA GTX 1080 GPU. Using high-resolution images of size 640×1280, the network achieves 31.3% mAP for mask prediction and 66.4% segmentation mIoU. However, the two-stage network proves computationally expensive, with inference times of 171 ms for high-resolution images. Although the network achieves accurate results, the computational demand is too high for practical applications such as autonomous driving. Therefore the input images are downsampled to 256×512 to reduce inference time. However, processing low-resolution fisheye images further decreases the detection range, which is particularly problematic given that objects in fisheye images already appear smaller compared to those in narrow field-of-view images. End-to-end network optimization with TensorRT is challenging

Method	mIoU
DeepLabv2-CRF [21]	70.4
Deep Layer Cascade [89]	71.1
ML-CRNN [43]	71.2
Adelaide context [94]	71.6
FRRN [125]	71.8
LRR-4x [53]	71.8
RefineNet [93]	73.6
Ladder DenseNet [80]	74.3
TuSimple [149]	80.1
ResNet-38 [157]	80.6
PSPNet [168]	81.2
DeepLabV3 [22]	81.3
DeepLabV3+ [24]	82.1
ISS-Fusion	72.7

Table 5.4: Cityscapes results on the *test* set.

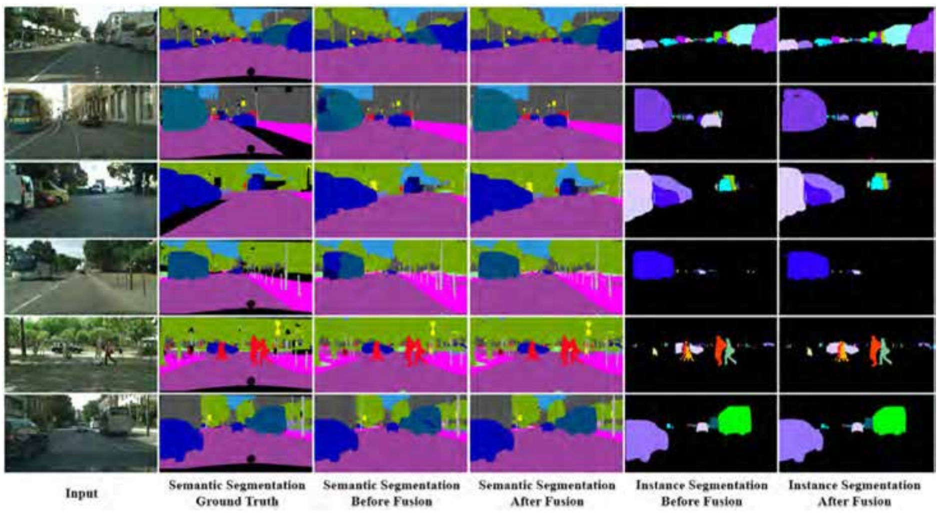

Input
Semantic Segmentation Ground Truth
Semantic Segmentation Before Fusion
Semantic Segmentation After Fusion
Instance Segmentation Before Fusion
Instance Segmentation After Fusion

Figure 5.5: Demo results for the ISS-Fusion network on Cityscapes images. The fusion process improves the segmentation quality.

for the two-stage network due to the presence of hand-crafted processing steps that must be separately implemented, such as those between the first and second object detection stages and between the second stage and the mask head. The complexities of the training and inference pipelines,

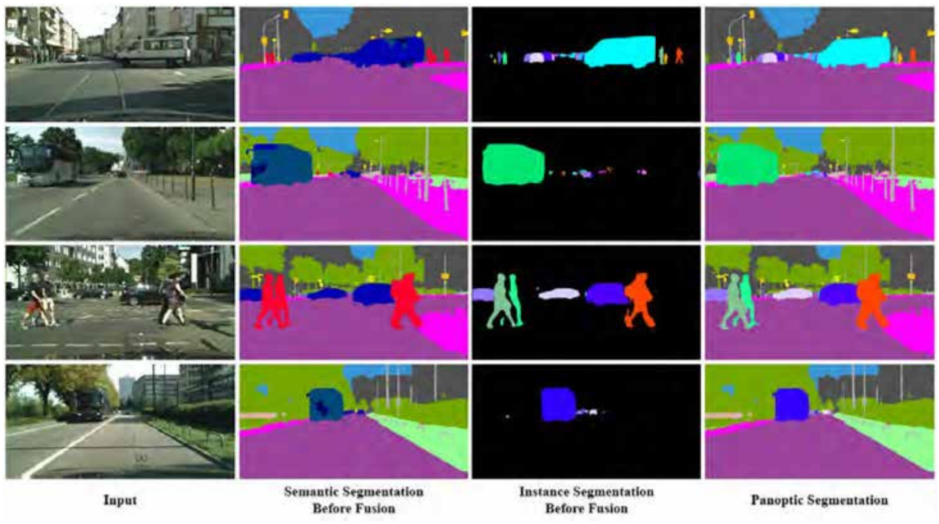

Input
Semantic Segmentation Before Fusion
Instance Segmentation Before Fusion
Panoptic Segmentation

Figure 5.6: Panoptic segmentation output from the ISS-Fusion network on Cityscapes images. The semantic and instance fusion scheme assigns to each pixel from the panoptic output a unique semantic class and instance identifier.

Resolution	mAP box	mAP mask	mIoU	Time (ms)
640 × 1280	**37.8**	**31.3**	**66.4**	171
256 × 512	29.7	25.2	61.8	**68**

Table 5.5: Evaluation of ISSNet on the UP-Drive fisheye images corresponding to front, left, back, right views. Time is measured on a NVIDIA GTX 1080 GPU.

coupled with the high inference time even at reduced resolutions, highlight the need for more efficient one-stage panoptic segmentation networks to be implemented in autonomous vehicles.

An example of a 2D perception system. Figure 5.7 illustrates an example of a 2D perception system architecture which processes input images from five cameras: four fisheye surround-view cameras providing 360° coverage around the vehicle and one front narrow field-of-view camera. The Data Flow Manager receives synchronized image samples from these cameras and selects the best temporally aligned set at the start of each perception processing cycle. The four fisheye images are first unwarped and undistorted before undergoing semantic segmenta-

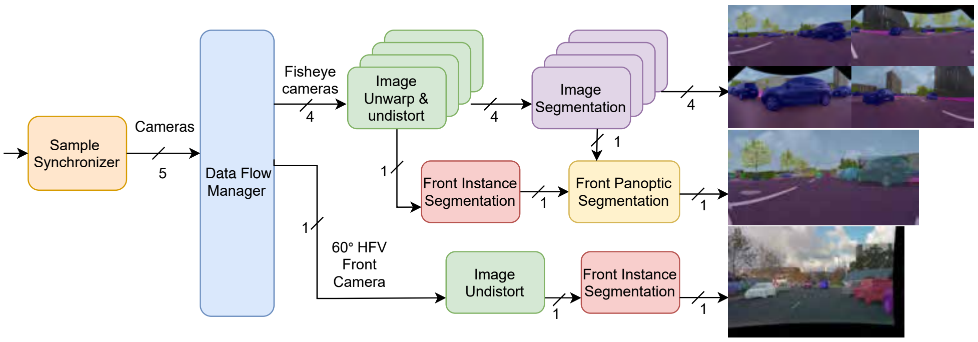


Figure 5.7: An example of a 2D panoptic perception system.

tion. For the front unwarped fisheye image, instance segmentation is additionally performed, followed by a fusion of semantic and instance segmentation outputs to produce the final panoptic output.

The instance segmentation network that is used in this system is the one-stage network RetinaMask[46] and the ERFNet [131] semantic segmentation network is selected. Both networks are trained on the UPDrive dataset, as detailed in earlier chapters. The post-processing step presented in the previous section is used to fuse the instance and semantic segmentation outputs into panoptic segmentation.

Table 5.6 presents the results of ablation studies for each segmentation module on the front unwarped fisheye images at a resolution of 640 $\times$ 1280 on the UP-Drive dataset. Using the INT8 model for semantic segmentation, the mean Intersection over Union (mIoU) is 65.1% when computed across all four view images, and 65.9% for the front view images alone. Incorporating the unified panoptic segmentation further improves the semantic segmentation performance by nearly 1%, increasing the mIoU from 65.9% to 66.8%. Additionally, the panoptic module enhances the instance segmentation mean Average Precision (mAP) by 0.3%. In terms of panoptic quality (PQ), the results show 42.4% PQ for all classes, with 42.2% for *stuff* classes and **43.0%** for *things* classes.

The inference time for the complete 2D perception pipeline is measured and summarized in Table 5.7. The preprocessing step, which includes image unwarping for the four fisheye images and image undistortion for all five images, is efficient and completes in 5 ms on the GPU. Semantic segmentation of the four unwarped fisheye images, performed using an INT8 quantized network, takes 36 ms. The most time-intensive operation is instance segmentation of the front unwarped fisheye image,

Model	mIoU	mAP mask	PQ	PQ_{st}	PQ_{th}	Time (ms)
Semantic segmentation	65.9	-	-	-	-	9
Instance segmentation	-	30	-	-	-	66
Panoptic fusion	66.8	30.3	42.4	42.2	43	5
Panoptic segmentation	66.8	30.3	42.4	42.2	43	80

Table 5.6: Ablation studies of each segmentation module on the front view unwarped fisheye image on the UP-Drive dataset. Semantic segmentation, instance segmentation and panoptic segmentation is evaluated on 640 $\times$ 1280 images. The networks are optimized with TensorRT. Time is measured on a NVIDIA GTX 1080 GPU.

Module	Time (ms)
Image unwarping and undistort $\times$ 5	6
Semantic segmentation $\times$ 4	36
Instance segmentation front 160° HFV image	66
Panoptic fusion front 160° HFV image	5
Instance segmentation front 60° HFV image	44
Total	157

Table 5.7: Time evaluation of the entire 2D semantic perception system on a NVIDIA GTX 1080 GPU.

which requires 66 ms. To reduce inference time, the resolution of the front 60° horizontal field-of-view (HFV) image is lowered, reducing its processing time to 44 ms. Overall, the entire pipeline runs in 157 ms. By omitting the instance segmentation for the front 160° HFV image and the panoptic fusion step, the pipeline achieves a faster runtime of 86 ms on a single NVIDIA GTX 1080 GPU. This solution based on a one-stage instance segmentation network is more suitable for deployment on a autonomous vehicle than a two-stage instance and semantic segmentation network due to the good trade-off between accuracy and inference time.

5.4 Panoptic Prototype Network

Method. This work [117] introduces an innovative end-to-end fully convolutional neural network named Panoptic Prototype tailored for panoptic segmentation, emphasizing fast inference, high accuracy, and practical deployment in real-time robotic applications. The central strategy for achieving efficient inference lies in adopting a one-stage object detector framework instead of a more computationally expensive two-stage approach, coupled with carefully designed lightweight network heads for panoptic segmentation.

Traditional two-stage instance segmentation models, such as Mask R-CNN [63], rely on Faster R-CNN [130] as their backbone. These models operate sequentially: the first stage generates regions of interest (RoIs), while the second stage uses operations like RoIAlign to localize features and predict object classes and instance masks. Although effective, the sequential nature and computational complexity of these two-stage methods hinder acceleration via inference engines (e.g., TensorRT [6]) and limit their viability for real-time systems.

To overcome these limitations, Panoptic Prototype builds on the one-stage FCOS object detector [142], which formulates object detection as a per-pixel prediction problem. FCOS is selected as the base architecture due to its impressive speed and accuracy, achieving state-of-the-art performance on several benchmarks. Its fully convolutional, proposal-free, and anchor-free design reduces the number of parameters, simplifying both the training and inference processes. This streamlined architecture significantly enhances computational efficiency, making it highly suitable for real-time deployment in robotic applications.

The network architecture, illustrated in Figure 5.8, comprises a shared backbone, an object detection head based on FCOS [142], and a panoptic segmentation head. To produce high-quality panoptic predictions, a dual-branch panoptic segmentation head that operates at high resolution is introduced: one branch is dedicated to generating semantic stuff masks and the other to producing instance masks. The instance branch introduces an instance-aware visual codebook for each image, which comprises a fixed set of prototype feature maps activated at specific instance locations. Leveraging object proposals, the panoptic head employs a learned weighting scheme to linearly combine the predicted instance prototype masks [17]. These combined instance masks are then seamlessly integrated with the stuff segments to generate the final panoptic output. The network is trained under supervision using an object detection loss, a semantic segmentation loss, and a panoptic segmentation loss.

Backbone. For feature extraction, the network utilizes the VoVNet2-39 backbone, chosen for its efficiency [85], in combination with

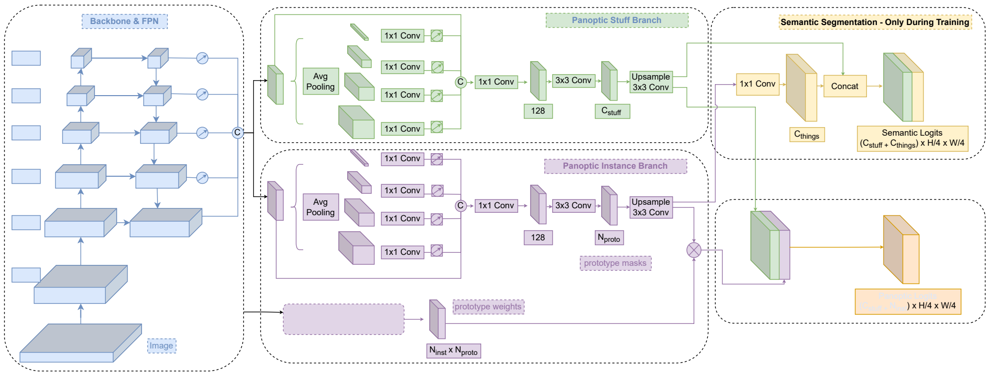


Figure 5.8: The panoptic segmentation network architecture. A shared backbone and FPN are used for feature detection. A dual-branch panoptic head performs *stuff* and instance mask prediction. Guided by object proposals, the network learns an instance-wise weighting scheme which is used for assembling the prototypes into instance masks. The panoptic head fuses *stuff* and instance masks into one coherent output.

a Feature Pyramid Network (FPN) [95]. While the original implementation employs a 5-level FPN with 256 channels per level, this design reduces the FPN size to 128 channels per level to improve efficiency.

Object Detection Head. The object detection head is based on the anchor-free, fully convolutional FCOS detector [142]. Built on top of the shared VoVNet2-39-FPN backbone, each FPN level is extended with a lightweight object regression branch and a shared branch for object classification and centerness. Both branches consist of two $[3 \times 3, 128]$ convolutional layers.

Panoptic Segmentation. The panoptic segmentation head is implemented with a dual-branch design with a lightweight architecture for *stuff* and prototype masks predictions.

Panoptic Stuff Branch. The panoptic *stuff* branch is designed to predict a mask for each *stuff* class. It begins with the multi-scale feature representation produced by the Feature Pyramid Network (FPN), which encodes features at five scales: $1/128, 1/64, 1/32, 1/16$, and $1/8$ of the original input resolution. Feature maps from all pyramid levels, each containing 128 channels, are upsampled to $1/8$ and concatenated. To model long-range dependencies, a Pyramid Pooling Module (PSP) [168] is applied on top of the concatenated features. The PSP module performs four parallel average pooling operations at different rates, producing feature maps of sizes $[1 \times 1], [2 \times 2], [3 \times 3]$, and $[6 \times 6]$. A $[1 \times 1, 128]$ convolution is then applied at each level, and the resulting features are upsampled to the 1/8 scale via bilinear interpolation. The output of the PSP module is generated by fusing the 4-level pyramid features with the input through concatenation, followed by a $[1 \times 1, 128]$ convolution. A $[3 \times 3, C_{\text{stuff}}]$ convolution, batch normalization, and ReLU activation are applied to predict the *stuff* class logits. These logits are further refined using $2\times$ bilinear interpolation and a $[3 \times 3, C_{\text{stuff}}]$ convolution to produce the final *stuff* masks. The resulting *stuff* logits, with a size of $C_{\text{stuff}} \times H/4 \times W/4$, contribute to the panoptic segmentation logits. To supervise the learning process, a semantic segmentation loss is introduced during training. For *things* logits, a $[1 \times 1, C_{\text{things}}]$ convolution is applied to the prototype masks. Finally, the *stuff* and *things* logits are concatenated to form the semantic segmentation output, which is optimized using a bootstrapped cross-entropy loss.

Prototype Masks Branch. Inspired by the instance segmentation network Yolact [17], a panoptic head is designed to generate a mask for each instance in the image by learning prototype masks and assembling them using a weighting scheme associated with each detected object. To produce the prototype masks, the FPN is extended with a branch similar to the panoptic *stuff* branch. Consequently, both branches share the backbone, the FPN, the upsampling operation to 1/8, and the concatenation step. The network then aggregates contextual information by applying a Pyramid Pooling Module (PSP) similar to the one used earlier. This is followed by a $[3 \times 3, N_{\text{proto}}]$ convolution, batch normalization, and ReLU activation, where N_{proto} represents the number of prototype masks. After extensive experimentation, $N_{\text{proto}} = 32$ was selected as the optimal number of prototypes. To capture small objects and fine details, the resolution is further increased to 1/4. This is achieved using a $2\times$ bilinear upsampling operation followed by a $[3 \times 3, N_{\text{proto}}]$ convolution. Notably, no direct supervision is enforced on the prototypes; instead, they are trained indirectly through the panoptic and semantic losses. Figure 5.9 illustrates the masks corresponding to *stuff* classes alongside a selection of instance prototype masks. The visualizations highlight how the network distributes attention across different regions in the prototype masks—some focus on instance regions, others on edges, and some on the image background.

Prototype Weights. The FCOS detector [142] predicts, at

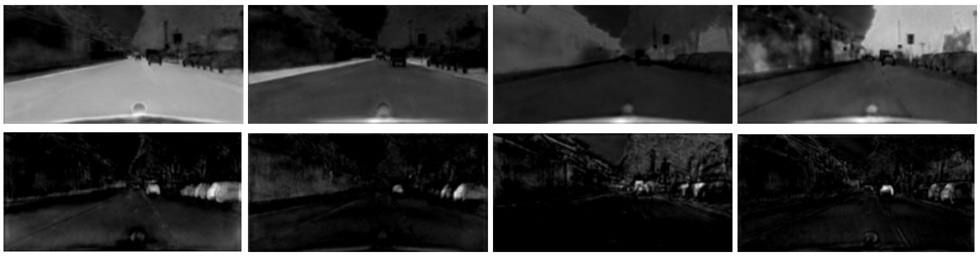


Figure 5.9: Top row: stuff segmentation masks. Bottom row: prototype instance masks. These masks are assembled into panoptic logits.

each foreground location of the 5-scale FPN, a 4D vector encoding the offsets from the four sides of the bounding box, a centerness value, and a semantic class. FCOS employs a shared object detection head across all FPN levels. This head consists of two branches: one for classification and centerness prediction, and the other for bounding box regression. In its original implementation, each branch comprises four $[3 \times 3, 256]$ convolutional layers. To improve inference speed, the number of convolutions in each branch is reduced to 2, and the number of channels is decreased to 128. To enable learning a weighting scheme for prototype assembly, the detection head is extended with an additional parallel branch that predicts a vector of N_{proto} values at each foreground location. Importantly, no explicit loss is applied to supervise the learning of these weights.

The final loss function is formulated as follows:

$$\begin{aligned}\mathcal{L} = \lambda_{box}\mathcal{L}_{box} + \lambda_{cls}\mathcal{L}_{cls} + \lambda_{centerness}\mathcal{L}_{centerness} \\ +\lambda_{semantic}\mathcal{L}_{semantic} + \lambda_{panoptic}\mathcal{L}_{panoptic}\end{aligned} \tag{5.4}$$

During training, the object detection loss weights for box regression, classification, and centerness are configured following the settings of FCOS [142]. The weights for the semantic loss and panoptic loss are set to $\lambda_{\text{semantic}} = 1$ and $\lambda_{\text{panoptic}} = 1$, respectively. Both the panoptic and semantic losses are implemented using the bootstrapped cross-entropy loss. The panoptic segmentation logits are constructed by combining *stuff* logits, taken directly from the panoptic *stuff* branch, and instance logits, generated by the prototype mask branch. To remove low-quality bounding boxes, multi-class Non-Maximum Suppression (NMS) with an IoU threshold of 0.5 is applied to resolve overlaps, retaining the top 50 scoring boxes. For each ground truth mask in the panoptic target, the predicted bounding box with the largest intersection-over-union (IoU) is assigned. This ensures that each ground truth mask is associated with only one prediction, and the order of instance masks in the panoptic logits matches the order of masks in the panoptic target. During training, ground truth bounding boxes are utilized to suppress values outside their regions by setting those locations to zero.

During inference, an additional filtering step is applied to the predicted bounding boxes. After performing Non-Maximum Suppression (NMS) and selecting the top 50 scoring boxes, the remaining boxes are sorted, and only those with confidence scores greater than 0.3 are retained. The panoptic segmentation logits are then assembled in the same manner as during training. To generate the final predictions, a softmax operation is applied over the panoptic segmentation logits to determine per-class assignments. If the maximum value lies within the first C_{stuff} channels, the corresponding pixel is classified as belonging to a *stuff* class, with its semantic class given by the index of the maximum value. Conversely, if the maximum is found within the subsequent N_{inst} channels, the pixel is classified as part of an instance mask, with the instance ID determined by the index and the semantic class derived from the bounding box class. For evaluation purposes, *stuff* masks with areas smaller than 1024 pixels are discarded. This step addresses the sensitivity of the Panoptic Quality (PQ) metric, particularly its Recognition Quality (RQ) component, to small and low-quality predictions.

Panoptic Masks Assembly. For each detected object, an instance mask is generated by performing a linear combination of the instance prototypes using the learned instance weights. To enable both addition and subtraction of prototype masks, the prototype weights are first remapped to the range $[-1, 1]$ using a tanh activation function. After the linear combination, noise outside the detected bounding box is suppressed by setting those locations to zero. The resulting instance logits are high-resolution, with a size of $N_{\text{inst}} \times H/4 \times W/4$. In the final step of the panoptic assembly process, the instance logits are concatenated with the *stuff* logits to produce the panoptic logits.

Experiments. The model is implemented using the PyTorch framework [113] and trained on a system equipped with four Tesla V100 GPUs. For inference, a single GPU is used with a batch size of 1. Experiments are conducted on the Cityscapes dataset [34]. The network is trained end-to-end with a batch size of one image per GPU. Optimization is performed using stochastic gradient descent (SGD) with a momentum of 0.9, weight decay of 10^{-4}, and a polynomial learning rate decay starting at 2×10^{-3}. The model is trained for 16k iterations. To improve generalization, data augmentation techniques are applied, including random horizontal flipping and scaling, where the shorter side of the image is randomly selected from $\{768, 896, 1024\}$.

The experimental results for the Prototype Panoptic network are presented on the urban driving Cityscapes dataset. Additionally, ablation studies are conducted to justify and motivate the design choices. The network is evaluated using the VoVNet2-39 backbone [85].

Ablation Studies. Ablation studies are conducted on the Cityscapes dataset to analyze the impact of various design choices. The baseline model employs a single decoder with a Pyramid Pooling Module (PSP) for contextual information, a $[1 \times 1]$ convolution for feature aggregation, and two parallel convolutions to reduce the channel dimensions for *stuff* and prototype masks: $[3 \times 3, C_{\text{stuff}}]$ and $[3 \times 3, 64]$, respectively. The output stride is set to $8\times$, and 64 prototype masks are predicted. This baseline configuration achieves a Panoptic Quality (PQ) score of 53.2%.

To improve performance, an additional decoder is introduced in parallel, allowing each branch to specialize in background and foreground features, respectively. This modification leads to a 1% improvement in PQ. Recognizing the importance of high-resolution feature learning for accuracy, an upsampling stage using bilinear interpolation followed by a $[3 \times 3]$ convolution is added. Reducing the output stride from $8\times$ to $4\times$ further improves PQ by 3.2%.

The effect of the number of prototype masks is also studied. Reducing the number of prototypes from 64 to 32 does not degrade performance, achieving the same PQ score of 57.3%. However, increasing the number of prototypes does not yield further improvements and may introduce redundant prototype masks while increasing computational cost. Conversely, using fewer prototypes leads to a decline in panoptic quality.

In conclusion, the final model adopts 32 prototype masks, which achieves the best trade-off between accuracy and computational efficiency.

Performance on Cityscapes. Table 5.9 presents a comparison of the panoptic and semantic segmentation accuracy and inference time of the one-stage Panoptic Prototype network against existing two-stage, proposal-based, and one-stage approaches. The reported results are based on lightweight backbones, such as ResNet-50 and MobileNetV2. Inference time includes the forward pass of the network and the Non-

Dec. × 1	Dec. × 2	4×	N_{proto}	PQ
✓			64	53.2
	✓		64	54.2
	✓	✓	64	57.3
	✓	✓	32	**57.3**
	✓	✓	16	56.9

Table 5.8: Ablation study on the Cityscapes *validation* set for Panoptic Prototype network: Baseline model has one decoder, 8× output stride, 64 prototype masks. **Dec.× 1:** shared decoder for *stuff* and prototype masks. **Dec.× 2:** separate decoders. **4×**: output stride = 4. $\mathbf{N}_{proto}$**:** number of prototypes.

Maximum Suppression (NMS) operation on a 1024 × 2048 Cityscapes image.

Compared to two-stage methods, the Panoptic Prototype network outperforms TASCNet [87], achieving better results while running twice as fast. It delivers comparable Panoptic Quality (PQ) to MTN Panoptic [116] and Panoptic-FPN [75]. However, methods like UPSNet [159] and Seamless Panoptic [126] achieve higher accuracy at the expense of significantly longer execution times.

In general, the network demonstrates strong performance in *stuff* classification, achieving $PQ_{\text{st}} = 62.4\%$. However, its instance mask predictions are less accurate compared to two-stage approaches. This limitation may arise in challenging scenes with dense crowds or significant occlusions, where the prototype masks struggle to localize individual instances. Additionally, the instance mask quality heavily depends on the object detector's accuracy, as masks are retained only within the detected bounding boxes. Misaligned, misclassified, or incorrectly sized bounding boxes can introduce artifacts into the panoptic and instance predictions.

When compared to other one-stage methods, this network is both the fastest, with an inference time of 82 ms, and the most accurate. It surpasses DeeperLab [162], FPSNet [37], and SSAP [48] in overall performance.

In Figure 5.10, visual results for panoptic and semantic segmentation are presented. The results demonstrate that the network effectively handles diverse scenarios, including occlusions and objects of various sizes. By generating masks from the 5-level Feature Pyramid Network (FPN) and Pyramid Pooling Module (PSP), the network captures multi-

Method	Backbone	PQ	PQ_{th}	PQ_{st}	mIoU	Time (ms)
Two-Stage						
TASCNet [87]	ResNet50-FPN	55.9	50.5	59.8	-	160
MTN Panoptic [35]	ResNet50-FPN	57.3	53.9	59.7	75.4	145*
Panoptic-FPN [75]	ResNet50-FPN	58.1	52.0	62.5	75.7	-
UPSNet [159]	ResNet50-FPN	59.3	54.6	62.7	75.2	**140**
Seamless Panoptic [126]	ResNet50-FPN	**60.3**	**56.1**	**63.3**	**77.5**	150
One-stage						
DeeperLab [162]	Wider MNV2	52.3	-	-	-	303
FPSNet [37]	ResNet50-FPN	55.1	48.3	60.1	-	98*
SSAP [48]	ResNet-50	56.6	49.2	-	-	130*
Panoptic Prototype	VoVNet2-39	**57.3**	**50.4**	**62.4**	**76.9**	**82**

Table 5.9: Comparative study with state-of-the-art panoptic segmentation networks on the Cityscapes *validation* dataset. Inference time is measured on a Tesla V100 GPU with batch size of 1. Inference time is approximated for entries marked with * based on their reported speed on other GPUs.

scale contextual information, enabling accurate pixel-level classification even for very large objects, such as the truck in column 4.

Simultaneously, the instance masks are produced at a high resolution, ensuring precision and the preservation of fine details. For example, in the last column, the pedestrians are well delineated despite significant occlusions in the scene.

