Răzvan ITU

Sisteme de percepție monoculară folosind inteligență și viziune artificială

Teză de doctorat

UTPRESS
Cluj-Napoca, 2021
ISBN 978-606-737-537-4

Răzvan ITU

SISTEM DE PERCEPȚIE MONOCULARĂ FOLOSIND INTELIGENȚĂ ȘI VIZIUNE ARTIFICIALĂ

Teză de doctorat

UTPRESS
Cluj - Napoca, 2021
ISBN 978-606-737-537-4

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

Director: ing. Călin Câmpean

Recenzia:

Prof.dr.ing. Ioan Salomie
Prof.dr.ing. Radu Dănescu
Prof.dr.ing. Mihai Micea
Prof.dr.ing. Ştefan Trăuşan-Matu
Conf.dr.ing. Tiberiu Mariţa

ISBN 978-606-737-537-4
Bun de tipar: 16.11.2021

Cuprins

Listă de figuri

Listă de tabele

Prefață

Cartea de față prezintă tehnici de percepție a mediului de trafic rutier, implementate folosind sisteme mobile bazate pe o singură cameră video (sistem de viziune monocular). Percepția mediului reprezintă un domeniu de interes activ, cu aplicații în domeniul roboticii, platformelor de roboți mobili sau domeniul auto. Producătorii de autovehicule utilizează sisteme bazate pe diferiți senzori pentru a percepe, măsura și estima scena de trafic pentru a lua decizii cu scopul de a îmbunătăți siguranța în trafic a conducătorului auto, a pasagerilor sau restul participanților din trafic (pietoni, alte vehicule). Aceste sisteme pot fi complexe și pot fi adăugate de producătorii de automobile, dar în cadrul acestei cărți am prezentat soluții care pot fi adăugate oricărui vehicul. Astfel de sisteme și algoritmi ar putea ajuta la creșterea siguranței în trafic și pot acționa preventiv pentru a evita anumite situații periculoase.

Această carte are la bază ideile și structura tezei de doctorat pe care am susținut-o în anul 2019 având același titlu. Teza de doctorat a fost elaborată sub îndrumarea prof.dr.ing. Radu Dănescu și a primit calificativul “magna cum laude”.

Aș dori să mulțumesc în mod deosebit conducătorului de doctorat, domnului profesor dr. ing. Radu Dănescu pentru îndrumarea oferită și pentru tot sprijinul acordat de-a lungul anilor, încă din vremea studenției când am colaborat pentru lucrarea de licență și apoi la masterat și în cele din urmă la doctorat, care mi-a oferit o anumită direcție și un scop mai bine definit și a observat un potențial care nu aș fi reușit să îl dezvolt singur. Totodată aș vrea să mulțumesc familiei mele și prietenilor apropiați pentru înțelegere, răbdare și motivare când a fost necesar, și nu în ultimul rând mulțumesc colegilor din grupul de cercetare, dar și colegilor de departament pentru sfaturi și ajutor ori de câte ori a fost nevoie.

1. Introducere și obiective

1.1 Introducere

Sistemele moderne de asistență a conducătorilor auto reprezintă un interes major atât pentru industria constructoare de mașini cât și pentru cercetători, dar și pentru beneficiari. Principalul interes este cel de a îmbunătăți siguranța participanților din trafic prin reducerea numărului de accidente. Principalele cauze ale accidentelor rutiere sunt determinate în general de factori umani: stări de somnolență cauzate de oboseală, conducerea sub efectul substanțelor interzise sau alcool, lipsa de atenție. În fiecare an există aproximativ 1.25 milioane de accidente mortale [ASIRT2018], iar la nivel de SUA sunt în scădere de la 50.000 accidente în 1975, la aproximativ 37.000 accidente în anul 2017 [NCSA2018]. Accesibilitatea unor platforme hardware, atât de percepție a mediului, cât și de procesare sau putere de calcul din ultimii ani a facilitat o puternică dezvoltare și un interes deosebit față de crearea unor noi soluții și arhitecturi pentru sisteme de asistență a conducătorilor auto.

Primele sisteme de conducere autonomă au fost dezvoltate încă din anii 1960, dar erau extrem de rudimentare. În continuare voi prezenta o scurtă retrospectivă a primelor tentative de a construi vehicule autonome.

1.1.1 Istoria vehiculelor autonome

Mașina autonomă dezvoltată în Anglia în anii 1960 [Waugh2013] folosea senzori magnetici instalați în vehicul, conectați la sistemul hidraulic al coloanei de direcție pentru a vira mașina. Pentru a avea sistemul funcțional sub asfalt este o "șină" de fier. Proiectul nu s-a dovedit fezabil pentru o scară largă deoarece necesită investiții mari pe partea de infrastructură. A fost totuși o primă tentativă de a avea un vehicul care să se deplaseze și să își mențină banda singur, deși era limitat la progresele tehnologice de atunci.

În 1968 firma Continental a prezentat un vehicul Mercedes 250 [Fehlhaber2018] care putea ajunge la viteze mari pe circuitul de test Contidrom, fără a avea un șofer la volan. Similar cu vehiculul dezvoltat de englezi, Mercedes utiliza senzori magnetici pentru a se menține de-a lungul unui câmp magnetic. Acest câmp magnetic era realizat în mod diferit, prin amplasarea unui fir direct pe carosabil, pe pista de testare. Sistemul dezvoltat de Continental era destinat pentru testarea de anvelope într-un mod cât mai eficient și fără nevoia unui șofer, pentru a reduce erorile la măsurători. Proiectul a fost sistat în 1974, dar în ultimii ani compania Continental a reluat cercetarea și oferă soluții moderne de tip ADAS.

O soluție mai modernă pentru vehicule autonome a fost prezentată în 1986 [Dickmanns1998], [Delcker2018], de către echipa condusă de cercetătorul Ernst Dickmanns. Sistemul autonom bazat pe un vehicul Mercedes folosea percepția vizuală folosind camere

video, analize probabilistice pentru luarea deciziilor de control. Vehiculul avea capabilitatea de a-și menține banda de circulație prin detecția benzilor folosind viziune artificială. Cercetarea în colaborare cu Daimler Benz a continuat până în anul 1995, în cadrul proiectului european Prometheus [Daimler2016] unde erau implicate mai multe firme din industrie (Bosch Gmbh) și universități. În 1994 a fost prezentat vehiculul VaMP și Vita-2 care au parcurs cu succes peste 1000 km în regim de autostradă cu intervenție umană foarte redusă.

Universitatea Carnegie Mellon a folosit pentru prima dată rețele neuronale cu scopul de a controla direcția de deplasare a unui vehicul în anul 1989, în proiectul Navlab, iar rețeaua a fost numită ALVINN (Autonomous Land Vehicule in a Neural Network) [Pomerleau1989]. Deși inițial, datorită constrângerilor hardware era limitat la o viteză de deplasare de aproximativ 5.6 km/h, după câteva iterații ale algoritmilor și după progresul tehnologic, în 1995 vehiculul Navlab era atingea viteze de 88 km/h. În același an vehiculul a parcurs cu succes 5000 km în SUA în condițiile în care în 98.2% din situații a acționat fără intervenție umană [Pomerleau1995], dar vehiculul era semi-autonom: direcția de deplasare era controlată de rețele neuronale în timp ce accelerația și frânare erau controlate de oameni (din motive de siguranță).

În 1996, proiectul universității din Parma numit ARGO, condus de Alberto Broggi folosea un vehicul Lancia pentru a detecta obiecte și benzi de circulație [Broggi1999]. Această soluție utiliza un sistem de stereo-viziune și senzori de poziționare prin satelit GPS. În anul 2000 Bosch a introdus tempomatul adaptiv folosind RADAR, iar în 2003 Toyota introduce pe piață un sistem de parcare lateral automat pe modelul Prius.

Progresele cele mai notabile au avut loc în urma desfășurării concursurilor Grand Challenge, organizate de DARPA (Defense Advanced Research Projects Agency) din cadrul armatei din SUA. Prima ediție din 2004 a constat în oferirea unui premiu de 1 milion de dolari echipei capabile să creeze un vehicul autonom care să parcurgă un traseu de 150 mile în deșertul Mojave. În acel an nici o echipă nu a reușit să termine traseul. În anul 2005 vehiculul autonom “Stanley” [Thrun2006] dezvoltat de o echipă din cadrul universității Stanford a câștigat competiția Grand Challenge II, parcurgând pentru prima dată și în cel mai scurt timp traseul din deșert. Alte 4 vehicule autonome au reușit să termine cursa cu succes. În 2007, a fost organizată competiția Grand Challenge III desfășurată într-un mediu urban, iar echipa câștigătoare a fost de la Carnegie Mellon University [Urmson2007].

În anii 2010, vehiculele autonome s-au dezvoltat intens și marii producători auto au prezentat câte o variantă proprie sau în colaborare cu unele companii mai mici. Mulți cercetători care au făcut parte din echipele care au concurat în competițiile DARPA au ajuns ulterior să formeze noi companii (Cruise, nuTonomy) sau să lucreze pentru cele existente din domeniu (Google, Daimler, Tesla) și să aducă noi inovații. De-a lungul anilor, tehnologiile dezvoltate în colective de cercetare au început să fie oferite și pe vehicule de serie. În continuare sunt prezentate câteva sisteme de asistență a conducătorilor.

1.1.2 Tipuri de sisteme de asistență

Există mai multe tipuri de sisteme active și pasive de siguranță a conducătorilor și a pasagerilor vehiculelor, cum ar fi: centura de siguranță, asistența la frânare (ABS - Anti Lock Braking System), sisteme de menținere a traiectoriei (ESP - Electronic Stability Program sau Traction Control), airbag-uri. De asemenea, au început să fie integrate și sisteme avansate de asistență a conducătorilor (ADAS - Advanced Driver Assistance Systems): menținerea benzii curente (Lane Keeping Assistant), sisteme de avertizare în caz de coliziuni frontale (Forward Collision Warning), tempomat adaptiv (Adaptive Cruise Control), sistem de asistență la schimbarea benzii de circulație (Lane Change Assistant) și multe altele.

1.1.3 Niveluri de automatizare

Diferite niveluri de autonomie au fost definite de-a lungul anilor de către cei implicați în această industrie. În general sunt considerate 6 grade de automatizare a vehiculelor, 0 fiind cel mai “slab”, unde sunt cele mai rudimentare sisteme de avertizare și asistența în care șoferul deține controlul complet al vehiculului, în timp ce nivelul 5 de automatizare reprezintă un sistem complet autonom, unde intervenția șoferului nu este necesară pe parcursul călătoriei. Aceste niveluri sunt definite in standardul SAE J3016 [SAE2018], care este acceptat in industrie ca fiind standardul (inclusiv de către NHTSA - “National Highway Traffic Safety Administration”).

Nivelul 0: Fără automatizare: sisteme ar putea oferi avertismente, dar nu au nici un control asupra vehiculului.
Nivelul 1: Asistența conducător auto: șoferul și sistemul automat au control împreună asupra vehiculului (de exemplu sistemul de tempomat automat, unde viteza este controlată de sistem, iar direcția de deplasare este controlată de șofer).
Nivelul 2: Automatizare parțială: sistemul automat poate prelua complet controlul vehiculului (accelerare, frânare și direcție de deplasare). Șoferul este responsabil pentru monitorizarea mediului înconjurător și trebuie să poată interveni în caz de o situație majoră.
Nivelul 3: Automatizare condițională: șoferul își poate lua ochii de la drum în anumite situații, dar este nevoie să revină la controlul vehiculului într-un interval de timp specificat de către producătorul auto.
Nivelul 4: Grad mare de automatizare: atenția șoferului nu este necesară. Acest nivel este definit doar în anumite scenarii unde vehiculul este capabil să navigheze autonom. În cazul unei situații neprevăzute vehiculul are capabilitatea de a gestiona situația sau de a parca vehiculul și de a solicita șoferului preluarea controlului.
Nivelul 5: Automatizare completă a procesului de conducere a unui vehicul: nu este necesară instalarea unui volan sau a pedalelor într-un vehicul.

1.1.4 Senzori vehicule autonome

Senzorii folosiți în vehicule autonome pot fi clasificați în două tipuri: senzori externi, montați în afara vehiculului și senzori interni, montați înăuntru. Senzorii externi sunt în general

de tip: LIDAR, RADAR, camere video (în spectrul vizibil de culoare), camere infraroșu sau cu termoviziune, senzori de poziționare prin satelit (GPS sau GLONASS) și de măsurare a inerției (IMU - Inertial Measurement Unit), senzori audio: care ar putea fi folosiți pentru a interpreta sunetul din scenă: de exemplu prezența unor vehicule de intervenție cu semnale acustice pornite, interpretarea sunetului scos de anvelope când rulează pe ploaie sau uscat. Senzorii interni în principal sunt folosiți pentru monitorizarea stării de atenție a conducătorului auto prin folosirea unei camere video sau chiar și a senzorilor audio.

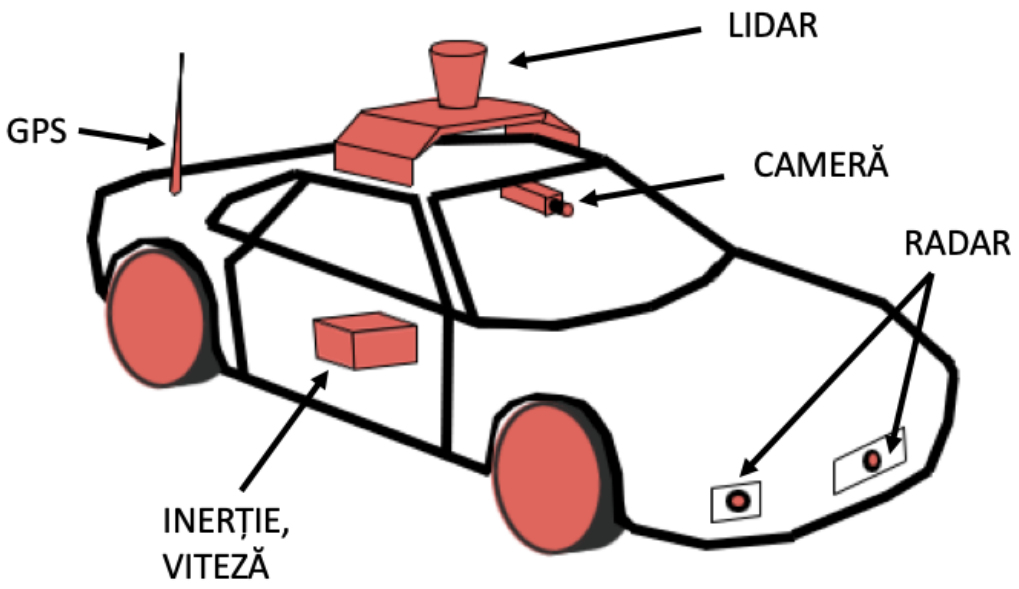


Figura 1. Tipuri de senzori folosiți în vehicule autonome.

Totodată, senzorii unui vehicul autonom îndeplinesc două roluri: măsurarea și percepția scenei 3D, dar și măsurarea stării curente a vehiculului. În general, starea curentă a vehiculului este dată de senzori externi adăugați: cum ar fi senzori de poziționare prin satelit sau de măsurare a inerției (IMU): giroscop, accelerometru sau senzor magnetic pentru determinarea orientării față de nordul magnetic. Din vehicul uneori se utilizează și informațiile obținute direct din unitatea de control a motorului (ECU - Engine Control Unit) prin protocolul de comunicare CAN (Controller Area Network), mai specific prin accesarea acestei rețele folosind standardul OBD (On Board Diagnostics).

Senzorii de tip LIDAR (Light/Laser Detection And Ranging) reprezintă o modalitate de măsurare a distanței folosind lumina, sub forma unor pulsații de laser. Tehnologia bazată pe senzori LIDAR a fost utilizată încă din anii 1960, când scannere laser au fost montate pe avioane. După introducerea standardului GPS în anii 1980, sistemele de măsurare folosind LIDAR au fost utilizate pentru măsurări geospațiale și teritoriale. Principiul de funcționare constă în iluminarea scenei sau a obiectului dorit și calcularea timpului necesar de întoarcere a pulsațiilor reflectate. Distanța este determinată folosind viteza luminii, de aceea aceste măsurători sunt denumite și “Time of Flight”. LIDAR este folosit în mod deosebit pentru generarea de măsurători 3D a scenei, fiind utilizate din ce în ce mai mult pentru vehicule autonome. Senzori de tip LIDAR moderni oferă puncte 3D (“point cloud”) a scenei pe o rază de 360 grade în jurul vehiculului, iar prin procesarea acestor puncte se pot determina obiectele atât statice, cât și în mișcare.

RADAR (RAdio Detection And Ranging) reprezintă o modalitate de măsurare a scenei prin utilizarea undelor radio. Principiul de funcționare este unul asemănător cu cel al LIDAR, iar măsurarea distanțelor se face tot folosind viteza luminii. Undele radio nu sunt utile pentru detecția obiectelor mici, de aceea este folosit pentru sisteme de tempomat adaptiv unde nu este necesară o percepție complexă a obiectului din față, ci este necesară doar informația despre distanța și viteza obiectului identificat. Determinarea vitezei se face folosind efectul Doppler și reprezintă un avantaj major față de LIDAR.

SONAR (SOund Navigation And Ranging) este o tehnică de percepție și navigare folosind propagarea undelor sonore. A fost utilizată inițial în domeniul navigației maritime, atât pe apă (nave) cât și sub (submarine, torpile). Acești senzori sunt utilizați și în unele platforme de vehicule autonome în combinație cu alți senzori, în principal datorită capabilității senzorilor SONAR de a nu fi afectați de apă. Senzorii SONAR sunt folosiți pentru percepție la distanțe scurte și medii, un exemplu uzual este folosirea lor ca și senzori de parcare pentru vehicule.

Tabel 1. Comparație intre senzorii de percepție ai unui vehicul autonom.

Tip senzor	Avantaje	Dezavantaje
LIDAR	- acuratețe și precizie în măsurarea scenei - poate fi folosit în orice moment al zilei și noaptea - măsurătorile nu prezintă distorsiuni geometrice	- nu funcționează în unele condiții meteo: ceață, ploaie - pulsațiile de lumină sunt afectate uneori de suprafețe foarte lucioase din cauza reflexiei - razele laser pot afecta vederea umană și alte aparate de măsurare (de exemplu camere foto) - oferă date “sparse” (nu “dense”) - tehnologie foarte scumpă, greu de interfațat și de interpretat datele, greu de integrat pe un vehicul existent - sistem compus din părți rotative, ceea ce înseamnă o fiabilitate pe termen lung mai scăzută
RADAR	- util pentru distanțe medii și lungi - nu are limitarea celorlalți senzori legată de vreme nefavorabilă - determinarea vitezei obiectelor folosind efectul Doppler	- probleme în distingerea obiectelor, inutil pentru cele de dimensiune redusă - informațiile oferite sunt greu de interpretat și folosit
SONAR	- ușor de folosit și accesibil ca și cost - funcționează și în apă, spre deosebire de restul	- precis și util doar pentru distanțe foarte scurte - timp de procesare mai redus datorită vitezei sunetului (față de viteza luminii) - rezoluție direcțională slabă - posibile interferențe cu alte unde sonore

Camere video	- pretabil pentru orice distanță, în configurație stereo (în funcție de disparitate), dar sunt folosite în general pentru distanțe scurte și medii - date "dense" ale scenei, se pot folosi și informații despre intensitatea de culoare sau textură	- necesită o calibrare precisă pentru a determina informații despre adâncime/distanță - afectate de distorsiuni geometrice cauzate de lentile - utilizare limitată în condiții meteo adverse

Camerele video sunt în general montate pe parbriz în spatele oglinzii retrovizoare. Pentru o percepție asupra adâncimii în scenă, a distanțelor până la obiecte se folosesc sisteme formate din două camere (stereo-viziune) sau uneori și mai multe camere. Vehiculele Tesla oferă un mod de condus autonom, iar sistemul este format în mare parte din camere video și RADAR [Tesla2019].

În afara detecției de obstacole și a măsurării distanței până la acestea, camerele pot fi folosite și pentru a detecta zona liberă ("driveable" sau "free space") dintr-o imagine din trafic, detecția semnelor de circulație, benzilor de circulație și a marcajelor de pe drum ("painted road obstacles") sau pentru monitorizarea șoferului.

1.1.5 Etape necesare în dezvoltarea unui vehicul autonom

În procesul complex de navigare autonomă a unui robot sau vehicul, se pot defini mai multe sarcini/etape principale:

- Măsurare ("sensing")
- Percepție ("perception")
- Localizare ("mapping")
- Planificare ("control"/"decision")

Procesul de măsurare are rolul de a achiziționa datele din senzorii interni și externi ai vehiculului. Această etapă implică de obicei echiparea vehiculului cu senzori și apoi calibrarea acestora într-un sistem de referință comun.

Partea de percepție este a doua etapă și este una dintre cele mai complexe. Percepția poate fi definită ca procesul de analizare și interpretare a datelor pentru a oferi informații relevante din scenă și pentru a înțelege scena: detecția de vehicule sau pietoni, direcția și viteza lor de deplasare (urmărirea de obstacole), numărul de benzi (urmărirea benzilor de circulație) sau tipurile de semne de circulație sau a culorii semaforului. Percepția s-ar putea clasifica în abordări bazate pe viziune artificială sau inteligență artificială, dar există și soluții care folosesc o combinație între cele două.

Localizarea și cartografierea mediului se referă la folosirea hărților existente pentru a identifica poziția vehiculului în harta, dar și analiza pozițiilor restul obstacolelor, precum și crearea de hărți noi (locale) pe măsură ce vehiculul avansează.

Planificarea reprezintă de asemenea o sarcină vitală în construirea unui vehicul autonom. Navigarea în trafic și planificarea manevrelor este condiționată de o percepție eficientă. Algoritmii de planificare necesită informații actualizate și robuste despre condițiile de trafic pentru a genera diferite acțiuni care să fie luate de către vehiculul autonom. Tot în această etapă se calculează și o predicție a intențiilor celorlalți participanți din trafic.

Vehiculele moderne se bazează pe o combinație de senzori. Fiecare tip de senzor are anumite avantaje sau dezavantaje (Tabel 1). De exemplu, unul dintre cele mai utilizate sisteme în industrie este Autopilot de pe vehiculele Tesla care folosește 6 camere video: în față sunt montate 3 camere: una cu un câmp vizual foarte mare (“wide angle”) utilă pentru distanțe scurte până în 60 metri, o cameră cu un câmp vizual normal pentru distanțe medii până în 150 metri și încă una cu deschidere angulară mică (“narrow”), utilă pentru distanțe mari până la 250 metri în față. În spate sunt încă 3 camere: două în colțuri și una montată central cu unghi mare de filmare. Sistemul Autopilot de la Tesla folosește și un radar montat în bară fată, dar și senzori cu ultrasunete montați pe lateral.

1.1.6 Motivație

Deși problema siguranței în traficul rutier este determinată și de o infrastructură bună, principalele cauze ale accidentelor sunt datorate coliziunilor dintre vehicule. Există un interes mare pentru le îmbunătăți, mai ales deoarece alte domenii de transport au fost deja parțial sau aproape complet automatizate și sunt capabile de transport de pasageri în regim autonom. De exemplu transportul aerian este în mare parte autonom, la fel ca și cel maritim sau chiar transportul public feroviar (în special cel subteran unde metrourile sunt autonome în multe țări). Totuși, vehiculele sunt supuse unor medii înconjurătoare mult mai complexe, iar gradul de percepție necesar trebuie să fie cât mai eficient, partea de control și decizie de asemenea trebuie să fie capabilă pentru a face față diferitelor situații extreme, pe lângă cele de trafic normal.

Vehiculele autonome, deși rezolvă o parte din problemele de siguranță pot introduce și noi probleme, în special dificultăți legate de etică și moralitate: cine e responsabil în cazul unui accident cu o mașină autonomă? Sau cine e responsabil în cazul în care sunt implicate două vehicule autonome, sau în situația în care un vehicul autonom are posibilitatea să salveze ori pasagerii vehiculului, ori pietonii din trafic (clasica problemă de etică: “trolley problem” [Foot1967]. Problema este una de moralitate, ce decizie e corectă, deoarece este foarte greu să alegi între viețile umane. Datorită acestor probleme, a complexității scenelor și traficului rutier din ce în ce mai aglomerat în marile centre urbane, consider că vehiculele autonome nu sunt încă fezabile în totalitate, dar sistemele avansate de prevenție, monitorizare și siguranță sunt într-adevăr de actualitate și au o importanță majoră.

Anumite state din SUA au acordat licențe speciale pentru companii pentru a-și testa pe drumuri publice sisteme autonome, începând cu Nevada în 2011, urmate de Florida și California în 2012. Waymo, companie din cadrul Google a avut cei mai mulți kilometri parcurși fără intervenție umană și acumulează undeva la 40.000 km / zi [Korosec2018], dar cumulat vehiculele Testa i-a depășit în iulie 2018 având un total de peste 1.9 miliarde de km [Lambert2018] conduși folosind sistemul Autopilot. Totodată, Waymo a primit în octombrie

2018 permisiunea de a opera vehicule autonome fără a avea o persoană la volan. Până la acea dată, companiile erau obligate să aibă o persoană în spatele volanului din motive de siguranță și pentru a putea interveni în situații limită.