5.5 Soft Attention Mask Network

Method. The Soft Attention Mask Network [119] introduces **AttentionPS**, a simple, fast, and accurate approach for generating coherent panoptic segmentation using an end-to-end trainable network. The key idea behind this method is to extract instance masks from the semantic segmentation output by leveraging high-resolution approximate representations for instances, referred to as *soft attention masks.*

The network utilizes a shared feature extraction backbone and incorporates three distinct heads: one for object detection, one for semantic segmentation, and one for instance-level attention masks. Inspired by proposal-free instance segmentation approaches [73, 109, 28], which generate instance masks by clustering pixels around their centers, the soft attention mask branch learns a spatial embedding space where pixels

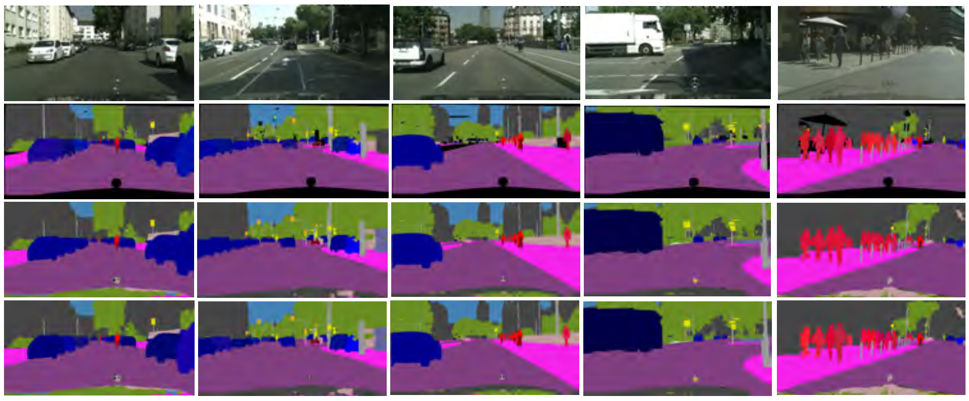


Figure 5.10: Semantic and panoptic segmentation results with the Panoptic Prototype network. From top to bottom: image, panoptic ground truth, semantic segmentation, panoptic segmentation. In the panoptic segmentation the color encodes the class and the instance identifier.

belonging to the same instance are attracted toward the bounding box center. Guided by object detections, these spatial embeddings are used to generate soft attention maps, which focus on *things* class segments derived from the semantic segmentation branch to produce panoptic instance masks. In practice, the semantic segmentation masks and the instance-aware soft attention masks exhibit pixel-wise coherence, allowing a simple element-wise multiplication operation to produce the final panoptic segmentation output.

The AttentionPS network is built on top of the anchor-free FCOS detector [142], which is extended with a novel panoptic segmentation head. The panoptic head predicts both semantic segmentation and instance-specific soft attention masks, which are activated at instance pixel locations. To generate the soft attention masks, the panoptic head learns the pixel offsets to the corresponding instance centers.

A high-level graphical representation of the pipeline is shown in Figure 5.11, while the detailed architecture of the network is illustrated in Figure 5.12.

Backbone. The network utilizes a shared backbone for feature extraction, paired with a Feature Pyramid Network (FPN) [95] for multi-scale feature representation. Specifically, a ResNet50-FPN [64] backbone is employed, where the FPN is designed with a lightweight architecture comprising 128 channels, as opposed to the default 256 channels used by

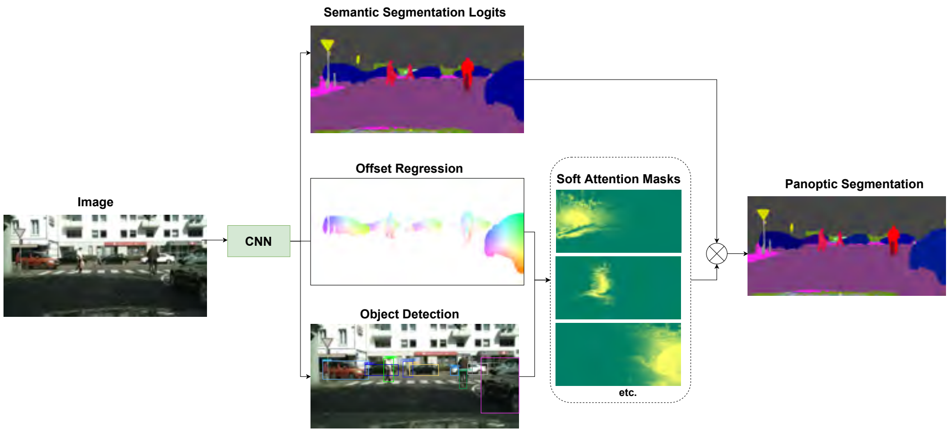


Figure 5.11: The fully convolutional network detects objects and predicts semantic segmentation and pixel offsets to the instance center. Bounding box predictions, along with pixels offsets, are used to generate instance specific soft attention masks. The panoptic output assembles *stuff* masks and *things* masks scaled by the soft attention maps. The final panoptic segmentation is achieved via per-pixel classification.

other panoptic segmentation networks. For obtaining increased accuracy, experiments are also conducted using the VoVNet2-39 backbone [85].

Semantic Segmentation. The network comprises of a lightweight segmentation head, which shares the same feature representation as the object detector. The semantic head processes the five-level FPN feature maps by upsampling them to a resolution of 1/8 and concatenating the results. To capture long-range contextual information, the Pyramid Pooling Module (PSP) [168] is applied on the concatenated features, which consist of 640 channels. The PSP module follows its default design: four parallel average pooling operations are performed, producing feature maps with sizes $[1 \times 1]$, $[2 \times 2]$, $[3 \times 3]$, and $[6 \times 6]$. These are followed by a $[1 \times 1, 640]$ convolution and upsampling back to 1/8 resolution. The outputs of the PSP module and its input are concatenated, and a $[1 \times 1, 128]$ convolution fuses the multi-scale features.

An upsampling stage is subsequently applied, comprising a $[3 \times 3, 128]$ depthwise separable convolution, Instance Normalization, ReLU activation, and $2\times$ bilinear interpolation. To further refine the features, two additional $[3 \times 3, 128]$ depthwise separable convolutions are employed. Finally, a $[1 \times 1, N_{\text{seg}}]$ convolution, followed by a per-pixel softmax operation, generates the class predictions. The semantic seg-

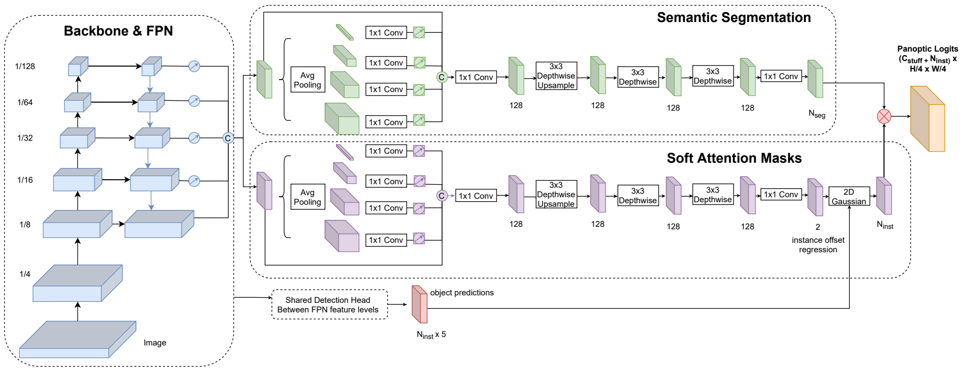


Figure 5.12: The network employs a shared backbone and FPN, which we extend with an object detection and classification head, semantic segmentation and panoptic segmentation head. The latter regresses foreground pixels to their corresponding instance centers. The object predictions along with the predicted instance center are used to generate instance-wise soft attention masks that guide the instance mask prediction within the panoptic head by re-weighting semantic segmentation logits. Panoptic segmentation is solved as a dense classification task.

mentation branch is supervised by minimizing the weighted bootstrapped cross-entropy loss [24], which assigns greater weight to smaller-sized instances.

Panoptic Segmentation. A novel panoptic segmentation head is proposed, which generates pixel-wise coherent instance-aware soft attention masks alongside the semantic segmentation output. By performing a simple element-wise multiplication between the semantic segmentation predictions and the instance-aware attention masks, the final panoptic segmentation output is obtained.

Several works [73, 28] formulate instance segmentation as the task of associating instance pixels $P = \{p_1, p_2, \ldots, p_n\}$ with their corresponding instance centroids $C = \{c_1, c_2, \ldots, c_k\}$, where k is the number of instances in the image. This association is achieved by predicting offset vectors $O = \{o_1, o_2, \ldots, o_n\}$, which represent the displacement between a pixel p_i and its corresponding instance centroid c_k, such that $c_k = p_i + o_i$. One common approach to identify instances involves clustering pixels based on their predicted centers and the corresponding true instance centers. However, the true locations of the instance centers are initially unknown. To address this, Cheng et al. [28] propose a keypoint-based representation, while Neven et al. [109] introduce an advanced loss function that learns these locations based on a seed map.

In AttentionPS, instance mask prediction is formulated as the task of predicting offset vectors $c_k = p_i + o_i$, mapping instance pixels to their respective instance centers. The shared backbone is extended with a soft attention mask decoder, which adopts the same architecture as the semantic segmentation decoder (illustrated in Figure 5.12). The primary modification involves replacing the final convolutional layer with a $[1 \times 1, 2]$ convolution to output pixel offsets along the x- and y-axes.

Unlike prior approaches that apply hard clustering to associate pixels with their nearest centers, this work avoids hard clustering. Since datasets typically include objects of varying sizes, the distance between instance pixels and their centers can span a wide range of scales. For large objects, learning offsets for pixels that are far from the center becomes particularly challenging. To mitigate this issue, the method introduces soft attention masks for each instance. These masks are computed using the true instance center, the predicted pixel offsets, and the object size, thereby relaxing the constraint of hard clustering and improving accuracy across objects of different scales.

To compute the soft attention masks, three components are required: the true instance centers, the pixel offset predictions, and the object size.

First, the true instance centers are derived from the predicted bounding boxes. The object detector outputs a set of bounding boxes $B = \{b_1, b_2, \ldots, b_k\}$, where each bounding box b is represented as:

$$b = \{(x_1, x_2, y_1, y_2), \text{centerness}\}, \quad s \in \{1, N_{\text{things}}\},$$

with (x_1, y_1) as the top-left corner coordinates, (x_2, y_2) as the bottom-right corner coordinates, and s denoting the semantic class of the object. The centerness value, which ranges between 0 and 1, is used during the Non-Maximum Suppression (NMS) step. Using the top-left and bottom-right coordinates, the true instance center is calculated as (x_k, y_k).

Second, the soft attention masks are computed using the pixel offset predictions from the panoptic head. For every pixel location (p_i, p_j) in the image that belongs to an instance, the network predicts an offset to the corresponding instance center. The predicted instance center is thus given by:

$$c_k = (p_i + o_i, p_j + o_j),$$

where o_i and o_j are the predicted offsets along the x- and y-axes, respectively.

Third, the object size is incorporated into the computation of the soft attention masks. The object size is determined from the height and width of the bounding box, $h \times w$, as provided by the object detector.

A soft attention mask A_k is defined for each detected object, using an instance-wise Gaussian kernel to transform the predicted offsets from the panoptic head into probabilities indicating pixel membership to the corresponding instance. Specifically, a 2-dimensional Gaussian function is employed, where the mean is located at the center of the bounding box and the standard deviation is determined by the size of the bounding box. The Gaussian function generates attention maps as follows:

$$A(p_i, p_j) = \exp\left(-\frac{2(p_i + o_i - x_k)^2}{w^2} - \frac{2(p_j + o_j - y_k)^2}{h^2}\right),$$

where $w = 2\sigma_w$ and $h = 2\sigma_h$, with σ representing the standard deviation.

The soft attention masks, modeled through this Gaussian function, indirectly encourage the predicted instance centers to align with the bounding box center while constraining them to reside within an elliptical region surrounding the center. The size of this elliptical region is determined dynamically by the bounding box dimensions and adapts to each object's size. This flexibility alleviates the strict requirement for offset vectors to point precisely at the instance center, a constraint that is particularly challenging for large objects where distant pixel locations are more prone to offset prediction errors.

Although the formulation relaxes the alignment constraint, it ensures that the predicted centers $(p_i + o_i, p_j + o_j)$ remain within the defined elliptical region with high probability at the bounding box center (x_k, y_k). This probability gradually decreases as the predicted center deviates from the true center, decaying towards the object's edges. A visualization of this concept is provided in Figure 5.13.

As illustrated in Figure 5.14, the soft attention masks assign high probabilities to instance pixels while assigning low probabilities elsewhere. To generate the instance masks, these soft attention masks are applied as filters on the semantic segmentation output. By performing an element-wise multiplication between the attention mask and the semantic segmentation, the pixels belonging to the corresponding instance are enhanced by increasing their values, while the logits of background pixels or pixels belonging to other instances are suppressed.

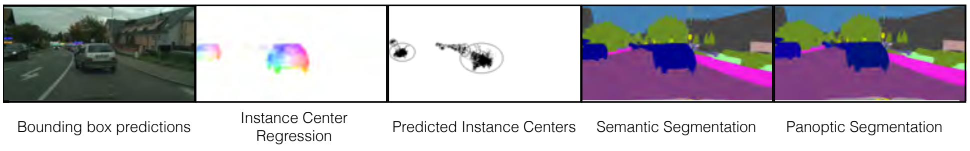


Figure 5.13: The network performs object detection and outputs bounding boxes and classes for each object. The spatial embedding branch regresses instance center offsets from each foreground pixel. In the instance center regression image, the intensity encodes the magnitude of the offset vector and the color indicates the orientation of the vector. By moving each pixel location with the predicted offsets, the predicted instance centers are obtained. The predicted instance centers are forced to lie within a region centered at the bounding box center with its size defined by the object's size.

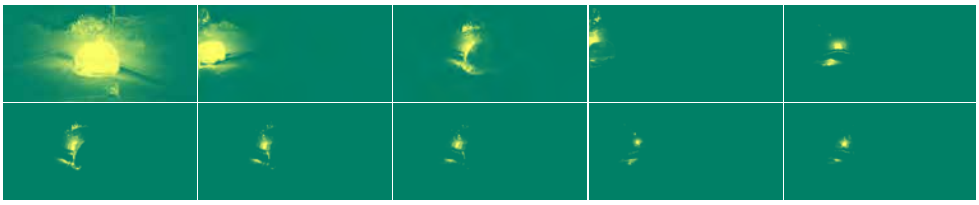


Figure 5.14: Using the predicted center offsets and the object proposals, soft attention masks for each instance are generated (the original image is in Figure 5.13). As the saturation of yellow pixels increases, the probability of belonging to the instance is larger. Green pixels have a low probability of belonging to the instance.

The final panoptic logits L_k for instance C_k are computed as follows:

$$L_k(p_i, p_j) = S(p_i, p_j)[s_k] \cdot A_k(p_i, p_j) \tag{5.5}$$

The semantic class of the box s_k is used for selecting the semantic logits S of a certain class, which are scaled by the attention map A. Panoptic logits will be computed for each detected instance.

Finally, the instance-wise panoptic logits are combined with the *stuff* logits derived from the semantic segmentation branch. A softmax operation is applied over the concatenated logits, and the weighted bootstrapped cross-entropy loss is minimized during training. It is important to note that the supervision of pixel offsets occurs solely through this loss.

Panoptic Segmentation Training. The panoptic output is represented as a tensor of size $(C_{\text{stuff}} + N_{\text{inst}}) \times \frac{H}{4} \times \frac{W}{4}$. The first C_{stuff} channels of the panoptic logits correspond to the semantic segmentation logits for *stuff* classes, while the following N_{inst} channels represent the instance logits. During training, N_{inst} equals the number of ground truth bounding boxes.

To generate the soft attention masks, the ground truth bounding boxes and corresponding object classes are utilized. It is essential to ensure that the order of the instance masks in the panoptic output matches the order used to construct the panoptic ground truth during training.

Panoptic Segmentation Inference. During inference, the predicted bounding boxes are utilized to generate the soft attention masks. First, Non-Maximum Suppression (NMS) is applied to the predictions across all FPN levels, and the top 100 scoring boxes with confidence scores above 0.3 are selected. The panoptic output is then constructed similarly to the training phase, but using the filtered object proposals. A softmax operation is applied over the panoptic logits. The channel corresponding to the maximum value determines the semantic class for *stuff* classes and the instance identifier for instances. The semantic class of an instance is derived from the predicted object class. For evaluation, considering the sensitivity of the Panoptic Quality (PQ) metric to *stuff* predictions [159], *stuff* masks with areas smaller than 1024 pixels for the Cityscapes dataset and 4096 pixels for the COCO dataset are ignored.

Pixel Offsets and Bounding Boxes. During training, as images are resized to multiple scales, pixel offsets are normalized by the image height for both the x- and y-directions. To ensure bounded offset predictions, a *tanh* activation is applied, constraining the values to the range $[-1, 1]$. This restriction implies that a location can regress a maximum offset equal to the height of the image in each direction. Additionally, both the bounding box center and the bounding box size are normalized by the image height.

Experiments. The AttentionPS network is trained end-to-end on a system equipped with four Tesla V100 GPUs using a single optimization step. Inference is performed on a single GPU with a batch size of 1. In all experiments, the backbone is initialized with ImageNet [38] pretrained weights [85]. The model is trained and evaluated on the Cityscapes [34] and COCO [97] datasets. For the Cityscapes dataset, the network is trained with a minibatch size of 4 images for 96k iterations, with the initial learning rate reduced by a factor of 0.1 at 76k and 88k iterations. On the COCO dataset, training is conducted with a batch size of 16 images for 270k iterations, with learning rate reductions at 210k and 250k iterations.

The total loss function is defined as the sum of the object detector losses (bounding box regression loss, object classification loss, and centerness loss), the semantic segmentation loss, and the panoptic segmentation loss. The loss function is optimized using Stochastic Gradient Descent (SGD) with a momentum of 0.9, weight decay of 1×10^{-4}, and a base learning rate of 0.01. A learning rate warm-up is applied for the first 1500 iterations.

During training, image augmentations are employed, including random horizontal flipping and scaling. For the Cityscapes dataset, the shorter side is randomly selected from the interval $[768, 1024]$, while for the COCO dataset, it is chosen from the interval $[640, 800]$.

Ablation Studies. The following section investigates key architectural design choices, the impact of different backbones, various loss functions, and the importance of soft attention masks. All ablation studies are conducted on the Cityscapes dataset, with inference times reported for images at a resolution of 1024×2048. The results are summarized in Table 5.10.

Network Backbone Ablation. As shown in Table 5.11, experiments are conducted with multiple backbones to assess their performance. Using the VoVNet2-39-FPNlite backbone, the network achieves the highest accuracy with a Panoptic Quality (PQ) of 59.7% and an inference time of 92 ms. Here, FPNlite refers to a lightweight version of the Feature Pyramid Network (FPN) with 128 channels, compared to the 256 channels in the original FPN.

Additionally, results using the ResNet50-FPNlite backbone are presented. This configuration achieves a competitive PQ score of 59.3% while offering a reduced inference time of 88 ms.

Network Heads Architecture Ablation. The panoptic segmentation network is built on the FCOS detector [142], sharing the same backbone. Experiments are conducted to compare the performance of using one versus two separate heads for semantic segmentation and instance offset regression, as summarized in Table 5.10.

Using a shared decoder for both semantic segmentation and instance offset regression achieves a Panoptic Quality (PQ) of 58.7%, a

Dec × 1	Dec × 2	Att Mean	Att Scale	Seg Loss	Reg Loss	Th Seg	PQ	mIoU	Time (ms)
	✓			✓		✓	53.1	75.4	92
	✓	✓		✓		✓	57.2	74.9	92
	✓	✓	✓	✓			57.6		91
	✓	✓	✓			✓	58.5	74.4	92
	✓	✓	✓	✓		✓	**59.7**	**76.4**	92
	✓	✓	✓	✓	✓	✓	59.6	76.2	92
✓		✓	✓	✓		✓	58.7	75.6	**87**

Table 5.10: Ablation study on the Cityscapes *validation* set with VoVNet2-39-FPNlite backbone. The following settings are used: **Dec × 1**: one decoder for both offsets regression and semantic segmentation tasks. **Dec × 2**: two decoders for each task. **Att mean**: attention mask is modeled by a Gaussian with standard deviation equal to 1. **Att scale**: use the bounding box size in the Gaussian formulation. **Seg loss**: use semantic segmentation loss. **Reg loss**: use offset center regression loss. **Th seg**: the classes in semantic segmentation are the *stuff* classes and *things* classes. Best results in terms of PQ, mIoU and speed are marked in bold. Time is measured on a NVIDIA Tesla V100 GPU.

mean Intersection over Union (mIoU) of 75.6%, and an inference time of 87 ms. However, semantic segmentation and instance offset regression may require distinct feature representations and contextual information. To address this, the backbone is extended with an additional decoder. This architectural change improves performance, increasing the PQ score by 1% and the mIoU by 0.8%, at the cost of a 5 ms overhead during inference.

In the baseline approach, the semantic segmentation branch classifies pixels into N_{seg} classes, encompassing both *things* and *stuff* classes. Since the attention masks guide the panoptic segmentation process by selecting instance-level pixels from the segmentation masks, it is hypothesized that merging all *things* classes into a single category could suffice. To test this, a simplified semantic segmentation head is implemented with $N_{\text{seg}} = N_{\text{stuff}} + 1$, where the network predicts *stuff* classes while merging *things* classes into one category.

However, results indicate that learning each *things* class separately is crucial. The network achieves a PQ score of 59.7% when using multiple *things* classes, compared to 57.6% with a single *things* category. This demonstrates that attention masks provide only rough instance cues for panoptic segmentation and benefit significantly from additional se-

Backbone	PQ	PQ_{th}	PQ_{st}	SQ	RQ	mIoU	Time (ms)
Full image size: 1024 × 2048							
ResNet50-FPNlite	59.3	52.8	64.1	80.0	72.7	76.0	88
VoVNet2-39-FPNlite	**59.7**	**52.9**	**64.7**	**80.2**	**73.1**	**76.4**	**92**
VoVNet2-57-FPNlite	60.4	53.6	65.4	80.3	73.9	76.8	132
Half image size: 512 × 1024							
VoVNet2-39-FPNlite	48.9	41.5	54.3	76.7	61.2	69.8	42

Table 5.11: Backbone evaluation on the Cityscapes *validation* set. Best results are marked in bold. Time is measured on a NVIDIA Tesla V100 GPU.

mantic class information, such as background-foreground separation and boundary delineation between different *things* segments.

SOFT ATTENTION MASKS ABLATION. Experiments are conducted to analyze the effect of soft attention masks, as summarized in Table 5.10. The network predicts both semantic segmentation and instance center offsets for each foreground pixel. In the baseline *no attention* model, hard pixel assignments are performed by associating each pixel with the closest predicted center, based on the offset values. The panoptic segmentation output is then built using the semantic segmentation logits and bounding box predictions. This baseline configuration achieves a Panoptic Quality (PQ) score of 53.1%.

Next, two variants of soft attention masks are introduced. Both approaches utilize a 2-dimensional Gaussian function applied on the instance center regression branch to compute the probability of a pixel belonging to an instance. A simplified Gaussian function with a fixed standard deviation of 1, where object size is not considered, achieves a PQ score of 57.2%, representing an improvement of over 4% compared to the hard clustering baseline.

However, due to the presence of objects of varying sizes in the dataset, it is essential to account for scale when creating the soft attention masks. By incorporating the object size into the Gaussian function, the constraint that pixel offsets must point exactly to the object center is relaxed. This design accommodates potential errors for distant pixels in large objects by defining a region around the bounding box center based on the object's size. Experimental results confirm the importance of this approach, yielding an additional 2.5% improvement in PQ.

Loss Functions Ablation. The baseline network is trained using object detection losses, as well as panoptic and semantic segmentation losses, with all losses equally weighted in the final loss. A series of ablation studies on the loss functions are summarized in Table 5.10.

First, the impact of the semantic segmentation loss is examined. Training the network without this loss results in a PQ score of 58.5% and an mIoU of 74.4%. Although the network successfully predicts *stuff* classes and instances using only the panoptic segmentation loss, the absence of direct supervision for semantic segmentation reduces the quality of boundary delineation between objects of different classes. By including the semantic segmentation loss, the network produces more accurate panoptic and semantic predictions, achieving 59.7% PQ and 76.4% mIoU.

Additionally, an instance offset regression loss is introduced on the spatial embedding branch using the L1 loss to pull pixels toward the bounding box center. However, no significant benefit is observed from this additional supervision, which may be attributed to the equal weighting of all losses.

Performance on Cityscapes. Table 5.12 presents a comparison of panoptic and semantic segmentation results on the Cityscapes validation set with other bottom-up, box-based two-stage, and one-stage methods. All methods are trained solely on the FINE annotations, and no test-time augmentation is applied. For a fair comparison, results are reported using a ResNet-50 backbone. Compared to two-stage approaches, the proposed network achieves results comparable to UPSNet [159] and Panoptic-FPN [75], while Seamless Panoptic [126] and EfficientPS [106] achieve higher accuracy. However, the AttentionPS method runs almost twice as fast as most two-stage networks. When compared to bottom-up approaches, the network ranks second behind Panoptic DeepLab [28]. By adopting a more powerful backbone, VoVNet2-39-FPNlite, the AttentionPS method matches the accuracy of Panoptic DeepLab.

Among single-stage methods, this network achieves the best results in terms of accuracy, outperforming FPSNet [37] by 4.2% PQ and DenseBox [67] by 0.5% PQ. In terms of inference speed, the method is slightly outperformed by Prototype Panoptic [117]; however, it achieves superior accuracy, with a PQ score difference of 2%.

Visual examples of the panoptic and instance segmentation results are provided in Figure 5.15.

In Table 5.12, the end-to-end inference time of the networks is reported. The measured time includes the forward pass of the network on

Method	Backbone	PQ	PQ_{th}	PQ_{st}	mIoU	Time (ms)
Bottom-up						
DeeperLab [162]	Xception-71	56.5	-	-	-	-
SSAP [48]	ResNet50-FPN	58.4	50.6	-	-	-
AdaptIS [137]	ResNet-50	59.0	55.8	61.3	-	-
Panoptic DeepLab [28]	ResNet-50	**59.7**	-	-	-	**117**
Box-based Two-Stage						
MTN Panoptic [116]	ResNet50-FPN	57.3	53.9	59.7	-	150
Panoptic-FPN [75]	ResNet50-FPN	58.1	52.0	62.5	75.7	-
UPSNet [159]	ResNet50-FPN	59.3	54.6	62.7	75.2	**140**
Seamless Panoptic [126]	ResNet50-FPN	60.3	56.1	63.3	77.5	150
EfficientPS [106]	ResNet50-FPN	**63.9**	**60.7**	**66.2**	**79.3**	166
Box-based One-Stage						
FPSNet [37]	ResNet50-FPN	55.1	48.3	60.1	-	98
Prototype Panoptic [117]	VoVNet2-39-FPNlite	57.3	50.4	62.4	-	**82**
DenseBox [67]	ResNet50-FPN	58.8	52.1	63.7	77.0	99
AttentionPS (ours)	ResNet50-FPNlite	59.3	52.8	64.1	76.0	88
AttentionPS (ours)	VoVNet2-39-FPNlite	**59.7**	**52.8**	**64.7**	**76.4**	92

Table 5.12: Comparative study with state-of-the-art two-stage and one-stage panoptic segmentation networks on the Cityscapes *validation* dataset. Inference time is measured on one Tesla V100 GPU with batch size of 1. Best results are marked in bold.

1024×2048 resolution images, as well as the Non-Maximum Suppression (NMS) step. All execution times are recorded on a Tesla V100 GPU with a batch size of 1.

The AttentionPS network, when using the ResNet50-FPNlite backbone, achieves the second fastest inference time of 88 ms. With the VoVNet2-39-FPNlite backbone, the execution time is 92 ms. While accurate instance segmentation is critical, achieving a favorable trade-off between accuracy and inference speed is equally important for practical applications. This approach strikes an effective balance between accuracy and speed, as it maintains a minimal accuracy drop compared to other methods while significantly improving inference speed.

2D Panoptic Perception. The AttentionPS panoptic image segmentation network can be seamlessly integrated into the 3D perception system of a self-driving car. In a prior work [146, 8], the 2D semantic perception solution served as a key component of the 3D perception pipeline, enabling the construction of a low-level representation of the environment by fusing semantic, instance, and geometric information. The primary objective of the 2D semantic perception system is to detect road

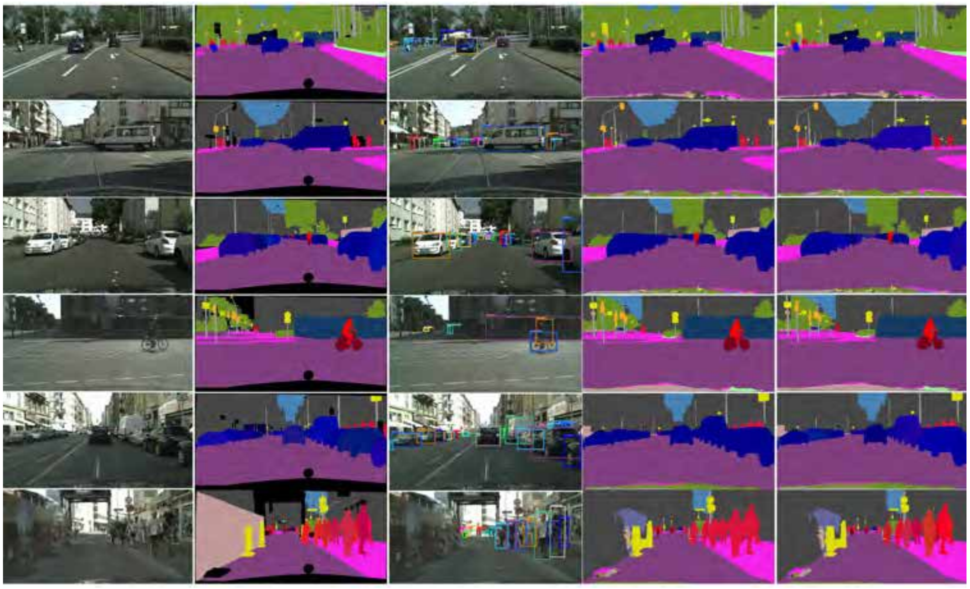


Figure 5.15: Semantic and panoptic segmentation results on the Cityscapes dataset for AttentionPS network. From left to right: image, panoptic segmentation ground truth, object detection, semantic segmentation and panoptic segmentation. In the panoptic segmentation the color encodes the semantic class and the instance identifier. The AttentionPS network can accurately segment objects of various sizes and can handle difficult scenarios with occlusions. Best viewed in color and zoom.

infrastructure along with both static and dynamic road users. This can be effectively achieved using the proposed panoptic image segmentation network, which delivers pixel-level and instance-level classifications. A low-level sensor fusion module can be used to associate the panoptic information from images with 3D point cloud data obtained from LiDAR. The resulting augmented 3D point cloud can then be further processed for the detection and classification of 3D objects. Additionally, the panoptic segmentation network can be extended to process multi-view images from a multi-camera system, providing full 360° coverage around the vehicle.