Deși sistemele dezvoltate de către producătorii auto vin echipate direct pe vehicule din fabrică, există în mod cert o nevoie de a face vehiculele actuale și cele mai vechi cât mai avansate. O modalitate de a adăuga funcții de siguranță constă în instalarea unor senzori de percepție și a unor module de calcul, iar una dintre cele mai ușoare metode este de a utiliza direct platforme mobile echipate cu mai mulți senzori (de exemplu dispozitive mobile). Dispozitivele mobile inteligente sunt omniprezente și vin echipate cu una sau mai multe camere video de diferite rezoluții, dar și cu putere de procesare din ce în ce mai mare. De asemenea, dispozitivele mobile moderne dispun de o varietate de senzori adiționali, cum ar fi: senzori de poziționare prin satelit (GPS, GLONASS), accelerometru, giroscop, senzor geomagnetic, compas, ceea ce înseamnă că ele pot fi utilizate pentru sisteme de asistență a conducătorilor auto sau ca și sisteme de analiză și monitorizare a traficului rutier. Șoferii pot plasa dispozitivele mobile în parbriz, astfel încât camera principala să fie orientată spre drum. Totuși analiza sau percepția a mediului înconjurător va necesita o calibrare a camerei cu scopul de a obține corespondențe între trăsăturile din scenă 3D a lumii și trăsăturile din imaginea camerei video. Acest pas de calibrare este de multe ori neglijat sau omis de către un utilizator obișnuit.

1.2 Obiective

În această carte sunt prezentate obiectivele principale din cadrul tezei de doctorat pe care am susținut-o, astfel principalul obiectiv constă in proiectarea și implementarea unui sistem de percepție a mediului de trafic rutier bazat pe viziune monoculară și senzori de măsurare a poziției și inerției care să fie capabil să ruleze în diferite condiții de iluminare, în scenarii de autostradă și mai ales în traficul urban.

Pentru îndeplinirea obiectivului general, am identificat o serie de obiective secundare:

1. crearea unui sistem de achiziție și procesare a datelor senzoriale
2. detecția obstacolelor:
   - identificarea prezenței obstacolelor în spațiul imaginii folosind segmentare semantică cu rețele neuronale convoluționale
   - extragere informații 3D din imaginea monoculară prin analiza imaginii cu efectul de perspectivă eliminat
3. urmărirea și estimarea pozițiilor obstacolelor folosind modele 3D:
   - urmărire folosind un filtru de particule cu hartă de ocupare
   - urmărire la nivel de cuboid
4. calibrarea automată a camerei
   - calibrare automată inițială
   - ajustarea dinamică a parametrilor camerei

Un sistem modern de asistență a șoferului necesită o percepție cât mai bună a scenei. Am ales folosirea unei singure camere video montată în vehicul în spatele parbrizului. Avantajul unui dispozitiv mobil (telefon sau tabletă) pentru achiziția de imagini este dat de faptul că oferă o rezoluție bună a imaginilor, dar și o gamă largă de senzori adiționali, totul integrat într-un dispozitiv de dimensiuni reduse care poate fi poziționat și manevrat ușor. Sistemul de achiziție constă în accesarea imaginilor din camera dispozitivului mobil și a senzorilor disponibili (GPS, accelerometru, giroscop, etc.) și procesarea lor direct pe mobil sau salvarea și trimiterea datelor pe un sistem mai performant pentru procesare mai rapidă.

Detecția obstacolelor presupune identificarea prezenței obstacolelor în spațiul imaginii și estimarea informațiilor 3D. Pentru soluția propusă în cadrul acestei teze de doctorat, am ales utilizarea segmentării semantice a imaginilor pentru identificarea obstacolelor și analiza imaginilor cu efectul de perspectivă eliminat pentru a extrage informațiile 3D despre obstacole. Segmentarea semantică a scenei din trafic reprezintă interpretarea imaginilor din camera monoculară și înțelegerea acestora. Pentru acest obiectiv am ales folosirea unor tehnici de procesare bazate pe inteligența artificială, mai specific folosirea unei rețele neuronale convoluționale capabilă să ofere informații despre zona de drum (“driveable road area”) din imagini. Această abordare este robustă și invariantă la raportul de aspect al imaginilor și oferă rezultate foarte bune, fiind necesar doar un pas de redimensionare a imaginilor de intrare (la dimensiunea cu care a fost antrenată rețeaua inițial). Am ales detecția drumului, a spațiului liber pe care se poate circula în siguranță, deoarece astfel se poate face presupunerea că tot ce nu este drum ar putea fi un obstacol. Presupunând că suprafața drumului este plată, în imaginile cu efectul de perspectivă eliminat se pot determina distanțele până la obstacole.

Zonele care nu fac parte din drum vor fi analizate și urmărite în timp folosind modele de percepție 3D, bazate pe filtre de particule. Este construită o hartă de ocupare a scenei bazată pe un filtru de particule, unde fiecare obstacol din scenă este reprezentat de un set de particule. Acesta este un model dinamic de reprezentare, deoarece fiecare particulă are o componentă de viteză, una de poziție în harta de ocupare. Aceste informații sunt folosite pentru predicția și actualizarea hărții, când sunt adăugate sau eliminate anumite particule ce reprezintă posibile obstacole. De asemenea sistemul folosește încă două rețele neuronale convoluționale: prima pentru a detecta vehiculele din scenă, iar a doua pentru a folosi detecțiile din prima rețea pentru a determina orientarea și dimensiunea obstacolelor. Se determină corespondența dintre vehiculele detectate din CNN și cele extrase din filtrul de particule și se fuzionează informațiile despre orientare și dimensiune.

Pentru a putea folosi camere video de la producători diferiți montate în vehicule diferite este nevoie de un sistem de calibrare cât mai eficient și cât mai puțin elaborat. Inițial am realizat un sistem de calibrare manual care necesită intervenția minimală a utilizatorului, iar apoi am prezentat abordări pentru calibrare automată unde nu este necesar nici un demers al utilizatorului. Astfel, un sistem capabil să se auto-calibreze este mult mai portabil și poate fi instalat mai ușor și în vehicule vechi care de obicei nu au nici un sistem de asistență a șoferului.

Prin integrarea tuturor componentelor într-un singur proiect, am reușit implementarea unui sistem de percepție monocular capabil să furnizeze informații 3D. Sistemul dezvoltat și prezentat în această lucrare are la bază idei și algoritmi care au fost publicați în jurnale, capitole de carte și la conferințe de specialitate.

1.3 Structura cărții

Această carte are la baza lucrarea de doctorat susținută de mine și este structurată în patru capitole, din care primul este o introducere în domeniu împreună cu motivația din spatele cercetării, urmată de o prezentare a obiectivelor. Cartea conține două capitole mari în care sunt prezentate contribuțiile: capitolul 2 care prezintă metode de detecție și urmărire a obstacolelor și capitolul 3 unde am descris modelul camerei și tehnici de calibrare a camerei. În capitolul 4 sunt exprimate concluziile și observațiile, urmate de bibliografie și anexa.

Capitolul 2 conține o introducere a metodelor de urmărire bazate pe probabilistică: filtrul Bayes, filtrul Kalman, filtrul Kalman extins sau filtrul de particule. Tot aici am prezentat o introducere în rețelelor neuronale convoluționale, am descris diferența între clasificare și regresie și apoi am prezentat funcțiile de activare necesare unei rețele artificiale, dar și modul de inițializare a ponderilor și ajustarea lor. Am descris funcțiile de cost și apoi segmentarea semantică folosind rețele neuronale convoluționale. Contribuțiile sunt prezentate începând cu secțiunea 2.12 unde am descris tehnici de percepție monoculară pentru roboți mobili sau vehicule inteligente. Secțiunea 2.13 reprezintă sistemul complet de percepție a traficului rutier având la bază o singură cameră, care folosește un filtru de particule bazat pe o hartă de ocupare dinamică a scenei, construită din imaginea din trafic segmentată de o rețea convoluțională neuronală și transformată în imagine cu efectul de perspectivă eliminat. Concluziile și contribuțiile sunt prezentate la finalul capitolului împreună cu publicațiile rezultate.

Capitolul 3 descrie modelul camerei, partea de fundamentare teoretică împreună cu parametri intrinseci și extrinseci, modul de calculare a distanței focale și algoritmi de eliminare a efectului de perspectivă în imagini. În secțiunea 3.5 am realizat o calculare automată a distanței focale pe dispozitive mobile. Secțiunea 3.7 descrie o soluție de calibrare manuală intuitivă pentru parametri extrinseci, în timp ce secțiunea 3.8 reprezintă o primă abordare de calibrare automată folosind dimensiunea unei benzi de circulație. Secțiunea 3.9 descrie pe larg un algoritm de calculare a punctului de fugă folosind procesarea imaginilor. În secțiunea 3.10 și 3.11 am prezentat o modalitate de a calcula punctul de fugă folosind rețele neuronale convoluționale, iar în ultima parte a capitolului (3.12) este descrisă o tehnică de generare a parametrilor extrinseci prin analiza traiectoriei vehiculelor din trafic. În cele din urmă sunt prezentate concluziile, contribuțiile originale și publicațiile.

Capitolul 4 descrie concluziile și principalele contribuții proprii. În final sunt referințele bibliografice.

2. Metode de detecție și urmărire a obstacolelor

Analiza secvențelor de imagini (video) reprezintă o activitate importantă în domeniul procesării imaginilor și a viziunii artificiale. Se poate împărți în trei sarcini principale: detecția obiectelor relevante (dorite), urmărirea lor cadru cu cadru și analiza traiectoriei lor. Toate cele trei sarcini pot fi implementate folosind metode clasice de calcul, dar și folosind rețele neuronale. Totuși, procesul de urmărire poate fi modelat și reprezentat mai bine utilizând metode convenționale bazate pe modele matematice bine definite care să reprezinte geometria obiectelor și modelul lumii și modelul dinamic de mișcare. Metodele uzuale sunt bazate sisteme modelate liniar expuse unor zgomote Gaussiene: filtre Kalman (simple sau extinse) sau sisteme neliniare: Monte Carlo (filtre de particule). În contextul unui vehicul autonom, detecția și apoi urmărirea obstacolelor reprezintă un pas esențial în implementarea sistemelor de siguranță, în special pentru asistență și avertizarea la coliziuni, deoarece se analizează mișcarea și traiectoriile celorlalți participanți la trafic (vehicule, pietoni, etc.).

Localizarea sau urmărirea se referă la implementarea unei funcții capabile să ofere informații despre poziția sau starea robotului mobil (sau a vehiculului) într-o scenă (cadru) și prin menținerea mai multor ipoteze despre locație, fiecare având probabilitate diferită. Astfel, din punct de vedere matematic procesul de urmărire poate fi definit ca o problemă abstractă de inferență probabilistică, după cum este prezentat pe larg și în cărțile [Forsyth2002], [Thrun2005] și [Danescu2019]. Ideea principală a abordării bazate pe probabilistică este de a estima starea curentă folosind date senzoriale (de măsurare). Estimarea stării se face într-un domeniu continuu și nu discret, de aceea valorile continue aleatorii din estimările probabilistice au o funcție densitate de probabilitate (PDF - “probability density function”), caracterizată de distribuția Gaussiană multi-variată (ecuația 1). În ecuația 1, μ reprezintă vectorul de medii, iar Σ este matricea de covarianță în timp ce x este un vector multidimensional.

$$p(x) = \det(2\pi\Sigma)^{-\frac{1}{2}} exp\left\{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right\} \quad (1)$$

Pentru a putea aplica într-un caz general ecuațiile de urmărire bazate pe inferența probabilistică se aproximează densitățile de probabilitate folosind funcții Gaussiene. Aceste funcții sunt preferate într-un context de urmărire deoarece sunt uni-modale, adică au un singur maxim, iar valoarea de adevăr despre o anumită stare va fi concentrată în jurul acelui maxim cu o anumită marjă de incertitudine.

Densitatea de probabilitate condiționată între este două variabile x și y independente, condiționată de proprietatea că y > 0 este definită în ecuația 2.

$$p(x \mid y) = \frac{p(x)\,p(y)}{p(y)} \quad (2)$$

Totodată se poate defini probabilitatea totală (ecuația 3).

$$p(x) = \sum_y p\ (x \mid y)\ p(y) \quad (3)$$

De asemenea, regula lui Bayes (ecuația 4) este importantă în acest context deoarece se poate defini probabilitatea *p(x | y)* folosind "inversa" *p(y | x)*.

$$p(x) = \frac{p(y \mid x)\ p(x)}{p(y)} \quad (4)$$

De exemplu dacă se calculează starea *x*, din datele senzoriale *y*, atunci probabilitatea *p(x)* este denumită distribuția de probabilitate "*prior*", iar probabilitatea *p(x | y)* este denumită "*posterior*". Probabilitatea *p(y)* nu depinde deloc de *x*, de aceea este folosită pentru normalizare și uneori notată cu η.

Regula lui Bayes poate fi folosită pentru a estima starea robotului (sau a vehiculului) într-un mediu care este perceput și măsurat prin senzori. Dinamica dintre robot și mediu este definită de: distribuția probabilistică a tranziției dintre stări și distribuția probabilistică a măsurătorilor. Prima funcție definește modul în care starea curentă a robotului este actualizată de-a lungul timpului (cadrelor), în timp ce măsurătorile precizează care măsurători *y* sunt generate într-o stare *x*. Măsurătorile pot fi privite ca *"proiecții zgomotoase"* ale stării curente.

Deplasarea robotului produce pierdere de informație, în timp ce percepția mediului (măsurarea) oferă informație nouă despre starea curentă. Urmărirea este un proces dificil deoarece mișcarea unui robot sau vehicul este inexactă, datorită factorilor stohastici care pot interveni într-o lume imperfectă. Această natură stohastică (aleatoare) a mediului este principalul motiv pentru care sunt folosite funcții de densitate de probabilitate pentru calculul stării curente bazate pe starea precedentă, în detrimentul folosirii unor funcții deterministe.

2.1 Filtrul Bayes

Estimarea recursivă Bayes sau filtrul Bayes este un algoritm probabilistic de estimare a funcției densitate de probabilitate recursiv, în care informația despre starea curentă ("belief") x_t la momentul (cadrul) *t*, calculată din informația despre starea precedentă x_{t-1}. Filtrul Bayes se poate implementa ca un proces în doi pași: predicție (ecuația 5) și corecție/actualizare (ecuația 6).

$$\overline{bel}(x_t) = \int p(x_t \mid u_t,\ x_{t-1})\ bel(x_{t-1})\ dx \quad (5)$$

$$bel(x_t) = \eta\ p(z_t \mid x_t)\ \overline{bel}(x_t) \quad (6)$$

În ecuațiile filtrului Bayes u_t reprezintă datele de control, z_t reprezintă datele de măsurare, ambele la momentul *t*. Așadar, avem un flux de observații (măsurători) *z* și de acțiuni (control) *u*. Modelul senzorial este dat de $p(z_t \mid x_t)$, cel de control de $p(x_t \mid u_t,\ x_{t-1})$, iar probabilitatea anterioară *"prior"* a stării sistemului este dată de $p(x_t)$. În ecuația de predicție, calculul integralei are la bază teoria probabilității complete (ecuația 3), dar modificată pentru

spațiul continuu. Starea curentă (*"posterior"*) a sistemului este notată cu *"bel"* (de la "belief"): $bel(x_t) = p(x_t \mid u_1, z_1, .. , u_t, z_t)$. În ecuația 6 *"η"* reprezintă o constantă folosită pentru normalizare.

Definirea stării curente ca fiind bazată complet pe starea precedentă este denumit proces Markov de ordinul întâi ("first order Markov process"), care în robotică este doar o aproximare.

Principalele aplicații și modificări ale estimării recursie Bayes sunt: filtrul Kalman și filtrul de particule care vor fi prezentate în secțiunile următoare.

2.2 Filtrul Kalman

Procesul de urmărire cu filtre Kalman a fost descris pe larg în comunitatea științifică și este aplicat în industrie în principal pentru localizarea, planificarea traiectoriei, monitorizarea și controlul vehiculelor, aeronavelor sau a roboților mobili sau în domeniul telecomunicațiilor pentru filtrarea și estimarea semnalelor, etc. Dezvoltat încă din 1959 de Peter Swerling [Swerling1959] și 1960 de Rudolf Kalman [Kalman1960][Kalman1961], algoritmul are rolul de a elimina zgomotul dintr-o serie de date folosind calcul recursiv. Este o tehnică folosită pentru a estima starea unui sistem liniar, unde starea sistemului poate fi modelată folosind o distribuție Gauss. Filtrele Kalman sunt ideale pentru sisteme într-o continuă schimbare, dinamice și au avantajul că nu necesită stocarea tuturor stărilor precedente ale sistemului, ci doar pe cea mai recentă (ultima stare). De aceea aceste filtre sunt rapide și pot fi folosite pentru urmărire în timp real. Principala limitare este legată de faptul că se poate folosi doar pentru sisteme modelate liniar. Pentru a depăși această limitare se pot folosi filtre Kalman extinse (EKF - Extended Kalman Filters) care pot modela și sisteme neliniare.

Filtrul Kalman folosește două etape: predicția și actualizarea (corecția) stării, ambele folosind ecuații liniare. În continuare voi prezenta ecuațiile de bază ale filtrului Kalman.

Ecuațiile de predicție a filtrului Kalman sunt următoarele: calcularea noii stări *x'* și calcularea covarianței erorii *P'*. Ecuațiile 7-8 reprezintă predicția într-un filtru Kalman.

$$x' = Fx + v \quad (7)$$
$$P' = FPF^T + Q \quad (8)$$

Formula completă a tranziției dintre stări este de fapt: $x' = Fx + Bu + v$. Matricea *B* numită matricea de control a intrării și *u* este vectorul de control. Folosind *Bu* se pot modela forțele interne care acționează asupra obiectului urmărit. De exemplu la urmărirea unui vehicul prin *Bu* se poate modela poziția acestuia ca urmare a accelerării și decelerării cauzate de motorul vehiculului.

Predicția stării se face folosind starea precedentă a sistemului, prin definirea unei funcții de tranziție între stări *F* la care se adaugă un zgomot v. Pentru a modela zgomotul din

proces se folosește matricea *P*, cu rolul de a modela incertitudinea (de exemplu un obiect ar putea accelera sau decelera de la o observație/măsurătoare la alta). Acest zgomot este definit ca o distribuție Gaussiană cu media zero și covarianța *Q*.

Pasul de actualizare/corecție folosește cea mai recentă măsurătoare pentru a corecta estimarea stării și incertitudinea sa. Ecuațiile 9-13 descriu pasul de actualizare (corecție) a filtrului Kalman.

$$y = z - Hx' \quad (9)$$
$$S = HP'H^T + R \quad (10)$$
$$K = P'H^T S^{-1} \quad (11)$$
$$x = x' + Ky \quad (12)$$
$$P = (I - KH)P' \quad (13)$$

În primul pas se compară starea curentă *x'* cu informațiile din senzor *z*. Filtrul Kalman trebuie "să știe" ce este măsurat și cum relaționează măsurătoarea cu starea curentă. Matricea *H* reprezintă modelul de observare și implementează această funcționalitate, practic va proiecta starea curentă a obiectului (certitudinea asupra stării curente *x*) la spațiul de măsurare al senzorului *z*. Incertitudinea cauzată de procesul de măsurare este definit de matricea *R* ca o distribuție Gaussiană cu media zero și covarianța *ω*. Matricea *K*, numită "Kalman gain" combină cele două incertitudini: ale datelor senzoriale din măsurătoare *R* și incertitudinea asupra stării *P'*. Următorul pas constă în actualizarea stării curente a sistemului ("posterior"), iar în final se calculează noua covarianță *P*, pentru viitorul pas de predicție.

2.3 Filtrul Kalman extins

Filtrul Kalman extins EKF ("Extended Kalman filter") este utilizat ca o variantă de a modela sisteme neliniare, unde nu se poate utiliza filtrul Kalman clasic. Sistemele neliniare sunt de fapt aproximate folosind ecuații liniare, prin serii Taylor. Funcția de măsurare *h(x)* este aproximată de o funcție liniară tangentă în locația unde este media funcției. Derivata funcției *h(x)* în funcție de x se numește Jacobian și va reprezenta o matrice unde sunt derivatele parțiale.

În cazul folosirii unui model liniar pentru predicție, ecuațiile filtrului EKF pentru predicție vor fi aceleași. Dacă la tranziția între stări este un model neliniar, atunci funcția *F* este înlocuită de derivatele sale parțiale (matricea Jacobiană) F_j. Ecuațiile 14-15 reprezintă pasul de predicție la filtrul Kalman extins.

$$x' = F_j x + v \quad (14)$$
$$P' = F_j P F_j^T + Q \quad (15)$$

Pentru pasul de actualizare a măsurătorii, ecuația de calculare a diferenței între starea actuală și cea măsurată devine:

$$y = z - h(x') \quad (16)$$

Ecuația 16 descrie formula de calcul a erorii între cele două stări: cea actuală *x'* și cea măsurată *z* folosind funcția de proiecție a măsurătorii *h*.

Funcția *h(x')* specifică proiecția între predicții și conține ecuațiile neliniare. De exemplu în cazul urmării unui vehicul unde se folosește o măsurătoare bazată pe RADAR, datele furnizate de senzor sunt sub forma polară (conțin unghiuri și orientări) și pentru transformarea datelor în sistemul de coordonate cartezian este necesară folosirea unor funcții neliniare (ex. arctangentă). În acest exemplu, funcția *h(x')* va reprezenta transformarea între cele două sisteme de coordonate. În continuare, datorită neliniarității ecuațiile filtrului Kalman clasic pentru calcularea erorii și a matricei *K* (Kalman gain) sunt modificate pentru a folosi derivatele parțiale ale *h(x')*, adică Jacobiană H_j. Actualizarea stării noi se face cu aceeași formulă, iar în ultimul pas calculul noii covariațe *P* se face folosind aceeași matrice Jacobiană H_j.

$$S = H_j P' H_j^T + R \quad (17)$$
$$K = P' H_j^T S^{-1} \quad (18)$$
$$x = x' + Ky \quad (19)$$
$$P = (I - KH_j)P' \quad (20)$$

În ecuațiile 17-20 sunt prezentate formulele de calcul pentru actualizarea măsurătorii într-un filtru Kalman extins. Ecuațiile 16-20 descriu pasul de măsurare (corecție) pentru filtrul Kalman extins.

2.4 Filtre de particule

Soluția pentru probleme non-Gaussiene este folosirea filtrelor de particule. Sisteme bazate pe filtre de particule, denumite și sisteme Monte Carlo secvențiale sunt metode (algoritmi) de tip Monte Carlo utilizați în domeniul de viziune artificială și robotică pentru urmărirea obiectelor sau localizarea roboților mobili. În literatura de specialitate există multe abordări pentru urmărire bazate pe filtre Kalman sau filtre Kalman extinse [Fod2002]. Aceste soluții se pretează unor sisteme liniare, uni-modale și cu zgomot Gaussian. Estimarea stării unui sistem modelat cu filtre Kalman nu oferă precizie și robustețe, deoarece deplasarea obiectelor în scenă nu este liniară. Totuși, filtrele Kalman extinse oferă posibilitatea de a estima mișcarea neliniară, dar folosind aproximări ale unor modele liniare. Principalul avantaj al filtrelor de particule în detrimentul celor bazate pe filtre Kalman este dat de multi-modalitate, ceea ce înseamnă că un filtru de particule poate menține mai multe ipoteze simultan [Gordon1993], deoarece o particulă reprezintă o ipoteză.

Etapele necesare pentru dezvoltarea unui filtru de particule sunt următoarele: inițializarea particulelor, predicția, actualizare ponderi și în final reeșantionarea particulelor. Predicția și actualizarea sunt asemănătoare cu cele din filtrul Bayesian. Filtrul de particule are la bază generarea unui număr total de *N* particule aleatorii, iar fiecare va avea o anumită pondere. În funcție de această pondere o particulă poate fi păstrată în viitor sau va fi eliminată.

Ponderea sau factorul de importantă a unei particule se calculează din eroarea/diferență dintre măsurătoarea prezisă și cea actuală (reală).

Alegerea unui set nou de particule din cele existente în funcție de ponderea lor se numește reeșantionare (“resampling”). Așadar, pentru a implementa un filtru de particule este nevoie de o modalitate de a seta o pondere unei particule și de o metodă de reeșantionare. Lucrarea [Hol2006] prezintă diferiți algoritmi pentru reeșantionare. Aceeași particulă ar putea fi extrasă de două sau mai multe ori din setul inițial, în funcție de ponderea sa. Particulele cu pondere mare vor avea o probabilitate mai mare de a fi păstrate. În general ponderile sunt normalizate, ceea ce înseamnă că suma lor va fi egală cu 1.

Inițializarea particulelor se poate face într-un mod aleator în spațiul de măsurare definit. Predicția constă în calcularea stării “posterior” a sistemului în momentul de timp următor și este definit ca o convoluție (sumă). Actualizarea definește starea “posterior” a sistemului în funcție de măsurătoarea curentă și constă în calcularea ponderilor noi asociate fiecărei particule. Astfel, datele de măsurare care pot fi din LIDAR, RADAR sau din imagini (trăsături relevante) vor fi folosite doar pentru actualizarea ponderilor particulelor și nu asupra predicției despre locația obiectelor urmărite. Totodată, zgomotul cauzat de măsurători este modelat folosind o distribuție Gaussiană și adăugat la fiecare măsurătoare.