The real-time performance of the AttentionPS panoptic image segmentation network is enabled by advancements in GPU hardware and neural network optimization, which facilitate high inference speeds while maintaining low computational costs, even on low-power GPUs. To achieve this, the TensorRT library [6] is employed for high-performance deep learning inference. TensorRT performs network optimizations, sig-

Backbon	Resolution	PQ	Time (ms)
ResNet50-FPNlite	1024 × 2048	59.3	70
VoVNet2-39-FPNlite	1024 × 2048	59.7	68
VoVNet2-39-FPNlite	512 × 1024	48.9	31

Table 5.13: Results on the Cityscapes *validation* set. Evaluation is done for the optimized network with TensorRT on the NVIDIA GTX 1080 GPU.

nificantly reducing inference time, and generates a high-performance runtime engine that can be seamlessly integrated into C++/CUDA projects and frameworks for automated driving, such as ADTF [1].

The inference time of the optimized panoptic segmentation network is measured on an NVIDIA GTX 1080 GPU, a less powerful device suitable for installation in self-driving cars, as demonstrated in previous work [146]. The results are presented in Table 5.13. With network optimization, real-time performance is achieved on the GTX 1080 GPU, processing images at 512×1024 resolution at over 30 frames per second, with no degradation in panoptic quality compared to the unoptimized network. The performance of the AttentionPS network can be further improved by utilizing more powerful GPUs.

Chapter 6

Video Panoptic Segmentation

Video panoptic segmentation extends the task of static image-based panoptic segmentation into the temporal domain, addressing the challenge of consistent pixel-wise classification across consecutive video frames. In addition to assigning semantic labels to *stuff* regions and instance IDs to *things* objects at each frame, video panoptic segmentation must ensure temporal coherence—tracking object instances as they move and evolve over time. In this chapter we review the methods and present in detail two different approaches.

6.1 Overview

Video panoptic segmentation is the task of predicting panoptic segmentation with consistent instance identifiers in a video. The task can be solved with the sub-tasks of panoptic segmentation and instance tracking. Kim *et al.* has recently introduced the task in [74] along with the baseline VPSNet network. VPSNet is built on top of the proposal-based two-stage panoptic segmentation network UPSNet [159]. In order to improve the current prediction, a pixel-level fusion module gathers features from the previous and next five frames, which are further aligned with optical flow and fused with spatio-temporal attention. For the tracking functionality, a MaskTrack head [161] is employed. The network incurs a high-computational cost and is not suitable for real-time applications since future frames are also considered for computation. SiamTrack [155] improves VPSNet by designing novel learning objectives that learn segment and pixel-wise temporal associations in a contrastive learning framework.

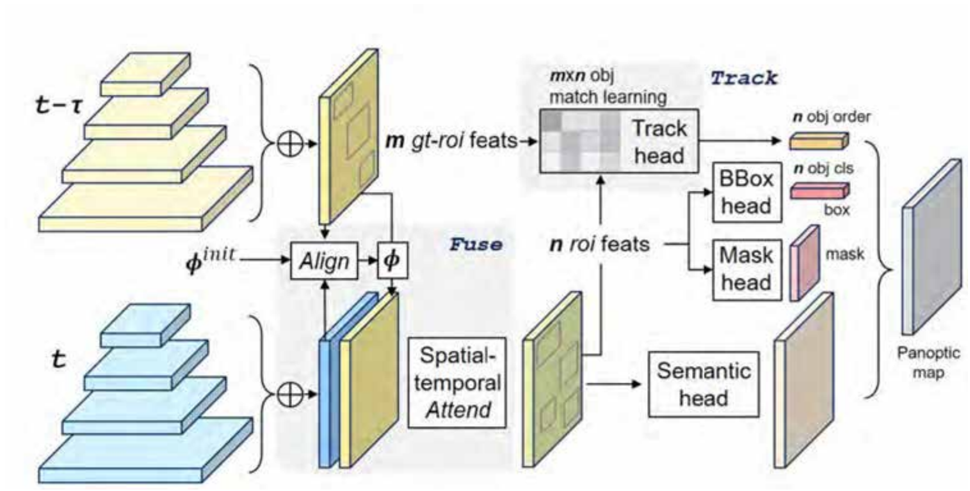


Figure 6.1: VPSNet [74] architecture.

6.2 VPSNet

Method. VPSNet [74], a method proposed for Video Panoptic Segmentation (VPS), extends panoptic segmentation into the video domain, addressing both spatial and temporal segmentation challenges. Unlike image panoptic segmentation, VPS requires assigning consistent semantic and instance labels to all pixels across consecutive video frames while maintaining temporal coherence of object instances. VPSNet builds on UPSNet [159], a state-of-the-art panoptic segmentation model, and incorporates temporal context through two novel modules: Fuse and Track. The architecture of VPSNet is displayed in Figure 6.1.

The Fuse module enhances pixel-level features by leveraging temporal information from neighboring video frames. Specifically, features from the target frame and a reference frame are extracted and aligned using a flow-based feature warping mechanism. The aligned features are then fused using a spatial-temporal attention mechanism, which selectively integrates useful information across frames. This fused feature representation improves downstream tasks like semantic segmentation and instance mask prediction.

The Track module focuses on object-level instance association across frames. It introduces a tracking branch inspired by MaskTrack [161], which computes similarity between region-of-interest (RoI) features of objects in the target and reference frames. To enhance discriminative power, the tracking branch operates on temporally fused RoI features, improving the ability to match and track object instances over time.

VPSNet integrates both modules into a unified framework that simultaneously performs object detection, semantic segmentation, mask prediction, and instance tracking. The network processes videos sequentially, generating a coherent sequence of panoptic segmentation outputs. At inference, the model applies class-agnostic Non-Maximum Suppression (NMS) to filter redundant detections and associates instance IDs across frames using the learned tracking affinity matrix.

Experiments. The experimental evaluation of VPSNet on Cityscapes demonstrates the effectiveness of the method in both image-level and video-level panoptic segmentation tasks. The evaluation focuses on two main aspects: per-frame panoptic segmentation and temporal consistency across frames.

VPSNet variants on Cityscapes-VPS	**Temporal window size**				**VPQ**
	k = 1	**k = 5**	**k = 10**	**k = 15**	
Track	61.6 / 54.9 / 66.5	54.3 / 39.9 / 64.9	50.7 / 34.6 / 62.4	47.8 / 30.7 / 60.4	53.6 / 40.0 / 63.6
FuseTrack (VPSNet)	**62.7** / 56.9 / 66.8	**56.9** / 44.5 / 65.9	**53.3** / 40.4 / 62.7	**51.4** / 36.9 / 61.9	**56.1** / 44.7 / 64.3

Table 6.1: Video panoptic segmentation results on Cityscapes-VPS validation set with VPSNet variants. Each cell contains VPQ / VPQ$^{\text{Th}}$ / VPQ$^{\text{St}}$ scores.

For image-level panoptic quality (PQ), VPSNet with the Fuse module is tested on the Cityscapes validation set, as seen in Table 6.2. The Fuse module leverages spatial-temporal features to enhance segmentation accuracy. The results show that VPSNet with Fuse outperforms existing state-of-the-art methods, such as UPSNet [159], by improving the PQ score by +1.0%. Specifically, when pretraining on the VIPER dataset [74], the model achieves further improvements, obtaining a PQ score of 62.2%. This highlights the complementary benefits of incorporating VIPER pretraining for image-level panoptic segmentation.

For video-level evaluation, the proposed Cityscapes-VPS dataset is used to assess temporal consistency through the Video Panoptic Quality (VPQ) metric. Results can be seen in Table 6.1. The full VPSNet model, integrating both the Fuse module and Track module, achieves the best VPQ performance. The combination of pixel-level feature fusion and object-level tracking improves VPQ by +1.1% over the Track-only variant.

Overall, VPSNet is the first method proposed in the literature

Method	**Backbone**	**PQ**	**PQ$^{\text{Th}}$**	**PQ$^{\text{St}}$**
UPSNet	ResNet-50	59.3	54.6	62.7
UPSNet+CO	ResNet-50	60.5	57.0	63.0
VPSNet-Base+CO	ResNet-50	60.6	57.0	63.2
VPSNet-Fuse+CO	ResNet-50	61.6	57.7	64.4

Table 6.2: Performance comparison of UPSNet and VPSNet variants on Cityscapes. Results include PQ, PQ$^{\text{Th}}$, and PQ$^{\text{St}}$ scores. "+CO" means models is pretrained on the COCO [97] dataset.

for this tasks and it offers a robust solution by effectively combining temporal feature fusion and instance-level tracking mechanisms.

6.3 VPS-Transformer

Method. VPS-Transformer [121] presents a novel approach to the task of video panoptic segmentation, which simultaneously predicts pixel-level semantic and instance segmentation while generating clip-level instance tracks. Video sequences contain abundant information, including temporal cues and motion patterns, that can be leveraged to achieve more accurate and consistent panoptic segmentation. While modeling temporal correlations between frames offers clear benefits, it introduces new challenges when processing video data. Temporal consistency across consecutive frames is generally assumed, but it can be disrupted by occlusions and the appearance of new objects in rapidly changing scenes. Therefore, temporal information must be carefully incorporated to avoid the inclusion of outdated or irrelevant information in the predictions [110, 70]. Moreover, extending single-frame segmentation methods to handle multiple video frames often leads to a significant increase in computational cost [74], resulting in overhead during both training and inference. Since efficiency is crucial for practical applications, maintaining a balance between performance and speed is essential.

For video panoptic segmentation, this work focuses on the following research directions:

1. Developing methods to model spatio-temporal relationships between frames for improved segmentation performance.

2. Enabling effective instance tracking within a multi-task video panoptic segmentation network.

3. Achieving an optimal trade-off between inference speed and segmentation accuracy.

A novel video panoptic segmentation method, named **VPS-Transformer**, is introduced with a focus on both efficiency and accuracy. The network employs a hybrid architecture that combines a convolutional backbone for single-frame panoptic segmentation with a novel video module based on an instantiation of the Transformer block [147]. Equipped with attention mechanisms, the Transformer module models spatio-temporal relationships between backbone output features from the current and past frames, enabling more accurate and consistent panoptic predictions. To ensure computational efficiency, a lightweight variant of the Transformer block is developed, achieving faster performance compared to the original implementation [147].

Several strategies are presented to factorize the attention operation of the Transformer across spatial and temporal dimensions, and their accuracy and efficiency are compared through extensive ablation studies. The enhanced feature representations provided by the Transformer module are processed by three convolutional decoders that recover the spatial resolution of the input image. Multiple prediction heads are employed to perform tasks such as semantic segmentation, instance center prediction, instance offset regression, and optical flow estimation.

To ensure consistent instance identifiers for the same objects across frames, a tracking module is implemented. This module uses mask propagation with optical flow and associates instance IDs by matching warped and predicted instance masks based on class labels and intersection over union (IoU). The proposed video panoptic segmentation network achieves a strong balance between speed and accuracy, with each newly introduced module carefully designed to maintain system efficiency.

The network can be trained in a weakly supervised regime using sparsely annotated datasets, as it does not require labels for previous frames. In comparison to existing methods, such as VPSNet [74] and ViP-DeepLab [127], which focus on improving panoptic segmentation quality at a high computational cost, VPS-Transformer achieves greater efficiency. VPSNet [74] employs a temporal fusion module based on optical flow, aggregating features from five neighboring past and future

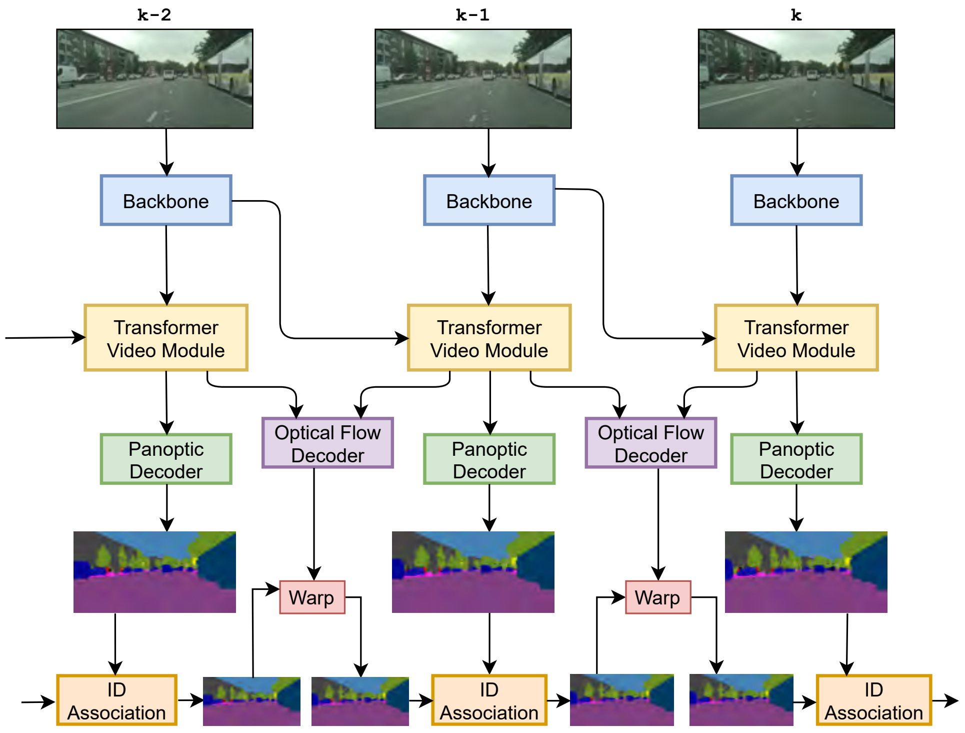


Figure 6.2: High level overview of the VPS-Transformer network, which processes video frames and outputs panoptic segmentation and consistent instance identifiers. The network has a Transformer based video module to model temporal and spatial relations among pixels from the current frame features and past frames features. Instance tracking is performed by warping the previous panoptic prediction with optical flow and associating the instance IDs with the current instance segmentation.

frames. In contrast, VPS-Transformer operates in an online fashion, processing only the current frame, while still delivering improved accuracy.

ViP-DeepLab [127] models video panoptic segmentation as a concatenation of image panoptic segmentation tasks. Compared to this approach, VPS-Transformer explicitly encodes spatio-temporal correlations through its Transformer module, achieving a performance boost while maintaining a lightweight design.

Baseline. The solution is built upon the bottom-up image panoptic segmentation network Panoptic-DeepLab [28]. The original implementation is modified by adopting a backbone output stride of 32×

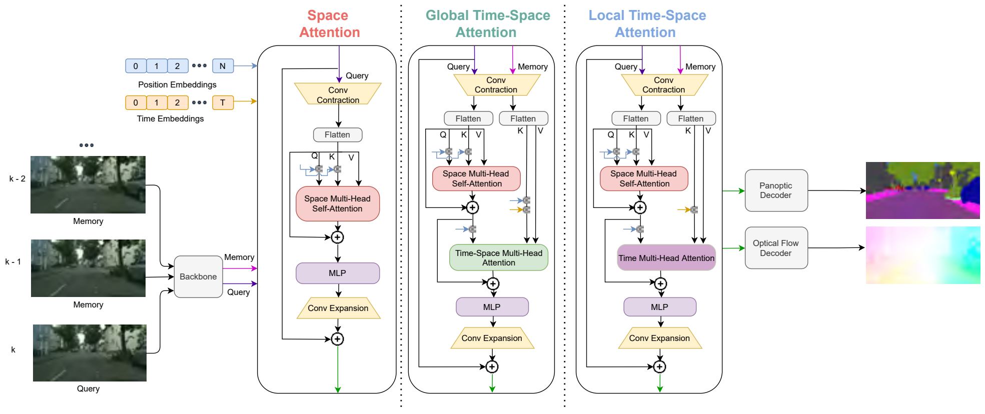


Figure 6.3: A Transformer video module is introduced between the backbone and the decoders for more accurate prediction. There are three variants of the module with various attention mechanisms: space attention, global time-space attention and local time-space attention.

and introducing an additional upsampling stage in the decoders. The network architecture is illustrated in Figure 6.2.

Lightweight Transformer Video Module. The VPS-Transformer network processes video frames sequentially, incorporating a video module between the backbone and the decoder to aggregate features from past frames and enhance the current feature representation. The past frames are referred to as *memory* frames, while the current frame is termed the *query* frame. The design of the video module is inspired by the original Transformer architecture [147], incorporating attention and self-attention mechanisms, along with a multi-layer perceptron (MLP).

The computational complexity of the self-attention block scales quadratically with the size of the input, as previously observed in [141, 138], particularly when processing high-dimensional data. To address this and reduce the complexity of the Transformer block, the Transformer is wrapped between two pointwise convolutions. The first convolution reduces the number of channels, and the second recovers the original dimensionality, as illustrated in Figure 6.3. Specifically, a 1×1 convolution is applied to the backbone output features, reducing the channel count from $C = 2048$ to $d = 1024$, and subsequently restoring it.

The Transformer processes sequences of input tokens of size $B \times N \times d$. To achieve this, the input query features $\mathbf{F} \in \mathbb{R}^{B \times H \times W \times d}$ are reshaped into a sequence of flattened tokens $\mathbf{f} \in \mathbb{R}^{B \times HW \times d}$, where B is the batch size, H and W are the height and width of the feature maps, and d is the embedding dimension. For the global time-space attention block, the memory features from the previous T frames, $\mathbf{M} \in \mathbb{R}^{B \times T \times H \times W \times d}$, are similarly reshaped into a sequence of tokens $\mathbf{m} \in \mathbb{R}^{B \times THW \times d}$. For the local time-space attention design, the memory tokens are reshaped to $\mathbb{R}^{BT \times HW \times d}$.

The spatial Transformer processes only the query frame and consists of a Multi-Head Self-Attention layer, Layer Normalization [13], and a one-hidden-layer MLP. For the Time-Space Attention design, an additional temporal Multi-Head Attention layer is incorporated, operating along the temporal dimension. Residual connections and Layer Normalization are applied after each attention module [13]. All attention blocks use the same number of heads, with experiments demonstrating the best performance using a single-head configuration.

The attention layer takes as input a query and a tuple of $(key, values)$, returning a weighted sum of the values based on the similarity between the query and key. In the self-attention module, the query, key, and values are derived from the same input $\mathbf{f}$ using linear projections with learnable weights $W^Q \in \mathbb{R}^{d \times d}$, $W^K \in \mathbb{R}^{d \times d}$, and $W^V \in \mathbb{R}^{d \times d}$. Since the Transformer architecture does not inherently include positional information, which is crucial for dense prediction, learned position embeddings are injected into both the keys and queries. For the time-space attention, learned temporal embeddings are also added.

Given the query, keys, and values packed as $\mathbf{Q} \in \mathbb{R}^{HW \times d}$, $\mathbf{K} \in \mathbb{R}^{HW \times d}$, and $\mathbf{V} \in \mathbb{R}^{HW \times d}$, the attention operation is formally defined as:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}}\right)\mathbf{V}. \tag{6.1}$$

The Transformer output sequence $\mathbf{o} \in \mathbb{R}^{B \times HW \times d}$ is obtained through a Multi-Layer Perceptron (MLP) with two fully connected layers and a GeLU activation [65] between them. The inner layer dimension of the MLP is set to d. Finally, the output is reshaped back to $\mathbf{m} \in \mathbb{R}^{B \times H \times W \times d}$, and the dimensionality is expanded using a pointwise convolution. A refined feature representation is achieved by applying a residual connection with the input, followed by a ReLU non-linearity.

Attention Variants. Several attention schemes are introduced for the Transformer video module, as illustrated in Figure 6.3. An example demonstrating how each scheme attends to different temporal and

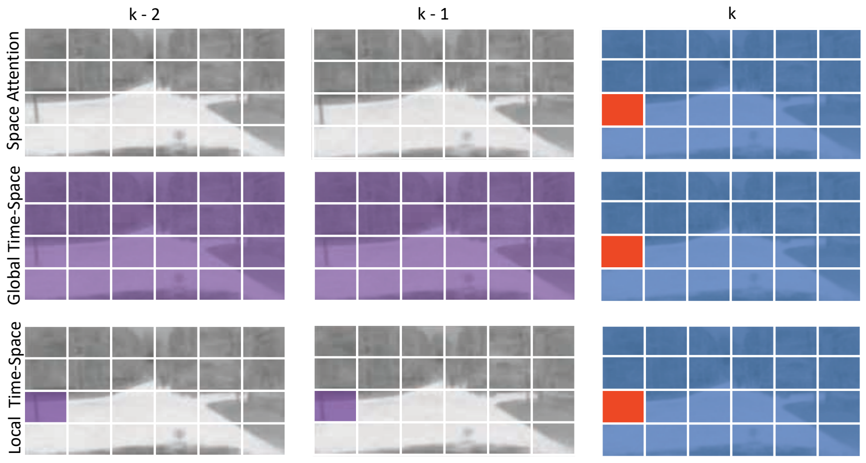


Figure 6.4: The three attention schemes studied in this work with three consecutive frames. The query token, shown in orange in frame k, attends to other tokens in the space and time dimension. When the space-time attention is factorized, blue is used for space and violet for time.

spatial tokens is provided in Figure 6.4.

Space Attention. In this configuration, the panoptic network processes only the query frame. The Transformer module employs spatial self-attention to model interactions between all positions within the query frame features. For each query location, this attention scheme performs $(H \cdot W)$ comparisons.

Global Time-Space Attention. This configuration utilizes features from past frames stored in memory. The attention operation is factorized across the query space and the memory time-space dimensions. First, a spatial Multi-Head Self-Attention is performed over the query frame. Following this, a Time-Space Multi-Head Attention module extracts global temporal correspondences by attending to every position in the query and the memory features. In this case, the query corresponds to the query frame, while the key and values are derived from the memory. For a given query spatial position, the Time-Space Multi-Head Attention requires $(H \cdot W \cdot T)$ comparisons.

Local Time-Space Attention. This module also models spatio-temporal attention but in a more computationally efficient manner compared to the global variant. After performing spatial Multi-Head Self-Attention, the Time Multi-Head Attention operates locally on the temporal dimension by attending only to tokens located at the same spatial position across frames. As a result, the Time Multi-Head Attention requires $(H \cdot W + T)$ comparisons for each query token.

Instance Tracking. Instance tracking is performed by matching the predicted instance masks with the warped instance masks from the previous frame. The warping process is achieved using optical flow. An optical flow decoder is implemented on top of the shared backbone. The first layer of the optical flow decoder is a correlation layer [40], which computes the correlation between features of the previous and current frames. The correlation at each location is calculated at the patch level within a neighborhood of size $d = 9$ around the location. The output of the correlation layer is a cost volume of size $H \times W \times d^2$, storing the matching costs. The remainder of the decoder follows a design similar to that of the instance decoder.

For unsupervised training of the optical flow, a photometric loss is employed to measure the photometric difference between the warped image and the actual image.

The instance ID association algorithm follows the approach described in [127]. Given the panoptic segmentation at frame t and the warped panoptic segmentation from frame $t-1$ to frame t, the intersection over union (IoU) is computed between each instance mask in the current panoptic segmentation and the warped panoptic segmentation, and vice versa. For each instance, the corresponding instance that shares the same semantic class and has the maximum IoU is stored. Additionally, the IoU must exceed a predefined threshold, set to 0.3. Two instances are considered matched if they point to each other, meaning they both have the maximum IoU. In such cases, the instance identifier from the warped panoptic segmentation is propagated to the matched instance in the current frame. Instances that are not matched are assigned a new instance identifier.

Experiments. The network is trained and evaluated on the Cityscapes-VPS dataset [74]. The training process is conducted in three stages.

In the first stage, the panoptic segmentation network is trained for image panoptic segmentation. The backbone is initialized with ImageNet pre-trained weights [38]. For Panoptic-DeepLab, the network is trained with a batch size of 8, following the training settings described in [28].

In the second stage, the backbone and the semantic and instance decoders are frozen, while the optical flow decoder and the Transformer module are trained from scratch. The linear layers of the Transformer module are initialized using Xavier initialization [55]. During this stage, the network is trained for a few epochs with a minibatch of 8 images to obtain a rough initialization. The Adam optimizer is employed with a polynomial learning rate decay, starting with a base learning rate of 1×10^{-3}. Image augmentation techniques are applied, including random horizontal flipping and random scaling with a factor in the range $[0.5, 2.0]$, relative to the original resolution of 1024×2048.

In the third stage, the CNN backbone and the Transformer module are fine-tuned, while batch normalization layers are fixed. The learning rate is set to 1×10^{-4}, and training continues until convergence. The network typically converges within 30k iterations.

Experimental results for the proposed VPS-Transformer network are provided on two datasets with pixel-level annotations: Cityscapes [34] and Cityscapes-VPS [74]. The inference time of the network is reported by measuring the forward pass and post-processing steps on an NVIDIA Tesla V100 GPU with a batch size of one.

Decoder	OS	PQ	mIoU	Time (ms)
Dual + depthwise ASPP	16	60.5	79.6	111
Single + depthwise ASPP	32	59.8	79.0	82
Single + ASPP	32	60.0	79.0	89
Dual + depthwise ASPP	32	**60.7**	**79.3**	**86**

Table 6.3: Ablation study of the baseline panoptic image segmentation Panoptic-DeepLab [28] network on the Cityscapes *validation* set. The following settings are varied: the number of decoders for instance and semantic segmentation, the type of convolutions in the ASPP context module and the output stride of the backbone. Time is measured on a NVIDIA Tesla V100 GPU.

Ablation Studies. Ablation studies are conducted to evaluate the improved version of the baseline network, attention variants, and Transformer module configurations.

Baseline Network. The panoptic image segmentation network Panoptic-DeepLab [28] serves as the baseline for the VPS-

Transformer video panoptic segmentation network. Since additional modules, such as the optical flow decoder and the Transformer video module, are incorporated into the network, the overall inference time increases. To maintain the efficiency of the video network, the inference time of the baseline image-level network is first reduced. Several experiments are performed on the Cityscapes dataset, as summarized in Table 6.3.

Throughout the study, the ResNet-50 backbone [64] is used. The original Panoptic-DeepLab architecture features two ASPP modules with depthwise convolutions for context aggregation, dual decoders for instance and semantic segmentation, and a backbone output stride of 16. This configuration achieves a Panoptic Quality (PQ) score of 59.7% with a forward pass time of 117 ms.

To reduce inference time, the backbone output stride is set to 32, and an additional upsampling stage is introduced in the decoder. This modification reduces the forward pass time to 86 ms and improves the PQ score by 1%. In another experiment, the instance segmentation decoder is removed, and the instance center head and offset regression head are placed on top of the semantic segmentation decoder, resulting in a PQ score of 59.8%.

Replacing the depthwise convolutions in the ASPP decoder with standard convolutions further increases the PQ score by 0.2%, at the cost of an additional 3 ms in inference time. The final configuration chosen as the baseline employs dual decoders, depthwise convolutions in the ASPP module, and a backbone output stride of 32. This configuration achieves the optimal trade-off between accuracy and efficiency.

Attention Variants. The proposed models with variants of the Transformer Video Module are compared on the Cityscapes-VPS dataset in terms of accuracy and efficiency, as summarized in Table 6.4. All models utilize a ResNet-50 backbone, and the runtime is measured both before and after the tracking step. For tracking, the runtime includes the optical flow decoder, warping process, and the instance ID association algorithms.

The baseline network, which excludes the Transformer Video Module, performs single-frame panoptic segmentation and achieves 63.0% PQ and 52.0% VPQ. Since video panoptic quality (VPQ) requires maintaining consistent instance IDs across video frames, a significant accuracy drop is observed when evaluated over larger temporal windows. By propagating instance IDs across frames, the tracking mod-

Models	$\text{VPQ}_k/\text{VPQ}_k^{Th}/\text{VPQ}_k^{St}$ for temporal window size k				VPQ	Time (ms)
	k = 1	k = 5	k = 10	k = 15		
Baseline (B) S = 0	63.0 / 52.1 / 70.9	51.1 / 27.3 / 68.5	48.0 / 21.2 / 67.5	45.9 / 17.4 / 66.7	**52.0 / 29.5 / 68.4**	86
B + Tracking	63.0 / 52.1 / 70.9	55.4 / 37.4 / 68.5	51.1 / 30.9 / 67.5	49.9 / 27.0 / 66.7	**55.1 / 36.8 / 68.4**	100
Local Time-Space Attention						
B + Transformer Video Module S = 1	64.8 / 54.9 / 72.0	52.9 / 30.4 / 69.2	49.6 / 24.0 / 68.1	47.5 / 20.4 / 67.1	53.7 / 32.5 / 69.1	97
B + Transformer Video Module S = 1 + Tracking	64.8 / 54.9 / 72.0	55.4 / 36.5 / 69.2	51.8 / 29.4 / 68.1	49.8 / 26.0 / 67.1	55.5 / 36.7 / 69.1	111
B + Transformer Video Module S = 2	64.7 / 54.7 / 72.0	53.1 / 30.8 / 69.3	49.8 / 24.5 / 68.2	47.7 / 20.8 / 67.2	53.8 / 32.7 / 69.1	98
B + Transformer Video Module S = 2 + Tracking	64.7 / 54.7 / 72.0	56.2 / 38.2 / 69.3	52.8 / 31.5 / 68.2	50.6 / 27.8 / 67.2	56.0 / 38.0 / 69.1	112
B + Transformer Video Module S = 3	64.7 / 54.7 / 71.8	53.0 / 30.9 / 69.1	49.7 / 24.4 / 68.0	47.6 / 20.5 / 67.3	**53.8 / 32.6 / 69.1**	99
B + Transformer Video Module S = 3 + Tracking	64.7 / 54.7 / 71.8	57.4 / 41.1 / 69.1	54.2 / 35.0 / 68.0	52.2 / 31.3 / 67.3	**57.1 / 40.5 / 69.1**	113
B + Transformer Video Module S = 4	64.6 / 54.6 / 71.9	53.0 / 30.8 / 69.2	49.9 / 24.7 / 68.2	47.7 / 21.0 / 67.0	53.8 / 32.7 / 69.0	100
B + Transformer Video Module S = 4 + Tracking	64.6 / 54.6 / 71.9	57.4 / 41.1 / 69.2	54.2 / 35.0 / 68.2	52.0 / 31.2 / 67.0	57.0 / 40.5 / 69.0	114
Global Time-Space Attention						
B + Transformer Video Module S = 1	64.8 / 54.7 / 72.1	53.3 / 31.0 / 69.4	49.9 / 24.6 / 68.3	47.8 / 20.9 / 67.3	**54.0 / 32.8 / 69.3**	98
B + Transformer Video Module S = 1 + Tracking	64.8 / 54.7 / 72.1	57.6 / 41.4 / 69.4	54.4 / 35.2 / 68.3	52.2 / 31.5 / 67.3	**57.3 / 40.7 / 69.3**	112
B + Transformer Video Module S = 2	64.7 / 54.8 / 72.0	53.2 / 31.0 / 69.3	49.9 / 24.7 / 68.2	48.0 / 21.2 / 67.3	53.9 / 33.0 / 69.3	101
B + Transformer Video Module S = 2 + Tracking	64.7 / 54.8 / 72.0	57.6 / 41.3 / 69.3	54.3 / 35.2 / 68.2	52.2 / 31.3 / 67.3	57.2 / 40.6 / 69.3	115
B + Transformer Video Module S = 3	64.7 / 54.8 / 71.9	53.1 / 30.9 / 69.3	50.0 / 24.7 / 68.4	47.9 / 21.2 / 67.4	54.0 / 32.9 / 69.2	105
B + Transformer Video Module S = 3 + Tracking	64.7 / 54.8 / 71.9	56.0 / 37.7 / 69.3	52.6 / 31.0 / 68.4	50.5 / 27.2 / 67.4	56.0 / 37.7 / 69.2	119
Space Attention						
B + Transformer Module S = 0	64.5 / 54.3 / 71.9	52.8 / 30.0 / 69.3	49.5 / 23.9 / 68.2	47.4 / 20.2 / 67.2	**53.6 / 32.1 / 69.2**	94
B + Transformer Module S = 0 + Tracking	64.5 / 54.3 / 71.9	57.2 / 40.5 / 69.3	54.0 / 34.2 / 68.2	51.7 / 30.3 / 67.3	**56.8 / 39.8 / 69.2**	108

Table 6.4: Video panoptic segmentation results on the Cityscapes-VPS dataset with various Transformer Video Module variants. Each cell shows $\text{VPQ}_k/\text{VPQ}_k^{Th}/\text{VPQ}_k^{St}$. VPQ is averaged over window size $k = \{1, 5, 10, 15\}$. VPQ_1 is equal to PQ. The number of input frames to the Transformer Video Module is varied from $S = 0$ to $S = 4$. With $S = 0$ the network processes only the current frame. Time is measured on a NVIDIA Tesla V100 GPU.

ule addresses this limitation and improves VPQ^{Th} by more than 3% for all temporal windows $k > 1$.

The Space Attention model, which incorporates the Transformer module with spatial self-attention, increases PQ by 1.5%, VPQ by 1.6%, and VPQ after tracking by 1.7%. This demonstrates the effectiveness of the Transformer module in refining the current frame features solely through spatial self-attention, without leveraging information from past frames. The computational overhead of this model is moderate, adding 12 ms to the baseline runtime when processing high-resolution images (1024×2048 pixels).

The Global Time-Space Attention model extends attention to spatio-temporal correlations, capturing interactions between every pair of pixels in the current frame and past frame features. This model achieves the highest performance among all variants, improving VPQ by 2.0% and VPQ after tracking by 2.6% compared to the baseline. The results confirm that incorporating past information enhances current predictions and improves instance ID consistency across frames. While the model introduces additional computational cost due to an extra Multi-Head Attention block and increased input token size, the best-performing variant uses only one past frame ($S = 1$) and adds 14 ms to the baseline runtime.

The Local Time-Space Attention model offers a more computationally efficient configuration of the Transformer Video Module. In this design, the Time Attention block operates temporally, attending to tokens from the same spatial position across frames. As a result, inference speed is minimally affected when processing additional frames. This module performs optimally with a larger memory size. With $S = 3$, the Local Time-Space Attention model improves VPQ by 1.8% before tracking and by 2.0% after tracking compared to the baseline. For both Time-Attention variants, during inference, past backbone features are stored in memory to avoid redundant feature re-computation when processing frames sequentially.

Memory Size. The effect of memory size is analyzed for both the Global Time-Space Attention and Local Time-Space Attention Transformers by varying the number of past frames from 1 to 4 ($S = \{1, 2, 3, 4\}$), as presented in Table 6.4. For both configurations, all memory sizes yield VPQ improvements over the baseline, ranging from 1.8% to 2.0% before tracking.

The Local Time-Space Attention module benefits significantly from long-term memory, achieving its highest score with a memory size of 3 frames. In this configuration, the tracking module further increases VPQ by 3.3%, highlighting improved temporal consistency across frames. With smaller memory sizes of 1 or 2 frames, the tracking module achieves slightly lower improvements of 2.2%.

In contrast, the Global Time-Space Attention module performs optimally with short-term memory. It achieves its best results with a memory size of 1 frame, as it effectively models complex pixel-wise correlations between the query frame and memory frames. With $S = 2$, the accuracy remains consistent, but for larger memory sizes, performance degrades due to the limited capacity of the Transformer module to model very long-range global correlations.

Transformer Configuration. The model equipped with the original Transformer [147] is compared to the proposed lightweight variant, as shown in Table 6.5. The original Transformer adheres to the standard implementation but excludes dropout. The lightweight variant demonstrates faster runtime and achieves higher video panoptic quality.

In Table 6.6, the results of the Space Attention model with vary-

Transformer module	k =1	k = 5	k = 10	k = 15	VPQ	Time (ms)
Original Transformer [147]	64.0	52.8	49.4	47.0	53.3	101
Proposed Transformer	**64.5**	**52.8**	**49.5**	**47.4**	**53.6**	**94**

Table 6.5: The proposed lightweight Transformer module versus the original Transformer [147] with spatial self-attention only. It obtains higher video panoptic quality and inference speed. Time is measured on a NVIDIA Tesla V100 GPU.

Channels	mIoU	PQ	VPQ	ΔT (ms)	Time (ms)
512	78.9	63.8	52.7	+4	90
768	79.1	64.0	52.9	+6	92
1024	**79.8**	**64.5**	**53.6**	+8	94

Table 6.6: The effect of varying the number of channels in the Transformer Video Module with *Space Attention*. The total inference time of the model and the time overhead over the baseline on the Cityscapes-VPS *val* set without tracking is reported. Time is measured on a NVIDIA Tesla V100 GPU.

ing numbers of channels in the Transformer module are presented. A 1×1 convolution is applied to the backbone features to reduce the number of feature maps from 2048 to a lower dimension. The remaining operations in the Transformer module, including the Attention and MLP blocks, maintain a constant dimensionality.

The results indicate that projecting to a higher number of channels improves accuracy. When using $d = 2048$ channels, a time overhead of 21 ms is measured. However, the configuration with $d = 1024$ channels is selected as it offers a favorable trade-off between accuracy and efficiency.

Qualitative Results. Figure 6.5 presents a comparison between the baseline panoptic image segmentation network and the proposed video counterpart. In the given example, the baseline network fails to correctly segment the motorcycle in the final frame. By incorporating the Transformer video module, temporal consistency is significantly improved, resulting in accurate segmentation of the motorcycle across all frames.

Comparison to the State-of-the-Art. Table 6.7 presents a comparison of the proposed VPS-Transformer network with state-of-the-

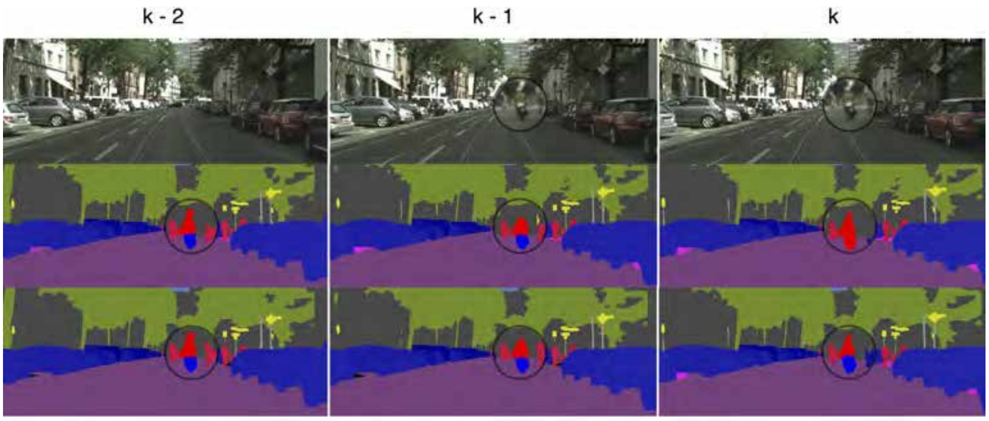


Figure 6.5: Qualitative results on a sequence of three consecutive frames. In the second row, the results of the baseline panoptic image segmentation network are shown, while the third row demonstrates the output of the Transformer video module using *Global Time-Space Attention* with $S = 1$. The encircled area containing the rider and motorcycle is zoomed in for improved visualization. Although the image-level panoptic segmentation network (second row) correctly segments the rider and motorcycle in the first two frames, it fails to segment the motorcycle in the third frame. In contrast, the Transformer video module ensures better temporal consistency, successfully segmenting both the rider and the motorcycle across all three frames (third row).

art methods on the Cityscapes-VPS dataset [74]. Both VPSNet [74] and the VPS-Transformer network are pretrained on the Cityscapes *fine* train dataset [34]. While VPSNet fuses features from 10 neighboring frames, including both past and future frames, the VPS-Transformer achieves superior performance with a 1.2% higher VPQ score when using the ResNet-50 backbone [64]. Additionally, the VPS-Transformer demonstrates significantly improved efficiency, running 7× faster.

With a more powerful backbone, HRNet-W48 [148], the VPS-Transformer further outperforms VPSNet, achieving a 3.7% higher VPQ score while remaining 4× faster. ViP-DeepLab [127], which is pretrained on the larger Mapillary Vistas dataset [108] and Cityscapes [34], utilizes a more complex architecture with a heavy backbone and test-time augmentations. Although ViP-DeepLab achieves better overall results, its inference time is not disclosed. However, it is estimated to exceed 400 ms, which corresponds to the runtime of its baseline panoptic image segmentation network, Panoptic-DeepLab with WR-41 [20].

Under comparable conditions, where both networks employ the ResNet-50 backbone, the VPS-Transformer achieves superior results with a +4.2% improvement in PQ and +4.5% in VPQ compared to ViP-DeepLab. Furthermore, the VPS-Transformer equipped with the HRNet-W48 backbone delivers competitive performance relative to the most powerful ViP-DeepLab model, while running at least twice as fast with an inference time of 185 ms.

Method	Backbone	PQ	VPQ	Time (ms)
VPSNet [74]	ResNet-50	62.7	56.1	770
ViP-DeepLab* [9]	ResNet-50	60.6	52.8	-
Baseline - Panoptic DeepLab [28]	ResNet-50	63.0	52.0	86
VPS-Transformer	ResNet-50	**64.8**	**57.3**	**112**
ViP-DeepLab [127]	WR-41	70.4	63.1	-
Baseline - Panoptic DeepLab [28]	HRNet-W48	66.1	55.1	168
VPS-Transformer	HRNet-W48	67.6	59.8	185

Table 6.7: Comparison to state-of-the-art video panoptic segmentation networks on the Cityscapes-VPS *val* set. VPSNet and the VPS-Transformer are pretrained on the Cityscapes *fine* train set. ViP-DeepLab is pretrained on Mapillary Vistas (MV) and on Cityscapes *fine* train set and uses the more complex WR-41 backbone. ViP-DeepLab* is pretrained on Cityscapes and is evaluated with the author's code. Time is measured on a NVIDIA Tesla V100 GPU.

The VPS-Transformer demonstrates a strong balance between accuracy and efficiency, achieving state-of-the-art performance in video panoptic segmentation while maintaining real-time inference speeds, making it highly suitable for real-world applications such as autonomous driving

Chapter 7

Monocular Depth Estimation

Monocular depth estimation enables the extraction of spatial depth information from single 2D images. This problem is inherently ill-posed, as the same 2D projection can arise from infinitely many 3D scenes. However, recent advances in deep learning have transformed the domain, empowering neural networks to leverage large datasets and learn sophisticated patterns that map image content to depth predictions. This chapter explores the most prominent methods for monocular depth estimation, with a detailed focus on two key approaches.

7.1 Overview

Supervised Monocular Depth Estimation. Depth estimation from a single image is an ill-posed problem since a 2D image can be generated from an infinity of 3D scenes. With the emergence of deep learning, Eigen *et al.* [42] formulated depth regression as a supervised learning problem. Since then, various improvements to network architectures [82, 41, 107, 154] and loss functions [167, 84] have been made. Xian *et al.* [88] models depth estimation as classification and obtains more robust results. However, the classification increases the complexity of the network and introduces challenges regarding the depth interval discretization. DORN [47] and SORD [39] propose improvements over the uniform discretization technique.

The aforementioned methods require ground truth depth, which is usually sparse depth from LiDAR scans. The difficulty to acquire ground truth has led to the development of weakly supervised methods that rely on weak labels such as relative depth [25], camera pose [165] or synthetic data [81, 12, 104]. Another line of research [91, 79] proposes the use of conventional structure-from-motion methods [135], that are usually computationally intensive, to generate pseudo labels. Knowledge distillation from stereo depth estimates [152, 62, 123, 114, 31] has also been recently exploited for improved depth predictions.

Self-Supervised Monocular Depth Estimation. The prohibitively large cost of collecting high-quality ground truth has led to the emergence of self-supervised monocular depth estimation, which unlocks the power of large-scale unlabeled datasets. Such approaches learn both the depth and ego motion, and embed 3D geometric constraints by using 3D reprojection models to synthesize consecutive images. More specifically, points from the target frame are back-projected in the camera coordinate system, displaced by the camera motion and reprojected onto adjacent source frames. In this way, the target image can be reconstructed from the source images, and the photometric difference between the target and synthesized image will be minimized during training.

Early approaches on self-supervised monocular depth estimation [49, 56] were inspired by auto-encoders and employed stereo pairs during training. The SfmLearner [171] was the first solution working on monocular image sequences by jointly training a depth and pose estimation network.

Current approaches address some of the issues of self-supervised monocular learning. Monodepth2 [57] handles the lack of ego motion with an auto-masking of stationary pixels and the occlusion problem with a minimum reprojection loss. Low-texture areas are often problematic when using the photometric loss, therefore feature-based reconstruction losses [136, 165] have been proved more robust. Formulating self-supervised depth estimation as a depth classification problem has been tackled in [58, 71]. Other works improve the network architecture [60] or include test-time refinement procedures [27, 19]. Feature representation learning is guided with semantic networks or single-view reconstruction auto-encoder networks in several approaches [61, 72, 140]. Guizilini *et al.* [61] enhances the feature representation with semantic guidance from a fixed teacher segmentation network using pixel-adaptive convolutions [140]. ManyDepth [153] uses multi-frame input at test-time for improved results.

For extracting potentially moving objects some methods [59, 102, 27, 128, 163] employ external optical-flow networks and design selective masking techniques to avoid propagating large errors in the training signal. Semantic and instance information can be used as well. Casser *et al.* [19] introduces a 3D object motion network that processes images filtered by instance masks. SGDepth [78] detects frames with moving objects based on semantic segmentation and removes them from the training set. [143] segments the object motion with semantic knowledge distillation.

7.2 MonoDepth2

Method. MonoDepth2 [57] is a pioneering self-supervised approach to monocular depth estimation that leverages innovative techniques to predict depth and camera motion using only monocular video sequences, eliminating the need for ground-truth depth supervision. The idea of MonoDepth2 is to jointly train a depth and a pose estimation network simultaneously. While during training sequential triplets from a video are required, during inference self-supervised monocular depth estimation methods process only a single frame.

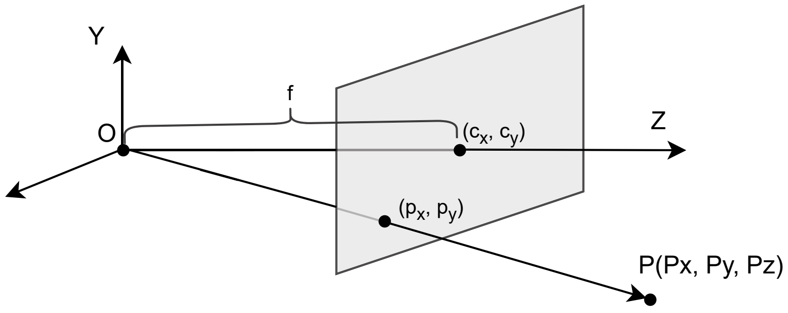


Figure 7.1: Perspective projection model.

To understand the mechanisms behind MonoDepth2, we first review the pinhole camera model and the perspective projection which is used in this work, as seen in Figure 7.1. The 3D camera coordinate system is defined with the optical center O as its origin. The X-axis is parallel to the horizontal axis of the image plane, the Y-axis is parallel to the vertical axis, and the optical axis Z is orthogonal to the image plane. The distance between the optical center and the image plane is referred to as the focal length f. The Z-axis intersects the image plane at the principal point, which has pixel coordinates (c_x, c_y).

To express the focal length in pixel coordinates, we scale the metric focal length f by the size of a pixel (s_x, s_y), obtaining f_x and f_y, the focal lengths in pixels. The intrinsic camera matrix K captures the focal lengths and the principal point as follows:

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{7.1}$$

Let $p(p_x, p_y)$ be a 2D point in the image, located at the intersection of the line PO and the image plane. The corresponding point in the camera coordinate system has metric coordinates (p_{xm}, p_{ym}, f). By applying the principle of similar triangles, we derive the following relationship:

$$\frac{f}{Z_c} = \frac{p_{xm}}{X_c} \tag{7.2}$$

$$p_{xm} = \frac{fX_c}{Z_c} \tag{7.3}$$

$$p_x = \frac{f_x X_c}{Z_c} \tag{7.4}$$

Similarly, we can derive the projection equation for the y-coordinate:

$$p_y = \frac{f_y Y_c}{Z_c} \tag{7.5}$$

Equations 7.2 and 7.5 assume the principal point (c_x, c_y) as the origin of the pixel coordinates. To move the origin to $(0, 0)$, we adjust the equations as follows:

$$p_x = \frac{f_x X_c}{Z_c} + c_x, \tag{7.6}$$

$$p_y = \frac{f_y Y_c}{Z_c} + c_y \tag{7.7}$$

To represent the projections using matrix operations, homogeneous coordinates are introduced. Homogeneous coordinates are obtained by appending a 1 to the original coordinates. In homogeneous form, the pixel coordinates p can be written as:

$$p = \begin{bmatrix} \frac{f_x X_c}{Z_c} + c_x \\ \frac{f_y Y_c}{Z_c} + c_y \\ 1 \end{bmatrix} \sim \begin{bmatrix} f_x X_c + c_x Z_c \\ f_y Y_c + c_y Z_c \\ Z_c \end{bmatrix} \tag{7.8}$$

Thus, the projection of a 3D point $P(X_c, Y_c, Z_c)$ in the camera coordinate system to a 2D pixel point $p(p_x, p_y)$ can now be expressed in matrix form:

$$\begin{bmatrix} f_x X_c + c_x Z_c \\ f_y Y_c + c_y Z_c \\ Z_c \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} \tag{7.9}$$

Consider a target image I_t and adjacent source images I_s, where $s = \{t-1, t+1\}$, captured by a moving camera. Let $M_{t\to s}$ represent the camera pose, which includes the 3D translation $T_{t\to s}$ and the rotation $R_{t\to s}$ between consecutive 3D scene positions:

$$M_{t\to s} = \begin{bmatrix} R_{t\to s} & T_{t\to s} \\ 0 & 1 \end{bmatrix} \tag{7.10}$$

The translation matrix $T_{t\to s}$ from the target to the source coordinate system is defined as:

$$T_{t\to s} = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \tag{7.11}$$

The rotation matrix can be expressed in terms of Euler angles. Let α, β, γ be the Euler angles corresponding to the rotations about the X, Y, and Z-axes, respectively, of the target camera coordinate system into the source camera coordinate system.

The rotation with α radians about the X-axis is given by:

$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) \\ 0 & \sin(\alpha) & \cos(\alpha) \end{bmatrix} \tag{7.12}$$

The rotation with β radians about the Y-axis is defined as:

$$R_y(\beta) = \begin{bmatrix} \cos(\beta) & 0 & \sin(\beta) \\ 0 & 1 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) \end{bmatrix} \tag{7.13}$$

The rotation with γ radians about the Z-axis is defined as:

$$R_z(\gamma) = \begin{bmatrix} \cos(\gamma) & -\sin(\gamma) & 0 \\ \sin(\gamma) & \cos(\gamma) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{7.14}$$

The complete rotation matrix $R_{t\to s}$ is obtained by sequentially applying the rotations around the X-, Y-, and Z-axes:

$$R_{t\to s} = R_x(\alpha)R_y(\beta)R_z(\gamma) \tag{7.15}$$

Monodepth2 uses the projection model and camera pose to synthesize a target frame from a source frame. Specifically, given a pixel p in the target frame, its corresponding position in 3D homogeneous coordinates P within the camera coordinate system can be computed through backprojection using the predicted target depth $Z_t(p)$:

$$P = \begin{bmatrix} K^{-1}Z_t(p)p \\ 1 \end{bmatrix} \tag{7.16}$$

where K is the intrinsic camera matrix.

Assuming a static scene and known camera motion, P can be reprojected into the source frame I_s after transforming it using the camera pose $M_{t\to s}$:

$$p' = \begin{bmatrix} K \,|\, 0 \end{bmatrix} M_{t\to s}P \tag{7.17}$$

where $M_{t\to s}$ represents the 3D rotation and translation between the target and source frames.

The target image I_t is synthesized, denoted as $I_{s\to t}$, by sampling the source image I_s at the projected coordinates p' using bilinear interpolation, as described in [56, 57]. This process is represented as:

$$I_{s\to t} = I_s\langle p'\rangle$$

To train the model, the per-pixel photometric reprojection error $\mathcal{L}_p$ between the target image I_t and the synthesized image $I_{s\to t}$ is minimized. To handle occlusions between views, the minimum reprojection loss over all source images is computed:

$$\mathcal{L}_p = \min_s pe(I_t, I_{s\to t}) \tag{7.18}$$

where pe represents a chosen photometric error metric, such as the L1 loss or Structural Similarity Index (SSIM).

The photometric error pe is defined as the weighted sum of the structural similarity index (SSIM) [151] and the L1 error:

$$pe(I_a, I_b) = \alpha \frac{1 - \mathrm{SSIM}(I_a, I_b)}{2} + (1 - \alpha) \left\| I_a - I_b \right\|_1 \tag{7.19}$$

where α is set to 0.85.

The SSIM between two images x and y is given by:

$$\mathrm{SSIM}(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{7.20}$$

where μ_x and μ_y are the means, σ_x^2 and σ_y^2 are the variances, and σ_{xy} represents the covariance of x and y. The constants $C_1 = 0.01^2$ and $C_2 = 0.03^2$ avoid division by 0.

To account for cases where the camera is stationary, which may manifest as "holes" of infinite depth in the predicted depth map, MonoDepth2 filters out pixels where the reprojection error of the synthesized image $I_{s\to t}$ is lower than that of the original source image I_s. The reprojection loss is then computed only for the remaining pixels.

An edge-aware smoothness loss is adopted to encourage local smoothness in regions with low image gradients:

$$\mathcal{L}_s = \left| \partial_x \hat{d}_t \right| e^{-|\partial_x I_t|} + \left| \partial_y \hat{d}_t \right| e^{-|\partial_y I_t|} \tag{7.21}$$

where $\hat{d}_t = d_t / \bar{d}_t$ is the inverse depth d_t normalized by its mean $\bar{d}_t$ over the image. Normalizing the inverse depth introduces scale-invariance and avoids very small depth values near zero, which could hinder valid 2D projections during training when using smaller depth scales.

The final depth loss is computed at four scales $1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}$, and averaged as follows:

$$\mathcal{L}_{md2} = \frac{1}{4} \sum_{s=0}^{3} \frac{\mathcal{L}_p + \gamma_s \mathcal{L}_s}{2^s} \tag{7.22}$$

where γ_s is set to 0.001.

Experiments. MonoDepth2 is trained on the KITTI dataset [51]. The model is implemented in PyTorch and trained for 20 epochs using the Adam optimizer. The batch size is set to 12, with an input/output resolution of 640 × 192. The learning rate is initialized to 10^{-4} for the first 15 epochs and then reduced to 10^{-5} for the remaining epochs.

Table 7.1 compares several baseline variants and the MonoDepth2 model for monocular depth estimation, evaluated using multiple metrics: Absolute Relative Error (Abs Rel), Squared Relative Error (Sq Rel), Root Mean Squared Error (RMSE), logarithmic RMSE (RMSE log), and accuracy thresholds $\delta < 1.25$, $\delta < 1.25^2$, and $\delta < 1.25^3$. The baseline method shows the highest errors, with an absolute relative error of 0.140 and RMSE of 5.512. Adding minimum reprojection loss significantly improves performance, reducing the absolute relative error to 0.122 and RMSE log to 0.199. Introducing automasking further improves results, achieving an absolute relative error of 0.124 and RMSE of 5.010, while incorporating full-resolution multi-scale supervision achieves similar improvements.

The MonoDepth2 model, which combines auto-masking, minimum reprojection loss, and full-resolution multi-scale supervision, achieves the best overall performance across all metrics. These results demonstrate that the combination of all three techniques leads to significant improvements in both error reduction and accuracy compared to the baseline variants.

Method	Auto-masking	Min. reproj.	Full-res multi-scale	Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^3$
Baseline				0.140	1.610	5.512	0.223	0.852	0.973
Baseline + min reproj.		✓		0.122	1.081	5.116	0.199	0.866	0.980
Baseline + automasking	✓			0.124	0.936	5.010	0.206	0.858	0.977
Baseline + full-res m.s.			✓	0.124	1.170	5.249	0.203	0.865	0.978
Monodepth2 (full)	✓	✓	✓	**0.115**	**0.903**	**4.863**	**0.193**	**0.877**	**0.981**

Table 7.1: Baseline variants and MonoDepth2 (full) performance on KITTI.

MonoDepth2 demonstrates the effectiveness of combining self-supervised techniques for monocular depth estimation. By integrating auto-masking, minimum reprojection loss, and full-resolution multi-scale supervision, it achieves significant improvements in both accuracy and error reduction compared to baseline methods. The model's ability to synthesize target images from source views, while accounting for occlusions and enforcing edge-aware smoothness, enables robust depth predictions without requiring ground-truth depth supervision. These results highlight MonoDepth2 as a state-of-the-art framework for self-supervised depth estimation, paving the way for practical applications in resource-constrained scenarios such as autonomous driving and robotics.