CONDENSATION [Isard1998] reprezintă o lucrare de referință unde este prezentat un sistem (“framework”) bazat pe filtre de particule potrivit pentru urmărire de obiecte folosind spațiul de imagine. Lucrarea [Dănescu2011] descrie o implementare bazată pe CONDENSATION, în contextul urmăririi de vehicule. Spațiul de măsurare este implementat sub forma unei hărți bi-dimensionale de ocupare împărțită în celule de dimensiune egală, unde fiecare celulă a hărții poate conține una sau mai multe particule. În pasul de predicție se calculează poziția următoare a obiectului urmărit prin folosirea unor modele dinamice de mișcare. Poziția fiecărei particule este actualizată în harta de ocupare, iar pentru a modela anumite incertitudini din lumea reală, fiecărei particule i se adaugă și un zgomot Gausian.

2.5 Introducere: inteligența artificială

Pentru îmbunătățirea sistemului de percepție a unei mașini autonome folosind o cameră monoculară am realizat detecția suprafeței de drum pe care autovehiculul se poate deplasa (“driveable road area” sau zonele fără obstacole de pe carosabil: “free space detection”). Metodele clasice de detecție a drumului folosind informație de culoare sau textură nu sunt utile în cazul unor scenarii complexe unde sunt diferite condiții de iluminare a carosabilului (umbre, canale, marcaje) sau în scenarii din traficul urban unde suprafața drumului este ocupată de restul de vehicule și este foarte slab vizibilă. Pentru a rezolva aceste dificultăți am ales implementarea unei soluții bazate pe segmentare semantică folosind rețele neuronale. În continuare voi prezenta conceptele de bază ale rețelelor neuronale artificiale și apoi rețele neuronale convoluționale pentru clasificarea imaginilor și segmentare semantică.

Există diferite definiții considerate acceptabile pentru inteligența artificială. Dacă se referă la sisteme cu agenți inteligenți, atunci se poate defini ca fiind capabilitatea unui sistem să perceapă mediul înconjurător și să ia decizii pentru a-și maximiza atingerea obiectivelor. La un mod mai general, inteligența artificială se referă la abilitatea unui sistem de a interpreta date relevante, de a învăța din ele și apoi să le folosească pentru a lua decizii. Inteligența artificială și folosirea rețelelor neuronale artificiale se poate referi si la procesul de a extrage informații relevante din date, cu intervenție umană cât mai redusă.

Domeniile principale de aplicabilitate pentru algoritmi de inteligență artificială sunt: recunoașterea și clasificarea imaginilor (ex: recunoaștere fețe), recunoaștere vocală, domeniul medical (diagnostice), vehicule autonome, asistenți digitali, sisteme de recomandare (muzică, filme, reclame) și multe altele. Principala problemă o reprezintă datele de intrare folosite precum și interpretarea lor: procesul de adnotare a datelor dar și proiectarea unor sisteme capabile să le folosească în mod corect. Sisteme bazate pe inteligență artificială și rețele neuronale au cunoscut o evoluție considerabilă în ultimii ani în special datorită sistemelor hardware capabile și accesibile ca și cost dezvoltate de companii precum Nvidia, Google, Intel și alții.

2.6 Rețele neuronale artificiale

Rețelele neuronale artificiale (en: “artificial neural networks” - ANN) au la baza modelul de neuron biologic: fiecare neuron artificial are asociat date de intrare care sunt ponderate și însumate. Ieșirea rețelei este dată de o funcție de activare sau de transfer, care activează anumiți neuroni pentru a produce o valoare scalară. În funcție de gruparea neuronilor artificiali, se pot obține mai multe variante de arhitecturi de rețele neuronale artificiale: “auto-encoder” [Bengio2009], rețele neuronale convoluționale (CNN) [LeCun1995], [Krizhevsky2012] sau rețele neuronale recurente (RNR), de exemplu rețele de tip LSTM [Hochreiter1997].

Modelul perceptron a fost propus în 1958 [Rosenblatt1958] de Frank Rosenblatt. În Figura 2 este ilustrat neuronul biologic și un model de perceptron. Modelul biologic are un corp celular pentru a procesa impulsurile (semnalele) recepționate de la dendrite și un axon pentru a trimite impulsuri spre alți neuroni. Asemănător, modelul perceptron are mai multe intrări pentru semnale, un mecanism de procesare al lor și o ieșire care se poate conecta la alți neuroni artificiali.

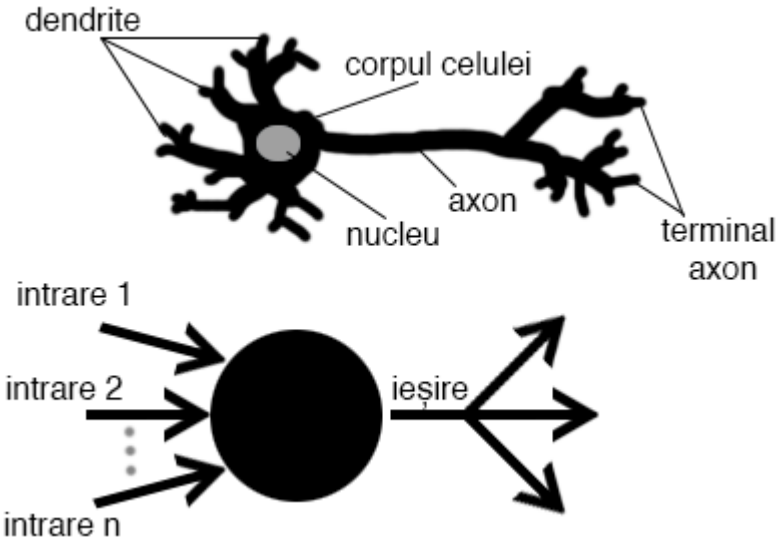


Figura 2. Neuronul biologic și perceptronul.

Perceptronul are un funcționalitate simplă: fiecare element din intrare este înmulțit cu o pondere iar la final toate rezultatele sunt adunate. Această sumă este numită pre-activare. La această sumă se adună o constantă, un deplasament (“bias”). Următorul pas constă în aplicarea unei funcții de activare pentru calculul din pasul anterior. Într-un perceptron, funcția de activare este numită și funcție prag, practic ieșirea acestui model este binară: “1” dacă rezultatul este peste un prag, altfel va fi “0”. Ponderile sunt numere reale cu rolul de a exprima importanța fiecărei intrări în raport cu ieșirea. Ecuația 21 reprezintă ecuația de calcul a sumei ponderate a intrărilor unui neuron.

$$y = \sum_{i=0}^{N} W_i X_i + b \qquad (21)$$

Acest model s-a dovedit ineficient pentru învățarea și recunoașterea unor date mai complexe, perceptronul fiind util doar pentru date separabile liniar (de exemplu punctele dintr-un plan 2D). În ecuația 21 se poate observa inclusiv faptul că funcția de actualizare a ponderilor unui neuron este asemănătoare cu ecuația unei drepte. În 1969 o carte numită “Perceptrons” [Minsky1969] a dovedit că era imposibil ca modelul simplu de perceptron să învețe funcționalitatea unei porți logice de tip ”XOR” (sau exclusiv). Aceste limitări majore au dus la o stagnare a dezvoltării rețelelor neuronale, iar abia în 1974 prin algoritmul “backpropagation” [Werbos1974] s-au adus îmbunătățiri majore în domeniu. Această lucrare de doctorat prezintă ideea folosirii a mai multor straturi în rețelele neuronale și propune folosirea unor funcții diferite de activare (transfer). În 1991 s-a publicat o lucrare [Hornik1991] referitoare la teorema aproximării universale, în care s-a dovedit matematic faptul că majoritatea funcțiilor continue pot fi aproximate folosind o rețea neuronală cu un singur strat ascuns și cu un număr finit de neuroni. Totodată, una din primele lucrări referitoare la teorema aproximării universale a fost publicată cu doi ani înainte, în 1989 [Cybenko1989] și a dovedit teorema pentru aproximarea funcțiilor de activare sigmoide.

Rețelele neuronale cu mai multe straturi, numite și perceptron multi-strat (“multilayer perceptron” MLP) conțin un nivel de intrare, unul sau mai multe niveluri ascunse (”hidden layers”) și un strat de ieșire. Așadar, aceste rețele vor produce un anumit rezultat în funcție datele de intrare care sunt pasate în primul nivel. De aceea, aceste rețele sunt denumite și

“feed-forward networks”: ieșirea unui nivel va reprezenta intrarea pentru următorul nivel și așa mai departe.

Spre deosebire de rețelele de tip “feed-forward” sau perceptron multi-strat, rețelele neuronale convoluționale (“convolutional neural networks” - CNN) pot captura și dependențe spațiale dintr-o imagine prin aplicarea unor anumite tipuri de filtre, numite kernel. Într-o abordare bazată pe CNN imaginea de intrare nu este transformată într-un vector uni-dimensional, ci este folosită direct având: lățime, înălțime și numărul de canale. Prin operația de convoluție cu un kernel de dimensiunea mai mică decât imaginea de intrare se va reduce informația din imagine și se păstrează doar trăsăturile importante. Convoluția unei imagini de dimensiune 5 x 5 cu un kernel 3 x 3 este ilustrată în Figura 3.

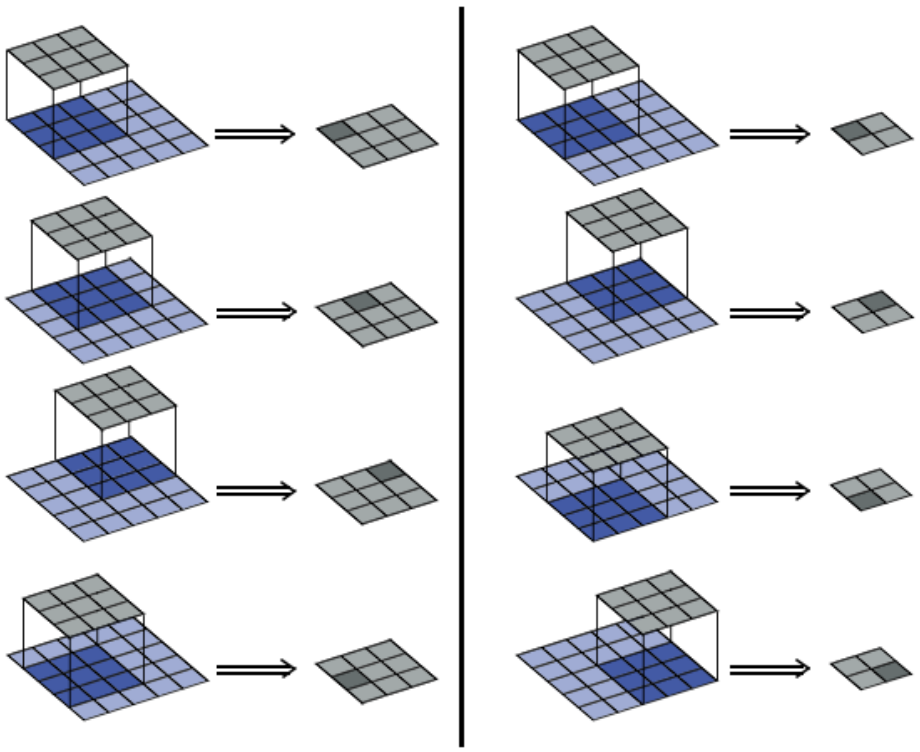


Figura 3. Exemplificare convoluție pe imaginea sursă de dimensiune 5 x 5 cu un kernel 3 x 3. În coloana din stânga: primele 4 convoluții (dintr-un total de 9) cu pas (“stride”) de 1, în coloana din dreapta: convoluție cu “stride” de 2 (4 operații în total).

Operația de convoluție se aplică din colțul din stânga sus al imaginii similar cu o mecanismul de fereastră glisantă, iar elementele din filtru sunt înmulțite cu cele din imagine și apoi însumate. Nucleul (”kernel”) este aplicat până se ajunge la capătul imaginii (lățimea ei) și apoi se reîncepe de la stânga, dar pornind cu linia a doua și așa mai departe până ce se parcurge întreaga imagine sursă. Deplasarea filtrului de la o convoluție la alta se poate face și cu un pas (“stride”) mai mare, dar în general este 1 (“no stride”). Figura 3 exemplifică în coloana din stânga primele 4 operații de convoluție cu un “stride” de 1 și în coloana din dreapta convoluția cu “stride” de 2. Se remarcă faptul că pasul (“stride”) modifică dimensiunea finală a imaginii. De asemenea, se poate aminti și faptul că se poate umple imaginea originală pentru a nu reduce dimensiunea sa, iar operația de umplere este denumită “padding”. Dacă imaginile

sunt pe mai multe canale (de exemplu imagini color pe 3 canale), atunci și nucleul va fi de aceeași dimensiune, se aplică operațiile la fel ca în variantă cu un singur canal, iar la final se însumează rezultatele pentru a obține o imagine (hartă) de trăsături numită “feature map”.

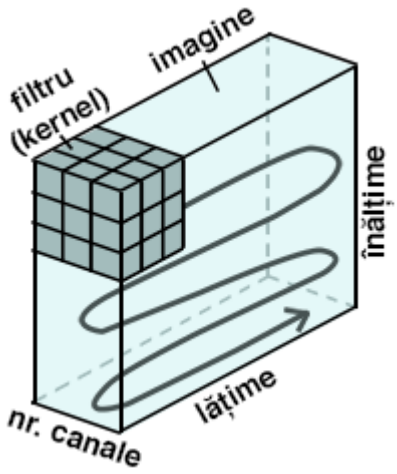


Figura 4. Deplasarea unui kernel într-o imagine în timpul operației de convoluție.

De asemenea, într-un strat convoluțional se pot aplica mai multe filtre. Prin generalizare, având o imagine de dimensiune n × n și un număr n_c de filtre de dimensiune k × k × n_c atunci dimensiunea imaginii de ieșire se poate calcula folosind ecuația 22, unde *s* reprezintă pasul (“stride”), *p* este valoarea de umplere (“padding”) și dimensiunea filtrului este notată cu *k*.

$$dim = \left[\frac{(n+2p-k)}{s} + 1\right] * \left[\frac{(n+2p-k)}{s} + 1\right] * n_c \qquad (22)$$

În rețele convoluționale neuronale mai există de asemenea și operațiunea de agregare (“pooling”), care în general este de două tipuri: agregarea maximelor (“max pooling”) și agregarea mediilor (“average pooling”). În prima variantă se calculează maximum pe o fereastră de căutare, iar în a doua se calculează media aritmetică din fereastră. Aceste două tehnici sunt folosite pentru a reduce și mai mult dimensiunea unei hărți de trăsături, în timp ce se păstrează activările (trăsăturile) dominante.

Straturile complet conectate (“fully connected layers” - FC) sunt adăugate de obicei la final în structura unei rețele neuronale convoluționale. O hartă de trăsături, adică ieșirea unui strat convoluțional este rearanjată într-un vector uni-dimensional (“flattened”) similar unei rețele “multi-layer perceptron”, unde fiecare neuron este interconectat. Lungimea acestui vector uni-dimensional este egală cu numărul de clase pe care rețeaua le identifică. Prin antrenare, ponderile neuronilor din stratul complet conectat sunt modificate pentru a obține o acuratețe cât mai mare pentru fiecare clasă care se antrenează în funcție de trăsăturile specifice. Printr-un nivel FC rețeaua va învăța combinațiile neliniare dintre trăsături, practic va face partea de clasificare dintr-o rețea convoluțională neuronală, în timp ce extragerea trăsăturilor este făcută de straturile de convoluție împreună cu cele de agregare.

Pentru a antrena rețele neuronale artificiale avem nevoie de o metrică de evaluare a performanței rețelei, mai specific de o funcție de cost. În cadrul acestui capitol voi prezenta funcțiile de cost folosite în mod uzual atât pentru clasificare, cât și pentru regresie.

2.7 Clasificare și regresie

Sisteme de învățare automată se pot împărți în două: învățare supervizată și nesupervizată. Conceptul de învățare supervizată se referă la utilizarea unei baze de date de antrenare pentru a genera o funcție de inferență care proiectează aceste date de intrare la cele de ieșire. Astfel, algoritmii de învățare vor genera predicții pe date noi de intrare, iar aproximarea funcției de inferență trebuie să fie cât mai precisă. Învățarea automată poate fi și nesupervizată unde algoritmii au ca intrare date care nu sunt adnotate și despre care nu se cunosc informații, iar învățarea se face prin diferite tehnici de agregare și asociere a datelor.

Rețelele neuronale artificiale folosesc învățare automată supervizată și se pot folosi pentru sarcini de clasificare sau regresie, în funcție de tipul de date folosit. Principala diferență între cele două este dată de domeniul funcției de inferență. Pentru regresie funcția de inferență este o funcție continuă, în timp ce ieșirea unei clasificări este o valoare discretă (care aparține unei categorii), funcția de inferență fiind definită în spațiul discret.

Algoritmii de regresie vor estima funcția de inferență *f* care proiectează datele de intrare *x* la valorile de ieșire numerice continue *y*. Valorile generate de funcția *f(x) = y* sunt pretabile pentru probleme de estimare a unor mărimi sau cantități. De exemplu pentru generarea unghiului de înclinare a camerei sau pentru estimarea coordonatelor punctului de fugă într-o imagine se poate folosi regresia.

Algoritmii de clasificare estimează funcția de inferență *f(x) = y* din valorile de intrare *x* și valorile de ieșire discrete *y*. Spațiul discret reprezintă categoria din care valorile y fac parte. De exemplu recunoașterea tipului de mijloc de transport este o problemă de clasificare, unde categoriile sunt definite clar: vehicule, autobuze, camioane, tramvai, bicicleta și așa mai departe.

Funcțiile de cost definite în cadrul acestui capitol sunt cele care vor face diferența la antrenarea eficientă a rețelelor neuronale, iar alegerea unei funcții de cost optime este condiționată de tipul de învățare supervizată: clasificare sau regresie.

2.8 Funcții de activare

În esență, un neuron artificial va calcula o sumă ponderată a intrării la care va adăuga un deplasament (“bias”), iar apoi printr-o funcție de activare se decide dacă neuronul este

declanșat (activat) sau nu. Există diferite funcții de activare, printre cele mai des utilizate fiind: funcția liniară, funcția sigmoidă, funcția liniară rectificată (“ReLU - Rectified Linear Unit” [Nair2010]), funcția tangentă hiperbolică (“tanh”). În continuare voi prezenta principalele funcții de activare folosite pentru rețele neuronale artificiale.

2.8.1 Funcția liniară

Funcția liniară propagă funcția fără a o altera: *g(f(x)) = f(x)*, practic neuronul este declanșat proporțional cu valoarea datelor de intrare (*f(x)*). Totuși, funcțiile de activare liniare nu sunt utile pentru rețele cu mai multe straturi, deoarece indiferent cum se realizează conexiunile între neuroni, funcția de activare a stratului final nu va fi altceva decât o funcție liniară compusă din funcțiile liniare din straturile precedente. Aceasta înseamnă că straturile pot fi reduse și înlocuite cu funcții liniare, iar întreaga rețea va deveni de fapt similară cu un “perceptron” și va avea limitările acestuia.

2.8.2 Funcția de activare sigmoidă

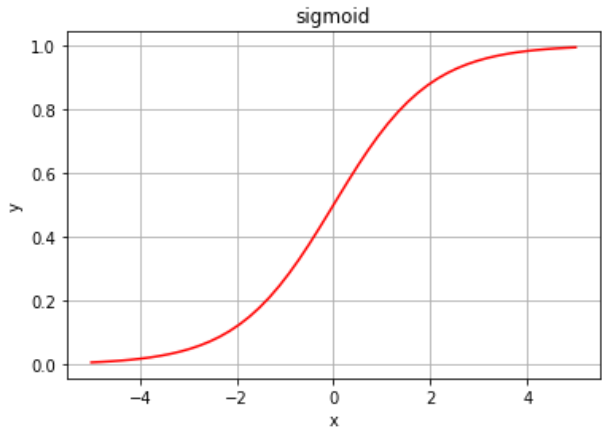


Figura 5. Funcția sigmoidă.

Funcția de activare sigmoida este o alternativă mai bună deoarece va avea valori în intervalul [0, 1], spre deosebire de funcția de activare liniară care are valori cuprinse între (-∞, +∞). Ecuația 23 reprezintă formula de calcul a funcției sigmoide.

$$S(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{e^x+1} \tag{23}$$

Un alt avantaj al funcției sigmoide este că e neliniară, astfel combinațiile între mai multe funcții sigmoide va fi neliniară. Derivata acestei funcții este de asemenea ușor de calculat, și este mai utilă decât cea a funcției liniare. Totuși, din figura 5 se poate observa că spre capetele intervalului răspunsul funcției sigmoide este nesemnificativ, ceea ce ar putea duce spre problema numită “vanishing gradient” - în practică aceasta înseamnă o învățare mai lentă a

rețelei deoarece uneori gradienții nu se schimbă suficient încât să determine o activare nouă a unui neuron.

2.8.3 Funcția de activare “tanh”

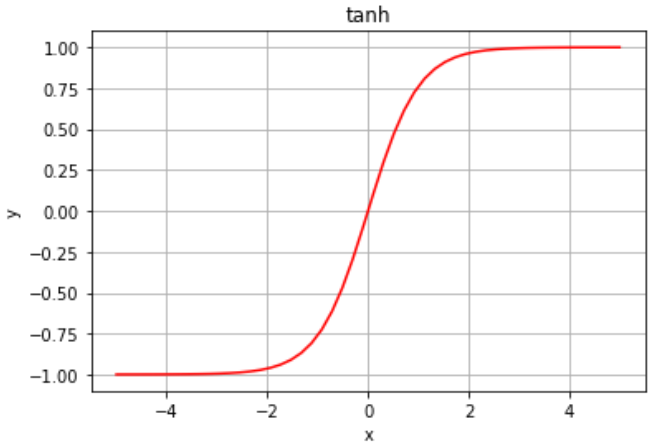


Figura 6. Funcția de activare “tanh”.

Funcția de activare “tanh” (ecuația 24) este similară cu cea sigmoidă. Practic este funcția sigmoidă scalată, deoarece are valori cuprinse în intervalul [-1, 1].

$$tanh(x) = \frac{2}{1+e^{-2x}} - 1 = 2\,sigmoid(2x) - 1 \qquad (24)$$

Avantajul folosirii “tanh” în detrimentul funcției de activare sigmoidă este că ieșirile funcției “tanh” sunt centrare în jurul valorii “0”. Funcția “tanh” este folosită pentru clasificare binară (între două clase) și va avea aceleași limitări la fel ca și funcția sigmoidă.

2.8.4 Funcția de activare ReLU (“rectified linear unit”)

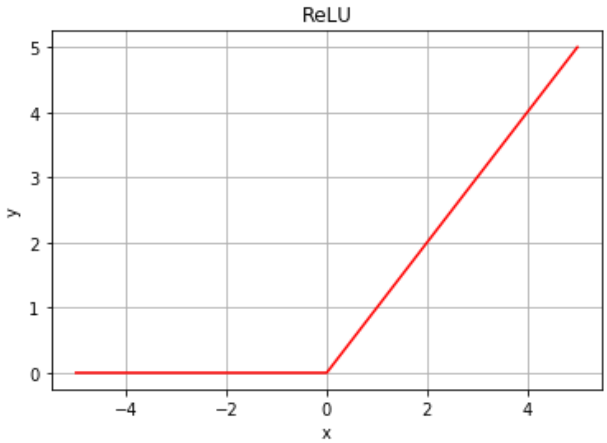


Figura 7. Funcția de activare ReLU.

Funcția de activare liniară rectificată ReLU (“rectified linear unit”) este de asemenea o funcție neliniară. Intervalul de valori este definit între [0, ∞), iar un avantaj față de funcția sigmoidă sau “tanh” este că reduce dimensiunea rețelei și a numărului de neuroni activi. Ieșirea funcției de activare va fi egală cu 0 pentru valori negative ale intrării. De asemenea, operațiile matematice nu sunt așa de complexe comparativ cu funcția sigmoidă sau “tanh”.

2.8.5 Funcția “softmax”

Funcția “softmax” poate fi utilizată cu rol de activare în rețele neuronale și este folosită pentru clasificare, având la bază funcția exponențială. Această funcție va transforma intrările numite “logits” în probabilități (suma lor fiind egală cu 1). Avantajul acestei funcții este dată de faptul că poate fi utilizată pentru clasificarea mai multor clase datorită intervalului de valori returnate de funcție: [0, 1]. Ecuația 25 reprezintă calculul funcției “softmax”.

$$S(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}} \tag{25}$$

Funcția “softmax” este de fapt o formă mai generalizată a funcției de activare sigmoidă.