7.3 SD-SSMDE

Method. Self-supervised monocular depth estimation relies on several assumptions that are not always true and hinder the learning performance. As a consequence, propagating correct training signals is still difficult for all pixels, photometric loss can be high in occluded areas or for moving objects, and low in uniform texture areas or for repetitive structures. Another problem is that the self-supervised depth estimation methods suffer from depth scale ambiguity which hinders their applicability in real-world systems. Specifically, the depth output of the network is relative depth, where the depth values in an image are in broad agreement with each other, however they differ from the real-world depth values by a scale factor. Also, each depth map requires a different scale factor, as the depth maps are not inter-frame scale consistent. The SD-SSMDE work [118] investigates

1. how to remove ground truth dependency with self-supervised monocular depth estimation
2. how to design a depth network architecture that improves the performance while being efficient
3. how to improve self-supervised depth training by propagating improved training signal
4. how to solve scale ambiguity

To address the first research question, a two-stage self-distillation training strategy for monocular depth estimation is introduced, referred to as **SD-SSMDE** [118]. In the first stage, the network is trained in a self-supervised regime by minimizing the photometric loss between consecutive views. The objective of this stage is to produce high-resolution pseudo depth labels that will serve as supervision for training a similar or more lightweight network in the second stage. To enhance the training signal, a filtering strategy based on 3D consistency between consecutive views is proposed to eliminate significant errors in pseudo labels.

To further improve performance, a novel depth network architecture is introduced. As self-supervised methods inherently suffer from scale ambiguity—providing relative depth rather than absolute depth—a scale factor is typically required to recover real-world depth values. Although median scaling with ground-truth data is commonly applied during testing, this approach contradicts the core motivation of self-supervised learning, which aims to avoid reliance on ground-truth annotations. To address this issue, the method incorporates scale into the pseudo labels, ensuring that depth predictions in the second stage are automatically scaled and inter-frame scale-consistent. The scale factor can be computed directly from depth predictions by estimating the camera height, as described in [160].

In the context of related works, self-distillation for monocular depth estimation has been explored in [124], where the student network is trained to mimic the teacher network's output distribution by learning its mean and variance. In contrast, this approach distills the student network using high-resolution hard pseudo labels, which are further refined through a 3D consistency-based filtering scheme applied between consecutive views. A significant contribution of this work, which has not been investigated in prior studies, is the resolution of scale ambiguity: the student network is trained using inter-frame scale-consistent pseudo labels that have been scaled to absolute depth values.

Self-Distillation Based Training Pipeline. SD-SSMDE is a two-stage training pipeline for monocular depth estimation, which relies solely on video frames and does not require ground-truth depth data. In the first stage, the camera pose network [57] and the depth teacher network are trained in a self-supervised manner using high-resolution images and the photometric loss. The trained depth network is then used to predict depth maps for the entire training set. As the depth outputs differ from real-world depth values by a scale factor, a scale recovery module [160] is employed to recover absolute depth values.

In the second stage, the camera pose network is fixed, and a new depth student network is instantiated, which can have the same or a more lightweight architecture compared to the teacher network. The student depth network is trained from scratch in a supervised regime using the generated pseudo labels. During this training phase, a mask is dynamically generated to filter out depth predictions with large errors, thereby excluding them from the loss computation. This filtering process assumes that the same scene captured in three consecutive frames should exhibit a high degree of 3D consistency when viewed from different perspectives. Depth locations with significant deviations between the 3D points in the camera coordinate system are identified and filtered out.

Depth Network Architecture. The encoder-decoder architecture for the depth network, utilized by both the teacher and stu-

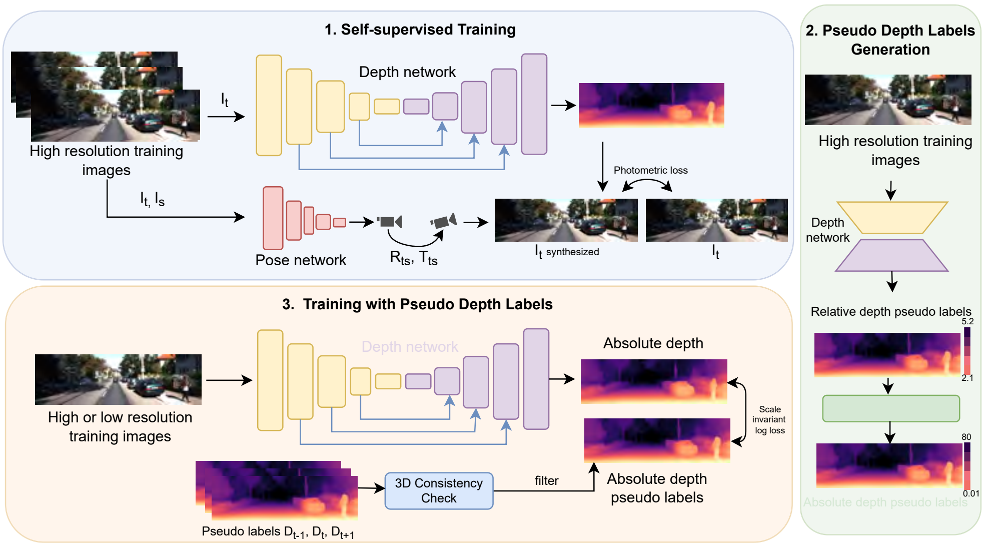

1. Self-supervised Training
High resolution training images
I_t
Depth network
I_t, I_s
Pose network
R_{ts}, T_{ts}
Photometric loss
I_t synthesized
I_t
2. Pseudo Depth Labels Generation
High resolution training images
Depth network
Relative depth pseudo labels
5.2
2.1
80
0.01
Absolute depth pseudo labels
3. Training with Pseudo Depth Labels
High or low resolution training images
Depth network
Absolute depth
Scale invariant log loss
Absolute depth pseudo labels
filter
3D Consistency Check
Pseudo labels D_{t-1}, D_t, D_{t+1}

Figure 7.2: The two-stage training framework. In the first stage, a depth estimation network and a camera pose network are trained in a self-supervised manner. In this setting, the photometric error between the target image I_t and the synthesized images from the adjacent source images I_s is minimized. Using the trained depth network, pseudo-labels for all the images in the training set are generated. Automatic scale recovery is performed in order to obtain absolute depth values. In the second stage, the depth network is trained from scratch to regress depth maps supervised by the previously generated pseudo-labels. In order to remove erroneous depth estimates from pseudo-labels, a consistency check is performed, for the same 3D point computed from different views.

dent networks, is based on the design of the Panoptic-DeepLab network [28, 127], with several modifications. The backbone operates with an output stride of 32 instead of 16. Context information is extracted from the backbone output using an Atrous Spatial Pyramid Pooling (ASPP) module [24], which employs parallel dilated depthwise separable convolutions [68]. The decoder comprises five upsampling stages, where the spatial resolution is progressively increased by a factor of two at each stage. Each upsampling stage consists of an upsampling operation, concatenation with low-level features from the backbone, and a $[5 \times 5, 256]$ depthwise separable convolution for feature fusion. The low-level features, from scales 1/16 to 1/2, are projected to channel dimensions of $\{128, 64, 32, 16\}$ before concatenation. Following the final upsampling stage, a $[5 \times 5, 64]$ depthwise separable convolution and a $2\times$ bilinear upsampling operation restore the spatial resolution to the original input size. Two additional convolutional layers, with kernel sizes $[5 \times 5, 32]$ and $[1 \times 1, 1]$, produce the final depth map.

During training, multi-scale depth predictions are employed at four scales, $\{1/8, 1/4, 1/2, 1\}$, with a $[1 \times 1, 1]$ convolution applied after feature fusion at each scale. The loss is computed using the multi-scale depth predictions, as described in [49, 56, 57]. A depiction of the network architecture is shown in Figure 7.3.

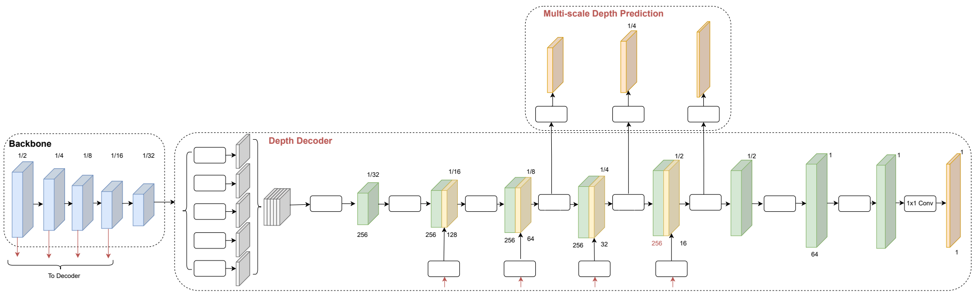


Figure 7.3: The depth network architecture

Self-Supervised Monocular Depth Estimation. In the first stage of training, the teacher network is trained in a self-supervised manner. The objective of self-supervised monocular depth estimation is to predict a depth map aligned to the input image without requiring ground-truth depth data during training. The approach leverages geometric projections to synthesize adjacent views based on the predicted depth. During inference, the network predicts depth from a single input image. However, during training, three consecutive frames are utilized: the target frame and its two adjacent source frames. Two separate networks—a depth estimation network and a camera pose estimation network—are jointly trained. The depth estimation network predicts the inverse of depth, which has been shown to be more robust for learning [49, 57]. To train the self-supervised teacher network, the photometric loss and smoothness loss, as employed by Monodepth2 [57], are adopted.

Scale Recovery in Pseudo Labels. The output of the self-supervised depth estimation network provides relative, up-to-scale depth. This means that while the depth values within an image are relative to one another, they differ from real-world measurements by an unknown scale factor. To address this issue, the scale recovery technique from [160] is employed to compute the scale factor. The scale recovery module determines the scale by estimating the relationship between the predicted camera height and the actual camera height. The first step involves identifying ground points in the scene. This is achieved by computing the surface normal for each 3D point and selecting points whose normalized normals are close to the ideal ground normal $\mathbf{n} = (0, 1, 0)^\top$, based on a similarity function. Once the ground points are identified, a set of camera heights is estimated for each corresponding 3D point. The depth scale factor is then calculated as the ratio between the real camera height and the median of the estimated camera heights. Finally, each pseudo label is scaled using the computed scale factor, resulting in absolute depth values and scale-consistent pseudo labels across frames.

Supervised Monocular Depth Estimation. In the second stage, the depth estimation task for training the student network is formulated as a supervised regression problem. The scale-invariant log loss, as introduced in [42, 84], is adopted:

$$\mathcal{L}_{sp} = \gamma \sqrt{\frac{1}{N}\sum_i d_i^2 - \frac{\lambda}{N^2}\left(\sum_i d_i\right)^2} \tag{7.23}$$

where $d_i = \log y_i - \log \bar{y}_i$, y_i is the predicted depth and $\bar{y}_i$ is the pseudo ground-truth depth. N represents the number of pixels with valid values and λ is a weighting factor. The range of the loss is scaled with γ in order to improve convergence.

During inference, depth values are directly predicted using the logits from the depth regression head. A sigmoid activation function is applied to the logits, and the resulting values are scaled by a constant factor, set to 80. This scaling is consistent for both the KITTI [51] and Cityscapes [34] datasets, where the typical depth range is $[0, 80]$ meters.

Filtering Errors in Pseudo Labels. To ensure the reliability of high-resolution pseudo depth labels, a 3D consistency check between consecutive views is performed, as illustrated in Figure 7.4. This check is valid because the pseudo labels become inter-frame consistent once scaled to absolute depth values. Reliable depth estimates are retained only for pixels with similar 3D coordinates across different views. This masking process is conducted on-the-fly during the second stage of training.

Assume the intrinsic matrix K and the camera pose M between the target and source coordinate systems are known, as provided by the pose network trained in the previous stage. Let p represent a pixel in the target image. The corresponding 3D point P can be obtained through backprojection using Equation 7.16. The 2D coordinates p' of this point in the source image are subsequently calculated using Equation 7.17. Since p' has real-valued coordinates, the source depth Z_s is sampled via bilinear interpolation as $Z_s\langle p'\rangle$ and backprojected into the source camera coordinate system as P'.

Next, the 3D point P' is displaced into the target camera's coordinate system using the camera pose transformation $M_{s\to t}$. The absolute difference between the z-axis coordinates of the two 3D points, P and P', is computed. If this difference is smaller than a predefined threshold T, the point is considered valid; otherwise, it is filtered out.

To account for potential occlusions in one of the source views, the minimum 3D consistency error between the target and adjacent source views is adopted. The resulting mask F is computed as the Iverson bracket:

$$F = \left[\min_s \left\| M_{s\to t} Z_s\langle p'\rangle K^{-1}p' - Z_t(p)K^{-1}p)\right\|_1 < T\right] \tag{7.24}$$

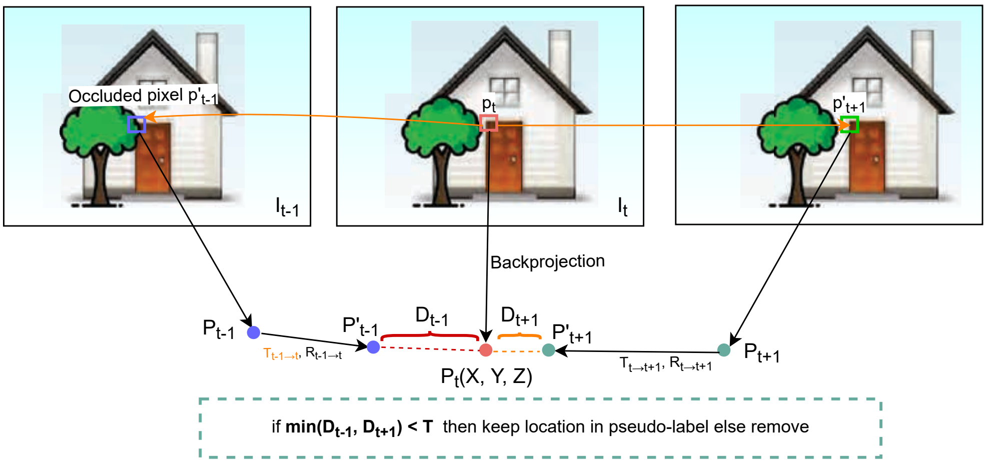


Figure 7.4: 3D consistency check which enables filtering out large errors in pseudo-labels.

For the depth prediction network, the backbone pretrained on Imagenet [38] is utilized, as introduced in [64]. In the self-supervised network, the inverse depth, which is the output of the sigmoid layer, is converted into depth within the interval $[0.1, 100]$ using

$$Z = \frac{1}{a\sigma + b}$$

The pose estimation network adopts a lightweight architecture [57] featuring a ResNet-18 backbone [64]. This network processes pairs of color images, namely the target and source, to predict the 6DOF camera pose, consisting of the translation vector and the rotation matrix expressed in terms of three Euler angles. During inference, the pose network is discarded.

Experiments. On KITTI [50], the networks are trained in both training stages with a minibatch of 12 images for 66k iterations, using the Adam optimizer and a base learning rate of 10^{-4}. In the first stage, step learning rate decay is applied, reducing the learning rate by a factor of 10 at 50k iterations. In the second stage, polynomial learning rate decay is employed, and the networks are trained for the same number of iterations. On Cityscapes, training is performed with a minibatch of 12 images for 12k iterations in the first stage and 30k iterations in the second.

During training, image augmentation techniques are applied, including random horizontal flipping and random color augmentation with settings from [57]. In the self-supervised training stage, the smoothness loss is weighted by 0.001, and α is set to 0.85 in the photometric loss. For the supervised loss, λ is set to 0.85, and γ is set to 10 [84]. The threshold T for 3D consistency masking is set to 1. Both the self-supervised and supervised losses are computed at four scales.

On the Cityscapes dataset [34], during evaluation, the original images with a resolution of 1024×2048 are center-cropped to 512×1664, as described in [153, 19]. During the training of the self-supervised teacher network, the images are center-cropped to 768×2048, and high-resolution pseudo-labels of the same size 768×2048 are generated. In the second stage of training, the cropping scheme used for evaluation is applied, and the images are further scaled to 128×416.

Ablation Study on KITTI. Ablation experiments are conducted to analyze the benefits of the self-distillation learning framework, filtering scheme, and scale recovery.

Self-distillation Based Learning Framework. Table 7.2 presents ablation experiments focusing on the self-supervised learning framework. The networks are trained on two image resolutions:

medium resolution 192×640 and high resolution 320×1024. In the first experiment, the depth network is trained in a supervised regime using the improved KITTI ground truth [145]. The depth network, equipped with a ResNet-50 backbone and the proposed decoder, is optimized using the scale-invariant logarithm loss [84] for depth regression. The training conditions and hyperparameters remain identical to those used for the self-supervised method. Next, the teacher network is trained in the self-supervised regime. Self-supervised methods inherently suffer from scale ambiguity, meaning that the output is not scaled to real-world values. Experiments are conducted with both ground truth median scaling [57], a common practice, and an automatic scale recovery method [160] during inference. The results show that adopting automatic scaling leads to increased error due to inaccuracies in the scale computed from the predicted depth maps. Using the best-performing model with a ResNet-50 backbone, trained on high-resolution images and automatic scaling, pseudo-labels are generated for the entire training set. In the second stage, the student depth network is supervised using these pseudo-labels. A 3D consistency check is applied to remove noisy estimates, resulting in sparser yet more accurate labels. The second-stage training yields improved results for both image resolutions. During inference, a fixed scale factor is applied to map the depth values into the range $[0, 80\text{m}]$.

Model	GT Scaling	Auto Scaling	Fixed Scaling	Resolution	AbsRel ↓	SqRel ↓	RMS ↓	RMSlog ↓	$\delta < 1.25$ ↑	$\delta < 1.25^2$ ↑	$\delta < 1.25^3$ ↑
Supervised (reference)			✓	192 × 640	0.097	0.645	4.296	0.180	0.892	0.964	0.983
Self-supervised (teacher)	✓			192 × 640	0.104	0.768	4.513	0.180	0.892	0.964	0.983
Self-supervised (teacher)		✓		192 × 640	0.108	0.795	4.655	0.192	0.878	0.959	0.981
Pseudo-supervised (student)			✓	192 × 640	**0.100**	**0.661**	**4.264**	**0.172**	**0.896**	**0.967**	**0.985**
Supervised (reference)			✓	320 × 1024	0.091	0.567	4.137	0.177	0.902	0.966	0.983
Self-supervised (teacher)	✓			320 × 1024	0.101	0.720	4.339	0.176	0.898	0.967	0.984
Self-supervised (teacher)		✓		320 × 1024	0.104	0.747	4.453	0.185	0.885	0.963	0.983
Pseudo-supervised (student)			✓	320 × 1024	**0.098**	**0.674**	**4.187**	**0.170**	**0.902**	**0.968**	**0.985**

Table 7.2: Ablation study for the self-distillation based two-stage self-supervised learning framework. Experiments are conducted using a ResNet-50 backbone, two image resolutions, and three scale recovery methods during inference. In the second stage, training is consistently performed on scaled pseudo-labels generated from the high-resolution self-supervised model, with 3D consistency check filtering applied.

Depth Decoder. Table 7.3 presents the results of the baseline Monodepth2 [57] and the SD-SSMDE teacher network, trained in a self-supervised regime using the photometric loss. The same loss functions as in [57] are employed; however, a different decoder for the depth network is proposed.

With both the lightweight ResNet-18 and the deeper ResNet-50 backbones [64], the SD-SSMDE approach achieves lower error compared to [57]. This improvement can be attributed to the proposed decoder, which better captures context due to the ASPP module [28] and the increased number of channels in each convolutional layer.

Model	Backbone	AbsRel ↓	SqRel ↓	RMS ↓	RMSlog ↓	$\delta < 1.25$ ↑
Monodepth2 [57]	ResNet-18	0.115	0.903	4.863	0.193	**0.877**
SD-SSMDE	ResNet-18	**0.112**	**0.854**	**4.839**	**0.190**	0.876
Monodepth2 [57]	ResNet-50	0.110	0.831	4.642	0.187	0.883
SD-SSMDE	ResNet-50	**0.104**	**0.768**	**4.513**	**0.180**	**0.892**

Table 7.3: Ablation study for the depth decoder of the self-supervised teacher network. By changing the decoder significant improvements are obtained compared to the Monodepth2 baseline [57] especially for the ResNet-50 backbone. The network is trained on medium resolution images.

Scale Recovery and Filtering. Table 7.4 presents ablation studies for the second part of the training pipeline, conducted using medium-resolution images. The goal is to evaluate the impact of training with scaled pseudo-labels. In the first experiment, the student depth network is trained with high-resolution pseudo-labels scaled using ground-truth median scaling. In this setting, where no error filtering is applied, the results improve over the self-supervised counterpart. This improvement is attributed to the self-supervised learning stage, where the network may become stuck in a local minimum due to the use of the reprojection loss. In contrast, training with labels, even noisy ones, enables the regression loss to guide the network toward a better minimum. In the second experiment, the network is trained using pseudo-labels scaled by an off-the-shelf scale recovery module [160], without using any ground-truth data. Interestingly, the results are comparable to those achieved with ground-truth scaled depth maps. Eliminating the reliance on ground-truth data is a significant advantage. Therefore, in the final experiment, the automatically-scaled pseudo-labels are filtered using a 3D consistency check to provide a more accurate training signal. As expected, training with higher-quality labels further improves the results.

Ablation Study on Filtering Threshold. Table 7.5 presents an ablation study on the threshold used in the filtering scheme, evaluating the depth error on pseudo-labels both before and after filtering. The filtering scheme results in pseudo-labels that are more accurate but also

Gt Scaled PS	Auto Scaled PS	Filtering	AbsRel ↓	SqRel ↓	RMS ↓	RMSlog ↓	$\delta < 1.25$ ↑
✓			0.102	0.716	4.351	0.177	0.887
	✓		0.103	0.729	4.457	0.181	0.881
	✓	✓	**0.100**	**0.661**	**4.264**	**0.172**	**0.896**

Table 7.4: In this stage, the student network is trained using high-resolution pseudo-labels (PS) generated by the self-supervised teacher network. The pseudo-labels are scaled either with ground truth or using an automatic scale recovery method [160]. Performing a 3D consistency check to filter out errors from the pseudo-labels proves to be beneficial.

sparser. A threshold of $T = 1$ provides the best balance between accuracy and density.

	Depth estimation error		Pseudo labels error		
Threshold (m)	AbsRel ↓	RMS ↓	% Filtered	AbsRel ↓	RMS ↓
no filtering	0.103	4.457	0	0.082	3.995
1.0	**0.100**	**4.264**	18	**0.069**	**3.245**
1.5	0.101	4.364	12	0.072	3.416

Table 7.5: Filtering scheme ablation. Comparison between student network training with or without pseudo label filtering on the KITTI test set. The error of pseudo labels is measured on the training set and the amount of 3D points that are filtered.

Effect of Scaled Pseudo-Labels. Table 7.6 demonstrates the advantage of using scaled pseudo-labels during training. By scaling the pseudo-labels with an automatic scale recovery method, absolute depth values as well as inter-frame scale-consistent depth labels are obtained. The experiments indicate that training with scaled pseudo-labels is essential for improved performance. Additionally, a scale variance analysis is conducted on the depth outputs of the student network. The scale is defined as the median of all individual ratios between the ground-truth depth and the medians of the predicted depth maps. The standard deviation of these individual scales, denoted as σ_{scale}, is reported, where a lower value signifies increased scale-consistent depth predictions across frames. The best results, including the most scale-consistent predictions, are achieved using the SD-SSMDE model with a ResNet-50 backbone trained on scaled pseudo-labels.

Model	PS	σ_{scale}	AbsRel ↓	SqRel ↓	RMS ↓	RMSlog ↓	$\delta < 1.25$ ↑
Monodepth2 [57] (R18)	-	0.093	0.109	0.623	4.136	0.154	0.873
SD-SSMDE (R50)	unscaled	0.100	0.109	0.494	3.591	0.141	0.888
SD-SSMDE (R18)	scaled	0.061	0.084	0.436	3.550	0.128	0.918
SD-SSMDE (R50)	scaled	**0.040**	**0.076**	**0.377**	**3.304**	**0.117**	**0.933**

Table 7.6: Scale variance analysis. Comparison on KITTI Eigen test split with improved ground truth [145] on 192×640 resolution. The student network learns from unscaled or automatically scaled [160] pseudo labels (PS). During inference, the standard deviation σ_{scale} of individual ground truth median scales is computed. All depth predictions are scaled with a fixed scale factor.

Results on KITTI. Table 7.7 compares the results of the SD-SSMDE approach with state-of-the-art methods that perform inference on a single image. When trained with medium-resolution images, the network with a ResNet-50 backbone outperforms all other methods and achieves a significant improvement over the baseline. With a ResNet-18 backbone on medium-resolution images, the results are comparable to those of FSRE-Depth [72]. Approaches such as [61, 78, 72] incorporate semantic segmentation guidance and rely on pre-trained semantic networks and pixel-level semantic annotations, which can be costly and challenging to acquire. In contrast, the SD-SSMDE network achieves the best results while being trained solely on monocular sequences without additional data. Another notable advantage of the method is the ability to eliminate the dependence on ground-truth data during inference by using a fixed scaling factor, with minimal or no loss in accuracy. However, a drawback of the two-stage training framework is the longer training time. Despite this, there is no additional computational cost during inference, which is crucial from a practical perspective. For high-resolution images, the method achieves the best overall scores.

Qualitative Comparison. Figure 7.5 presents a qualitative comparison between Monodepth2 [57] and the results from the SD-SSMDE student network. The observations indicate that the SD-SSMDE network produces more accurate depth maps, particularly on surfaces such as the ground and vehicles.

Results on Improved KITTI. Table 7.8 presents the results on the KITTI Eigen set using the improved ground truth [145]. Experiments are conducted with both a fixed scaling factor and ground truth

Method	Backbone	Sem	Resolution	AbsRel ↓	SqRel ↓	RMS ↓	RMSlog ↓	$\delta < 1.25$ ↑	$\delta < 1.25^2$ ↑	$\delta < 1.25^3$ ↑
GeoNet [163]	ResNet-50		192 × 640	0.153	1.328	5.737	0.23	0.802	0.934	0.972
DF-Net [173]	ResNet-50		192× 640	0.146	1.182	5.215	0.213	0.818	0.943	0.978
Guizilini *et al.* [61]	ResNet-50	✓	192 × 640	0.113	0.831	4.663	0.189	0.878	0.971	0.983
SGDepth [78]	ResNet-50	✓	192 × 640	0.112	0.833	4.688	0.190	0.884	0.961	0.981
Monodepth2 [57]	ResNet-50		192 × 640	0.110	0.831	4.642	0.187	0.883	0.962	0.982
FSRE-Depth [72]	ResNet-50	✓	192 × 640	0.102	0.675	4.393	0.178	0.893	0.966	0.984
SD-SSMDE	ResNet-50		192 × 640	**0.100**	**0.661**	**4.264**	**0.172**	**0.896**	**0.967**	**0.985**
Shu *et al.* [136]	ResNet-50	✓	320 × 1024	0.104	0.729	4.481	0.179	0.893	0.965	0.984
SD-SSMDE	ResNet-50		320 × 1024	**0.098**	**0.674**	**4.187**	**0.170**	**0.902**	**0.968**	**0.985**
Guizilini *et al.* [61]	ResNet-18	✓	192 × 640	0.117	0.854	4.714	0.191	0.873	0.963	0.981
Monodepth2 [57]	ResNet-18		192 × 640	0.115	0.903	4.863	0.1F93	0.877	0.959	0.981
SGDepth [78]	ResNet-18	✓	192 × 640	0.113	0.835	4.693	0.191	0.879	0.961	0.981
Poggi *et al.* [124]	ResNet-18		192 × 640	0.111	0.863	4.756	0.188	0.881	0.961	0.982
HR-Depth [103]	ResNet-18		192 × 640	0.109	0.792	4.632	0.185	0.884	0.962	0.983
FSRE-Depth [72]	ResNet-18	✓	192 × 640	**0.105**	**0.722**	4.547	0.182	**0.886**	**0.964**	**0.984**
SD-SSMDE	ResNet-18		192 × 640	0.106	0.751	**4.485**	**0.180**	0.885	**0.964**	**0.984**
Monodepth2 [57]	ResNet-18		320 × 1024	0.115	0.882	4.701	0.190	0.879	0.961	0.982
SGDepth [78]	ResNet-18	✓	384 × 1280	0.107	0.768	4.468	0.186	0.891	0.963	0.982
HR-Depth [103]	ResNet-18		320 × 1024	0.106	0.755	4.472	0.181	0.892	0.966	0.984
FSRE-Depth [72]	ResNet-18	✓	320 × 1024	0.102	**0.687**	4.366	0.178	0.895	**0.967**	0.984
SD-SSMDE	ResNet-18		320 × 1024	**0.101**	0.700	**4.332**	**0.174**	**0.895**	0.966	**0.985**

Table 7.7: Comparison with the state-of-the-art on KITTI Eigen test set [41]. Methods in the table use only a single image during inference. *Sem* denotes the use of semantic segmentation. Best results are in bold.

median scaling. The SD-SSMDE approach outperforms the Monodepth2 baseline [57] across both scaling methods, with ResNet-18 and ResNet-50 backbones, and on both medium and high resolutions. The largest performance difference is observed when using a fixed scaling factor. Monodepth2 does not enforce scale consistency between depth predictions, resulting in significant scale variance across frames. Consequently, the error increases when applying a single scale factor to all predictions. In contrast, the SD-SSMDE model achieves better performance by learning from scale-consistent pseudo-labels, which ensures inter-frame scale-consistent depth predictions. This property enables the use of a single scaling factor for all depth predictions with minimal accuracy loss compared to ground truth median scaling.

Results on Cityscapes. Table 7.9 presents the evaluation of the SD-SSMDE models with a ResNet-50 backbone on the Cityscapes dataset, comparing the results with state-of-the-art methods. Additionally, the generalization capability of the model trained on KITTI, without any fine-tuning, is assessed. Compared to Monodepth2 and other competing methods, this approach achieves superior performance across all metrics.

Inference Time. Table 7.10 reports the inference time measured on an NVIDIA Tesla V100 GPU and the number of multiply-add computations (MACs), computed using the PyTorch 1.7.1 frame-

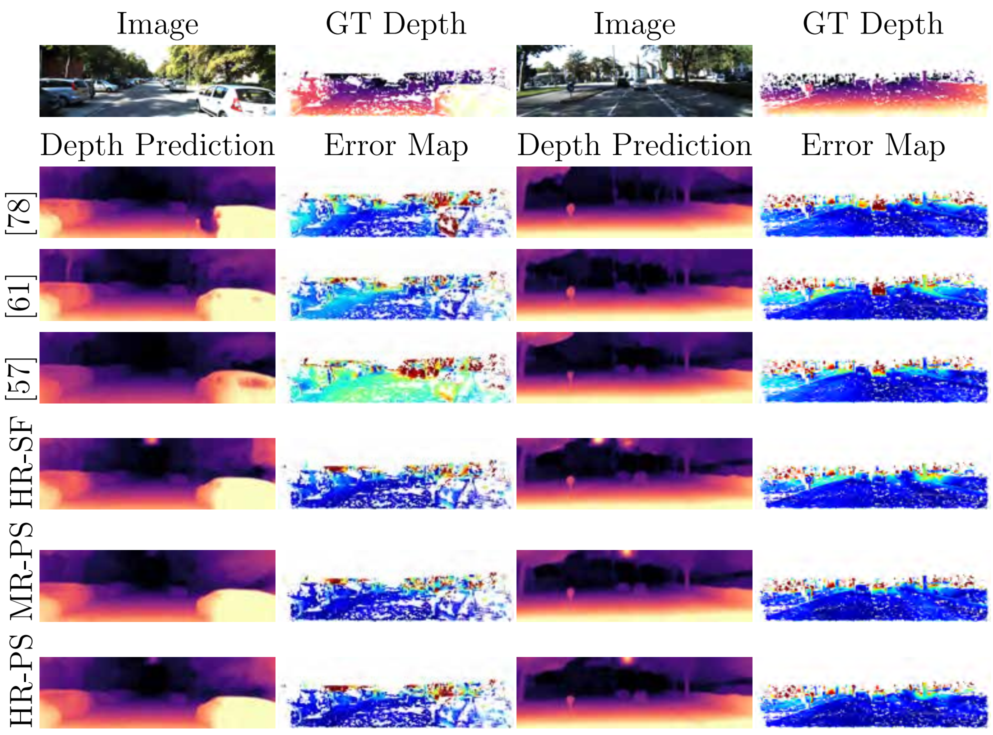


Figure 7.5: **Qualitative Results on KITTI Eigen Test Set.** The qualitative results on the KITTI Eigen test set with improved ground truth are compared against [57, 78, 61]. Both [78] and [61] incorporate external data, such as semantic segmentation, and use medium-resolution images. Monodepth2 [57] with a ResNet-50 backbone serves as the baseline for high-resolution images (HR). HR-SF represents the output from the first stage of training, *i.e.*, the self-supervised teacher network trained on high-resolution images 320×1024, which is subsequently used for generating pseudo-labels. MR-PS and HR-PS correspond to the outputs from the second stage of training of the student network, supervised using pseudo-labels on medium-resolution and high-resolution images, respectively. The HR-PS network produces the highest-quality depth maps, as reflected in Table 7.7.

work and the THOP library[1]. Comparisons are made against the Monodepth2 baseline [57] and FSRE-Depth [72]. The SD-SSMDE model achieves higher accuracy than Monodepth2 but is slightly more compu-

[1]https://github.com/Lyken17/pytorch-OpCounter

Method	Scaling	Backbone	Resolution	AbsRel ↓	SqRel ↓	RMS ↓	RMSlog ↓	$\delta < 1.25$ ↑	$\delta < 1.25^2$ ↑	$\delta < 1.25^3$ ↑
Monodepth2 [57]	GT	ResNet-18	192 × 640	0.084	0.481	3.757	0.129	0.923	0.985	0.996
SD-SSMDE	GT	ResNet-18	192 × 640	0.079	0.399	3.442	0.121	0.929	0.986	0.997
SD-SSMDE	GT	ResNet-50	192 × 640	**0.072**	**0.347**	**3.219**	**0.111**	**0.941**	**0.990**	**0.998**
Monodepth2 [57]	Fixed	ResNet-18	192 × 640	0.104	0.561	3.961	0.147	0.882	0.980	0.995
SD-SSMDE	Fixed	ResNet-18	192 × 640	0.084	0.436	3.550	0.128	0.918	0.985	0.997
SD-SSMDE	Fixed	ResNet-50	192 × 640	**0.076**	**0.377**	**3.304**	**0.117**	**0.933**	**0.988**	**0.997**
Monodepth2 [57]	GT	ResNet-18	320 × 1024	0.085	0.450	3.542	0.126	0.925	0.987	0.996
SD-SSMDE	GT	ResNet-18	320 × 1024	0.072	0.344	3.255	0.112	0.940	0.989	0.998
SD-SSMDE	GT	ResNet-50	320 × 1024	**0.068**	**0.311**	**3.077**	**0.106**	**0.947**	**0.991**	**0.998**
Monodepth2 [57]	Fixed	ResNet-18	320 × 1024	0.096	0.504	3.691	0.137	0.903	0.984	0.996
SD-SSMDE	Fixed	ResNet-18	320 × 1024	0.077	0.370	3.338	0.118	0.931	0.988	0.997
SD-SSMDE	Fixed	ResNet-50	320 × 1024	**0.074**	**0.338**	**3.144**	**0.112**	**0.939**	**0.990**	**0.998**

Table 7.8: The evaluation is conducted on the KITTI Eigen set using the improved ground truth [145]. During inference, the depth predictions are scaled either using the ground truth median (GT) or a fixed scale factor. For comparison, Monodepth2 [57] is evaluated using the authors' provided code.

Model	Train	Test	AbsRel ↓	SqRel ↓	RMS ↓	RMSlog ↓	$\delta < 1.25$ ↑	$\delta < 1.25^2$ ↑	$\delta < 1.25^3$ ↑
Struct2Depth 2 [19]	C	C	0.145	1.737	7.280	0.205	0.813	0.942	0.976
Monodepth2 [57]	C	C	0.129	1.569	6.876	0.187	0.849	0.957	0.983
Videos in the Wild [59]	C	C	0.127	1.330	6.960	0.195	0.830	0.947	0.981
Li *et al.* [86]	C	C	0.119	1.290	6.980	0.190	0.846	0.952	0.982
Choi *et al.* [32]	C	C	0.115	1.125	6.584	0.195	0.857	0.963	0.986
SD-SSMDE (teacher - GT scaling)	C	C	0.117	1.090	6.468	0.176	0.856	0.964	0.990
SD-SSMDE (student - fixed scaling)	C	C	0.114	1.017	5.949	0.169	0.870	0.967	0.990
SD-SSMDE (student - GT scaling)	C	C	**0.110**	**0.988**	**5.953**	**0.165**	**0.876**	**0.970**	**0.991**
Monodepth2 [57]	K	C	0.153	1.785	8.590	0.234	0.774	0.926	0.976
SD-SSMDE (student - fixed scaling)	K	C	**0.143**	**1.635**	**8.441**	**0.221**	**0.789**	**0.931**	**0.980**

Table 7.9: Evaluation on Cityscapes. Evaluation of models on the Cityscapes dataset, trained on Cityscapes (C) or on KITTI (K).

tationally intensive. In contrast, the model is significantly more efficient than FSRE-Depth, which provides a comparable absolute relative error. FSRE-Depth is slower due to its reliance on a semantic segmentation network for cross-task feature refinement.

The self-distillation concept is worth exploring in the context of self-supervision. As demonstrated by SD-SSMDE, a two-stage training framework that generates high-resolution pseudo-labels, which are further used to supervise a lightweight depth network, brings significant improvements. Improving the accuracy of pseudo-labels through filtering and the use of metric-scaled pseudo labels are also important from a practical perspective.

Such a monocular depth estimation network could be used in autonomous vehicles in scenarios where cost, hardware constraints, or environmental conditions limit the use of more expensive and complex

Model	Backbone	Resolution	MACs	Time (ms)	AbsRel ↓
Monodepth2 [57]	ResNet-18	192 × 640	**8.0**	**11**	0.115
FSRE-Depth [72]	ResNet-18	192 × 640	20.4	18	**0.105**
SD-SSMDE	ResNet-18	192 × 640	10.8	12	0.106
Monodepth2 [57]	ResNet-18	320 × 1024	**21.4**	**12**	0.115
FSRE-Depth [72]	ResNet-18	320 × 1024	54.5	29	0.102
SD-SSMDE	ResNet-18	320 × 1024	28.8	15	**0.101**
Monodepth2 [57]	ResNet-50	192 × 640	**16.6**	**15**	0.110
FSRE-Depth [72]	ResNet-50	192 × 640	32.0	29	0.102
SD-SSMDE	ResNet-50	192 × 640	18.6	17	**0.100**
SD-SSMDE	ResNet-50	320 × 1024	44.3	22	0.098

Table 7.10: Inference time and MACs for the SD-SSMDE depth network. Time is measured on a NVIDIA Tesla V100 GPU.

depth-sensing systems such as LiDAR or stereo cameras. Additionally, monocular depth networks can complement other sensors by providing depth predictions. Their lightweight and scalable architectures make them ideal for real-time deployment on resource-constrained platforms, enhancing overall perception systems in autonomous vehicles.

Chapter 8

Depth-aware Video Panoptic Segmentation

Depth-aware video panoptic segmentation is an emerging task in computer vision that extends traditional panoptic segmentation by incorporating spatial depth information to enhance scene understanding. While conventional video panoptic segmentation focuses on classifying all pixels into semantic and instance categories across temporal sequences, the depth-aware variant introduces the third dimension—depth—to provide richer geometric context. This enables a more comprehensive perception of dynamic environments, particularly in applications like autonomous driving. Given the novelty of this task, the existing literature remains limited. In this chapter two works will be discussed in detail.

8.1 ViP-DeepLab

Method. ViP-DeepLab [127] has recently introduced the task, the metrics, as well as the baseline network. ViP-DeepLab processes concatenated image pairs and extends Panoptic DeepLab [28] with a next-frame instance center offsets decoder and a monocular depth estimation decoder. For instance tracking, a stitching algorithm is proposed, which temporally propagates instance identifiers across instances with significant overlap. Monocular depth estimation is implemented as dense regression [42] and trained in a fully-supervised regime.

The Video Panoptic Segmentation Network in ViP-DeepLab is an extension of Panoptic-DeepLab, originally developed for image-level panoptic segmentation. The network performs semantic segmentation, instance center prediction, and center regression simultaneously. For video sequences, ViP-DeepLab processes two consecutive frames at a time, denoted as frame t and $t+1$. The method predicts the instance center offsets for both frames relative to the object centers in the first frame. This design allows the model to associate pixels belonging to the same object across frames, ensuring temporally consistent segmentation and object tracking. By leveraging center regression to propagate instance information, ViP-DeepLab efficiently handles dynamic scenes in video data. A depiction of the method can be found in Figure 8.1.

The stitching algorithm in ViP-DeepLab plays a crucial role in maintaining temporal consistency of instance IDs across entire video sequences. For each pair of consecutive frames, the outputs are split into left (P_t) and right (R_t) panoptic predictions, corresponding to frame t and $t+1$. To propagate IDs, the algorithm matches instance regions between the two frames based on their mask Intersection over Union (IoU). If regions have the same semantic class and the highest IoU with each other, the instance ID is transferred. Any unmatched regions are treated as new instances. This simple IoU-based stitching approach is robust to object motion and occlusions, enabling ViP-DeepLab to track objects consistently, even when they undergo significant displacement between frames.

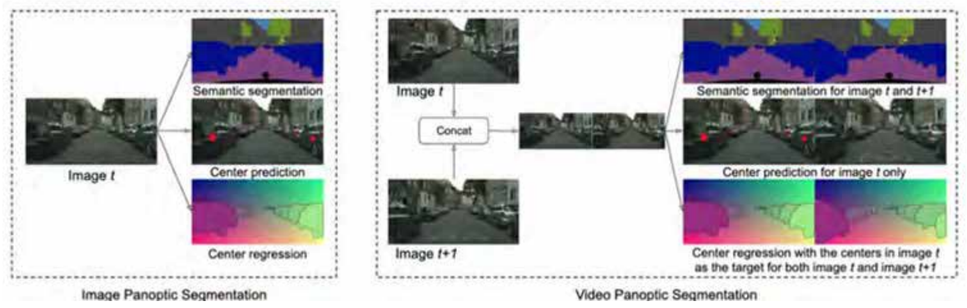

Semantic segmentation
Image t
Center prediction
Center regression
Image Panoptic Segmentation
Image t
Concat
Image $t+1$
Semantic segmentation for image t and $t+1$
Center prediction for image t only
Center regression with the centers in image t as the target for both image t and image $t+1$
Video Panoptic Segmentation

Figure 8.1: Comparing image panoptic segmentation with video panoptic segmentation, ViP-DeepLab builds on the insight that video panoptic segmentation can be effectively represented as a sequence of concatenated image panoptic segmentations. In this approach, center regression serves as an offset map, linking each pixel to its corresponding object center. Image from [127].

The monocular depth estimation task in ViP-DeepLab is framed as a dense regression problem. A depth prediction head is added on top of the semantic segmentation decoder to predict per-pixel depth values. The network learns to estimate depth using a combination of scale-invariant loss and relative squared error, ensuring accurate predictions across varying distances and scales. The depth loss function L_{depth} is defined as follows:

$$L_{\text{depth}}(d, \hat{d}) = \frac{1}{n}\sum_i \left(\log d_i - \log \hat{d}i\right)^2 - \frac{1}{n^2}\left(\sum i \log d_i - \log \hat{d}i\right)^2 + \left(\frac{1}{n}\sum i \left(\frac{d_i - \hat{d}_i}{d_i}\right)^2\right)^{0.5} \tag{8.1}$$

where d_i is the ground-truth depth, $\hat{d}_i$ is the predicted depth, and n is the total number of pixels. The loss combines scale-invariant logarithmic error and relative squared error, encouraging precise and consistent depth predictions.

Experiments. The experiments are conducted on two datasets: Cityscapes-DVPS and SemKITTI-DVPS, as seen in Table 8.1. In the case of Cityscapes-DVPS, the evaluation uses four values of k, namely $k = \{1, 2, 3, 4\}$, as the dataset provides 6 annotated frames per video sequence. For the SemKITTI-DVPS dataset, the evaluation employs larger k values: $k = \{1, 5, 10, 20\}$, which is possible due to the longer video sequences available in this dataset. These larger values allow the evaluation of longer-term temporal consistency. The results show that as the clip length k increases, performance drops are more pronounced on Cityscapes-DVPS compared to SemKITTI-DVPS. For example, the drop in DVPQ for Cityscapes-DVPS from $k = 1$ to $k = 2$ is 7%, whereas the corresponding drop for SemKITTI-DVPS from $k = 1$ to $k = 5$ is only 3.2%. This difference is likely due to the higher annotation frame rate in SemKITTI-DVPS, which simplifies the task of predicting offsets for subsequent frames. Despite the evaluation using larger k values in SemKITTI-DVPS, the temporal consistency of predictions remains relatively stable.

8.2 MonoDVPS

Method. MonoDVPS [122] is a novel network proposed for the DVPS task. To utilize large amounts of unlabeled data, the network em-

DVPQ$_\lambda^k$ on Cityscapes-DVPS	$k = 1$	$k = 2$	$k = 3$	$k = 4$	Average
$\lambda = 0.50$	68.7 | 61.4 | 74.0	61.7 | 48.5 | 71.3	58.4 | 42.1 | 70.2	56.3 | 38.0 | 69.5	61.3 | 47.5 | 71.2
$\lambda = 0.25$	66.5 | 60.4 | 71.0	59.5 | 47.6 | 68.2	56.2 | 41.3 | 67.1	54.2 | 37.3 | 66.5	59.1 | 46.7 | 68.2
$\lambda = 0.10$	50.5 | 45.8 | 53.9	45.6 | 36.9 | 51.9	42.6 | 31.7 | 50.6	40.8 | 28.4 | 49.8	44.9 | 35.7 | 51.5
Average	61.9 | 55.9 | 66.3	55.6 | 44.3 | 63.8	52.4 | 38.4 | 62.6	50.4 | 34.6 | 61.9	**55.1** | **43.3** | **63.6**
DVPQ$_\lambda^k$ on SemKITTI-DVPS	$k = 1$	$k = 5$	$k = 10$	$k = 20$	Average
$\lambda = 0.50$	54.7 | 46.4 | 60.6	51.5 | 41.0 | 59.1	50.1 | 38.5 | 58.5	49.2 | 36.9 | 58.2	51.4 | 40.7 | 59.1
$\lambda = 0.25$	52.0 | 44.8 | 57.3	48.8 | 39.4 | 55.7	47.4 | 37.0 | 55.1	46.6 | 35.6 | 54.7	48.7 | 39.2 | 55.7
$\lambda = 0.10$	40.0 | 34.7 | 43.8	37.1 | 30.3 | 42.0	35.8 | 28.3 | 41.2	34.5 | 26.5 | 40.4	36.8 | 30.0 | 41.9
Average	48.9 | 42.0 | 53.9	45.8 | 36.9 | 52.3	44.4 | 34.6 | 51.6	43.4 | 33.0 | 51.1	**45.6** | **36.6** | **52.2**

Table 8.1: Results of ViP-DeepLab [127]. DVPQ$_\lambda^k$ on Cityscapes-DVPS and SemKITTI-DVPS. Each cell shows DVPQ$_\lambda^k$ — DVPQ$_\lambda^k$-Thing — DVPQ$_\lambda^k$-Stuff. λ is the threshold of relative depth error, and k is the number of frames.

ploys self-supervised training for depth estimation and semi-supervised learning for video panoptic segmentation. In the semi-supervised setting, both labeled and unlabeled data are leveraged to train a better-performing model. Within the complex multi-task training framework, various techniques are introduced to enhance performance across all subtasks. Loss balancing is explored to ensure accuracy improvements for all tasks, while panoptic guidance is utilized to minimize depth estimation errors.

Since self-supervised depth estimation assumes a static scene, moving objects can introduce photometric errors and corrupt the training signal. To overcome this limitation, the proposed approach introduces a novel moving object masking strategy, which relies on panoptic segmentation maps from consecutive frames to identify and exclude moving object pixels from the photometric loss computation. Additionally, three loss terms are introduced to further refine depth prediction by addressing depth discontinuities at panoptic edges. The first is a *panoptic-guided smoothness loss* [134], which enforces depth smoothness for neighboring pixels within panoptic segments. The second is a *panoptic-guided edge discontinuity loss*, designed to enforce significant depth changes at panoptic edges. Finally, the semantic-guided triplet loss [72] is adapted to the panoptic domain for improved depth refinement. The training framework is illustrated in Figure 8.2.

Baseline Network. The MonoDVPs network is built on top of the panoptic image segmentation network Panoptic DeepLab [28], which is extended to handle video data. The network incorporates a shared backbone and dual decoders for semantic and instance segmentation, along with three heads for semantic prediction, instance center

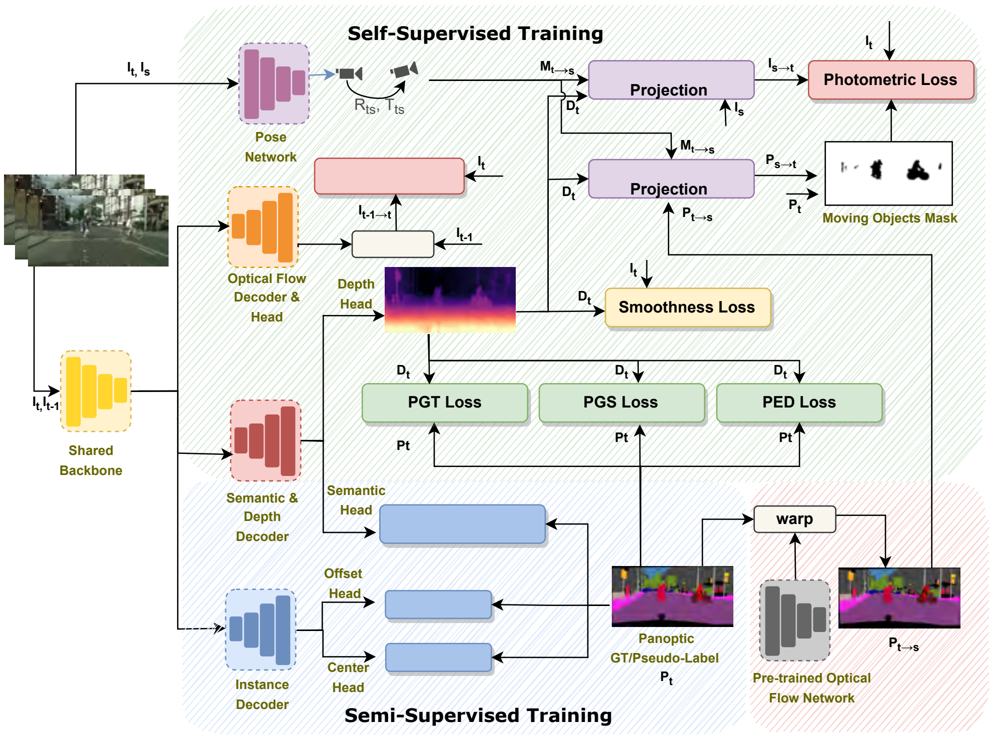
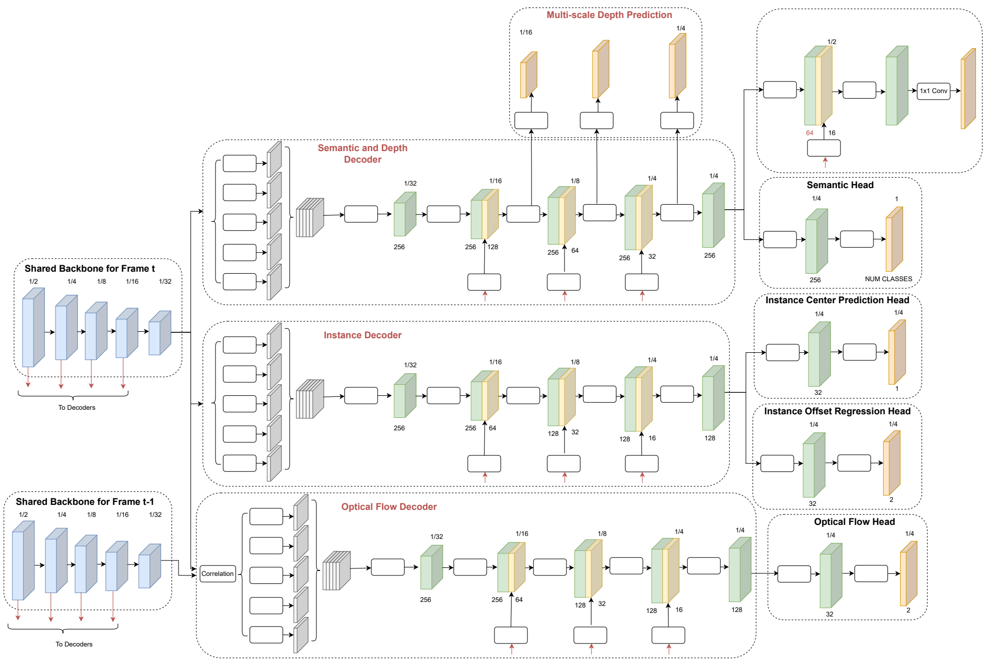


Figure 8.2: The MonoDVPS depth-aware video panoptic segmentation network employs a mixed training regime, where depth, optical flow, and ego motion are trained in a self-supervised manner. Panoptic segmentation is semi-supervised using a combination of ground truth and pseudo-labels. Several loss functions are introduced, including the panoptic-guided triplet loss (PGT), panoptic-guided smoothness loss (PGS), and panoptic-guided edge discontinuity loss (PED), to enhance depth training. Additionally, a novel moving objects mask, computed using panoptic labels, is utilized to mask the photometric loss.

prediction, and instance offset regression. The final panoptic prediction is achieved by grouping class-agnostic foreground pixels to the nearest center based on the predicted offsets to form instances, which are then merged with the semantic segmentation.

The network is further extended with an optical flow decoder, similar to the one used in the VPSTransformer network [121], to facilitate instance tracking. Instance tracking is performed by matching the current instance predictions with warped instance masks from the previous frame using the predicted optical flow. The optical flow decoder is trained in a self-supervised manner by minimizing the photometric loss between the current frame and the warped previous frame. The overall network architecture is illustrated in Figure 8.3.

Self-Supervised Monocular Depth Estimation. The semantic decoder is extended with a depth prediction head, which includes a $[5\times 5, 64]$ depthwise separable convolution, followed by bilinear interpolation, concatenation with low-level features, and $[5\times 5, 32]$ and $[1\times 1, 1]$ convolutions. The approach utilizes multi-scale depth prediction and image reconstruction across four scales with output strides of 2, 4, 8, and 16 relative to the original image resolution. In practice, the network learns the inverse of depth, as it has been shown to be more robust [57]. The minimum reprojection loss and stationary pixels masking techniques from MonoDepth2 [57] are adopted to further enhance the self-supervised depth estimation.



Multi-scale Depth Prediction
Semantic and Depth Decoder
Semantic Head
Instance Decoder
Instance Center Prediction Head
Instance Offset Regression Head
Optical Flow Decoder
Optical Flow Head
Shared Backbone for Frame t
Shared Backbone for Frame t-1
To Decoders
Correlation
1x1 Conv
NUM CLASSES

Figure 8.3: The MonoDVPS network architecture.

Semi-supervised Panoptic Segmentation. The Panoptic DeepLab [28] image panoptic segmentation network with HRNet-W48 [148] is utilized to generate pseudo-labels for the unlabeled data in the Cityscapes-DVPS [127] train set. The initial train set includes human-annotated labels for every fifth frame in a 30-frame video sequence. Following the approach of Naive-Student [20], test-time augmentations, such as horizontal flips and multi-scale inputs with scales ranging from 0.5 to 2.0 at intervals of 0.25, are applied to enhance the quality of pseudo-label predictions. For Cityscapes-DVPS images, pixels belonging to the ego-car are labeled as void and ignored during training.

Improving Depth with Panoptic Guidance. Two primary mechanisms are proposed to enhance the performance of depth estimation with panoptic guidance. First, based on the observation that panoptic segmentation edges align strongly with depth map edges, three panoptic-guided losses are introduced. Second, motion masks are generated using consecutive panoptic labels and applied to the photometric loss to refine the training signal. Figure 8.4 presents visual results of the proposed panoptic-guided mechanisms.

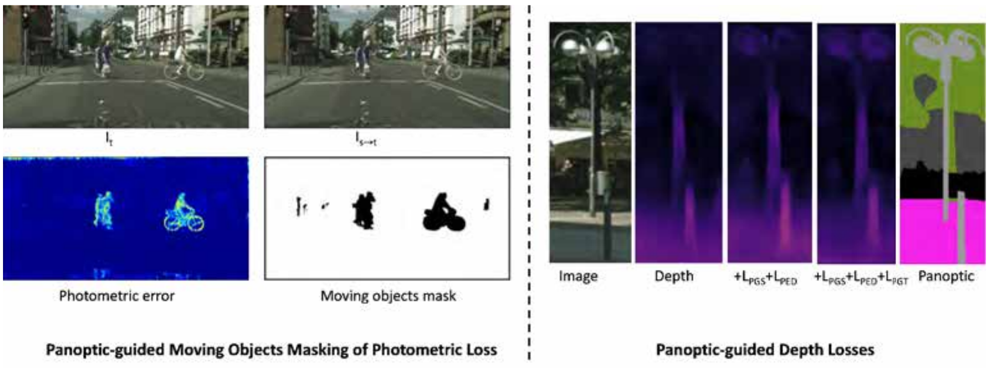

I_t
$I_{s\rightarrow t}$
Photometric error
Moving objects mask
Panoptic-guided Moving Objects Masking of Photometric Loss
Image
Depth
$+L_{PGS}+L_{PED}$
$+L_{PGS}+L_{PED}+L_{PGT}$
Panoptic
Panoptic-guided Depth Losses

Figure 8.4: Moving objects masking and panoptic-guided losses. On the left is illustrated the high photometric loss for moving objects which corrupts the training signal and the moving objects mask. On the right, panoptic-guided depth losses improve the depth prediction.