2.9 Inițializarea ponderilor și ajustarea lor

În procesul de antrenare a rețelelor neuronale de tip convoluțional, se urmărește reducerea (minimizarea) erorii. Există o varietate de funcții de cost pentru minimizare a erorii (“loss function”), dar în general mecanismul pentru a găsi minimul este implementat de algoritmul “gradient descent” (GD), iar modalitatea de a propaga ponderile din fiecare strat din rețea se face prin algoritmul “backpropagation”. Rolul acestor funcții de optimizare (GD sau altele) este de a minimiza funcția de cost: găsirea unei valori cât mai mici pentru funcția de cost în timp ce acuratețea are o valoare mai mare. De exemplu, dacă se proiectează toate valorile unei funcții de cost tridimensională vom obține un model 3D (ilustrat în Figura 8). Rolul funcției de optimizare GD este a găsi cel mai “jos” punct din modelul 3D, mai specific găsirea minimului global.

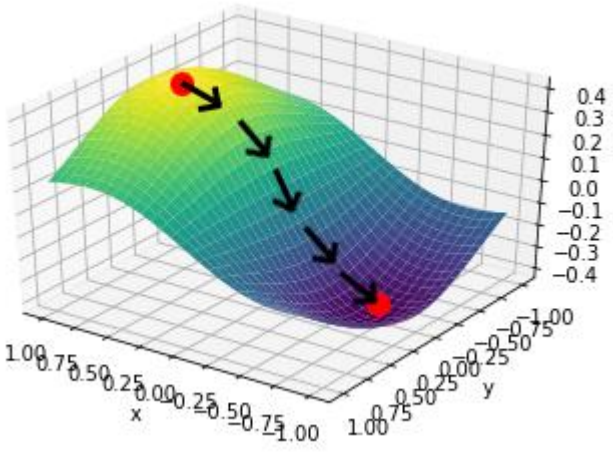


Figura 8. Exemplu de funcție proiectată în spațiul 3D și găsirea valorii minime.

Direcția cu rata de creștere cea mai mare a unei funcții este dată de gradientul funcției. Pașii algoritmului de optimizare "gradient descent" sunt următorii: pentru a găsi minimul funcției de cost se face o parcurgere a funcției în direcția opusă a gradientului până când valoarea funcției scade și se iterează până când valoarea de minim nu se mai schimbă. Ponderile și deplasamentele sunt inițializate la început cu valori aleatorii.

Pentru găsirea minimului local al unei funcții într-un punct, direcția de căutare este dată de negativul valorii gradientului în acel punct, care este înmulțită cu o constantă γ, numită rată de învățare. Gradientul într-un punct al unei funcții reprezintă un vector care indică direcția celei mai mari rate de creștere a funcției, iar modulul gradientului reprezintă rata de creștere (magnitudinea). Ecuația 26 reprezintă modul de calcul pentru "gradient descent" în care gradientul funcției *x(k)* este notat cu ∇f.

$$x^{(k+1)} = x^{(k)} - \gamma \nabla f(x^{(k)}) \qquad (26)$$

Într-o rețea artificială neuronală pentru a minimiza eroarea funcției de cost în timpul antrenării este necesar un mecanism de calculare a erorii în funcție de imaginea de intrare și de ponderile și deplasamentele din fiecare neuron al rețelei. Calcularea erorii fiecărui neuron se face începând de la ultimul strat al rețelei și apoi penultimul și așa mai departe, prin folosirea regulii "lanțului". O problemă majoră este că parametrii și ponderile din straturi nu sunt conectate direct la funcția de cost. Rezultatul din funcția de cost va depinde de ponderea din stratul precedent, astfel că pentru a putea actualiza ponderile din straturile precedente e nevoie să se înceapă de la ultimele straturi din rețea, deoarece aici funcția de cost este apropiată de ponderi ("weight") și deplasamente ("bias") și e nevoie să se parcurgă în sens invers rețeaua (de aici și numele de propagare înapoi).

Considerând o rețea neuronală artificială cu *L* straturi, în care A_L este predicția rețelei (ieșirea ultimei activări din rețea din ultimul strat), *Y* este "label" (din setul de antrenare). Rețeaua are funcția de activare sigmoidă (ecuația 27), iar un neuron este caracterizat de ecuația 28.

$$A_N = \sigma(Z_N) \quad (27)$$

$$Z_N = W_N X + b_N \quad (28)$$

Ecuațiile principale pentru a implementa algoritmul “backpropagation” pentru acest exemplu sunt următoarele:

$\frac{\partial J}{\partial Z_L} = \frac{\partial J}{\partial A_L} \cdot \frac{\partial A_L}{\partial Z_L} = \frac{1}{m}(A_L - Y) \odot \sigma'(Z_L)$, unde $\sigma'(x) = \sigma(x)\big(1 - \sigma(x)\big)$, iar $\odot$ reprezintă înmulțirea matricilor element cu element (29)

$$\frac{\partial J}{\partial Z_i} = \frac{1}{m} W_{i+1}^T \cdot \frac{\partial J}{\partial Z_{i+1}} \odot \sigma'(Z_i) \quad (30)$$

$$\frac{\partial J}{\partial W_i} = \frac{\partial J}{\partial Z_i} \cdot A_{i-1}^T \quad (31)$$

$\frac{\partial J}{\partial b_i} = \frac{1}{m} \sum_i \frac{\partial J}{\partial Z_i}[i]$, unde $\frac{\partial J}{\partial Z_i}[i]$ este coloana *i* din $\frac{\partial J}{\partial Z_i}$ (32)

Folosind regula “lanțului” (“chain rule”) se calculează derivata parțială (gradientul) pentru funcția de cost *J* în funcție de neuronul Z_L (ecuația 29). În ecuația 30, dacă se cunoaște derivata parțială a funcției de cost J în funcție de Z_{i+1}, se poate determina derivata parțială a *J* în funcție de Z_i. Gradienții W_i și b_i (ecuațiile 31 și 32) sunt cei care sunt folosiți pentru actualizări în fiecare neuron al rețelei. Având ecuațiile 29-32 se pot actualiza ponderile și deplasamentele fiecărui strat din rețea în momentul antrenării, începând de la ultimul strat al rețelei. Motivul principal pentru a utiliza “backpropagation” folosind “gradient descent” se datorează eficienței computaționale în calculul derivatelor parțiale pentru fiecare strat.

În timpul antrenării abia după ce se parcurg toate datele se face actualizarea ponderilor, ceea ce nu este neapărat o problemă pentru bazele de date mici. În cazul bazelor de date mari acest lucru ar însemna o învățare foarte lentă, deoarece se așteaptă parcurgerea tuturor datelor pentru o singură actualizare a “gradient descent”. O posibilă soluție este dată de folosirea unui lot mic de date alese aleator pentru a calcula o medie a gradientului și a actualiza ponderile. Se procedează asemănător pentru restul datelor de antrenare, iar când s-a parcurs întreg setul de date înseamnă că s-a terminat o epocă de antrenare. Această variantă a GD care folosește mini-loturi de date se numește “stochastic gradient descent” (SGD) și are avantajul de a oferi o performanță sporită la antrenarea rețelei. Diferența între cele două variante este ilustrată în Figura 9.

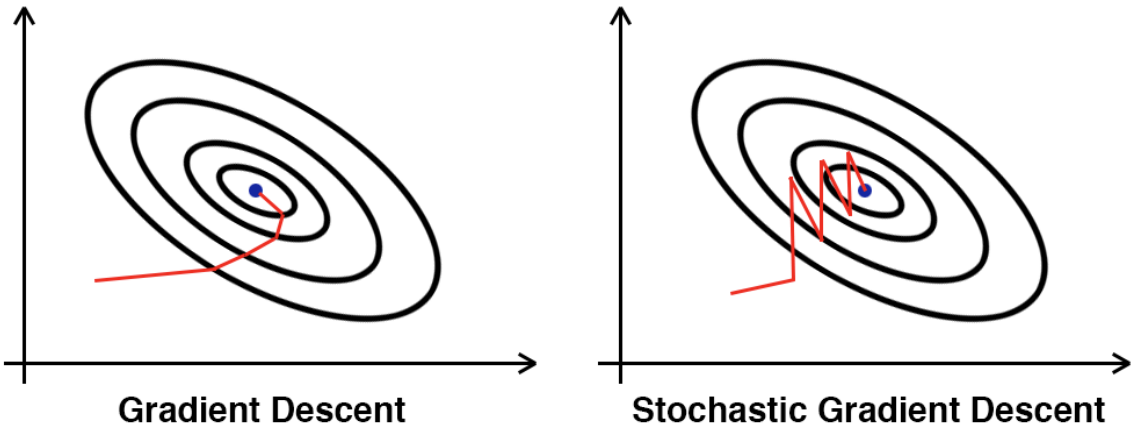


Figura 9. Gradient Descent versus Stochastic Gradient Descent.

Cantitatea cu care se ajustează ponderile după ce se calculează derivata este dată de rata de învățare (“learning rate”), un parametru care se setează inițial (de obicei valori mici 0.00001). Rata de învățare se poate ajusta automat în timpul antrenării rețelei folosind anumiți algoritmi de optimizare (de exemplu Adam [Kingma2015], o extensie a algoritmului SGD).

În termeni specifici, o epocă (“epoch”) se referă la o iterație pe toate datele de antrenare. Astfel, un pas de antrenare reprezintă o actualizare a gradientului. Dimensiunea lotului de date (“batch size”) dintr-o epocă se poate seta în funcție de resursele hardware disponibile, mai exact în funcție de memoria GPU disponibilă.

Normalizarea lotului (“batch normalization”) are rolul de a îmbunătăți stabilitatea rețelei și reduce timpii de antrenare. Prin normalizarea lotului practic se normalizează ieșirile stratului precedent de activare prin scăderea mediei lotului curent și apoi împărțirea la deviația standard a lotului. Astfel, intrarea unui nou strat al rețelei este normalizată prin scalarea și ajustarea ieșirilor funcției de activare din stratul precedent. Există situații în care trăsăturile pot avea valori cuprinse în intervale numerice mici (ex: 0-10) sau alte trăsături pot fi cuprinse în intervale mult mai mari (0-5000), iar în aceste situații este utilă normalizarea acestor valori. Prin normalizare se va obține o îmbunătățire considerabilă a vitezei de antrenare. Ponderile din straturile de după normalizarea lotului nu sunt optime, iar algoritmul “stochastic gradient descent” va anula această normalizare în situația în care va ajuta pentru a reduce/minimiza funcția de cost.

2.10 Funcții de cost pentru rețele neuronale

În rețelele neuronale, eroarea este calculată ca fiind diferența dintre ieșirea reală (“ground truth”) și ieșirea rețelei (“prediction”). Funcția folosită pentru determinarea acestei erori se numește funcție de cost sau de minimizare a erorii la învățare, numită și “loss” în engleză. Această funcție are rolul de a ajuta în procesul de învățare supervizată a rețelelor neuronale. Pentru aceleași date de intrare, folosirea unor funcții de cost diferite va produce rezultate diferite și va influența performanța rețelei mai ales în procesul de antrenare. În funcție

de tipul de antrenare supervizată sunt alese funcțiile de cost, unele sunt pretabile pentru regresie, iar altele sunt utile pentru clasificare sau segmentare de imagini.

Funcțiile de cost utilizate pentru regresie sunt cele care pot oferi valori numerice continue, cum ar fi: “Mean Squared Error” (MSE), “Mean Absolute Error” (MAE) sau variații ale acestora sau alte funcții asemănătoare.

2.10.1 Funcția de cost MSE și RMSE

Funcția de cost “Mean Squared Error” (MSE) este des utilizată deoarece este ușor de implementat și funcționează bine pe diferite tipuri de date. Pentru a determina MSE se calculează diferența între predicții și datele originale, apoi se ridică la pătrat rezultatul (ecuația 33). Se aplică acest calcul pentru toată baza de date și apoi se normalizează datele.

$$\mathcal{L} = \frac{1}{n}\sum_{i=1}^{n}(y^{(i)} - \hat{y}^{(i)})^2 \quad (33)$$

RMSE (“root mean squared error”) este o variație a funcției MSE, unde se aplică și operația de radical de ordinul 2 înainte de normalizare. MSE este numită și “L2”, iar uneori formula diferă deoarece datele nu sunt normalizate (ecuația 34).

$$\mathcal{L} = \sum_{i=1}^{n}(y^{(i)} - \hat{y}^{(i)})^2 \quad (34)$$

2.10.2 Funcția de cost MAE

Funcția de cost “Mean Absolute Error” diferă față de MSE deoarece nu se folosește ridicarea la pătrat, iar la diferența între date se ia valoarea absolută (modulul).

$$\mathcal{L} = \frac{1}{n}\sum_{i=1}^{n}\left|y^{(i)} - \hat{y}^{(i)}\right| \quad (35)$$

MAE și MSE sunt două funcții asemănătoare, dar MSE este mai ușor de folosit în calculul gradientului. Totuși, un avantaj al MAE în detrimentul funcției de cost MSE este dat de faptul că MAE este mai puțin sensibilă la valori extreme (“outlier”) deoarece nu se folosește ridicarea la pătrat. În mod similar cu MAE este funcția de cost “L1”, care este denumită și “least absolute errors” deoarece minimizează suma diferențelor absolute valorile prezise de rețea și datele inițiale (ecuația 36).

$$\mathcal{L} = \sum_{i=1}^{n}\left|y^{(i)} - \hat{y}^{(i)}\right| \quad (36)$$

2.10.3 Funcția de cost “Huber”

Funcția de cost Huber publicată în [Huber1964] este asemănătoare cu funcția “L1”, iar modul de calcul este prezentat în ecuația 37.

$$\mathcal{L}_\delta\big(y, f(x)\big) = \begin{cases} \frac{1}{2}(y - f(x))^2 \text{ , dacă } |y - f(x)| \le \delta \\ \delta|y - f(x)| - \frac{1}{2}\delta^2, \text{ altfel} \end{cases} \quad (37)$$

În ecuația 37 “δ” reprezintă un parametru care poate fi setat. Dacă se alege $\delta = 1$, atunci se obține o variație a funcției Huber numită “smooth L1”, care a fost folosită în lucrarea [Girshick2015]. Autorii menționează faptul că această funcție de cost este mai puțin sensibilă la valori extreme (“outlier”).

2.10.4 Funcția de cost “hinge” sau “multi-class SVM loss”

Funcția de cost “hinge” este folosită pentru clasificare și are proprietatea de a penaliza predicțiile când acestea sunt incorecte, dar și când sunt corecte dar nu au o probabilitate suficient de mare. Formula de calcul pentru funcția de cost “hinge” sau “SVM loss” este definită în ecuația 38, unde ”s” reprezintă vectorul de scor ($s_i = f(x_i, W)$), iar ”x_i” reprezintă imaginea și ”y_i” reprezintă adnotarea (“label”).

$$\mathcal{L}_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1) \quad (38)$$

În teorie, probabilitatea (scorul) predicției clasei corecte ar trebui să fie mai mare decât suma probabilităților predicțiilor claselor incorect clasificate, cu o anumită marjă de siguranță (în general 1).

2.10.5 Funcția de cost “cross entropy” și “binary cross entropy”

Pentru situația în care este nevoie de o clasificare este utilizată funcția de cost “cross entropy” (denumită și “log loss”) sau “binary cross entropy”. Pentru antrenarea de rețele de segmentare semantică poate fi utilizat calculul de entropie Shannon [Shannon1949] (o măsură a incertitudinii) la nivel pixel. Folosind această metodă, se caută minimizarea erorii dintre valoarea fiecărui pixel din predicție și valoarea efectivă din clasa de etichete. Fiecare clasă este reprezentată o imagine de etichetă separată și de obicei stocată într-un vector. Dacă există doar două clase, atunci funcția de cost este denumită “binary cross-entropy” deoarece se face o clasificare binară. Totodată, funcția “cross entropy” este folosită împreuna cu funcția de activare “softmax” pentru a calcula eroarea.

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N} y_i \log\big(p(y_i)\big) + (1 - y_i) \log(1 - p(y_i)) \quad (39)$$

În ecuația 39 este prezentat modul de calcul pentru funcția de cost “binary cross entropy”, unde ”y_i” reprezintă adnotarea (“label”) și poate fi “0” sau “1”, iar $p(y_i)$ reprezintă predicția rețelei (probabilitatea ca ”y_i” să fie parte din clasă).

Se menține o învățare egală la nivelul fiecărui pixel din imagine deoarece funcția de “cross entropy” evaluează predicțiile pentru fiecare vector de pixeli individual și apoi calculează o medie pentru toți pixelii. Totodată, se pot introduce erori la învățare în situațiile în care datele de intrare nu sunt echilibrate și există o discrepanță între numărul de imagini de antrenare pentru fiecare clasă în parte. În aceste situații, procesul de antrenare a rețelei va fi afectat și dominat de clasă cu cele mai multe date. Lucrarea [Ronneberger2015] prezintă o schemă de ponderare a funcțiilor de minimizare a erorilor, unde pixelii la marginea obiectelor din clase diferite vor avea o pondere mai mare. Aceeași idee a fost prezentată inițial și în [Long2015] cu scopul de a contracara situația claselor dezechilibrate din datele de intrare.

2.10.6 Funcția de cost IoU

O altă metrică populară pentru calcularea erorii la învățare este calcularea indexului Jaccard cunoscut și sub denumirea de “Intersect over Union” (IoU), care favorizează predicțiile rețelei care se suprapun cel mai bine peste etichetele din datele de antrenare. Această metrică va calcula intersecția împărțită la reuniunea dintre ieșirea rețelei convoluționale (predicția) și eticheta din intrare a rețelei (“label”). Calculul se face pentru fiecare clasă de obiect segmentat separat. De exemplu în situația în care este o singură clasă (ex. zona de drum), predicția rețelei va fi o imagine în care zona corespunzătoare clasei de obiect segmentat (ex. zona de drum) va avea valori mari (de obicei apropiate de 1). Imaginea “label” va avea pixeli adnotați în care clasa de obiect este reprezentată cu valoarea 1, iar restul este marcat cu 0. Intersecția se va calcula între cele două imagini doar pentru pixelii de clasă de obiect, la fel și pentru reuniune. Rezultatul calculului IoU este în intervalul 0 și 1, unde 1 reprezintă o suprapunere perfectă între imagini. Astfel, pentru a minimiza eroarea la antrenare a unei rețele neuronale, este necesar să minimizăm negativul acestui rezultat. Ecuația 40 prezintă modul de calculare al indexului Jaccard.

$$J = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|} \tag{40}$$

Similar cu IoU se poate utiliza și coeficientul Sorensen-Dice [Sørensen1948], [Dice1945]. Formula de calcul este prezentată în ecuația 41.

$$SD = \frac{2|A \cap B|}{|A| + |B|} \tag{41}$$

Aceste două metrici (IoU și coeficientul SD) sunt cel mai adesea folosite în rețele de segmentare semantică a imaginilor.

2.11 Segmentare semantică folosind rețele neuronale convoluționale

Folosirea unor tehnici de segmentare semantică este utilă pentru a avea o imagine de ansamblu a scenei din traficul rutier, dar este utilă și pentru analiza imaginilor medicale pentru oferirea de diagnostice: de exemplu identificarea celulelor canceroase în imagini microscopice [Mocan2018], tomografice sau rezonanță magnetică RMN. Spre deosebire de metodele tradiționale, segmentarea semantică se referă la asignarea fiecărui pixel din imagine la o clasă de obiect. În mod clasic, segmentarea s-a realizat folosind tehnici de procesare a imaginilor pentru a genera segmente de linii (sau clustere/poligoane), dar fără a avea informații despre clasa obiectului. Segmentarea semantică are rolul de a înțelege clasele de obiecte dintr-o scenă și are la bază detecția și clasificarea trăsăturilor relevante din imagini. Detecția acestor trăsături se poate face prin folosirea informațiilor de intensitate ale pixelului, histograma orientării gradienților ”HoG” [Dalal2005], trăsături ”SIFT” [Lowe2004], ”Harrs” [Harris1988], ”Fast” [Rosten2005], etc. Înainte de popularitatea metodelor folosind inteligența artificială (“deep learning”), în literatură s-au folosit tehnici bazate pe “random forest classifiers” [Shotton2011] sau “texton forest” [Shotton2008]. Cele mai simple metode au la bază aplicarea de praguri în imagini de tip grayscale, în special cu aplicații în domeniul medical unde imaginile sunt capturate folosind computere tomograf (CT) sau RMN. Alte metode clasice folosesc informațiile din muchii sau tehnici de creștere a regiunilor [Nock2004]. “K-means clustering” reprezintă o tehnică de învățare nesupervizată care a fost utilizată pentru segmentare [Dhanachandra2015], în timp ce pentru învățare supervizată s-a folosit “Support Vector Machines” (SVM).

O primă abordare bazată pe rețele neuronale convoluționale a fost [Ciresan2012] unde autorii au folosit ferestre mai mici (“patch-uri”) pentru a genera predicții pe imaginea de intrare. Această constrângere era cauzată de necesitatea unei imagini fixe de intrare pentru CNN-ul de clasificare deoarece folosea niveluri “fully connected”. Rețelele neuronale pentru segmentare semantică au luat amploare după apariția rețelei “Fully Convolutional Networks” (FCN) publicată în lucrarea [Long2015], care oferă predicții dense la nivel de pixel fără a utiliza straturi complet conectate (“fully connected layers”).

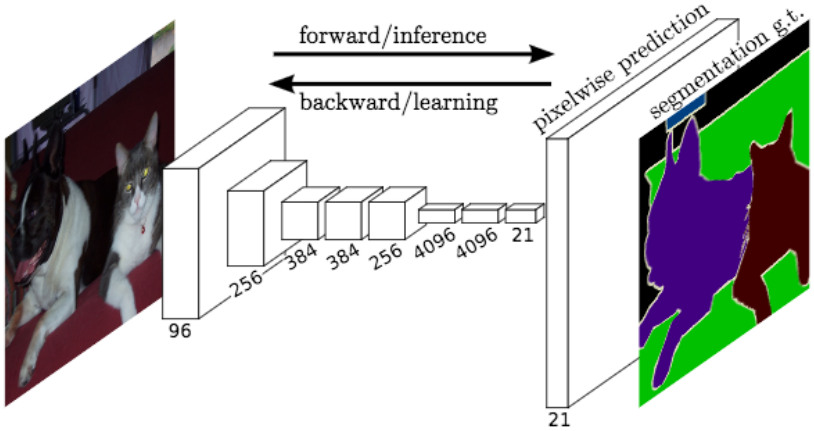


Figura 10. Arhitectura rețelei FCN. Sursă imagine: [Long2015].

Ieșirea rețelei va avea o dimensiunea identică cu imaginea de intrare, dar numărul de canale al imaginii de ieșire (predicția) va fi egal cu numărul de clase. În locul folosirii interpolării bi-liniare, FCN utilizează niveluri de “deconvoluție” sau convoluție transpusă (Figura 11) pentru a construi segmentarea. Deși operația de “deconvoluție” nu este inversul operației normale de convoluție, cel puțin nu din punct de vedere matematic, aceasta este utilă pentru rețele de segmentare semantică. În figura 11 este ilustrată o operație de convoluție transpusă, unde dintr-o imagine de intrare de dimensiune 2 x 2 prin umplere (“padding” în zonele punctate este copiată informația din imaginea inițială sau se umple cu valori de zero) se obține o imagine de 7 x 7, iar după convoluția cu un nucleu de 3 x 3 se obține imaginea finală de dimensiune 5 x 5.

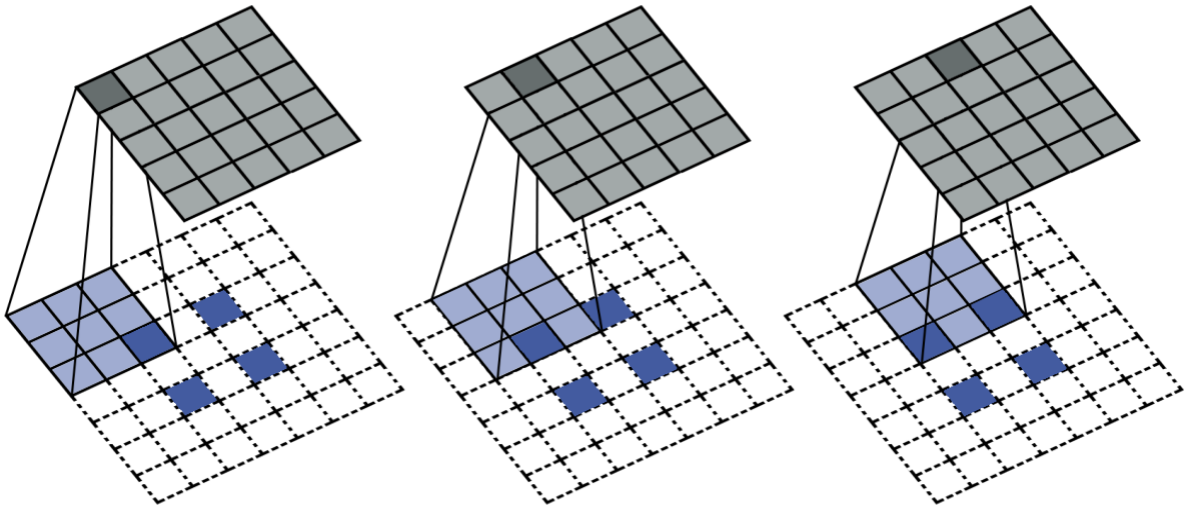


Figura 11. Convoluție transpusă (“deconvoluție”).

Primele niveluri de convoluție reduc rezoluția imaginii de intrare cu un factor de 32, ceea ce duce la dificultăți în partea de deconvoluție. Pentru a rafina rezultatele sunt utilizate conexiuni directe între niveluri de intrare care conțin trăsături la rezoluție mare și nivelurile de deconvoluție.

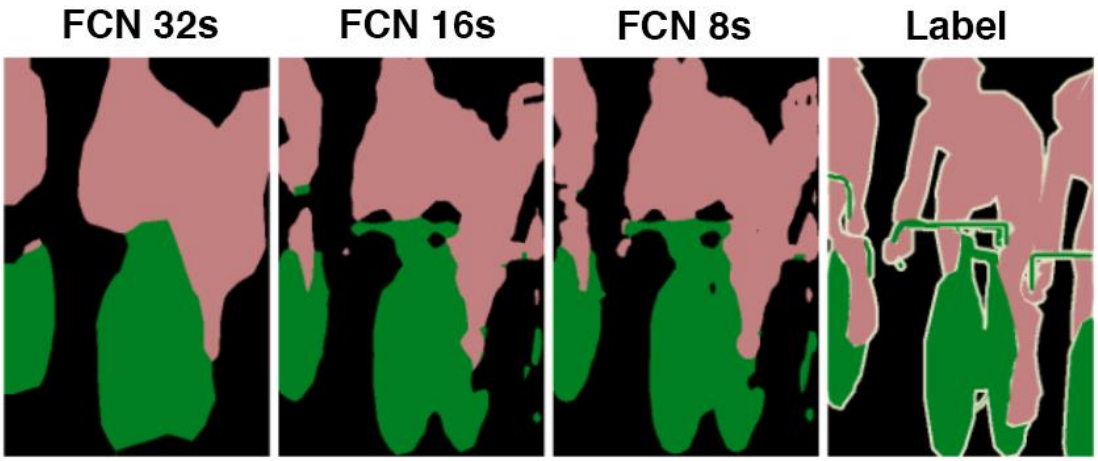


Figura 12. Efectul adăugării de conexiuni directe din lucrarea FCN [Long2015]. Primele 3 imagini reprezintă rafinarea predicției rețelei folosind conexiuni directe cu niveluri convoluționale cu deplasament (“stride”) de 32, 16 sau 8. Sursă imagine: [Long2015].

Nivelurile de agregare (“pooling”) au rolul de a reduce informațiile despre localizare, dar rețin date despre context și crește câmpul vizual (“field of view”). Pentru a genera o hartă de predicție cât mai bună, este necesară informația despre locația fiecărui pixel, de aceea se introduc aceste legături directe între primele și ultimele niveluri. Aceste soluții sunt numite arhitecturi de tip “encoder-decoder” și rezolvă problema localizării pixelilor fiecărei clase în harta de predicție. În afară de FCN, rețeaua U-Net publicată în [Ronneberger15] reprezintă o altă soluție populară bazată pe “encoder-decoder” și a fost folosită inițial pentru analiza imaginilor din domeniul medical. În U-Net partea de decodare este extinsă, astfel încât rețeaua este simetrică: are număr egal de straturi în partea de codificare cât și în partea de decodificare.

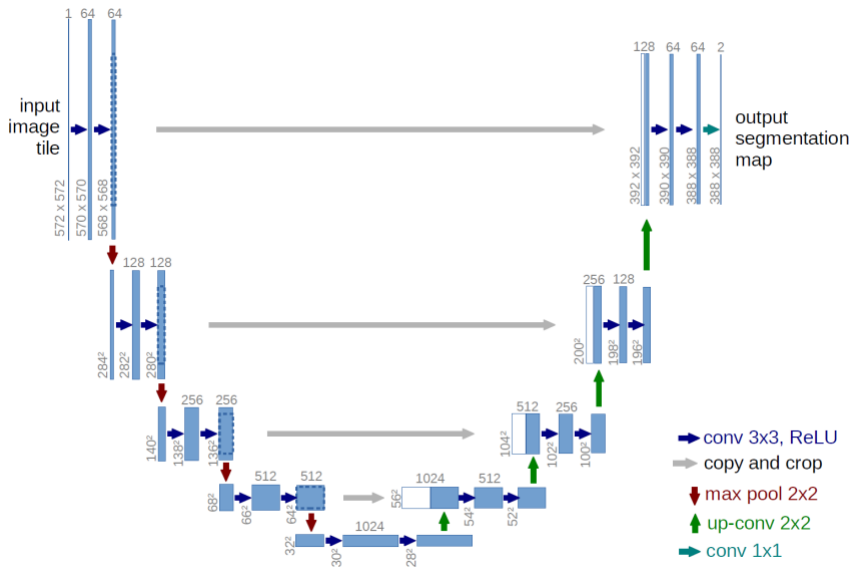


Figura 13. Arhitectura rețelei U-Net. Sursă imagine: [Ronneberger15].

Partea de codificare are rolul de a reduce dimensiunea spațială a imaginilor prin folosirea nivelurilor de tip “pooling”, în timp ce reconstrucția detaliilor obiectelor și a poziției lor în harta de predicție este realizată de partea de decodare. Unele legături directe între primele și ultimele niveluri din rețeaua neuronală facilitează antrenarea și recuperarea informației despre localizare a obiectelor. Totodată, legăturile directe între nivelurile de la codificare și decodificare au rolul de a extrage trăsăturile importante și relevante ale obiectelor la diferite dimensiuni (scale), ceea ce ajută la antrenare și apoi la predicții în diferite scenarii unde dimensiunile obiectelor variază. Spre deosebire de FCN, metoda prezentată în [Badrinarayanan2017] (rețeaua SegNet) este mai eficientă din punct de vedere al utilizării memoriei deoarece utilizează indicii din nivelurile de agregare “max-pooling” în loc de cei ai trăsăturilor. În general pentru partea de codificare se poate utiliza orice tip de rețea neuronală antrenată pentru clasificarea de imagini, cum ar fi: rețeaua VGGNet din lucrarea [Simonyan2014], ResNet publicată în [He2016], DenseNet din lucrarea [Huang2017], MobileNet publicată în [Howard2017], etc. Fiecare prezintă diferite avantaje și dezavantaje, de obicei se poate alege o arhitectură care va favoriza precizia predicției în detrimentul vitezei de execuție sau a memoriei folosite. Performanța acestor arhitecturi de rețele neuronale depinde și de tipul de augmentare folosită, dar și de domeniul de aplicabilitate (datele de intrare). Autorii lucrării [Long2015] (rețeaua FCN) au raportat îmbunătățiri modeste dacă au folosit augmentare, în schimb în lucrarea [Ronneberger15] (rețeaua U-Net) autorii au consemnat faptul că augmentarea datelor este un proces esențial pentru a obține performanțe sporite.

În literatura de specialitate au fost prezentate și o varietate de arhitecturi avansate bazate pe U-Net, unde înșiruirea clasică de niveluri de convoluție (un “bloc” de operații de convoluții) a fost înlocuită cu alte straturi (“blocuri”) mai complexe. O astfel de abordare mai avansată a fost prezentată în lucrarea [Drozdzal2015], unde nivelurile de convoluție stivuite au fost înlocuite cu blocuri reziduale, care introduc conexiuni directe între convoluțiile din același bloc (locale). Aceste legături directe la nivel de convoluții permit o antrenare mai eficientă a unor arhitecturi de rețele mai complexe bazate pe niveluri mai multe. Blocurile reziduale au fost propuse inițial în lucrarea [He2016] (rețeaua ResNet) de la Microsoft care a și câștigat în 2016 competiția de clasificare a imaginilor “ImageNet” cu o acuratețe de 96.4%.

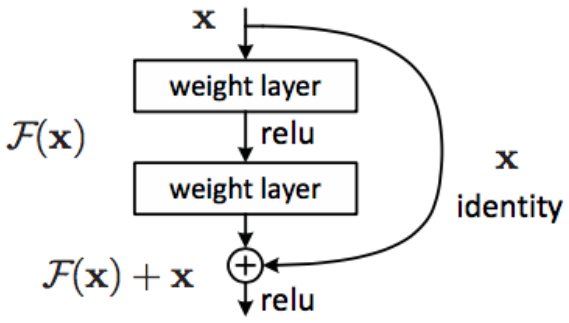


Figura 14. Bloc rezidual folosit în rețeaua ResNet, sursă imagine: [He2016].

Lucrarea [Jegou2016] oferă o variantă și mai complexă a U-Net, în care se folosesc blocuri dense, idee propusă inițial în lucrarea [Huang2017] (rețeaua DenseNet), care au proprietatea de a transfera trăsături primare (“low level”) din primele niveluri, alături de

trăsături elaborate (“high level”) din nivelurile ulterioare, ceea ce permite o reutilizare foarte eficientă a acestor trăsături în procesul de segmentare. În această abordare, partea de decodare (“upsampling”) nu folosește conexiuni directe între intrarea și ieșirea unui bloc dens pentru a nu utiliza prea multă memorie în timpul învățării.

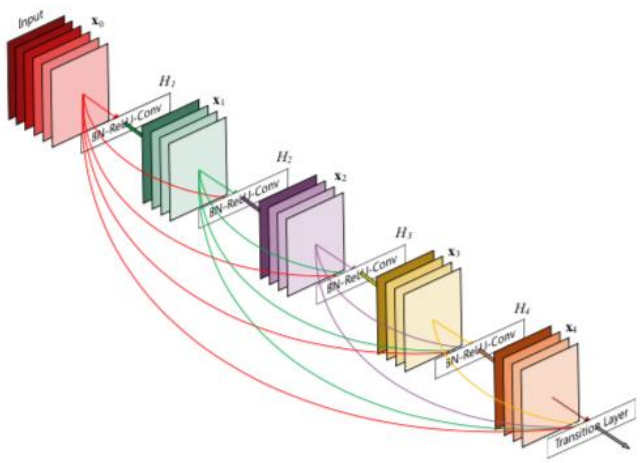


Figura 15. Exemplu de bloc dense. Sursă imagine: [Huang2017].

Ieșirea dintr-un nivel “tradițional” care conține operații de convoluție înșiruite, urmate de agregare și de funcții de activare se poate exprima folosind ecuația 42.

$$x_L = F_L(x_L - 1) \tag{42}$$

Ecuația 43 descrie ieșirea unui bloc rezidual, iar ecuația 44 reprezintă ieșirea unui bloc dens în care *[..]* reprezintă operația de concatenare.

$$x_L = F_L(x_{L-1}) + x_{L-1} \tag{43}$$
$$x_L = F_L([x_0, x_1, \dots, x_{L-1}]) \tag{44}$$

O alternativă la convoluțiile transpuse este folosirea convoluțiilor dilatate, numite și “atrous” în literatură [Yu2015]. Convoluțiile dilatate permit creșterea câmpului vizual fără a reduce dimensiunile spațiale. De asemenea, se introduce un nou parametru pentru aceste operații, numit rata de dilatare, care are rolul de a defini spațierea valorilor din kernel. Un exemplu de convoluție dilatată este ilustrat în Figura 16, unde este folosită o rată de dilatare egală cu 2.

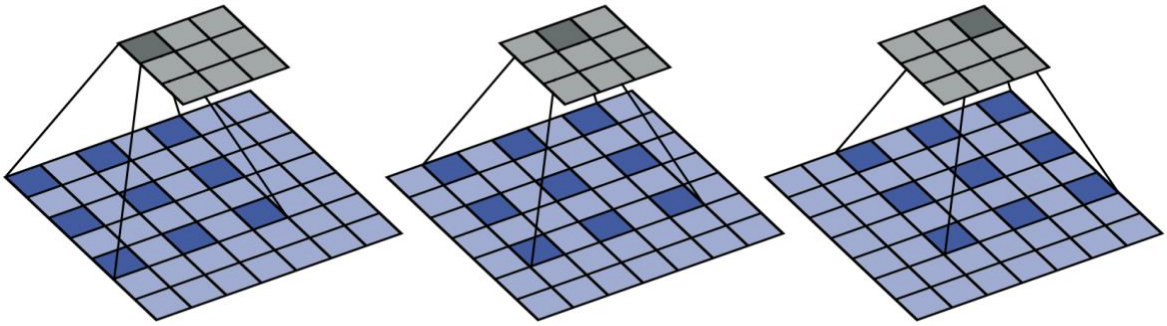


Figura 16. Convoluție dilatată (“atrous”) cu rata de dilatare 2.

O abordare des folosită este folosirea post-procesării pentru a rafina harta de predicție a segmentării. În literatura de specialitate s-a folosit tehnica bazată pe “Conditional Random Fields” (CRF) [Krähenbühl2011], care folosește intensitatea imaginii sursă pentru îmbunătățirea predicției. Rețeaua DeepLab V1 publicată în [Chen2016a] și DeepLab V2 din lucrarea [Chen2016b] reprezintă abordări în care s-au utilizat niveluri de convoluții dilatate și un nivel de CRF. De asemenea lucrarea introduce și posibilitatea de procesare a imaginilor la diferite rezoluții (scale) prin ceea ce se numește “atrous spatial pyramid pooling” (ASPP). Modelele de rețea bazate pe “image pyramid” aplică operațiunea de “pooling” sau aplică în paralel convoluție dilatată pe diferite rezoluții ale imaginii de intrare. Aceste modele îmbunătățesc precizia segmentării prin exploatarea informațiilor despre trăsăturile din imagini de la diferite scale. DeepLab V3 publicată în [Chen2017] introduce o variantă îmbunătățită a operației de “spațial pooling” (ASPP) și utilizează un modul pentru aplicarea acestor operațiilor de convoluții dilatate în cascadă. În 2018 a fost publicată lucrarea [Chen2018] (rețeaua DeepLab V3+) de Google, care setează un nou record pe baza de date Pascal Visual Object Challenge [Everingham2010]. Versiunea 3+ explorează și folosirea de convoluții separabile pe adâncime (“depthwise separable convolutions”). Acest tip de convoluție a fost utilizată cu succes în abordările necesare pentru detecția de obiecte: rețeaua Xception din [Chollet2017] sau rețeaua MobileNet publicată în [Howard2017].

Tabel 2 reprezintă un top al lucrărilor prezentate mai sus, evaluate folosind baza de date Pascal Visual Object Challenge și metrica “intersect over union” (scorul reprezintă procentul de pixeli clasificați corect).

Tabel 2. Top modele de rețele neuronale pentru segmentare semantică, calculate pe baza de date Pascal Visual Object Challenge.

Model CNN	Scor evaluare
SegNet	59.9
FCN 8s	62.2
DeepLab V2 CRF	79.7
DeepLab V3	85.7
DeepLab V3+	87.8

Separarea între mai multe instanțe ale obiectelor din aceeași clasă este numită “instance segmentation”, iar pentru obiectivul propus în această teză nu este necesară, deoarece am folosit o singură clasă (cea de drum) iar separarea între mai multe zone de drum într-o imagine nu este utilă și nici relevantă.

Detecția obiectelor poate fi realizată prin ”instance segmentation”, dar și prin generarea unor chenare de încadrare pentru fiecare obiect (”bounding box”), iar determinarea acestor chenare se poate face folosind o imagine segmentată. Una din cele mai importante abordări pentru detecție de obiecte este prezentată în rețeaua R-CNN din lucrarea [Girshick2014]. Rețeaua R-CNN detectează diferite tipuri de obiecte (vehicule, cicliști, pietoni, etc.) și generează chenare de încadrare în jurul lor. Această soluție constă în două etape de procesare: propunerea de regiuni (2000 în total) și extragerea trăsăturilor folosind o rețea neuronală convoluțională. Trăsăturile extrase de CNN sunt clasificate de câte un SVM antrenat pentru fiecare clasa de obiect în parte. Propunerea de regiuni de obiecte este realizată folosind algoritmul ”selective search” publicat în [Uijlings2013], care folosește o imagine segmentată (folosind abordarea din [Felzenszwalb2004]). Din imaginea segmentată se extrag chenare de încadrare pentru fiecare regiune (segmentată) și apoi se grupează zonele asemănătoare în funcție de culoare și textură. R-CNN are dezavantajul de a avea timpi lungi de antrenare și de predicție, din cauza celor 2000 de regiuni care sunt procesate indiferent de imaginea de intrare.

Limitările de performanță au fost rezolvate în lucrarea [Girshick2015] (rețeaua Fast R-CNN), unde imaginea sursa este folosită ca intrare pentru o rețea convoluțională. Pe harta de trăsături rezultată se caută regiuni de interes (“region of interest” - ROI) folosind ”selective search”. Următoarea operație este una de agregare a zonelor de interes, numită ”ROI pooling” care folosește operația ”max pooling” pentru a converti trăsăturile dintr-o regiune de interes într-o hartă de trăsături de dimensiune $H * W$ (hiper-parametri care se pot ajusta înainte de antrenarea rețelei). Rezultatul stratului ”ROI pooling” este apoi folosit într-o rețea complet conectată iar în final se face regresie liniară pentru ajustarea chenarului de încadrare și se aplică operația ”softmax” pentru clasificarea chenarului.

Rețeaua Faster R-CNN publicată în [Ren2015] introduce o îmbunătățire a celor precedente prin introducerea unei rețele care propune regiuni numită “region proposal network” (RPN) în locul ultimului nivel convoluțional din R-CNN.

Rețeaua RPN are ca intrare o imagine de trăsături (“feature map”) de obicei din ultimul strat convoluțional și are ca ieșire un set de propuneri de obiecte, mai specific de chenare de încadrare (“bounding box”). RPN conține un nivel de clasificare cât și unul de regresie. Ideea principală este de a folosi o fereastră glisantă de dimensiune variabilă (în cazul lucrării Faster R-CNN este setată la 3 x 3) și de a genera chenare de încadrare în aceste ferestre folosind această mini-rețea RPN. Faster R-CNN a introdus și conceptul de “ancore” (“anchors”) care reprezintă de fapt punctul central al unui chenar de încadrare.

Antrenarea unei rețele folosind ”ancore” este mai ușoară, mai ales pe partea de implementare. De exemplu predicția unui chenar de încadrare caracterizat de coordonatele xmin, ymin, xmax, ymax este aproape imposibilă pentru o rețea neuronală, deoarece dimensiunea imaginilor și a chenarelor de încadrare, dar și a rapoartelor de aspect variază foarte mult. O altă problemă este dată de constrângerile care trebuie aplicate coordonatelor, de exemplu xmin trebuie să fie mai mic decât xmax. Astfel, abordarea cea mai eficientă este de a oferi predicții de valori de ajustare (”offset”) care trebuie aplicate unor chenare de încadrare de referință. Dacă avem un chenar de încadrare caracterizat de x, y (coordonatele centrului chenarului) și w, h (dimensiunea pe lățime și înălțime a chenarului), atunci rețeaua va oferi predicții asupra Δx, Δy și Δw, Δh.

Pe harta de trăsături din ultimul nivel convoluțional se aplică un algoritm de tip fereastră glisantă (”sliding window”), iar rețeaua Faster R-CNN generează câte 9 chenare de încadrare (“anchors”) în fiecare fereastră de glisare: 3 chenare la dimensiuni diferite, fiecare din ele la 3 proporții diferite (de exemplu: 1:1, 1:2 și 2:1). Numărul maxim de predicții în fiecare fereastră glisantă este notat cu k, ceea ce înseamnă că RPN va oferi 4*k predicții din stratul de regresie care codifică datele despre coordonatele celor k chenare de încadrare, iar din stratul de clasificare vor fi *2*k* predicții (una pentru probabilitatea de a fi de obiect și una pentru probabilitatea de a fi parte din fundal). În Figura 17 este exemplificată mini-rețeaua RPN pentru o fereastră glisantă de dimensiune 3 x 3.

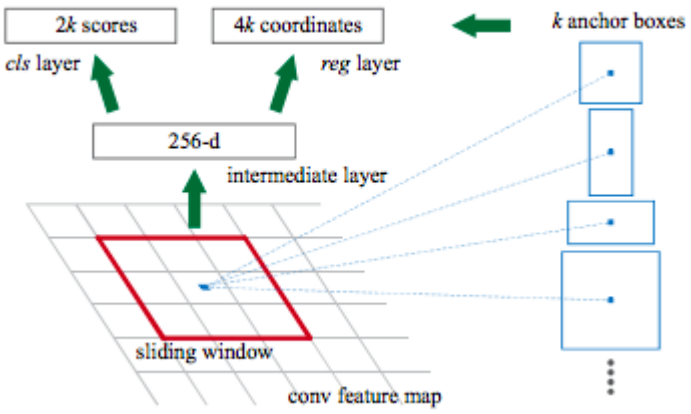


Figura 17. Rețeaua RPN introdusă în lucrarea Faster R-CNN. Sursă imagine: [Ren2015].

Rețeaua RPN are un strat convoluțional cu 512 canale, cu un nucleu de convoluție de 3 x 3 cu ”padding” de 1, urmat de două convoluții cu nucleu de 1 x 1, una pentru clasificare

având funcția de cost ”binary cross entropy” și una pentru regresia chenarului de încadrare cu funcția de cost ”smooth L1”. Arhitectura rețelei Faster R-CNN care folosește RPN este ilustrată în Figura 18.

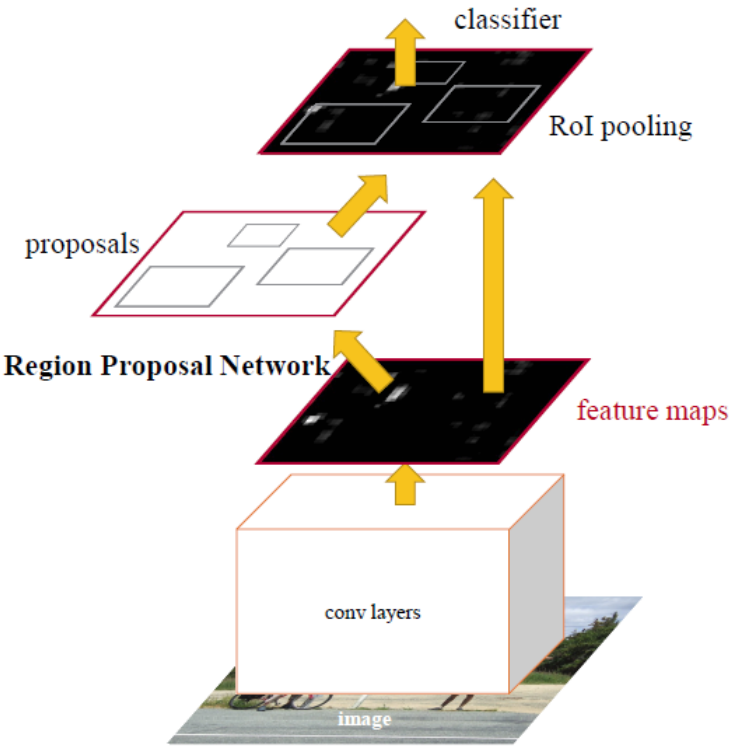


Figura 18. Arhitectura rețelei Faster R-CNN folosind RPN și agregarea zonelor de interes (“ROI pooling”). Sursă imagine: [Ren2015].

Chenarele de încadrare rezultate din RPN vor reprezenta zonele de interes care sunt agregate folosind “ROI pooling” unde se împarte chenarul de încadrare în zone mai mici și se păstrează cele cu valoarea maximă, similar cu operația “max pooling”.

Rețeaua Mask R-CNN publicată în [He2017] extinde rețeaua Faster R-CNN pentru task-ul de segmentare semantică prin adăugarea a 2 niveluri de convoluții după nivelul de agregare “ROI pooling”.

2.12 Percepție monoculară

2.12.1 Percepție monoculară folosind dispozitive mobile inteligente

Sisteme de percepție a mediului folosind dispozitive mobile inteligente au fost propuse de-a lungul anilor. Unele aplicații disponibile pentru testare sunt: iOnRoad, Drivea, Movon FCW sau Comma.ai. iOnRoad [iOnRoad] este una dintre primele soluții de asistență augmentată a conducerii folosind dispozitive mobile Android sau iOS. Această soluție oferă avertizări sonore și vizuale în cazul unei posibile coliziuni cu obstacolul din fața vehiculului.

Aplicația folosește camera dispozitivului și senzorul de poziționare prin satelit GPS pentru a calcula timpul până la coliziune (“time to collision”), iar dacă această valoare scade sub un prag, atunci se emit avertizările pentru șofer. Aplicația oferă și avertizări în cazul în care vehiculul iese din banda curentă (“lane departure warning”). O aplicație similară este Drivea [Drivea], care oferă aceleași tipuri de avertizări: coliziune cu obstacole, ieșirea din banda curentă și avertizări asupra vitezei mari de deplasare. Movon FCW [Movon] reprezintă o abordare apărută după primele două prezentate, și este una dintre puținele care detectează și vehicule care se îndreaptă din sens opus și oferă informații despre distanță până la vehiculele detectate. Există de asemenea și dispozitive dedicate care se pot monta în vehicul ce oferă avertizare de coliziuni sau la ieșirea de pe banda de circulație, de exemplu Garmin Dash Cam 65W [Garmin]. Software-ul Comma.ai [Santana2016], apărut recent este unul dintre puținele care se bazează exclusiv pe rețele neuronale convoluționale pentru a procesa obstacolele și pentru a detecta benzile de circulație. Partea de percepție a mediului este implementată pe dispozitive mobile Android și oferă predicții pentru zonele de obstacole din imagine, dar oferă și informații despre banda curentă de circulație, precum și predicția unghiului de virare al vehiculului în funcția de caracteristicile benzii de circulație detectată. Principala limitare a acestui sistem este dată de existența unui drum marcat corespunzător astfel încât să ofere predicții corecte pentru unghiul de virare, de aceea acest sistem este fezabil doar în anumite scenarii, cum ar fi autostrăzile sau drumurile expres.