Panoptic-guided Smoothness Loss. A smoothness loss term [134] is adopted to enforce similar depth values for adjacent pixels within a panoptic segment. This loss is derived from $\mathcal{L}_s$, as defined in equation 7.21, which assumes depth smoothness unless an image edge is present. However, many image edges do not correspond to depth edges, leading to potential misalignment. In contrast, depth and panoptic edges exhibit strong alignment. A panoptic edge is identified where a change in the panoptic identifier occurs between adjacent locations. Based on this observation, the depth differences for adjacent locations within a panoptic segment are minimized. The panoptic-guided smoothness loss is defined as follows:

$$\mathcal{L}_{pgs} = |\partial_x \hat{d}_t|(1 - \partial_x P_t) + |\partial_y \hat{d}_t|(1 - \partial_y P_t) \tag{8.2}$$

where P_t represents the panoptic ground truth label, ∂P_t are the panoptic edges, $\hat{d}_t$ is the mean normalized inverse depth and $\partial_x \hat{d}_t$ is the depth gradient. The latter is computed as the depth difference on the x and y axis for adjacent pixels. For two adjacent pixels (p_0, p_1), the $\partial_x P_t(p_0, p_1)$ is defined as the Iverson bracket, which is equal to 1 when the adjacent pixels have different panoptic identifiers and 0 otherwise:

$$\partial_x P_t(p_0, p_1) = [P(p_0) \neq P(p_1)] \tag{8.3}$$

Panoptic-guided Edge Discontinuity Loss. Based on the observation that adjacent pixels across the panoptic edge may have large depth discontinuities, the following panoptic-guided edge discontinuity term is introduced:

$$\mathcal{L}_{ped} = \partial_x P_t e^{-|\partial_x \hat{d}_t|} + \partial_y P_t e^{-|\partial_y \hat{d}_t|} \tag{8.4}$$

This loss enforces a gradient peak in the depth map at panoptic edges, where different panoptic identifiers exist for adjacent pixels. The exponential function is minimized to ensure large depth differences between adjacent pixels along the panoptic edge. The panoptic-guided edge discontinuity loss extends the work of [164] from the semantic domain to the panoptic domain. While this loss shares similarities with the panoptic-guided alignment approach proposed by [134], the latter is designed for single-task depth networks and does not operate in the context of a multi-task panoptic and depth network.

Panoptic-guided Triplet Loss. The semantic-guided triplet loss [72] is extended to the panoptic domain. This loss is based on the idea that pixels across panoptic edges should exhibit large depth differences. The original formulation using semantic edges [72] has limitations, as instances with the same semantic class are grouped into one segment, causing the absence of edges between instances. In contrast, panoptic maps contain instance edges, which are better aligned with depth edges.

The triplet loss is formulated as follows: the panoptic segmentation map is divided into 5×5 patches, and patches that do not intersect panoptic edges are discarded. For the remaining patches, a triplet loss is defined and applied in the feature representation space on normalized depth feature maps at four scales before the final $[1 \times 1, 1]$ convolution. Features within each patch are categorized into three classes: anchor, positive (P_i^+), and negative (P_i^-). The anchor is located at the center of the patch, while positive features share the same panoptic label as the anchor, and negative features have a different panoptic label. The triplet loss increases the L2 distance d_i^- between the anchor and the negative features while reducing the L2 distance d_i^+ between the anchor and the positive features within a patch. The triplet loss, with a margin m, is defined as follows:

$$\mathcal{L}_{pgt} = \max(0, m + d_i^+ - d_i^-) \tag{8.5}$$

$$d_i^s = \frac{1}{|P_i^s|} \sum_{j \in P_i^s} \sqrt{(F_d(i) - F_d(j))^2}, s \in \{+, -\} \tag{8.6}$$

where $F_d(i)$ is the depth feature of the anchor, $F_d(j)$ is the depth feature of positive or negative depth features inside a patch.

By minimizing the triplet margin loss, the distance between positive and negative features is ensured to be larger than the margin m.

Panoptic-guided Motion Masking. Self-supervised depth estimation assumes a static scene, modeling only the ego motion. However, this assumption does not account for object motion, which corrupts the training signal with artificially high photometric loss on moving objects. To address this issue, a novel scheme is proposed to detect moving objects using the panoptic labels of consecutive frames, where instance identifiers are temporally consistent. The goal is to define a moving object mask, which assigns a value of 0 to regions containing potentially moving objects in the target frame I_t or the geometrically warped source frames $I_{s \to t}$, and 1 otherwise.

To compute the moving object mask, the panoptic segmentation pseudo ground truth for the target frame is utilized. Since the panoptic pseudo-labels are not temporally consistent, panoptic labels for adjacent source frames are synthesized from the target panoptic label to ensure consistent instance identifiers across frames. This is achieved by employing an external pre-trained optical flow network [169] to warp the target panoptic map P_t to the source $\hat{P}_{t \to s}$. The use of optical flow provides the advantage of modeling both ego and object motion.

An occlusion mask $O_{t \to s} = [\exp(-|I_s - \hat{I}_{t \to s}|) > r]$ is introduced to remove occluded pixels, where $[\cdot]$ represents the Iverson bracket.

Subsequently, the predicted depth and geometric projection model from equation 7.17 are employed to reconstruct the target panoptic map $P_{s\to t}$ using nearest neighbor interpolation. The reconstructed panoptic map is formulated as follows:

$$P_{s\to t} = (O_{t\to s}\hat{P}_{t\to s})\langle p'\rangle \tag{8.7}$$

where p' is the location in the source frame of pixel p in the target frame.

Next, the consistency between the reconstructed panoptic map $P_{s\to t}$ and the true target panoptic map P_t is measured, filtered by the instance masks corresponding to potentially moving object classes. Since the geometric projection model accounts only for ego-motion, a high level of consistency between $P_{s\to t}$ and P_t is expected in static scenes, while reduced consistency is expected for moving objects. Consistency is measured as the intersection over union (IoU) between instance masks with the same panoptic identifier in $P_{s\to t}$ and P_t.

A threshold T is defined for the IoU, such that if the IoU is lower than T, the instance is considered a moving object. Pixel locations corresponding to moving objects are excluded from the photometric loss computation. In practice, optimal results are achieved using a linear scheduling for the threshold T. Instead of a fixed value, an initial threshold of $T = 0.7$ is set, which linearly decreases with each iteration. The intuition behind this approach is that, at the beginning of training, the network focuses on learning from static pixels, but as training progresses, it incorporates more potentially noisy samples to account for potential warping errors.

Two masks are obtained, corresponding to the left and right source images, indicating locations in the target frame and one of the geometrically warped source frames that contain potential moving objects. The steps of this process are described in Algorithm 2.

Losses. During training, the optimization involves nine loss functions. Instead of simply summing the loss terms, which is suboptimal, each loss term is balanced with a weighting factor to control its scale in the final objective:

$$\begin{split}\mathcal{L}_{\text{total}} = \gamma_{\text{depth}}\mathcal{L}_{\text{depth}} + \gamma_{\text{sem}}\mathcal{L}_{\text{sem}} \\ + \gamma_{\text{instance}}\mathcal{L}_{\text{instance}} + \gamma_{\text{optical}}\mathcal{L}_{\text{optical}}\end{split} \tag{8.8}$$

The depth loss is defined as a combination of the photometric

Algorithm 2 Panoptic-guided Motion Masking

```
Input: Panoptic segmentation $P_t$, Target image $I_t$, Source image $I_s$,
    iteration $i$, total number of iterations $e$, threshold $T$
Output: Moving object mask $M_t$
  Compute optical flow $V \leftarrow$ Optical Flow$(I_t, {}_s)$
  Warp $P_t$ using optical flow $V$ to source $\hat{P}_{t\to s}$
  Backproject depth using $x \leftarrow K^{-1}D_t(p)p$
  Displace $x$ using the ego motion to $x' \leftarrow M_{t\to s}x$
  Project $x'$ to the source image $p' \leftarrow Kx'$
  Compute occlusion mask $O_{t\to s} \leftarrow [\exp(-|I_s - \hat{I}_{t\to s}|) > r]$
  Geometrically synthesize the target $P_{s\to t} \leftarrow (O_{t\to s}\hat{P}_{t\to s})\langle p'\rangle$
  for $id \in instanceIDs(P_t)$ do
      if IoU$(P_t(id), P_{s\to t}(id) < T * (1 - i/e)$ then
          Paste $P_t(id)$ and $P_{s\to t}(id)$ in $M_t$
      end if
  end for
```

loss $\mathcal{L}_{\text{photo}}$ from equation 7.18, smoothness loss $\mathcal{L}_{\text{s}}$ from equation 7.21, panoptic-guided smoothness loss $\mathcal{L}_{\text{pgs}}$, panoptic-guided edge discontinuity loss $\mathcal{L}_{\text{ped}}$, and panoptic-guided triplet loss $\mathcal{L}_{\text{pgt}}$:

$$\begin{aligned}\mathcal{L}_{\text{depth}} = \gamma_{\text{photo}}\mathcal{L}_{\text{photo}} + \gamma_{\text{s}}\mathcal{L}_{\text{s}} + \gamma_{\text{pgs}}\mathcal{L}_{\text{pgs}} \\ + \gamma_{\text{ped}}\mathcal{L}_{\text{ped}} + \gamma_{\text{pgt}}\mathcal{L}_{\text{pgt}}\end{aligned} \tag{8.9}$$

Following [28], the instance loss $\mathcal{L}_{\text{instance}}$ is defined as the weighted sum of mean squared error (MSE) for the instance center prediction head and L1 loss for the center offset head. Instance weights are set as in [28].

The weighting factors are set as follows: $\gamma_{\text{sem}} = 1$, $\gamma_{\text{depth}} = 50$, $\gamma_{\text{instance}} = 1$, $\gamma_{\text{optical}} = 10$, $\gamma_{\text{photo}} = 1$, $\gamma_{\text{s}} = 0.001$, $\gamma_{\text{pgs}} = 0.01$, $\gamma_{\text{ped}} = 0.0001$, and $\gamma_{\text{pgt}} = 0.1$. These weights are chosen to ensure that the main losses have similar magnitudes.

Panoptic 4D Point Cloud. The depth output of the network represents relative up-to-scale depth. While depth values within an image are broadly consistent with each other, obtaining metric-scale depth requires multiplying by a scale factor. Additionally, each depth map necessitates a unique scale factor, as the depth maps are not inter-frame scale consistent. To recover the true depth, a common practice

[57] is to perform per-image median scaling: each predicted depth map is scaled using the ratio between the median of the ground truth and the predicted depth values.

Once the depth maps are scaled to real-world values, a panoptic 3D point cloud is generated. To compute the 3D point in the camera coordinate system for each pixel in the image, the depth map is back-projected. Since the panoptic segmentation output is aligned with the depth map, each 3D point is augmented with the panoptic identifier of its corresponding pixel, resulting in the panoptic 4D point cloud.

To eliminate the dependency on ground truth, the scale factor can be computed directly from the predicted depth map. For instance, the scale factor can be determined as the ratio between a known camera height and a computed camera height, where the camera height is derived as the median or average height of all 3D points labeled as road. For network evaluation, the ground truth median scaling method is adopted.

Experiments. The ResNet-50 [64] backbone is adopted for the depth-aware video panoptic segmentation network. The network is pre-trained on the Cityscapes dataset [34] for image panoptic segmentation.

To compute the final depth values, the inverse depth at the highest resolution, corresponding to 1/2 of the original image resolution, is activated by a sigmoid layer σ and converted to depth using the formula $Z = 1/(a\sigma + b)$, where a and b map the depth values to the interval $[0.1, 100]$.

The pose estimation network follows the design in [57], utilizing a ResNet-18 backbone and a decoder to predict the 6DOF camera pose, including the translation vector and rotation matrix represented as Euler angles. The input to the camera pose network consists of pairs of source and target images, and the network is supervised exclusively through the photometric loss. During inference, the pose estimation network is discarded.

Training is conducted with a minibatch size of 4 images for 30k iterations, using the Adam optimizer with a base learning rate of 1e-3 for decoders and heads, and 1e-4 for the backbone, employing a polynomial learning rate decay schedule. Image augmentation techniques, including random horizontal flipping and random color augmentation (brightness, contrast, saturation, and hue jitter), are applied. For Cityscapes-DVPS, an image resolution of 1025×2049 is used, while for SemKITTI-DVPS, the resolution is 385×1281.

Multi-task Learning Ablation. Table 8.2 presents the results of single-task baselines for panoptic segmentation and depth estimation, as well as the performance of the MonoDVPS multi-task network. In the baseline multi-task learning setup, where the depth loss weight $\gamma_{\text{depth}} = 1$, a decrease in accuracy is observed for both panoptic segmentation and depth estimation compared to the single-task baselines. To address this, the loss magnitudes are balanced by setting $\gamma_{\text{depth}} = 50$, which achieves the best PQ and absRel metrics.

To further enhance depth prediction, panoptic-guided losses are incorporated during training, enabling the multi-task network to match the performance of the single-task depth baseline. Since self-supervised depth estimation does not rely on depth ground truth, it can be trained on large-scale unlabeled datasets. In contrast, panoptic segmentation is formulated as a supervised learning problem and requires labeled images.

For the Cityscapes-DVPS dataset, which consists of 30-frame video snippets with only every fifth frame annotated for video panoptic segmentation, the training set is expanded from 2400 to 14100 image-annotation pairs. This is achieved by generating panoptic pseudo-labels for all previously unlabeled frames. Training the network on the extended dataset results in significant performance improvements for both panoptic segmentation and depth estimation.

Model	PQ ↑	absRel ↓
Panoptic only	63.5	-
Depth only	-	0.098
MTL Baseline $\gamma_{depth} = 1$	62.9	0.151
+ Loss Balancing $\gamma_{depth} = 100$	63.2	0.102
+ Loss Balancing $\gamma_{depth} = 50$	63.6	0.102
+ Panoptic-guided depth	63.6	0.098
+ Extended train set	**66.5**	**0.082**

Table 8.2: Multi-task Learning (MTL). Comparison between single task and several multi-task training settings.

Panoptic-guided Depth Ablation. Table 8.3 presents an extensive ablation study on depth estimation. The study first compares depth estimation trained in a supervised versus self-supervised regime within a multi-task learning setting. For supervised training, depth estimation is formulated as a regression problem, utilizing the scale-invariant log loss from [42]. As anticipated, supervised depth estimation outperforms self-supervised depth estimation across all metrics.

To narrow this performance gap, several enhancements to the self-supervised training process are proposed. The multi-task loss is balanced by increasing the depth loss weight, and panoptic-guided losses $\mathcal{L}_{\text{PGS}}$, $\mathcal{L}_{\text{PED}}$, and $\mathcal{L}_{\text{PGT}}$ are introduced to reduce depth error. Additionally, a moving object masking scheme is designed to avoid corrupting the training signal in regions containing moving objects, further improving performance.

Finally, the training set is expanded from 2400 to 14410 frames by incorporating additional data. This extension significantly reduces the depth error, demonstrating the critical importance of a large dataset for effective self-supervised depth training.

Model	absRel ↓	sqRel ↓	RMS ↓
Self-Supervised Depth Only	0.098	0.731	4.919
MTL Supervised Depth	0.070	0.368	3.675
MTL Self-Supervised Depth	0.106	0.841	5.270
+ Loss Balancing	0.102	0.767	5.034
+ $\mathcal{L}_{PGS}$ + $\mathcal{L}_{PED}$ + $\mathcal{L}_{PGT}$	0.099	0.747	4.988
+ Moving Objects Masking	0.098	0.701	4.864
+ Extended dataset	**0.082**	**0.515**	**4.198**

Table 8.3: Panoptic-guided depth evaluation in a multi-task setting. Ablation study for loss balancing, panoptic-guided depth losses and moving objects masking.

Moving Object Masking for Improved Depth Ablation. For each instance in frame t, the IoU is calculated between its mask in the reconstructed panoptic label $P_{s \to t}$ and its mask in P_t. The geometric projection model used to generate $P_{s \to t}$ assumes a static scene and accounts only for ego-motion. Consequently, a high overlap is observed between instance masks for static objects, whereas moving objects exhibit low overlap due to unmodeled object motion.

A threshold T is defined such that instances with an IoU below this threshold are classified as moving objects, and the corresponding pixels are excluded from the photometric loss computation. Table 8.4 reports experimental results using $T = \{0.3, 0.5, 0.7\}$ and a linear scheduling approach. Errors arising from optical flow warping, geometric reconstruction, or occlusions can affect the IoU computation. As a result, a high threshold ($T = 0.7$) removes too many instances, while a low threshold ($T = 0.3$) is overly permissive. Linear scheduling achieves the best balance between panoptic and depth performance, with $T = 0.5$ providing comparable results.

IoU threshold	PQ ↑	absRel ↓
0.3	63.2	0.099
0.5	63.5	0.098
0.7	63.9	0.102
linear	**63.6**	**0.098**

Table 8.4: Ablation study on the IoU threshold used to determine if an object is moving. *Linear* means that the IoU is decreased linearly from 0.7 with each training iteration.

Depth-aware Video Panoptic Segmentation. Depth-aware video panoptic segmentation (DVPQ) results on Cityscapes-DVPS are reported in Table 8.5. As expected, DVPQ decreases with larger temporal window sizes k due to reduced temporal consistency, and with lower thresholds λ for depth absRel. A greater performance drop is observed in DVPQ-Things compared to DVPQ-Stuff when decreasing λ across all k, suggesting that depth errors are more pronounced on instance pixels than on *stuff* pixels.

MonoDVPS is also trained in a fully supervised regime for both depth and video panoptic segmentation (MonoDVPS S-MDE), resulting in higher DVPQ compared to the multi-task network trained under a self-supervised depth regime (MonoDVPS Average). Relative to MonoDVPS S-MDE, depth errors are higher for instance pixels and lower for *stuff* pixels, as indicated by the lower DVPQ-Things and higher DVPQ-Stuff metrics.

In Table 8.6, a comparison is presented between the MonoDVPS network trained on the original and extended training sets, and the concurrent ViP-DeepLab [9] on the DVPS task. When trained on the original Cityscapes-DVPS training set, the MonoDVPS network achieves a DVPQ score of 43.4. By utilizing the extended dataset with panoptic pseudo-labels, the DVPQ score improves to 48.8. The MonoDVPS network surpasses ViP-DeepLab [9] with the ResNet-50 backbone in terms of DVPQ, while also demonstrating faster performance.

DVPQ^k_λ on Cityscapes-DVPS	k = 1	k = 2	k = 3	k = 4	DVPQ Average
MonoDVPS $\lambda = 0.50$	65.9 | 55.7 | 73.3	59.0 | 43.0 | 70.6	55.8 | 36.9 | 69.5	53.5 | 32.5 | 68.8	58.6 | 42.0 | 70.6
MonoDVPS $\lambda = 0.25$	59.3 | 45.4 | 69.4	53.0 | 34.2 | 66.7	50.2 | 28.9 | 65.7	48.5 | 26.1 | 64.7	52.8 | 33.7 | 66.7
MonoDVPS $\lambda = 0.10$	39.0 | 23.7 | 50.0	35.1 | 17.5 | 47.9	33.4 | 14.5 | 47.1	32.5 | 13.1 | 46.6	35.0 | 17.2 | 48.0
MonoDVPS Average	54.7 | 41.6 | 64.2	49.0 | 31.6 | 61.7	46.5 | 26.8 | 60.8	44.8 | 23.9 | 60.0	**48.8** | **31.0** | **61.7**
MonoDVPS S-MDE	57.2 | 48.4 | 63.6	51.0 | 37.0 | 61.0	47.9 | 31.0 | 60.0	45.7 | 27.0 | 59.3	**50.4** | **35.9** | **61.0**
ViP-DeepLab (WR-41) [127]	61.9 | 55.9 | 66.3	55.6 | 44.3 | 63.8	52.4 | 38.4 | 62.6	50.4 | 34.6 | 61.9	**55.1** | **43.3** | **63.6**
ViP-DeepLab* (ResNet-50) [9]	47.4 | 38.8 | 53.7	44.0 | 28.1 | 51.6	39.0 | 23.3 | 50.5	37.5 | 20.2 | 50.0	42.0 | 27.6 | 51.5

DVPQ^k_λ on SemKITTI-DVPS	k = 1	k = 5	k = 10	k = 20	DVPQ Average
MonoDVPS $\lambda = 0.50$	48.7 | 44.7 | 51.7	43.0 | 33.3 | 50.0	41.6 | 30.7 | 49.6	40.4 | 28.4 | 49.2	43.4 | 34.2 | 50.1
MonoDVPS $\lambda = 0.25$	45.3 | 39.7 | 49.4	39.8 | 28.9 | 47.8	38.5 | 26.7 | 47.2	37.6 | 25.0 | 46.8	40.3 | 30.0 | 47.8
MonoDVPS $\lambda = 0.10$	35.9 | 28.0 | 41.6	31.6 | 20.0 | 40.0	30.6 | 18.4 | 39.4	29.8 | 17.3 | 39.0	32.0 | 21.0 | 40.0
MonoDVPS Average	43.3 | 37.5 | 47.5	38.1 | 27.4 | 46.0	36.9 | 25.2 | 45.4	36.0 | 23.6 | 45.0	**38.6** | **28.4** | **46.0**
ViP-DeepLab [127] (WR-41)	48.9 | 42.0 | 53.9	45.8 | 36.9 | 52.3	44.4 | 34.6 | 51.6	43.4 | 33.0 | 51.1	**45.6** | **36.6** | **52.2**

Table 8.5: Depth-aware video panoptic segmentation on Cityscapes-DVPS and SemKITTI-DVPS. Each cell shows DVPQ^k_λ\| DVPQ^k_λ-Things \| DVPQ^k_λ-Stuff. k is the number of frames and λ is the threshold of relative depth error. MonoDVPS S-MDE is the network trained in a fully supervised regime for both panoptic and depth. All networks use the ResNet-50 backbone. ViP-DeepLab uses the heavier WR-41 backbone, Mapillary Vistas pretraining and test-time augmentations. ViP-DeepLab* with the ResNet-50 backbone is evaluated using the author's code and pretrained model.

Model	Backbone	DVPQ	DVPQ-Things	DVPQ-Stuff	Time (s)
MonoDVPS	ResNet-50	**48.8**	**31.0**	**61.7**	**0.11**
MonoDVPS*	ResNet-50	43.4	26.2	55.9	0.11
ViP-DeepLab [9]	ResNet-50	42.0	27.6	51.5	0.18

Table 8.6: DVPS evaluation on Cityscapes-DVPS. MonoDVPS* is the network trained on the reduced training set (without extension). ViP-DeepLab with ResNet-50 was evaluated with the author's code [9]. Time is measured on a NVIDIA Tesla V100 GPU.

Video Panoptic Segmentation. Table 8.7 compares the MonoDVPS network with state-of-the-art methods for video panoptic segmentation. ViP-DeepLab [127], utilizing the WR-41 [20] backbone and pretrained on Mapillary Vistas [108], is designed for accuracy and achieves state-of-the-art performance across all metrics. However, ViP-DeepLab's reliance on costly test-time augmentations results in a slow inference speed of 54 seconds per frame on a NVIDIA Tesla V100 GPU. When evaluated with the same ResNet-50 backbone, the proposed MonoDVPS network surpasses all other methods, including ViP-DeepLab, in terms of both VPQ and inference time.

Model	Backbone	k = 1	k = 2	k = 3	k = 4	VPQ ↑	Time (s)
VPSNet [74]	ResNet-50	62.7	56.9	53.3	51.3	56.1	0.77
Siam-Track [155]	ResNet-50	64.6	57.6	54.2	52.7	57.3	0.22
VPS-Transformer [121]	ResNet-50	64.8	57.6	54.4	52.2	57.3	0.11
ViP-DeepLab* [9]	ResNet-50	60.6	53.1	49.9	47.7	52.8	0.17
ViP-DeepLab [127]	WR-41	**70.4**	**63.6**	**60.1**	**58.1**	**63.1**	54*
MonoDVPS (ours)	ResNet-50	66.5	59.6	56.3	54.0	59.1	0.10

Table 8.7: Video panoptic segmentation on Cityscapes-DVPS. k is the window size used for evaluation. In this paper $k = \{1, 2, 3, 4\}$ is equivalent to $k = \{1, 5, 10, 15\}$ from [74, 155, 121]. ViP-DeepLab* is evaluated with the author's code. Inference time is measured on a Tesla V100.

Qualitative Results. Figure 8.5 presents panoptic and depth visualizations for two consecutive frames. The proposed approach demonstrates overall strong qualitative results on both the Cityscapes-DVPS and SemKITTI-DVPS datasets.

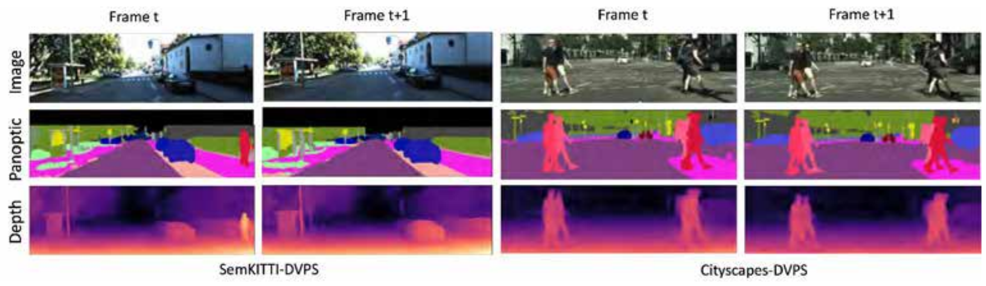


Figure 8.5: Qualitative Results. Video panoptic and depth predictions on SemKITTI-DVPS and Cityscapes-DVPS.

SemKITTI-DVPS Results. Table 8.5 reports the evaluation of the MonoDVPS network with self-supervised depth on the SemKITTI-DVPS dataset. While ViP-DeepLab, utilizing the WR-41 backbone and test-time augmentations, achieves superior results, the proposed network would similarly benefit from a heavier backbone and these computationally expensive operations.

Compared to the results on the Cityscapes-DVPS dataset, smaller drops in DVPQ_k^λ are observed as the absolute relative depth threshold λ decreases on SemKITTI-DVPS. For instance, the difference $\text{DVPQ}_1^{0.5}$ – $\text{DVPQ}_1^{0.25}$ is 6.6% on Cityscapes-DVPS, while it is 3.4% on

SemKITTI-DVPS. This effect is even more pronounced for smaller values of λ. A possible explanation for Cityscapes-DVPS being more sensitive to λ is its increased complexity, with a larger number of instances per image and more challenging scenarios, resulting in higher depth errors on instances compared to the background.

Chapter 9

Conclusions

Creating an advanced system for autonomous driving requires development and research of key technologies: robust surround view environment perception, accurate life-long metric localization and mapping, scene understanding, motion prediction, planning and navigation. This book has explored the critical role of visual perception in enabling autonomous vehicles to navigate complex environments. Deep learning has revolutionized visual environment perception, enabling robust and accurate identification of objects, semantic regions, and measuring depth from a single image. Tasks like semantic segmentation, instance segmentation, panoptic image and video segmentation and depth estimation have demonstrated their effectiveness in extracting contextual and spatial details essential for building a 3D or even 4D environment representation, across space and time, which is further used for navigation. We have also discussed the challenges that arrise with these tasks related to the integration of such algorithms in an autonomous vehicle. As such, the perception system of the automated vehicle needs to be robust and accurate and should run in real-time on the limited hardware resources available on the vehicle. For each individual task, we review the literature and describe in more detail important methods, with a focus on methods proposed in the author's PhD thesis ”Deep Learning-based Visual Perception for Autonomous Driving”.

In Chapter 3, semantic segmentation is explored comprehensively, emphasizing its role as a foundational task in visual perception for autonomous vehicles. The chapter delves into various approaches, including traditional fully convolutional networks and more advanced architectures such as ERFNet [131] and DeepLabV3+ [24], which excel in balancing computational efficiency with accuracy. Additionally, it introduces transformer-based networks, which represent a significant leap in the field by leveraging self-attention mechanisms to model long-range dependencies and global context.

In Chapter 4, we explore instance segmentation, an advanced perception task that assigns unique identifiers to objects within an image while also assigning pixel-level semantic masks. We highlight the two dominant approaches: top-down methods, which rely on object detection frameworks, and bottom-up methods, which segment and cluster pixels based on similarities. Two top-down methods are discussed in detail: Mask R-CNN [63] which is a two-stage, very accurate but slower instance segmentation network and RetinaMask [46] which is a one-stage network with faster inference speed, but slightly lower accuracy. The RetinaMask network has been also trained and tested on fisheye images from a 360° camera system. From our findings, the detection range for pedestrians on fisheye images is up to 20 meters due to the distortions introduced by the lens. That is why it is important to implement a solution that combines fisheye cameras and narrow field-of-view cameras especially for the front area, in order to extend the detection range.