În cadrul acestei teze de doctorat, voi prezenta abordări diferite pentru a crea un sistem de percepție monocular, folosind în principal dispozitive mobile dotate cu camere foto și senzori adiționali care pot fi folosiți pentru a crea o reconstrucție a scenei. Folosirea unui singur senzor de percepție (o singură cameră foto) este principalul obiectiv, deoarece aceste sisteme monoculare sunt mai accesibile ca și cost decât cele bazate pe camere stereo, care necesită o calibrare și poziționare fixă în vehicul (poziție care nu poate fi schimbată, fără a fi necesară o nouă calibrare), sau sisteme bazate pe senzori adiționali de percepție, cum ar fi: senzori LIDAR sau RADAR, care momentan sunt mai costisitoare decât unele mașini, astfel încât nu sunt fezabile și accesibile pentru publicul larg. Având sisteme de percepție se pot asista șoferii în procesul de conducere al vehiculelor, astfel încât să se evite diferite situații neplăcute ce ar putea produce accidente, majoritatea cauzate din neatenție sau neadaptarea la condițiile de drum. Un alt avantaj al unui sistem monocular, este ușurința cu care poate fi adăugat unui vehicul, nefiind necesar ca acest sistem să fie instalat de către producătorul auto în fabrică.

2.12.2 Extragere informații 3D din imagini

O direcție de cercetare cu aplicații în robotică și mai ales mașini autonome este extragerea informațiilor 3D din imagini achiziționate cu o singură cameră video, care se poate face prin adăugarea de senzori adiționali pentru estimarea adâncimii sau direct din analiza imaginii (folosind rețele neuronale artificiale sau analiza imaginii cu efectul de perspectivă eliminat).

Abordările clasice de estimare a datelor de adâncime se bazează în general pe sisteme formate din două sau mai multe camere (ex: stereo-viziune [Bertozzi1998]), senzori (LIDAR, RADAR, etc.), sau din analiza mișcării obiectelor împreună cu scena (“structure from motion”). Senzorii bazați pe LIDAR nu sunt eficienți pentru obiecte aflate în mișcare, totodată

suprafața obiectelor afectează măsurătorile (zone transparente, reflective sau foarte întunecate). Senzorii bazați pe lumină structurată, cum este senzorul Kinect creat de Microsoft, nu funcționează în exterior și oferă o rază de acțiune limitată. Folosirea datelor stereo, adică perechi de imagini obținute folosind camere calibrate cu disparitate și parametri cunoscuți evită limitările altor tipuri de senzori. Aceste date sunt mai ușor de obținut la un cost foarte redus. Principalele dezavantaje ale unui sistem bazat pe două sau mai multe camere (sau senzori) sunt date de faptul că necesită spațiu mai mult pentru a le monta într-un vehicul, consumă mai multă energie și necesită o putere de procesare mai mare. Soluția este de a reduce numărul senzorilor folosiți, dar estimarea adâncimii din imagini capturate cu o singură cameră reprezintă o provocare. Unii autori au folosit camera monoculară împreună cu un senzor bazat pe laser [Zhuang2003] pentru estimarea adâncimii pentru navigarea unui robot mobil.

Principalele limitări ale soluțiilor bazate pe o singură cameră au început să fie rezolvate prin folosirea rețelelor artificiale. Majoritatea soluțiilor de învățare supervizată necesită date despre adâncime (“depth”) asociate cu fiecare imagine de intrare, ceea ce înseamnă folosirea unor sisteme care au fost foarte bine setate și calibrate. De aceea, aceste soluții supervizate sunt folosite în general doar pe anumite baze de date, cum ar fi “KITTI Vision” de la Karlsruhe Institute of Technology publicată în [Geiger2012]. Astfel, estimarea adâncimii este tratată ca o problemă de regresie supervizată, unde aceste metode învață să estimeze hărți de adâncime din imagini color. Bazele de date cu informații despre adâncime sunt imperfecte și crearea lor necesită mult timp, iar datorită senzorilor folosiți sunt de obicei și costisitoare. Lucrarea [Xie2016] folosește o rețea neuronală convoluțională pentru a genera imaginea din dreapta având ca intrare imaginea din stânga (dintr-o bază de date stereo, unde sunt perechi de imagini stânga - dreapta sincronizate și capturate simultan în scenă). În [Garg2016], autorii prezintă o abordare de învățare nesupervizată în care generează harta de adâncime folosind CNN. O abordare cu rezultate mai bune și un proces de antrenare simplificat este prezentată în [Godard2017], unde se generează două imagini de adâncime: cea obținută din deplasamentul pixelilor din imaginea stângă în raport cu imaginea dreaptă și cea dreaptă la stânga, simultan dintr-o singură imagine color de intrare. Această lucrare introduce și o funcție de cost numită “left-right consistency” care facilitează procesul de antrenare. Principalul dezavantaj al acestor metode este că nu tratează corect obiectele aflate în mișcare. Totuși, soluții mai recente au depășit aceste limitări și oferă inclusiv informații despre mișcarea vehiculului în scenă sau chiar a obiectelor [Casser2018].

De altfel, extragerea informațiilor 3D dintr-o imagine se poate face prin folosirea imaginii cu efectul de perspectivă eliminat (“inverse perspective mapping” - IPM). Presupunând că suprafața drumului este plată, distanța până la obstacolele din scena de trafic se poate estima direct din imaginile IPM, iar abordările folosite de mine sunt prezentate în secțiunea următoare.

2.12.3 Detecția vehiculelor în imagini IPM din sisteme de percepție monoculară

Am folosit diferiți algoritmi de procesare de imagini pentru a detecta obstacole în scenă direct din imaginile cu efectul de perspectivă eliminat. Abordările prezentate în această secțiune reprezintă contribuții originale și au fost diseminate în lucrări publicate la conferințe

[Itu2014], jurnale [Danescu2016a] sau capitole de carte [Danescu2016b]. În aceste abordări, obstacolele sunt detectate prin segmentarea imaginii de niveluri de gri (“grayscale”) și apoi filtrate, iar în final sunt procesate. O primă variantă de a găsi zonele de obstacole presupune găsirea unei valori medii de intensitate pentru sosea și aplicarea unei operații de binarizare (”thresholding”) cu prag fix în funcție de această valoare. Se alege o zonă dreptunghiulară din fața vehiculului în imaginea cu efectul de perspectivă eliminat și apoi se calculează intensitatea medie pe acea zonă. Generarea imaginii IPM este descrisă în secțiunea 3.6. O astfel de abordare are avantajul de a fi eficientă din punct de vedere computațional, dar va fi limitată în cazul unor umbre de pe drum, caz în care ele ar putea fi identificate incorect ca fiind obstacole.

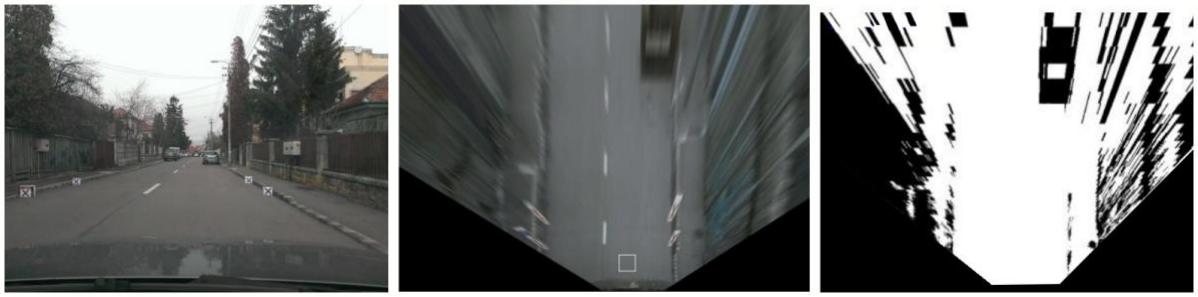


Figura 19. Stânga: imaginea de intrare, mijloc: imaginea IPM color, dreapta: imaginea IPM binară, segmentată folosind pragul fix calculat din imaginea IPM color.

O abordare mai eficientă constă în scanarea zonei din fața vehiculului în imaginea IPM folosind raze cu originea în punctul *f*, unde e montată camera foto. Deoarece obiectele în imaginile IPM vor fi dispuse radial, se poate folosi această proprietate. Astfel, pentru fiecare linie cu originea în punctul *f* se va salva într-o listă $g_\alpha(d)$ intensitatea nivelului de gri din imaginea IPM, unde α reprezintă unghiul liniei, iar *d* este distanța. Pentru fiecare distanță posibilă sunt calculate trei valori ale intensității: în zona proximală (μ_α,P), zona din mijloc (μ_α,M) și o zonă distală (μ_α,D) (ecuațiile 45, 46 și 47).

$$\mu_{\alpha,P}(d) = \frac{1}{d-3-d\_min_\alpha+1}\sum_{k=d\_min_\alpha}^{d-3} g_\alpha(k) \quad (45)$$

$$\mu_{\alpha,D}(d) = \frac{1}{d-d\_max_\alpha-d-3+1}\sum_{k=d+3}^{d\_max_\alpha} g_\alpha(k) \quad (46)$$

$$\mu_{\alpha,M}(d) = \frac{1}{7}\sum_{k=d-3}^{d+3} g_\alpha(k) \quad (47)$$

Ecuația 48 reprezintă funcția binară pentru determinarea unui obstacol la distanță *d*, de-a lungul liniilor de scanare în imaginea IPM.

$$\omega_\alpha(d) = \begin{cases} 1, \text{dacă } \left(\mu_{\alpha,P}(d) - \mu_{\alpha,M}(d) > \sigma\right) \text{ și } \left(\left(\mu_{\alpha,D}(d) - \mu_{\alpha,M}(d) > \sigma\right) \text{ sau } \left(\mu_{\alpha,P}(d) - \mu_{\alpha,D}(d) > \sigma\right)\right) \\ 0, \text{altfel} \end{cases} \quad (48)$$

Unde σ reprezintă deviația standard a intensităților din imaginea IPM, o măsură a contrastului imaginii. Figura 20 reprezintă rezultatele scanării radiale pentru găsirea obstacolelor.

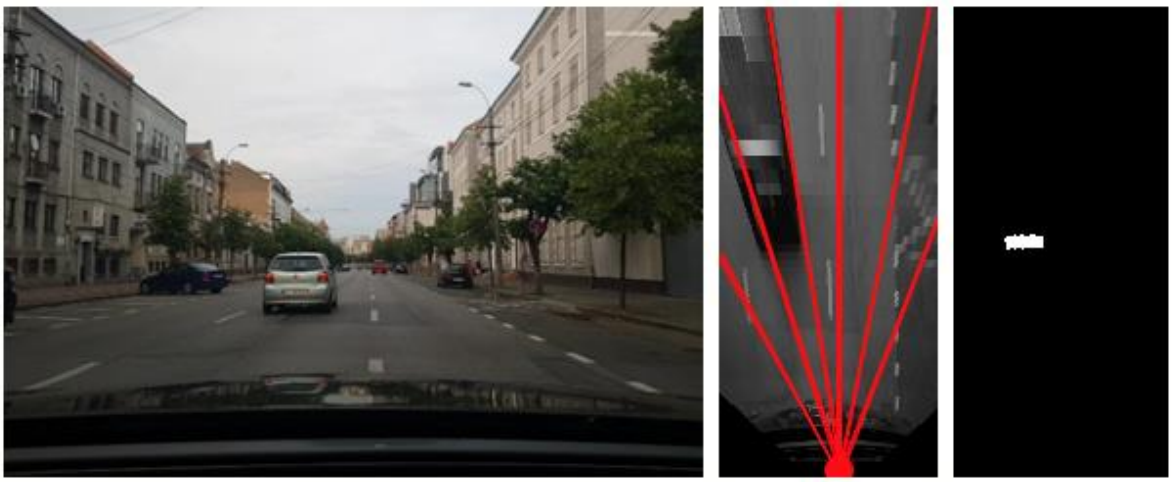


Figura 20. Rezultatul scanării radiale: stânga: imaginea de intrare color, mijloc: imaginea IPM, dreapta: segmentare obstacole.

În mod intuitiv, ecuația 48 exprimă faptul că punctul de contact al unui obstacol pe carosabil este mai închis ca intensitate de culoare decât zona de asfalt dinaintea lui, dar și mai închis decât intensitatea culorii de după acesta. Umbra de sub un obstacol va corespunde acestei proprietăți, dar cu limitarea că această abordare este validă doar în anumite condiții de iluminare, în special în timpul zilei.

Pentru îndeplinirea obiectivelor propuse pe parcursul acestei teze de doctorat, vehiculele detectate folosind analiza imaginii cu efectul de perspectivă eliminat vor fi urmărite în timp folosind tehnici și algoritmi tradiționali prin care se pot defini unele constrângeri și modelări geometrice.

2.13 Framework de percepție monoculară

Pentru îndeplinirea obiectivelor propuse în această teză și pentru a construi un sistem de percepție monocular, am ales identificarea prezenței obstacolelor prin segmentarea semantică folosind rețele neuronale artificiale și extragerea informațiilor 3D din scenă prin analiza imaginii cu efectul de perspectivă eliminat. Segmentarea semantică a zonei de drum este utilă pentru a detecta orice tip de obstacol din scenă, nefiind limitată doar la vehicule sau pietoni. Practic, din detecția zonei libere de drum se poate considera că tot ce nu este drum este obstacol și mai departe poate fi interpretat și urmărit în timp. Pentru a extrage informațiile 3D din imaginea IPM este nevoie de o calibrare a camerei, iar tehnicile principale folosite de mine sunt descrise pe larg în capitolul 3.

Fluxul de procesare al sistemului propus de mine este ilustrat în figura următoare:

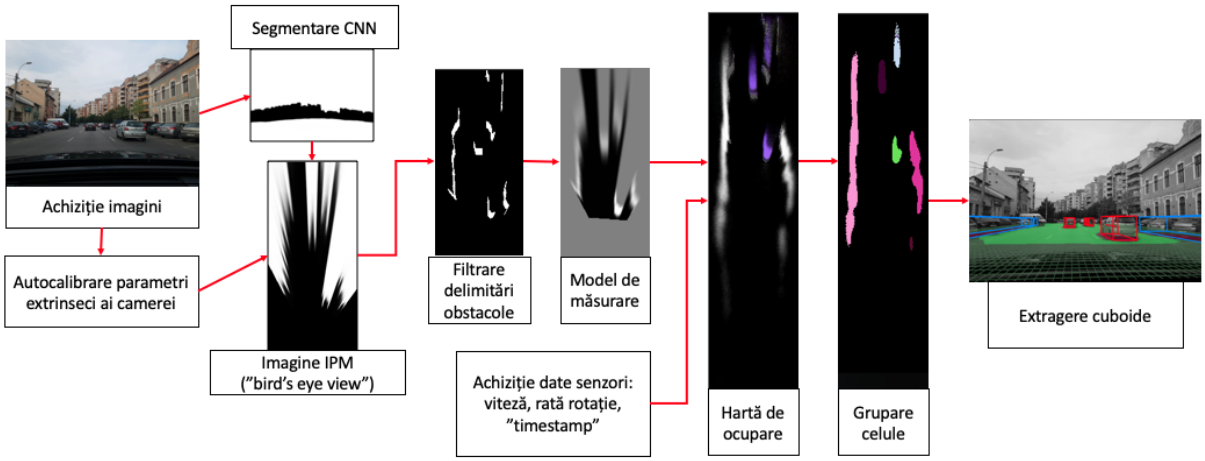


Figura 21. Fluxul de procesare al sistemului de percepție monoculară.

Primul pas constă în achiziția imaginilor și auto-calibrarea parametrilor extrinseci ai camerei, care sunt folosiți pentru a genera o imagine cu efectul de perspectivă eliminat din imaginea segmentată de rețeaua neuronală. Următorul pas constă în detecția delimitării obstacolelor, iar din această imagine se generează un model de măsurare care este folosit pentru a crea particule într-o hartă de ocupare. În ultimul pas celulele asemănătoare sunt grupate, iar din ele se extrag cuboide care sunt afișate în imaginea inițială a scenei de drum.

Principalul obiectiv descris în acest capitol se referă la proiectarea unui sistem monocular de percepție a scenelor de trafic rutier. Am ales folosirea rețelelor neuronale convoluționale care au rezultate foarte bune pentru astfel de sarcini, dar necesită foarte multe date de antrenare. Partea de urmărire este implementată folosind un filtru de particule. Ținând cont de aceste aspecte, obiectivele principale din cadrul acestui capitol sunt următoarele:

- colectarea de date noi folosind dispozitive mobile inteligente și crearea unei baze de date cu secvențe din traficul rutier urban
- centralizarea bazelor de date existente pentru interpretarea automată a scenelor din traficul rutier
- alegerea unei arhitecturi de rețea neuronala convoluțională și modificarea ei, antrenarea și reglajul parametrilor de învățare pentru detecția zonei de drum și segmentarea semantică a scenei
- integrarea rezultatelor din CNN într-un cadru de procesare ("framework") probabilistic pentru percepția scenei și urmărirea obstacolelor (estimarea parametrilor de poziție și viteză a obiectelor relevante din scenă)

2.13.1 Baze de date existente

Pentru sisteme avansate de asistență pentru șoferi și realizarea obiectivelor propuse, am folosit date din două surse principale: baze de date internaționale disponibile pentru

comunitatea de cercetare, care au avantajul că sunt adnotate cu obiecte detectate, distanțe, și alte rezultate de referință, și date achiziționate de noi, unde putem adăuga informații suplimentare, sau putem acoperi scenarii diferite. Bazele de date existente au fost create de diferite instituții de cercetare și educație, de aceea ele vor avea imagini de dimensiuni diferite și datele de etichetă stocate în formate diferite. În Tabel 3 este prezentată o comparație din mai multe puncte de vedere (dimensiune, scenarii) a bazelor de date existente pentru analiza automată a imaginilor de trafic.

Tabel 3. Prezentare comparativă a bazelor de date cu imagini din traficul rutier.

Baza de date	Număr secvențe	Număr imagini	Mai multe orașe	Condiții meteo diferite	Momente diferite ale zilei	Scenarii diferite
KITTI	22	14999	Nu	Nu	Nu	Da
Cityscapes	50	5000	Da	Nu	Nu	Nu
Mapillary	-	25000	Da	Da	Da	Da
BDD100K	100000	120.000.000	Da	Da	Da	Da

Pentru determinarea suprafeței de drum, am ales antrenarea pe mai multe baze de date cu imagini din trafic adnotate: Cityscapes, Kitti, Mapillary Vistas și Berkeley Deep Drive.

Baza de date “CityScapes” [Cordts2016] constă în scene din traficul rutier urban și conține 5000 de imagini adnotate cu 30 de clase de obiecte diferite. Adnotările sunt stocate în fișiere de tip ”json” unde sunt descrise coordonatele poligoanelor care formează fiecare obiect de clasă. Datele au fost preluate în 50 de orașe diferite din Germania și Elveția (ex: Aachen, Koln, Dusseldorf, Zurich), pe parcursul a mai multor luni, astfel încât avem scene de trafic din perioada verii, toamnei și a primăverii. Pentru antrenare datele sunt din traficul rutier în 2975 de imagini etichetate și cu 30 de clase de obiecte diferite. Setul de validare conține 500 de imagini și adnotări, iar setul de testare conține 1525 imagini și etichetele sunt descrise doar ca și poligoane în fișierele ”json” (nu există imagini etichetate). Imaginile sunt stocate pe 3 canale, de dimensiune 2048 x 1024 pixeli (un raport de 2:1). În toate imaginile adnotate culoarea clasei drumului este: (128, 64, 128).

Baza de date “KITTI” [Geiger2012] reprezintă o bază de date complexă și oferă date de antrenare pentru algoritmi de detecție a drumului, reconstrucție stereo, odometrie vizuală sau flux optic, detecție și urmărire de obiecte 3D. Datele au fost preluate într-un oraș din Germania, Karlsruhe, iar setul de date pentru detecția drumului conține 289 imagini de antrenare și 290 imagini de test. Drumul este marcat folosind culoarea: (255, 0 255) în imaginile adnotate, iar imaginile au dimensiunea 1392 x 512 pixeli. Doar imaginile de antrenare au și imaginea etichetată, cele 290 imagini de test nu au informații despre adnotarea corectă. KITTI oferă de altfel și o bază de date pentru segmentare semantică, unde codificarea pe culori a claselor este identică cu cea folosită în CityScapes. Baza de date pentru

segmentare semantică are 200 de imagini de antrenare și imagini etichetate (adnotate) și încă 200 de imagini de test fără adnotări. Dimensiunea imaginilor este de 1242 x 375 pixeli.

Baza de date ”Mapillary Vistas” [Neuhold2017] este folosită pentru segmentare semantică și conține un total de 25.000 imagini și adnotări din orașe diferite, la momente diferite ale zilei și în condiții meteo variate. Dimensiunea imaginilor este de 3840 x 2160 pixeli. Baza de date este structurată în 18.000 imagini de antrenare, 5000 imagini de testare și 2000 de imagini pentru validare, toate fiind adnotate. Zona de drum în spațiul de culoare RGB este adnotată la fel ca și în CityScapes: (128, 64, 128).

Berkeley Deep Drive publicată în [Yu2018] este o bază de date recent apărută și oferă imagini de antrenare și adnotări pentru detecția obiectelor (autobuze, mașini, bicicliști, semne de circulație, etc), segmentare semantică (“driveable area”) sau marcajele de pe carosabil. Baza de date Berkeley pentru segmentare conține 10.000 imagini etichetate, din care 7000 de imagini de antrenare adnotate, 2000 de imagini de validare cu adnotări și 1000 de imagini de test (fără adnotări). Dimensiunea imaginilor este de 1280 x 720 pixeli pe 3 canale color (RGB). Imaginile de etichetă sunt de aceeași dimensiune, unde fiecare clasă are asignată câte o culoare diferită. Pentru drum, culoarea este: (128, 64, 128).

2.13.2 Achiziția imaginilor

Sistemul de achiziții a datelor a fost scris în limbajul de programare Java pentru dispozitive mobile Android. Acesta folosește următoarele componente hardware pentru achiziția de secvențe din trafic:

- camera foto
- accelerometru
- giroscop
- senzor geomagnetic
- senzorii de poziționare prin satelit: GPS sau GLONASS (în funcție de dispozitivul folosit)

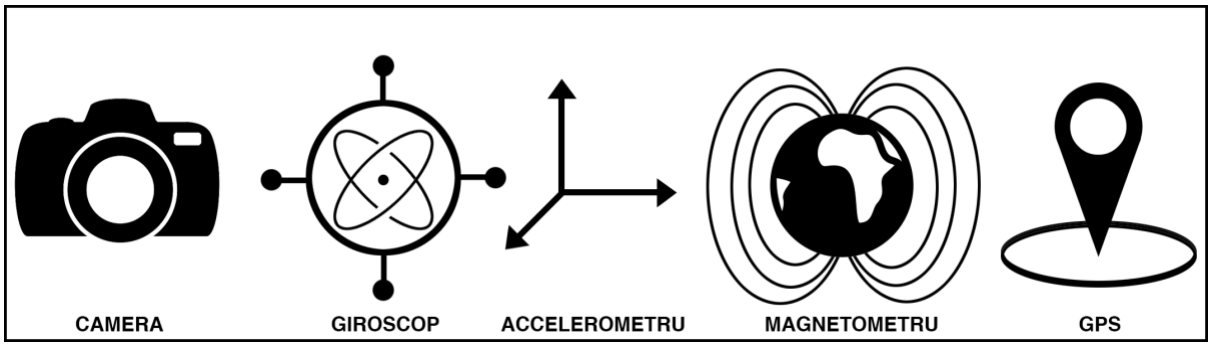


Figura 22. Senzorii folosiți în sistemul de achiziție.

Pentru achiziția de imagini este folosită cameră foto principală a dispozitivului (în general cea din spate), deoarece aceste camere au un senzor mai bun. Imaginile sunt salvate

în format RGB și informația despre intensitatea pe fiecare canal de culoare este stocată pe 8 biți. Restul de senzori enumerați mai sus sunt utilizați pentru a stoca informații adiționale din timpul deplasării vehiculului: date despre accelerația și viteza de deplasare, poziția (exprimată în coordonate de latitudine și longitudine), orientarea față de nordul magnetic (din magnetometru) și orientarea locală a dispozitivului (obținută din giroscop).

Toate aceste informații sunt stocate în memoria internă și sincronizate folosind un marcaj de timp ("timestamp") în modul următor: se salvează imaginea având ca nume marcajul de timp în format UNIX la momentul scrierii, iar simultan este creat un fișier text cu același nume conținând următoarele informații:

- timestamp: marcaj de timp UNIX
- accX, accY, accZ: accelerația gravitațională în jurul axelor x, y și z (m/s^2)
- gyroX, gyroY, gyroZ: rata de rotație a dispozitivului în jurul axelor x, y și z (rad/s)
- magX, magY, magZ: rotația din senzorul geomagnetic în jurul axelor x, y și z (rad)
- yaw: unghiul de girație în jurul axei x (rotație laterală a vehiculului de-a lungul axei drumului) exprimat în radiani
- pitch: unghiul de înclinare a dispozitivului mobil (rad)
- roll: unghiul de rotire în jurul axei y (rotația dispozitivului mobil pe axa verticală a drumului) exprimat în radiani
- yawRate: rata de rotație în jurul axei x (aceeași cu gyroX)
- speedMs: viteza de deplasare exprimată în metri / secundă
- lat, lng: coordonatele de latitudine și longitudine obținute din senzorul de poziționare prin satelit (GPS sau GLONASS)

Cele 3 unghiuri de rotație ("yaw", "pitch" și "roll") ale dispozitivului mobil au fost obținute din valorile senzorului geo-magnetic și cel de accelerație gravitațională. De asemenea, aceste trei valori sunt stocate într-un șir circular de dimensiune 10, iar apoi filtrate folosind media aritmetică pentru a atenua eventuale zgomote apărute în cazul trepidațiilor din timpul deplasării pe carosabil. Această abordare permite o procesare a datelor în mod offline pe un sistem desktop. Există posibilitatea de a procesa datele și în timp real pe Android, abordare publicată în [Danescu2016a] și [Danescu2016b].

Figura 23. Modulul de achiziție bazat pe dispozitive Android.

La momentul scrierii tezei am avut un total de peste 115.200 cadre din traficul rutier urban în Cluj-Napoca și pe autostrada din proximitatea orașului, achiziționate cu un dispozitiv mobil Samsung S8 Plus la o rată de medie 14 cadre per secundă. Există în total 27 secvențe diferite, luate în momente diferite ale zilei și inclusiv în condiții meteo diferite (înnorat, senin sau ploaie). Secvențele achiziționate cu sistemul propriu sunt ilustrate în Figura 24, unde fiecare traseu este marcat pe hartă din punctele de coordonate GPS.

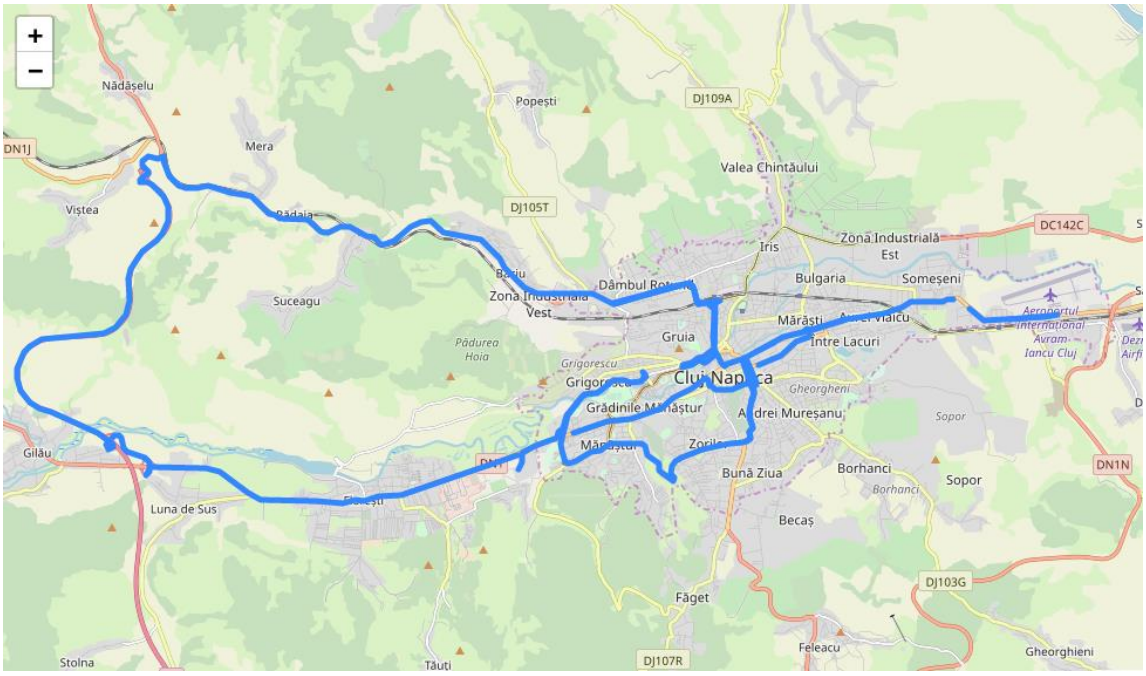


Figura 24. Traseul tuturor secvențelor achiziționate cu sistemul propriu în orașul Cluj-Napoca.

2.13.3 Segmentarea semantică a drumului folosind CNN

În sistemul dezvoltat de mine, pentru segmentarea semantică am folosit în total un set de 31964 imagini din cele 4 baze de date publice. Am folosit o singură clasă de etichete, cea a suprafeței de drum. În urma eliminării imaginilor de bazele de date care au mai puțin de 2500 de pixeli adnotați ca și suprafața de drum a rezultat un set de 28366 imagini de antrenare și 3534 imagini de validare. Deoarece bazele de date au un raport diferit între înălțimea și lățimea imaginilor sursă, am ales folosirea unei dimensiuni standard. Astfel toate imaginile au fost scalate la dimensiunea 256 x 256 pixeli.

Sistemul de predicție a zonei de drum pe care un vehicul se poate deplasa folosește la bază o rețea U-Net [Ronneberger2015] modificată de mine, care primește date de intrare perechi de imagini de dimensiune 256 x 256 pixeli, iar ieșirea rețelei, predicția (imaginea segmentată) va fi tot sub forma unei imagini de dimensiune 256 x 256 pixeli. Imaginile sunt încărcate de pe hard disk folosind funcții din OpenCV. Fiecare canal reprezintă informația de intensitate pentru albastru, galben sau roșu, în intervalul 0 - 255, astfel datele sunt de tipul “uint8” (date de tip întreg pe 8 biți). Imaginile eticheta sunt încărcate pe 1 canal, ele având valoarea intensității 255 pentru zona de drum (“driveable area”) și cu valoarea intensității 0 pentru tot ce nu reprezintă carosabil. Aceste imagini sunt transformate în matrice de valori reale (numere în virgulă mobilă) scalate în intervalul (0...1), un format mai potrivit pentru antrenarea și validarea rețelelor neuronale. Un exemplu de date de intrare pentru rețea este ilustrat în Figura 25.

Figura 25. Exemple de imagini folosite la antrenarea rețelei. Stânga: imagini color ale scenei, dreapta: imaginile etichetă cu carosabilul. Sursă imagini: [Yu2018].

Deoarece funcțiile neliniare de activare vor avea ca rezultat date de tip “float”, nu este nici o diferență sesizabilă dacă datele de intrare ar fi fost de tip întreg, iar oricum algoritmul de “backpropagation” necesită ca datele sa fie valori ce aproximează cât mai bine numere reale, astfel că folosirea tipului “float” va duce la o învățare mult mai eficientă și la fel de rapidă. În plus, tipul de date “float” pe 32 biți este optimizat foarte bine la nivel hardware pe arhitecturile moderne de CPU și GPU.

Am ales utilizarea unei rețele de tip “encoder-decoder” deoarece implementările care folosesc convoluții dilatate au o utilizare foarte mare a memoriei, iar rețelele de tip U-net sunt foarte eficiente pentru acest tip de proiect. Pentru funcția de cost (“loss”) am utilizat funcția ”binary cross entropy”, dar în timpul antrenării am monitorizat și coeficientul Sorensen-Dice. Antrenarea rețelei convoluționale funcționează mai bine dacă se folosesc date mai multe. Generarea de date de antrenare se poate face ușor prin folosirea unor tehnici de augmentare pentru extindere bazei de date inițiale. În sistemul propus de mine am folosit următoarele tehnici de augmentare: generare de variații de intensitate aleatorii ale imaginii de intrare în spațiul de culoare HSV, translatarea imaginii sau rotirea și scalarea aleatoare, dar și oglindirea imaginii pe orizontală.

Întreaga parte de antrenare și predicție folosind rețele neuronale a fost scrisă în limbajul de programare Python și folosind software-ul TensorFlow [Tensor] și Keras [Keras]. De asemenea, pentru a facilita procesul de augmentare și pre-procesare a imaginilor am folosit și software-ul OpenCV [OpenCV].

Rețeaua neuronală convoluțională de tip U-Net modificată a fost inspirată din [Ronneberger2015] și [Keng2017] și are următoarea structură: 5 straturi de codificare, un strat central și 5 straturi de decodificare. Un strat de codificare este caracterizat de următoarele operații:

- convoluție cu nucleu de dimensiune 3 x 3 în primele trei niveluri de codificare (d1, d2, d3) și nucleu de 5 x 5 la ultimele doua niveluri de codificare (d4, d5)
- normalizarea lotului (“batch normalization”)
- funcție de activare de tip “ReLU” (Rectified Linear Unit)
- convoluție cu nucleu de dimensiune 3 x 3 în primele trei niveluri de codificare (d1, d2, d3) și nucleu de 5 x 5 la ultimele doua niveluri de codificare (d4, d5)
- normalizarea lotului (“batch normalization”)
- funcție de activare de tip “ReLU”
- operație de agregare “max pooling” cu pas (“stride”) de 2 x 2

Stratul central al rețelei este compus din:

- convoluție cu nucleu 3 x 3
- normalizarea lotului (“batch normalization”)
- funcție de activare de tip “ReLU”
- convoluție 3 x 3
- normalizarea lotului
- funcție de activare de tip “ReLU”

Un strat de decodificare are următoarele operații:

- deconvoluție (“up sampling”) cu nucleu 2 x 2

- concatenare la nivel de canale (de exemplu primul strat de decodificare ”d1” este concatenat cu ultimul din codificare ”u5”)
- deconvoluție cu nucleu 3 x 3 în primele trei niveluri de codificare (u1, u2, u3) și nucleu de 5 x 5 la ultimele doua niveluri de codificare (u4, u5)
- normalizarea lotului
- funcție de activare de tip “ReLU”
- deconvoluție cu nucleu 3 x 3 în primele trei niveluri de codificare (u1, u2, u3) și nucleu de 5 x 5 la ultimele doua niveluri de codificare (u4, u5)
- normalizarea lotului
- funcție de activare de tip “ReLU”
- deconvoluție 3 x 3
- normalizarea lotului
- funcție de activare de tip “ReLU”

Ieșirea rețelei va fi dată de rezultatul a încă unui strat convoluțional cu nucleu 1 x 1 și cu o funcție de activare de tip sigmoidă. O reprezentare grafică a nivelurilor rețelei este ilustrată în Figura 26.

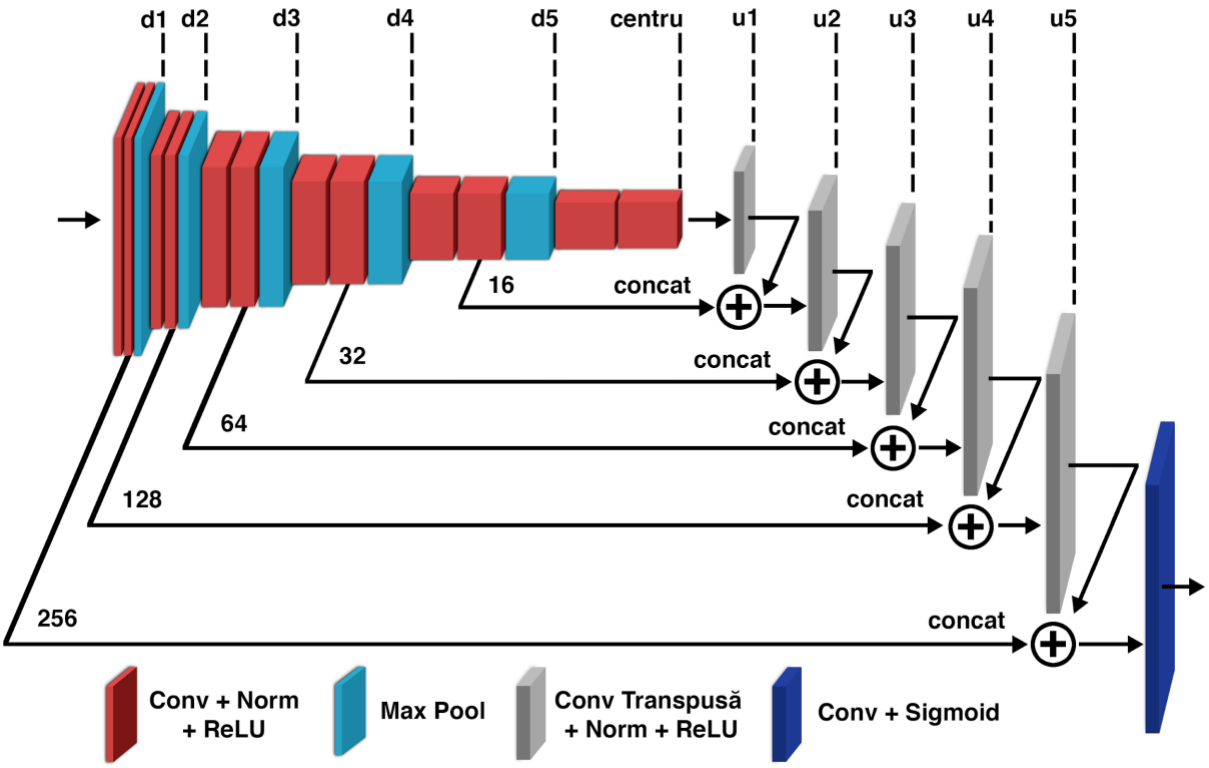


Figura 26. Arhitectura rețelei U-net modificată.

Antrenarea am efectuat-o într-un maxim de 50 de epoci (“epochs”) și cu o dimensiune a lotului de date (“batch size”) de 16. Astfel la 28366 imagini de antrenare am folosit 1772 pași de antrenare (“steps per epoch”), iar pentru validare am folosit un total de 220 de pași (“validation steps”). Timpul de antrenare a unei epoci este în medie 460 secunde (~336 ms / pas) pe un sistem desktop echipat cu 2 plăci video de tip Nvidia 1080 Ti. O analiză a

minimizării funcției de cost în timpul antrenării timp de 50 de epoci este prezentată în Figura 27. Se poate observa oprirea antrenării după 34 de epoci, deoarece am folosit un proces de monitorizare care este declanșat dacă nu se îmbunătățește rezultatul după 5 epoci consecutive. În grafic se observă și faptul că rezultatul funcției de cost scade, iar metrica IoU ("dice coeff") converge spre valoarea 1, ceea ce înseamnă o suprapunere foarte bună între imaginea de predicție a rețelei și imaginea eticheta de antrenare.

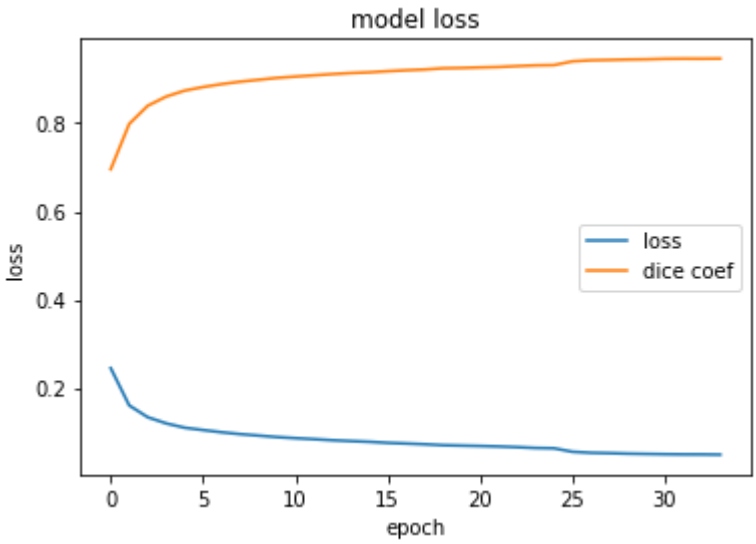


Figura 27. Rezultatul antrenării după 34 de epoci: funcția de cost ("loss") scade, iar coeficientul Sorensen-Dice ("dice coeff") converge spre valoarea 1.

O predicție a rețelei convoluționale neuronale pe o imagine din setul de testare din baza de date CityScapes este prezentată în Figura 28.

Figura 28. Exemplu de predicție a rețelei. Stânga: imaginea de intrare color, centru: rezultatul segmentării (predicția), dreapta: imaginea etichetată ("ground truth"). Sursă imagine: [Cordts2016].

Putem observa capabilitatea rețelei de a distinge precis între zona de carosabil și alte obiecte sau obstacole din scenă. De asemenea, această soluție produce rezultate foarte bune și pe imagini capturate cu sistemul propriu de achiziție, după cum se poate observa în Figura 29. Aceste imagini sunt complet diferite față de cele din setul de antrenare.

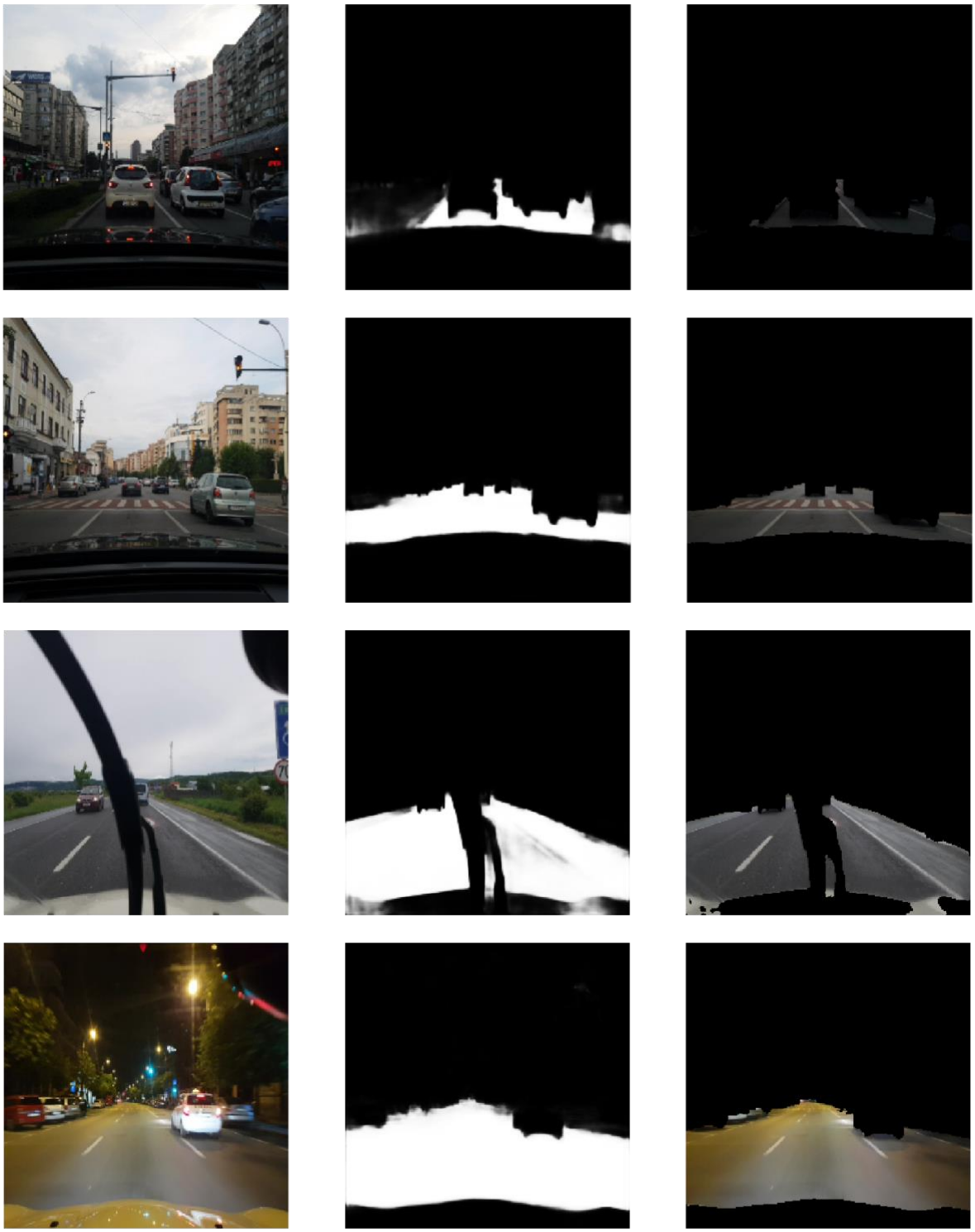


Figura 29. Rezultatele segmentării: stânga: imaginea color de intrare, centru: imaginea segmentată (predicția rețelei), dreapta: imaginea de intrare având ca mască imaginea segmentată.

Totodată, în Figura 29 se poate remarca și performanța bună a rețelei în scenarii de trafic complexe unde sunt multe vehicule în scenă. Această delimitare precisă a ce este carosabil și ce nu este foarte utilă pentru crearea unui sistem de percepție monocular unde

nu avem informații despre adâncime sau alte informații din scenă. Astfel, am ales folosirea acestei zone “free space” pentru a crea un sistem de percepție monocular folosind un filtru de particule bazat pe o hartă de măsurători. Probabilitatea predicției este dată de intensitatea imaginii, astfel că o probabilitate de 1 (maximă) ca un pixel să fie în clasa corectă va fi egală cu valoarea intensității egală cu 255 (alb).

Din testele efectuale am observat de asemenea faptul că datorită folosirii unor praguri mari pentru variația saturației și a intensității în spațiul de culoare HSV din timpul antrenării, rețeaua este capabilă să genereze predicții corecte chiar dacă imaginea de intrare este de tip “grayscale”. Acest lucru reprezintă un avantaj, deoarece rețeaua se poate folosi și cu baze de date din trafic mai vechi, unele bazate pe sisteme de stereo-viziune, care de obicei sunt “grayscale”.

Timpul de predicție mediu pe seturile de validare a fost de 15 milisecunde pe un sistem dotat cu 2 plăci video Nvidia 1080 Ti și de 410 milisecunde pe un CPU Intel i7 6700K. Pentru evaluarea performanței am folosit setul de validare CityScapes care conține 500 imagini și adnotări, dar și setul de validare al bazei de date BDD care are 965 imagini și adnotări. Am calculat următoarele metrici pentru evaluarea și analiza performanței: IoU (“intersect over union”), numărul de pixeli ai clasei clasificați corect (“true positive” - TP), numărul de pixeli din afara clasei care sunt clasificați incorect ca parte din clasa (“false positive” - FP), pixeli din afara clasei clasificați corect (“true negative” - TN) și pixeli ai clasei care sunt clasificați incorect că parte din altă clasă (“false negative” - FN). Pe baza acestora se pot calcula următoarele statistici: rata de detecții corecte a pixelilor de clasă (“true positive rate” - TPR) sau “sensitivity” care e o măsură a proporției de pixeli TP care sunt detectați corect, rata de detecții corecte a pixelilor care nu sunt parte din clasa (“true negative rate” - TNR) sau “specificity” reprezintă proporția de pixeli TN care sunt detectați corect. Am calculat și rata de detecție a pixelilor FP (“false positive rate” - FPR), acuratețea (ACC - numărul de pixeli clasificați corect împărțit la numărul total de pixeli), precizia (PPV) proporția de rezultate corecte (numărul de pixeli TP împărțit la TP și FP) și “F1 score”, o măsură a acurateței unui test, care este media armonică dintre ACC (acuratețe) și TPR (“sensitivity”). Rezultatele pe setul de date de validare CityScapes este prezentat în Tabel 4.