In Chapter 5, we discuss panoptic segmentation, a task that integrates semantic segmentation and instance segmentation into a unified framework. This task enhances the ability of perception systems to simultaneously classify amorphous regions (e.g., roads, sky) and individual objects (e.g., vehicles, pedestrians) while also identifying them. Panoptic segmentation can be solved using two separate tasks: semantic segmentation and instance segmentation. Since the semantic and instance segmentation networks process the images on separate paths, their results are independent and might be mismatched in terms of pixel-level semantic labels. A solution to this problem is a fusion scheme [35] between the instance and semantic output, which ensures a unique semantic and instance label per pixel and increases the accuracy of the results at the same time. Moreover, end-to-end one-stage panoptic segmentation networks are the most suitable for deployment in autonomous vehicles, because they meet the requirements of fast inference speed and high accuracy. Also, it is important from a deployment point of view, to enable the use of acceleration engines such as TensorRT and to design fully-convolutional networks. Two one-stage networks are discussed, Panoptic Prototype [117], which proposes the automatic learning of prototype masks guided by object detections and AttentionPS [119] in which a novel panoptic head is introduced: object detections along with the instance center offsets are used for generating instance-specific soft attention maps. The panoptic output is obtained by applying the attention maps to the semantic segmentation. From a practical point of view, both networks are fast and can be further optimized to run in more than 30 frames per second on less powerful GPUs, which can be installed on a vehicle. We also discuss a fast bottom-up panoptic segmentation network, Panoptic-DeepLab [28], which formulates the task as semantic segmentation and instance segmentation as offset regression to predicted object centers. A more accurate but slow alternative represents the ISS-Fusion [35] network which extends the instance segmentation network Mask R-CNN [63] with a semantic segmentation branch, and proposes a fusion algorithm between the two ouputs as a post-processing step.

In Chapter 6, we start with an overview of video panoptic segmentation networks, which extend the panoptic segmentation task with instance ID tracking across frames. Two networks are discussed: VPSNet [74] and VPS-Transformer [121], the latter introducing a spatio-temporal transformer module inside a convolutional network and a optical-flow based instance ID tracking branch for improved performance, while keeping the processing time low.

In Chapter 7, we discuss monocular depth estimation methods. In the overview section, both supervised and self-supervised approaches are reviewed. Self-supervised methods jointly learn depth and camera pose from three consecutive frames. Based on geometric assumptions, the current frame can be synthesized from the adjacent frames, and a photometric loss is minimized. The major advantage of self-supervised methods is that they do not need ground truth data, only the raw images. The focus in this chapter is on self-supervision and the method MonoDepth2 [57] and the self-distillation framework SD-SSMDE [118] are discussed in detail.

In Chapter 8, we present two methods for depth-aware video panoptic segmentation: ViP-DeepLab [9] and MonoDVPS [122]. This task extends video panoptic segmentation by incorporating depth estimation. While ViP-DeepLab integrates a supervised depth estimation branch, MonoDVPS employs self-supervised depth estimation.

This work highlights how individual components like segmentation, depth estimation, and video analysis coalesce into a unified framework for perception. Together, these tasks contribute to a semantic 4D understanding of the environment, laying the foundation for more intelligent planning, navigation, and interaction capabilities for autonomous driving.

While significant progress has been achieved, the field of autonomous driving is far from reaching its pinnacle. Future research should explore the integration of multi-modal perception systems, leveraging data from LiDAR, radar, and cameras to achieve redundancy and robustness under adverse conditions. Additionally, advancements in unsupervised and semi-supervised learning could reduce dependency on large labeled datasets, enabling faster adoption in diverse scenarios.

Bibliography

[1] Automotive data and time-triggered framework. https://www.elektrobit.com/products/automated-driving/eb-assist/adtf/.

[2] Mobility report. https://static.tti.tamu.edu/tti.tamu.edu/documents/mobility-report-2021.pdf.

[3] Nhtsa. /urlhttps://www.nhtsa.gov/.

[4] Nvidia cuda® deep neural network library (cudnn). /url-https://developer.nvidia.com/cudnn.

[5] Onnx. https://onnx.ai/.

[6] Tensorrt. https://developer.nvidia.com/tensorrt.

[7] United nations. https://www.un.org/uk/desa/68-world-population-projected-live-urban-areas-2050-says-un.

[8] Urban parking and driving h2020 european project (up-drive). https://up-drive.ethz.ch/.

[9] Vip-deeplab. https://github.com/google-research/deeplab2.

[10] Who road traffic injuries. https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries.

[11] Anurag Arnab and Philip HS Torr. Pixelwise instance segmentation with a dynamically instantiated network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 441–450, 2017.

[12] Amir Atapour-Abarghouei and Toby P Breckon. Real-time monocular depth estimation using synthetic data with domain adaptation via image style transfer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2800–2810, 2018.

[13] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[14] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12):2481–2495, 2017.

[15] Min Bai and Raquel Urtasun. Deep watershed transform for instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5221–5229, 2017.

[16] Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Cyrill Stachniss, and Jurgen Gall. Semantickitti: A dataset for semantic scene understanding of lidar sequences. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9297–9307, 2019.

[17] Daniel Bolya, Chong Zhou, Fanyi Xiao, and Yong Jae Lee. Yolact: Real-time instance segmentation. In *Proceedings of the IEEE international conference on computer vision*, pages 9157–9166, 2019.

[18] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6154–6162, 2018.

[19] Vincent Casser, Soeren Pirk, Reza Mahjourian, and Anelia Angelova. Unsupervised monocular depth and ego-motion learning with structure and semantics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019.

[20] Liang-Chieh Chen, Raphael Gontijo Lopes, Bowen Cheng, Maxwell D Collins, Ekin D Cubuk, Barret Zoph, Hartwig Adam, and Jonathon Shlens. Naive-student: Leveraging semi-supervised

learning in video sequences for urban scene segmentation. In *European Conference on Computer Vision*, pages 695–714. Springer, 2020.

[21] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017.

[22] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.

[23] Liang-Chieh Chen, Yi Yang, Jiang Wang, Wei Xu, and Alan L Yuille. Attention to scale: Scale-aware semantic image segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3640–3649, 2016.

[24] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818, 2018.

[25] Weifeng Chen, Zhao Fu, Dawei Yang, and Jia Deng. Single-image depth perception in the wild. *Advances in Neural Information Processing Systems*, 29:730–738, 2016.

[26] Xinlei Chen, Ross Girshick, Kaiming He, and Piotr Dollár. Tensormask: A foundation for dense object segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2061–2069, 2019.

[27] Yuhua Chen, Cordelia Schmid, and Cristian Sminchisescu. Self-supervised learning with geometric constraints in monocular video: Connecting flow, depth, and camera. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7063–7072, 2019.

[28] Bowen Cheng, Maxwell D Collins, Yukun Zhu, Ting Liu, Thomas S Huang, Hartwig Adam, and Liang-Chieh Chen. Panoptic-deeplab:

A simple, strong, and fast baseline for bottom-up panoptic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12475–12485, 2020.

[29] Bowen Cheng, Ishan Misra, Alexander G Schwing, Alexander Kirillov, and Rohit Girdhar. Masked-attention mask transformer for universal image segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1290–1299, 2022.

[30] Bowen Cheng, Alex Schwing, and Alexander Kirillov. Per-pixel classification is not all you need for semantic segmentation. *Advances in neural information processing systems*, 34:17864–17875, 2021.

[31] Hyesong Choi, Hunsang Lee, Sunkyung Kim, Sunok Kim, Seungryong Kim, Kwanghoon Sohn, and Dongbo Min. Adaptive confidence thresholding for monocular depth estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12808–12818, 2021.

[32] Hyesong Choi, Hunsang Lee, Sunkyung Kim, Sunok Kim, Seungryong Kim, Kwanghoon Sohn, and Dongbo Min. Adaptive confidence thresholding for monocular depth estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 12808–12818, October 2021.

[33] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258, 2017.

[34] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016.

[35] Arthur Daniel Costea, Andra Petrovai, and Sergiu Nedevschi. Fusion scheme for semantic and instance-level segmentation. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 3469–3475. IEEE, 2018.

[36] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 764–773, 2017.

[37] Daan de Geus, Panagiotis Meletis, and Gijs Dubbelman. Fast panoptic segmentation network. *IEEE Robotics and Automation Letters*, 5(2):1742–1749, 2020.

[38] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[39] Raul Diaz and Amit Marathe. Soft labels for ordinal regression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4738–4747, 2019.

[40] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2758–2766, 2015.

[41] David Eigen and Rob Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Proceedings of the IEEE international conference on computer vision*, pages 2650–2658, 2015.

[42] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. *arXiv preprint arXiv:1406.2283*, 2014.

[43] Heng Fan, Xue Mei, Danil Prokhorov, and Haibin Ling. Multi-level contextual rnns with attention model for scene labeling. *IEEE Transactions on Intelligent Transportation Systems*, 19(11):3475–3485, 2018.

[44] Clement Farabet, Camille Couprie, Laurent Najman, and Yann LeCun. Learning hierarchical features for scene labeling. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1915–1929, 2012.

[45] Horatiu Florea, Andra Petrovai, Ion Giosan, Florin Oniga, Robert Varga, and Sergiu Nedevschi. Enhanced perception for autonomous driving using semantic and geometric data fusion. *Sensors*, 22(13):5061, 2022.

[46] Cheng-Yang Fu, Mykhailo Shvets, and Alexander C Berg. Retinamask: Learning to predict masks improves state-of-the-art single-shot detection for free. *arXiv preprint arXiv:1901.03353*, 2019.

[47] Huan Fu, Mingming Gong, Chaohui Wang, Kayhan Batmanghelich, and Dacheng Tao. Deep ordinal regression network for monocular depth estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2002–2011, 2018.

[48] Naiyu Gao, Yanhu Shan, Yupei Wang, Xin Zhao, Yinan Yu, Ming Yang, and Kaiqi Huang. Ssap: Single-shot instance segmentation with affinity pyramid. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 642–651, 2019.

[49] Ravi Garg, Vijay Kumar Bg, Gustavo Carneiro, and Ian Reid. Unsupervised cnn for single view depth estimation: Geometry to the rescue. In *European Conference on Computer Vision*, pages 740–756. Springer, 2016.

[50] A Geiger, P Lenz, and R Urtasun. Are we ready for autonomous driving? *The kitti vision benchmark suite, "in IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.

[51] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.

[52] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3354–3361. IEEE, 2012.

[53] Golnaz Ghiasi and Charless C Fowlkes. Laplacian pyramid reconstruction and refinement for semantic segmentation. In *European conference on computer vision*, pages 519–534. Springer, 2016.

[54] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[55] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

[56] Clément Godard, Oisin Mac Aodha, and Gabriel J Brostow. Unsupervised monocular depth estimation with left-right consistency. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 270–279, 2017.

[57] Clément Godard, Oisin Mac Aodha, Michael Firman, and Gabriel J. Brostow. Digging into self-supervised monocular depth prediction. In *The International Conference on Computer Vision (ICCV)*, 2019.

[58] Juan Luis GonzalezBello and Munchurl Kim. Forget about the lidar: Self-supervised depth estimators with med probability volumes. *Advances in Neural Information Processing Systems*, 33:12626–12637, 2020.

[59] Ariel Gordon, Hanhan Li, Rico Jonschkowski, and Anelia Angelova. Depth from videos in the wild: Unsupervised monocular depth learning from unknown cameras. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8977–8986, 2019.

[60] Vitor Guizilini, Rares Ambrus, Sudeep Pillai, Allan Raventos, and Adrien Gaidon. 3d packing for self-supervised monocular depth estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2485–2494, 2020.

[61] Vitor Guizilini, Rui Hou, Jie Li, Rares Ambrus, and Adrien Gaidon. Semantically-guided representation learning for self-supervised monocular depth. In *International Conference on Learning Representations*, 2020.

[62] Xiaoyang Guo, Hongsheng Li, Shuai Yi, Jimmy Ren, and Xiaogang Wang. Learning monocular depth by distilling cross-domain stereo networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 484–500, 2018.

[63] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[64] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[65] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

[66] Heiko Hirschmuller. Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2):328–341, 2007.

[67] Rui Hou, Jie Li, Arjun Bhargava, Allan Raventos, Vitor Guizilini, Chao Fang, Jerome Lynch, and Adrien Gaidon. Real-time panoptic segmentation from dense detections. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8523–8532, 2020.

[68] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[69] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[70] Samvit Jain, Xin Wang, and Joseph E Gonzalez. Accel: A corrective fusion network for efficient semantic segmentation on video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8866–8875, 2019.

[71] Adrian Johnston and Gustavo Carneiro. Self-supervised monocular trained depth estimation using self-attention and discrete disparity volume. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4756–4765, 2020.

[72] Hyunyoung Jung, Eunhyeok Park, and Sungjoo Yoo. Fine-grained semantics-aware representation enhancement for self-supervised monocular depth estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12642–12652, 2021.

[73] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7482–7491, 2018.

[74] Dahun Kim, Sanghyun Woo, Joon-Young Lee, and In So Kweon. Video panoptic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9859–9868, 2020.

[75] Alexander Kirillov, Ross Girshick, Kaiming He, and Piotr Dollár. Panoptic feature pyramid networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6399–6408, 2019.

[76] Alexander Kirillov, Kaiming He, Ross Girshick, Carsten Rother, and Piotr Dollár. Panoptic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9404–9413, 2019.

[77] Alexander Kirillov, Evgeny Levinkov, Bjoern Andres, Bogdan Savchynskyy, and Carsten Rother. Instancecut: from edges to instances with multicut. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5008–5017, 2017.

[78] Marvin Klingner, Jan-Aike Termöhlen, Jonas Mikolajczyk, and Tim Fingscheidt. Self-supervised monocular depth estimation: Solving the dynamic object problem by semantic guidance. In *European Conference on Computer Vision*, pages 582–600. Springer, 2020.

[79] Maria Klodt and Andrea Vedaldi. Supervising the new with the old: learning sfm from sfm. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 698–713, 2018.

[80] Ivan Kreso, Sinisa Segvic, and Josip Krapac. Ladder-style densenets for semantic segmentation of large natural images. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 238–245, 2017.

[81] Jogendra Nath Kundu, Phani Krishna Uppala, Anuj Pahuja, and R Venkatesh Babu. Adadepth: Unsupervised content congruent adaptation for depth estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2656–2665, 2018.

[82] Iro Laina, Christian Rupprecht, Vasileios Belagiannis, Federico Tombari, and Nassir Navab. Deeper depth prediction with fully convolutional residual networks. In *2016 Fourth International Conference on 3D Vision (3DV)*, pages 239–248. IEEE, 2016.

[83] Hei Law and Jia Deng. Cornernet: Detecting objects as paired keypoints. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 734–750, 2018.

[84] Jin Han Lee, Myung-Kyu Han, Dong Wook Ko, and Il Hong Suh. From big to small: Multi-scale local planar guidance for monocular depth estimation. *arXiv preprint arXiv:1907.10326*, 2019.

[85] Youngwan Lee and Jongyoul Park. Centermask: Real-time anchor-free instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13906–13915, 2020.

[86] Hanhan Li, Ariel Gordon, Hang Zhao, Vincent Casser, and Anelia Angelova. Unsupervised monocular depth learning in dynamic scenes. *arXiv preprint arXiv:2010.16404*, 2020.

[87] Jie Li, Allan Raventos, Arjun Bhargava, Takaaki Tagawa, and Adrien Gaidon. Learning to fuse things and stuff. *arXiv preprint arXiv:1812.01192*, 2018.

[88] Ruibo Li, Ke Xian, Chunhua Shen, Zhiguo Cao, Hao Lu, and Lingxiao Hang. Deep attention-based classification network for robust prediction. In *Asian Conference on Computer Vision*, pages 663–678. Springer, 2018.

[89] Xiaoxiao Li, Ziwei Liu, Ping Luo, Chen Change Loy, and Xiaoou Tang. Not all pixels are equal: Difficulty-aware semantic segmentation via deep layer cascade. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3193–3202, 2017.

[90] Yanwei Li, Xinze Chen, Zheng Zhu, Lingxi Xie, Guan Huang, Dalong Du, and Xingang Wang. Attention-guided unified network for panoptic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7026–7035, 2019.

[91] Zhengqi Li and Noah Snavely. Megadepth: Learning single-view depth prediction from internet photos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2041–2050, 2018.

[92] Justin Liang, Namdar Homayounfar, Wei-Chiu Ma, Yuwen Xiong, Rui Hu, and Raquel Urtasun. Polytransform: Deep polygon transformer for instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9131–9140, 2020.

[93] Guosheng Lin, Anton Milan, Chunhua Shen, and Ian Reid. Refinenet: Multi-path refinement networks for high-resolution semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1925–1934, 2017.

[94] Guosheng Lin, Chunhua Shen, Anton Van Den Hengel, and Ian Reid. Efficient piecewise training of deep structured models for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3194–3203, 2016.

[95] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.

[96] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[97] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[98] Shu Liu, Jiaya Jia, Sanja Fidler, and Raquel Urtasun. Sgn: Sequential grouping networks for instance segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3496–3504, 2017.

[99] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8759–8768, 2018.

[100] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[101] I Loshchilov. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[102] Chenxu Luo, Zhenheng Yang, Peng Wang, Yang Wang, Wei Xu, Ram Nevatia, and Alan Yuille. Every pixel counts++: Joint learning of geometry and motion with 3d holistic understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(10):2624–2641, 2019.

[103] Xiaoyang Lyu, Liang Liu, Mengmeng Wang, Xin Kong, Lina Liu, Yong Liu, Xinxin Chen, and Yi Yuan. Hr-depth: High resolution self-supervised monocular depth estimation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(3):2294–2301, May 2021.

[104] Nikolaus Mayer, Eddy Ilg, Philipp Fischer, Caner Hazirbas, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. What makes good synthetic training data for learning disparity and optical flow estimation? *International Journal of Computer Vision*, 126(9):942–960, 2018.

[105] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *2016 4th international conference on 3D vision (3DV)*, pages 565–571, 2016.

[106] Rohit Mohan and Abhinav Valada. Efficientps: Efficient panoptic segmentation. *International Journal of Computer Vision*, 129(5):1551–1579, 2021.

[107] Arsalan Mousavian, Hamed Pirsiavash, and Jana Košecká. Joint semantic segmentation and depth estimation with deep convolutional networks. In *2016 Fourth International Conference on 3D Vision (3DV)*, pages 611–619. IEEE, 2016.

[108] Gerhard Neuhold, Tobias Ollmann, Samuel Rota Bulò, and Peter Kontschieder. The mapillary vistas dataset for semantic understanding of street scenes. In *International Conference on Computer Vision (ICCV)*, 2017.

[109] Davy Neven, Bert De Brabandere, Marc Proesmans, and Luc Van Gool. Instance segmentation by jointly optimizing spatial embeddings and clustering bandwidth. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8837–8845, 2019.

[110] David Nilsson and Cristian Sminchisescu. Semantic video segmentation by gated recurrent flow propagation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6819–6828, 2018.

[111] George Papandreou, Tyler Zhu, Liang-Chieh Chen, Spyros Gidaris, Jonathan Tompson, and Kevin Murphy. Personlab: Person pose estimation and instance segmentation with a bottom-up, part-based, geometric embedding model. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 269–286, 2018.

[112] Adam Paszke, Abhishek Chaurasia, Sangpil Kim, and Eugenio Culurciello. Enet: A deep neural network architecture for real-time semantic segmentation. *arXiv preprint arXiv:1606.02147*, 2016.

[113] Adam Paszke, Sam Gross, Soumith Chintala, and Gregory Chanan. Pytorch: Tensors and dynamic neural networks in python with

strong gpu acceleration. *PyTorch: Tensors and dynamic neural networks in Python with strong GPU acceleration*, 6:3, 2017.

[114] Rui Peng, Ronggang Wang, Yawen Lai, Luyang Tang, and Yangang Cai. Excavating the potential capacity of self-supervised monocular depth estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15560–15569, 2021.

[115] Sida Peng, Wen Jiang, Huaijin Pi, Xiuli Li, Hujun Bao, and Xiaowei Zhou. Deep snake for real-time instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8533–8542, 2020.

[116] Andra Petrovai and Sergiu Nedevschi. Multi-task network for panoptic segmentation in automated driving. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 2394–2401. IEEE, 2019.

[117] Andra Petrovai and Sergiu Nedevschi. Real-time panoptic segmentation with prototype masks for automated driving. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, pages 1400–1406. IEEE, 2020.

[118] Andra Petrovai and Sergiu Nedevschi. Exploiting pseudo labels in a self-supervised learning framework for improved monocular depth estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1578–1588, 2022.

[119] Andra Petrovai and Sergiu Nedevschi. Fast panoptic segmentation with soft attention embeddings. *Sensors*, 22(3):783, 2022.

[120] Andra Petrovai and Sergiu Nedevschi. Semantic cameras for 360-degree environment perception in automated urban driving. *IEEE Transactions on Intelligent Transportation Systems*, 23(10):17271–17283, 2022.

[121] Andra Petrovai and Sergiu Nedevschi. Time-space transformers for video panoptic segmentation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 925–934, 2022.

[122] Andra Petrovai and Sergiu Nedevschi. Monodvps: A self-supervised monocular depth estimation approach to depth-aware

video panoptic segmentation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3077–3086, 2023.

[123] Andrea Pilzer, Stephane Lathuiliere, Nicu Sebe, and Elisa Ricci. Refine and distill: Exploiting cycle-inconsistency and knowledge distillation for unsupervised monocular depth estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9768–9777, 2019.

[124] Matteo Poggi, Filippo Aleotti, Fabio Tosi, and Stefano Mattoccia. On the uncertainty of self-supervised monocular depth estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3227–3237, 2020.

[125] Tobias Pohlen, Alexander Hermans, Markus Mathias, and Bastian Leibe. Full-resolution residual networks for semantic segmentation in street scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4151–4160, 2017.

[126] Lorenzo Porzi, Samuel Rota Bulo, Aleksander Colovic, and Peter Kontschieder. Seamless scene segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8277–8286, 2019.

[127] Siyuan Qiao, Yukun Zhu, Hartwig Adam, Alan Yuille, and Liang-Chieh Chen. Vip-deeplab: Learning visual perception with depth-aware video panoptic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3997–4008, 2021.

[128] Anurag Ranjan, Varun Jampani, Lukas Balles, Kihwan Kim, Deqing Sun, Jonas Wulff, and Michael J Black. Competitive collaboration: Joint unsupervised learning of depth, camera motion, optical flow and motion segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12240–12249, 2019.

[129] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[130] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[131] Eduardo Romera, José M Alvarez, Luis M Bergasa, and Roberto Arroyo. Erfnet: Efficient residual factorized convnet for real-time semantic segmentation. *IEEE Transactions on Intelligent Transportation Systems*, 19(1):263–272, 2017.

[132] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[133] Samuel Rota Bulò, Lorenzo Porzi, and Peter Kontschieder. In-place activated batchnorm for memory-optimized training of dnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5639–5647, 2018.

[134] Faraz Saeedan and Stefan Roth. Boosting monocular depth with panoptic segmentation maps. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3853–3862, 2021.

[135] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4104–4113, 2016.

[136] Chang Shu, Kun Yu, Zhixiang Duan, and Kuiyuan Yang. Feature-metric loss for self-supervised learning of depth and egomotion. In *European Conference on Computer Vision*, pages 572–588. Springer, 2020.

[137] Konstantin Sofiiuk, Olga Barinova, and Anton Konushin. Adaptis: Adaptive instance selection network. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7355–7363, 2019.

[138] Aravind Srinivas, Tsung-Yi Lin, Niki Parmar, Jonathon Shlens, Pieter Abbeel, and Ashish Vaswani. Bottleneck transformers for visual recognition. In *Proceedings of the IEEE/CVF Conference*

on Computer Vision and Pattern Recognition, pages 16519–16529, 2021.

[139] Robin Strudel, Ricardo Garcia, Ivan Laptev, and Cordelia Schmid. Segmenter: Transformer for semantic segmentation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 7262–7272, 2021.

[140] Hang Su, Varun Jampani, Deqing Sun, Orazio Gallo, Erik Learned-Miller, and Jan Kautz. Pixel-adaptive convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11166–11175, 2019.

[141] Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. Efficient transformers: A survey. *arXiv preprint arXiv:2009.06732*, 2020.

[142] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 9627–9636, 2019.

[143] Fabio Tosi, Filippo Aleotti, Pierluigi Zama Ramirez, Matteo Poggi, Samuele Salti, Luigi Di Stefano, and Stefano Mattoccia. Distilled semantics for comprehensive scene understanding from videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4654–4665, 2020.

[144] Jonas Uhrig, Eike Rehder, Björn Fröhlich, Uwe Franke, and Thomas Brox. Box2pix: Single-shot instance segmentation by assigning pixels to object boxes. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 292–299. IEEE, 2018.

[145] Jonas Uhrig, Nick Schneider, Lukas Schneider, Uwe Franke, Thomas Brox, and Andreas Geiger. Sparsity invariant cnns. In *2017 International Conference on 3D Vision (3DV)*, pages 11–20. IEEE, 2017.

[146] Robert Varga, Arthur Costea, Horatiu Florea, Ion Giosan, and Sergiu Nedevschi. Super-sensor for 360-degree environment perception: Point cloud segmentation using image features. In *2017*

IEEE 20th International Conference on Intelligent Transportation Systems (ITSC), pages 1–8. IEEE, 2017.

[147] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

[148] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, et al. Deep high-resolution representation learning for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 43(10):3349–3364, 2020.

[149] Panqu Wang, Pengfei Chen, Ye Yuan, Ding Liu, Zehua Huang, Xiaodi Hou, and Garrison Cottrell. Understanding convolution for semantic segmentation. In *2018 IEEE winter conference on applications of computer vision (WACV)*, pages 1451–1460. IEEE, 2018.

[150] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7794–7803, 2018.

[151] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[152] Jamie Watson, Michael Firman, Gabriel J Brostow, and Daniyar Turmukhambetov. Self-supervised monocular depth hints. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2162–2171, 2019.

[153] Jamie Watson, Oisin Mac Aodha, Victor Prisacariu, Gabriel Brostow, and Michael Firman. The temporal opportunist: Self-supervised multi-frame monocular depth. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1164–1174, 2021.

[154] Diana Wofk, Fangchang Ma, Tien-Ju Yang, Sertac Karaman, and Vivienne Sze. Fastdepth: Fast monocular depth estimation on

embedded systems. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6101–6108. IEEE, 2019.

[155] Sanghyun Woo, Dahun Kim, Joon-Young Lee, and In So Kweon. Learning to associate every segment for video panoptic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2705–2714, 2021.

[156] Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018.

[157] Zifeng Wu, Chunhua Shen, and Anton Van Den Hengel. Wider or deeper: Revisiting the resnet model for visual recognition. *Pattern recognition*, 90:119–133, 2019.

[158] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M Alvarez, and Ping Luo. Segformer: Simple and efficient design for semantic segmentation with transformers. *Advances in neural information processing systems*, 34:12077–12090, 2021.

[159] Yuwen Xiong, Renjie Liao, Hengshuang Zhao, Rui Hu, Min Bai, Ersin Yumer, and Raquel Urtasun. Upsnet: A unified panoptic segmentation network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8818–8826, 2019.

[160] Feng Xue, Guirong Zhuo, Ziyuan Huang, Wufei Fu, Zhuoyue Wu, and Marcelo H Ang. Toward hierarchical self-supervised monocular absolute depth estimation for autonomous driving applications. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2330–2337. IEEE, 2020.

[161] Linjie Yang, Yuchen Fan, and Ning Xu. Video instance segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5188–5197, 2019.

[162] Tien-Ju Yang, Maxwell D Collins, Yukun Zhu, Jyh-Jing Hwang, Ting Liu, Xiao Zhang, Vivienne Sze, George Papandreou, and Liang-Chieh Chen. Deeperlab: Single-shot image parser. *arXiv preprint arXiv:1902.05093*, 2019.

[163] Zhichao Yin and Jianping Shi. Geonet: Unsupervised learning of dense depth, optical flow and camera pose. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1983–1992, 2018.

[164] Pierluigi Zama Ramirez, Matteo Poggi, Fabio Tosi, Stefano Mattoccia, and Luigi Di Stefano. Geometry meets semantics for semi-supervised monocular depth estimation. In *Asian Conference on Computer Vision*, pages 298–313. Springer, 2018.

[165] Huangying Zhan, Ravi Garg, Chamara Saroj Weerasekera, Kejie Li, Harsh Agarwal, and Ian Reid. Unsupervised learning of monocular depth estimation and visual odometry with deep feature reconstruction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 340–349, 2018.

[166] Rui Zhang, Sheng Tang, Yongdong Zhang, Jintao Li, and Shuicheng Yan. Scale-adaptive convolutions for scene parsing. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2031–2039, 2017.

[167] Zhenyu Zhang, Zhen Cui, Chunyan Xu, Zequn Jie, Xiang Li, and Jian Yang. Joint task-recursive learning for semantic segmentation and depth estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 235–251, 2018.

[168] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2881–2890, 2017.

[169] Shengyu Zhao, Yilun Sheng, Yue Dong, Eric I Chang, Yan Xu, et al. Maskflownet: Asymmetric feature matching with learnable occlusion mask. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6278–6287, 2020.

[170] Sixiao Zheng, Jiachen Lu, Hengshuang Zhao, Xiatian Zhu, Zekun Luo, Yabiao Wang, Yanwei Fu, Jianfeng Feng, Tao Xiang, Philip HS Torr, et al. Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6881–6890, 2021.

[171] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G Lowe. Unsupervised learning of depth and ego-motion from video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1851–1858, 2017.

[172] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. *arXiv preprint arXiv:2010.04159*, 2020.

[173] Yuliang Zou, Zelun Luo, and Jia-Bin Huang. Df-net: Unsupervised joint learning of depth and flow using cross-task consistency. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 36–53, 2018.