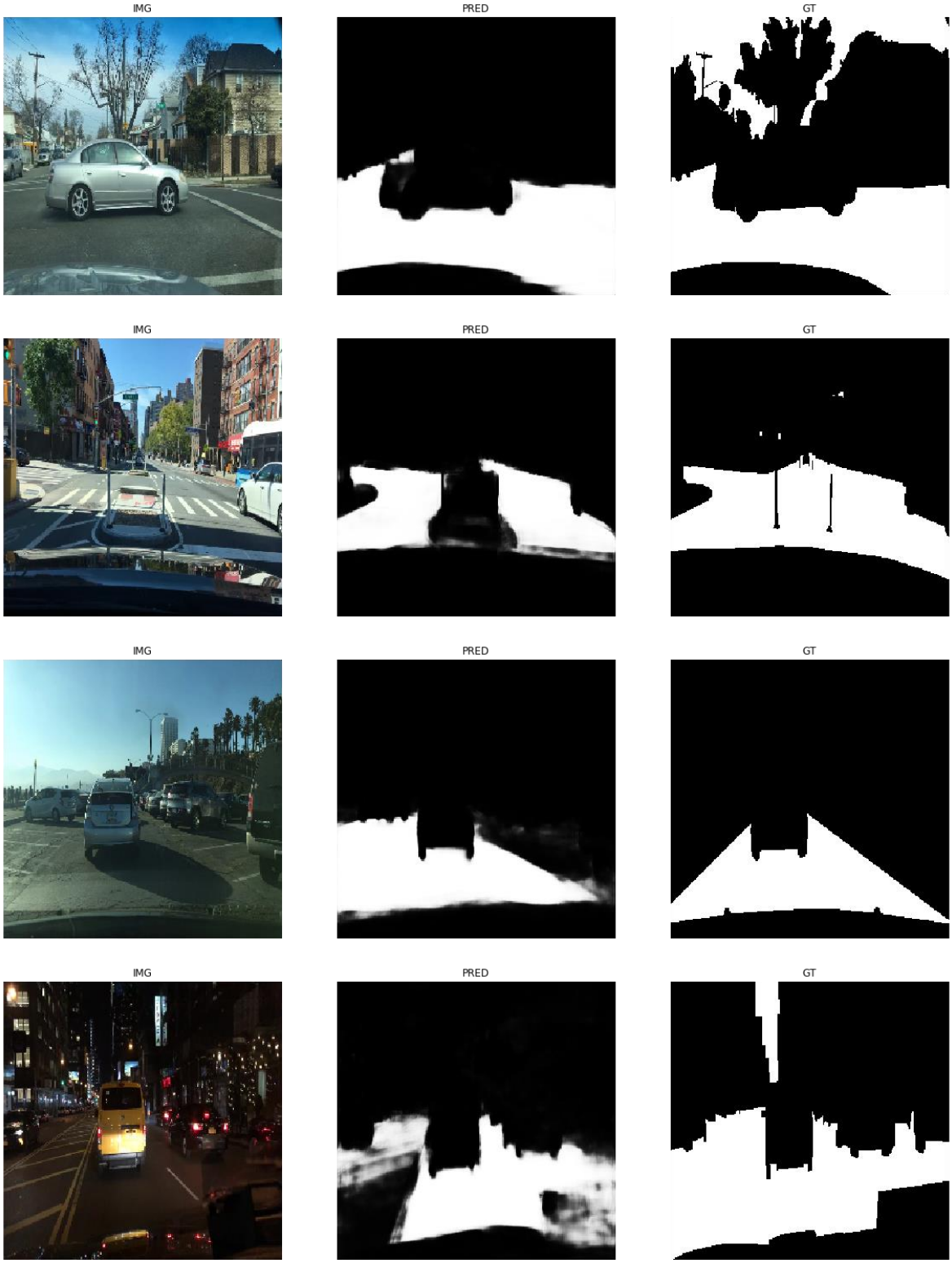
Tabel 4. Comparație între rezultatele obținute pe setul de validare CityScapes complet și cel filtrat.

	CityScapes complet	CityScapes filtrat
TPR	0.955701892257	0.95438612877
TNR	0.982337011493	0.98192927691
FPR	0.0176629885072	0.0180707230898
PPV	0.96130156874	0.964383092247
ACC	0.973773363955	0.972594330276
F1	0.949254206254	0.959358567961
IoU	0.911535794241	0.9218915742

În Tabel 4 am prezentat o analiză între setul de validare complet (500 imagini) și același set filtrat (483 imagini). Din setul inițial am eliminat imaginile în care adnotările nu sunt relevante pentru detecția de drum: de exemplu sunt imagini în care zona de drum (“driveable”) nu este marcată corespunzător sau chiar deloc (Figura 30). În aceste situații rețeaua dezvoltată de mine oferă o predicție deși imaginea de adnotare (“ground truth”) nu este corectă, iar acest lucru va afecta metricile și statisticile.

Figura 30. Exemple de imagini din setul de validare CityScapes în care adnotarea pentru drum nu este realizată (coloana din dreapta), iar rețeaua reușește să identifice corect zone de drum pe care se poate conduce (în centru) - vehiculele parcate confirmă faptul că zona poate fi considerată “driveable”.

În setul de validare al bazei de date Berkeley Deep Drive (BDD) am identificat de asemenea erori la adnotări. Acest set prezintă abateri foarte grave, deoarece există imagini în care zona de cer este codificată la fel ca și cea de drum, rezultând imagini de adnotări complet eronate. Diferite astfel de exemple sunt ilustrate în figura de mai jos.


IMG
PRED
GT
IMG
PRED
GT
IMG
PRED
GT
IMG
PRED
GT

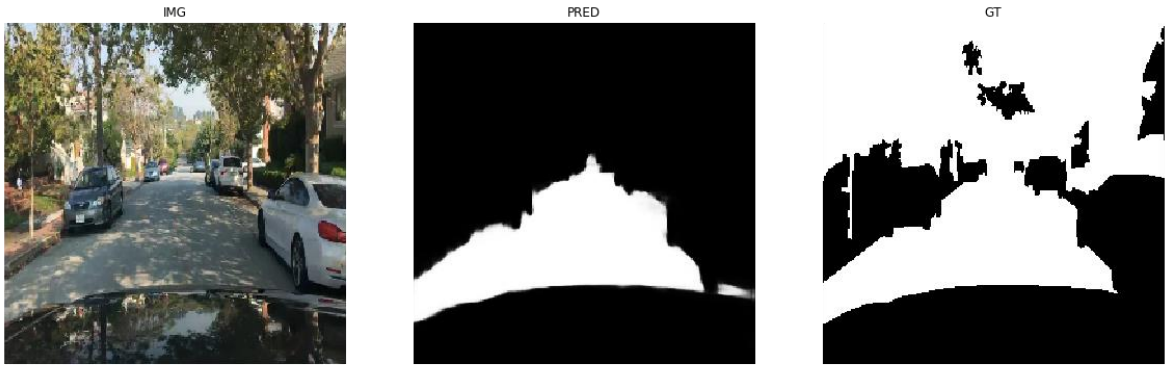


Figura 31. Exemple de adnotări eronate din setul de validare al bazei de date Berkeley Deep Drive. În unele imagini se poate observa că zona de cer este marcată ca și carosabil, în timp ce alte imagini au adnotări incomplete sau parțiale. Se poate remarca faptul că în aceste situații rețeaua oferă predicții foarte apropiate de realitate. Sursă imagini: [Yu2018].

Majoritatea imaginilor de acest gen au fost eliminate și a rezultat un set de date filtrat având un total de 938 imagini, în timp ce setul complet de validare conține 965 imagini. Analiza statistică este prezentat în Tabel 5, unde se poate observa și diferența de precizie și acuratețe.

Tabel 5. Comparație între rezultatele obținute pe setul de validare Berkeley Deep Drive complet și cel filtrat.

	BDD complet	BDD filtrat
TPR	0.961435813189	0.965269719435
TNR	0.983920917501	0.985030022461
FPR	0.0160790824994	0.0149699775389
PPV	0.935984230719	0.941434547361
ACC	0.978937005997	0.980868157789
F1	0.944925543374	0.950524922466
IoU	0.90792355061	0.9173262331

Am efectual aceeași analiză și pe setul de date KITTI și KITTI Semantic, iar rezultatele sunt în tabelul următor.

Tabel 6. Analiza segmentării pe setul de validare KITTI și KITTI Semantic.

	KITTI	KITTI SEMANTIC
TPR	0.9130502755	0.8611070882
TNR	0.9635749148	0.949929774
FPR	0.03642508525	0.05007022597
PPV	0.8561038004	0.8395499963
ACC	0.953883329	0.9292096384
F1	0.8836605262	0.8501919162
IoU	0.840760074	0.7881281115

Pentru aceste două baze de date ale Karlsruhe Institute of Technology rezultatele nu sunt la fel de bune, dar la o analiză a predicțiilor sub un anumit scor se poate observa că există anumite adnotări incomplete, unde rețeaua oferă uneori un rezultat mai bun decât ce e marcat în setul de date. O serie de astfel de exemple sunt ilustrate în Figura 32, Figura 33 și Figura 34.

Figura 32. Rețeaua oferă predicție corectă și asupra zonei de drum de pe sensul opus, dar are dificultate cu zona de trotuar ce este clasificată greșit. Sursă imagini: [Geiger2012].

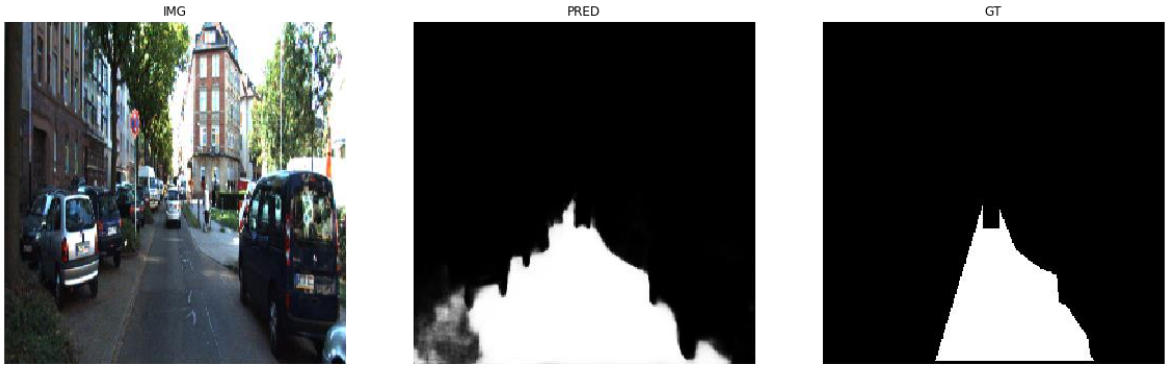


Figura 33. Zona de drum este clasificată complet până la limita vehiculelor parcate (centru) față de masca originală (dreapta). Sursă imagini: [Geiger2012].

Figura 34. Rețeaua are anumite dificultăți în zone de umbră și iluminare neuniformă a scenei. Sursă imagini: [Geiger2012].

Setul de validare Mapillary conține de asemenea unele imagini adnotate greșit, după cum se poate observa în figura următoare:

Figura 35. Exemplu de adnotare eronată în baza de date Mapillary în care zona de trotuar este marcată ca fiind zonă de drum (imaginea din dreapta). Sursă imagine: [Neuhold2017].

Pe setul de validare al bazei de date Mapillary scorul IoU este 0.87. De altfel, am identificat o bună parte de imagini adnotate la fel de eronat și în seturile de antrenare, în special în baza de date BDD. Se poate remarca faptul că uneori cifrele rezultate din statistici s-ar putea să nu reflecte întotdeauna realitatea, iar aceste rețele neuronale convoluționale sunt dependente în mare parte de un număr cât mai mare de date adnotate corect. Rețeaua prezentată în această secțiune reușește să nu "învețe" greșit aceste adnotări eronate, dar totuși are uneori dificultăți în detecția zonei de drum, în special unde sunt zone de pavaje sau de iluminare foarte proastă (cel mai adesea cu soarele direct în față). Uneori și partea de trotuar este clasificată incorect, iar o parte din aceste detecții proaste este prezentată în figura de mai jos:

Figura 36. Exemplu de predicții eronate ale rețelei de segmentare. În centru este ilustrată predicția rețelei, iar în dreapta este imaginea “ground truth”. Sursă imagini: [Cordts2016], [Yu2018].

Majoritatea dintre aceste erori se pot rezolva prin suplimentarea de date corecte (în special pentru zone de pavaje) sau prin extinderea rețelei pentru predicții multi-clasă și antrenarea mai multor clase (de exemplu antrenare drum și trotuar separat). Totuși, în

condițiile în care anumite imagini din setul de antrenare sunt greșite, rețeaua dezvoltată de mine a obținut o valoare a metricii IoU de 0.91 și 0.90 pe cele două seturi de validare filtrate (CityScapes și BDD).

Tabel 7. Comparație cu metode similare pe setul de validare din baza de date CityScapes.

Model CNN	**Scor ”IoU” drum**
DeepLab [Chen2018]	0.986
E-Net [Paszke2016]	0.974
CNN multi-class teză	0.922
CNN teză	0.911

În Tabel 7 am prezentat rezultatele de evaluare pe setul de validare din baza de date CityScapes [Cordts2016]. Rezultatele obținute de mine sunt comparabile cu cele din literatura de specialitate. Diferența de scor este dată de faptul că imaginile de antrenare folosite în alte soluții sunt de minim două ori mai mari: de obicei de la 500x500 pixeli sau mai mari, față de 256x256 pixeli cât am folosit eu. Dimensiunea imaginilor de intrare are un efect direct proporțional asupra timpului de predicție, de exemplu pe imagini de 512x512 pixeli timpul de predicție este de 33-35ms, în timp ce pe imaginile 256x256 timpul este de 15ms. Totodată, rezultatele obținute în cadrul acestei teze sunt fără post-procesare (de exemplu CRF), pe imaginea segmentată se aplică doar o binarizare cu prag fix. Totodată, există și posibilitatea ca rețelele cu scor foarte mare de IoU să reflecte o învățare eronată pe imaginile adnotate greșit (se face de fapt ”overfitting”). În tabelul de mai sus am prezentat și scorul obținut prin antrenarea aceleași rețele extinsă (”CNN multi-class”) încât să prezică un total de patru clase (drum, trotuar, vehicule și cer), iar rezultatul este cu un procent mai mare decât rețeaua antrenată să prezică doar clasa de drum.

Scopul soluției prezentate în această teză este de a avea o segmentare suficient de bună și de rapidă încât să poată fi folosită ca hartă de măsurare a scenei într-un filtru de particule.

2.13.4 Urmărire prin particule folosind o hartă de ocupare

Imaginile segmentate folosind tehnicile descrise în secțiunea anterioară sunt integrate într-un framework probabilistic, bazat pe o hartă dinamică a scenei din jurul autovehiculului. Din detecția zonei libere de drum, putem considera că tot ce nu este drum reprezintă un posibil obstacol și poate fi interpretat și urmărit în timp. Astfel, această soluție este utilă pentru a detecta orice tip de obstacol, nefiind limitată doar la pietoni sau vehicule. Pentru implementarea acestei funcționalități am ales folosirea unei hărți de ocupare bazată pe un filtru de particule. Această soluție reprezintă o implementare proprie a ideilor prezentate în lucrarea [Danescu2011].

Sisteme bazate pe filtre de particule sunt folosite în sisteme pentru roboți mobili sau vehicule autonome pentru urmărirea obiectelor sau pentru localizare. Abordările bazate pe filtre de particule sunt intense din punct de vedere computațional, dar dezvoltarea soluțiilor hardware și creșterea puterii de procesare din ultimii ani facilitează crearea unor sisteme în timp real. Aceste abordări pot fi folosite pentru estimarea sistemelor neliniare, multi-modale.

Harta de ocupare bazată pe filtre de particule este o reprezentare dinamică a scenei care are efectul de perspectivă eliminat. În această implementare obstacolele din scenă sunt compuse din particule, fiecare având o poziție individuală și un vector de viteză. Harta de ocupare este structurată în celule de 20 cm x 20 cm. Probabilitatea unei celule de a fi ocupată este dată de numărul de particule din acea celulă. Particulele din această soluție au un rol dublu: ele formează ipoteze (similar cu algoritmul CONDENSATION [Isard1998]) sau pot fi considerate elemente de bază a mediului, cu proprietatea că particulele se pot deplasa dintr-o celulă în alta, astfel reușind o reprezentare dinamică a scenei înconjurătoare. Prin deplasarea particulelor se va realiza și partea de predicție a stării scenei.

Crearea și migrarea particulelor se face având la bază o măsurătoare a scenei cu efectul de perspectivă eliminat (IPM), în acest caz reprezentată de o hartă binară de ocupare în care obstacolele sunt reprezentate cu alb și fundalul cu negru. Generarea hărții binare de ocupare se poate face folosind tehnica descrisă în secțiunea 2.12.3, unde imaginea scenei cu efectul de perspectivă eliminat este binarizată utilizând un prag adaptiv (sau fix) și apoi analizată folosind linii radiale care pornesc din centrul camerei. Totuși, sistemul propus în această teză folosește direct imaginea de drum segmentată de CNN din care apoi este eliminat efectul din perspectivă. Această imagine este apoi analizată folosind tehnica din secțiunea 2.12.3: se scanează zona din fața vehiculului propriu începând din punctul în care este montată camera prin folosirea unor linii radiale. De-a lungul acestor linii se va găsi punctul de contact al unui obstacol cu asfaltul (ecuațiile 45-48). Un exemplu de hartă binară de ocupare este ilustrat în figura următoare:

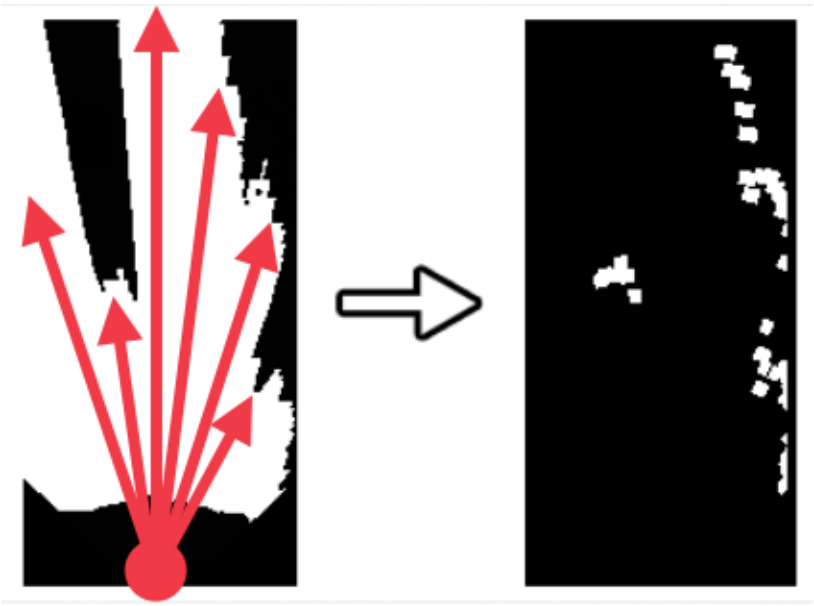


Figura 37. Harta binară de ocupare (dreapta), obținută din imaginea de drum segmentată (stânga). Sunt ilustrate și o parte din razele de-a lungul cărora se caută tranziții de intensitate.

Sistemul de coordonate folosit în sistemul implementat va avea axa Z aliniată longitudinal cu vehiculul și axa e orientată înspre față, iar axa X este transversală cu vehiculul și orientată spre dreapta. Obstacolele din scenă sunt reprezentate de un set de particule definit astfel:

$$S = \{p_i \mid p_i = (c_i, r_i, vc_i, vr_i), i = 1 \dots N_S\} \quad (49)$$

Fiecare particulă are o poziție în harta de ocupare și o componentă de viteză. Poziția unei particule este definită de rândul *"r_i"*, corespunzător distanței de-a lungul axei Z a scenei 3D și o coloană *"c_i"*, reprezentând poziția laterală pe axa X, iar viteza este definită de *"vr_i"* și *"vc_i"* (componente de viteză pentru fiecare axă a hărții de ocupare). Numărul total de particule din scenă *"Ns"* nu este cunoscut inițial și depinde de numărul de obstacole detectate în scenă, dar acest număr (Ns) este actualizat de fiecare dată când un nou obstacol este descoperit prin măsurare.

Probabilitatea unei celule "C" de a fi ocupată de un obstacol (ecuația 50) este estimată că raportul dintre numărul de particule a căror poziție coincide cu poziția celulei "C" și numărul total de particule permise pentru o singură celulă *Nc*.

$$P_O(C) = \frac{|\{p_i \in S \mid r_i = r_c, c_i = c_c\}|}{N_C} \quad (50)$$

Parametrul *"Nc"* reprezintă numărul total de particule dintr-o celulă *"C"* suportat de către sistem și este constant. În unele implementări pe dispozitive mobile l-am setat la o valoare redusă, de exemplu 50, în timp ce pe un sistem desktop poate fi setat mai mare: 200 de particule totale într-o celulă. Alegerea acestei valori constante reprezintă un compromis între acuratețea dorită și performanța sistemului. O valoare mare înseamnă că pentru o celulă se mențin simultan mai multe ipoteze de viteză, prin urmare sistemul de urmărire va avea o estimare mai bună a vitezei și poate gestiona mai bine obiectele care se mișcă rapid în scenă. Cu toate acestea viteza algoritmului va scădea, deoarece numărul total de particule din scenă va fi direct proporțional cu *"Nc"*.

Dacă presupunem că există doar un singur obiect într-o celulă din harta de ocupare, putem calcula viteza celulei ca fiind egală cu viteza medie a particulelor asociate acesteia (ecuația 51):

$$(vc_c, vr_c) = \frac{\sum_{p_i \in S, x_i = x_c, z_i = z_c} (vc_i, vr_i)}{|\{p_i \in S \mid r_i = r_c, c_i = c_c\}|} \quad (51)$$

Astfel, populația de particule este suficient de reprezentativă pentru densitatea de probabilitate a ocupării și a vitezei pentru întreaga hartă de ocupare. Ipotezele cu viteză multiplă pot fi menținute simultan pentru o singură celulă, iar incertitudinea de ocupare este reprezentată de numărul variabil de particule asociate celulei. Algoritmul de urmărire care utilizează modelul lumii are la bază crearea, deplasarea și distrugerea particulelor din celule, un proces condus de informațiile de măsurare (harta de ocupare binară).

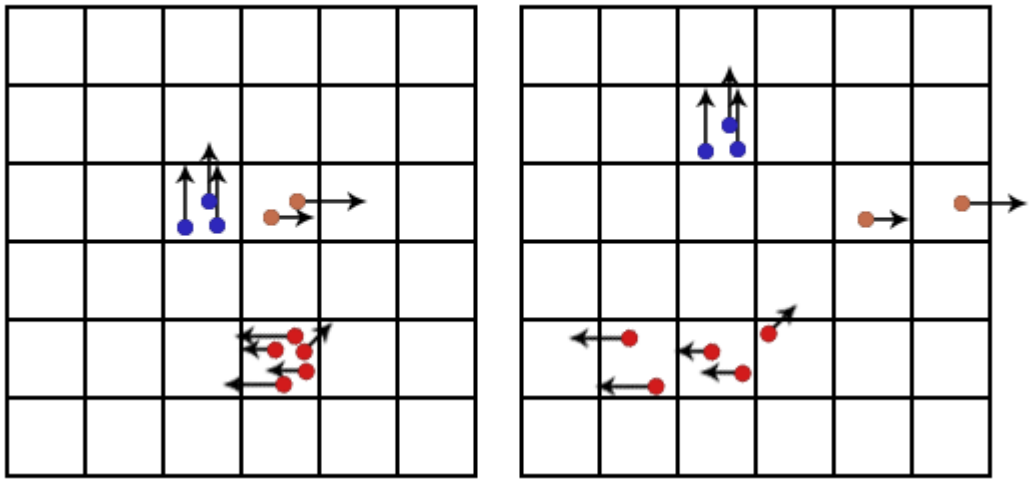


Figura 38. Migrarea particulelor dintr-o celulă în alta în harta de ocupare.

Primul pas al algoritmului de urmărire este cel de predicție și este aplicat pe fiecare particulă. Poziția particulelor din set este modificată (Figura 38) în funcție de viteza lor (ele se mișcă în concordanță cu vectorul propriu de viteză) și în funcție de parametri de mișcare ai vehiculului propriu citiți din senzorii vehiculului sau ai dispozitivului mobil: GPS, accelerometru și giroscop (ecuația 52).

$$d = \frac{2v\Delta t sin\frac{\psi}{2}}{\psi} \tag{52}$$

În ecuația 52 Δt reprezintă intervalul de timp dintre măsurători, în care vehiculul se deplasează o distanță d, având o viteză de deplasare v, iar Ψ este rata de rotație a vehiculului propriu (”yaw rate”). O particulă din harta de ocupare este deplasată folosind ecuația 53, unde d_c și d_r sunt calculate din distanța d (ecuația 52) ajustată în funcție de rata de rotație în raport cu dimensiunea celulei.

$$\begin{bmatrix} c_n \\ r_n \end{bmatrix} = \begin{bmatrix} cos\psi & -sin\psi \\ sin\psi & cos\psi \end{bmatrix} \begin{bmatrix} c \\ r \end{bmatrix} - \begin{bmatrix} d_c \\ d_r \end{bmatrix} \tag{53}$$

Modelarea incertitudinii modelului de mișcare în raport cu mișcarea obiectelor din lumea reală este realizată prin adăugarea unor valori aleatorii la poziția și viteza fiecărei particule (δc ,δr ,δvc și δvr), având efectul unei difuzii stohastice (ecuația 54).

$$\begin{bmatrix} c \\ r \\ v_c \\ v_r \end{bmatrix} = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_n \\ r_n \\ v_c \\ v_r \end{bmatrix} + \begin{bmatrix} \delta c \\ \delta r \\ \delta vc \\ \delta vr \end{bmatrix} \tag{54}$$

Al doilea pas al algoritmului este procesul de actualizare, mai specific procesarea informațiilor măsurate și se bazează pe harta binară de ocupare a celulelor creată prin procesarea imaginii IPM.

Informațiile din măsurare sunt folosite pentru a pondera particulele și apoi pentru a le reeșantiona într-un singur pas. Prin ponderare și reeșantionare, particulele dintr-o celulă se

pot multiplica sau distruge, iar întreg procesul este descris în lucrarea [Danescu2011]. Ultimul pas este de a calcula viteza și apoi de a estima probabilitatea unei celule de a fi ocupată de un obstacol.

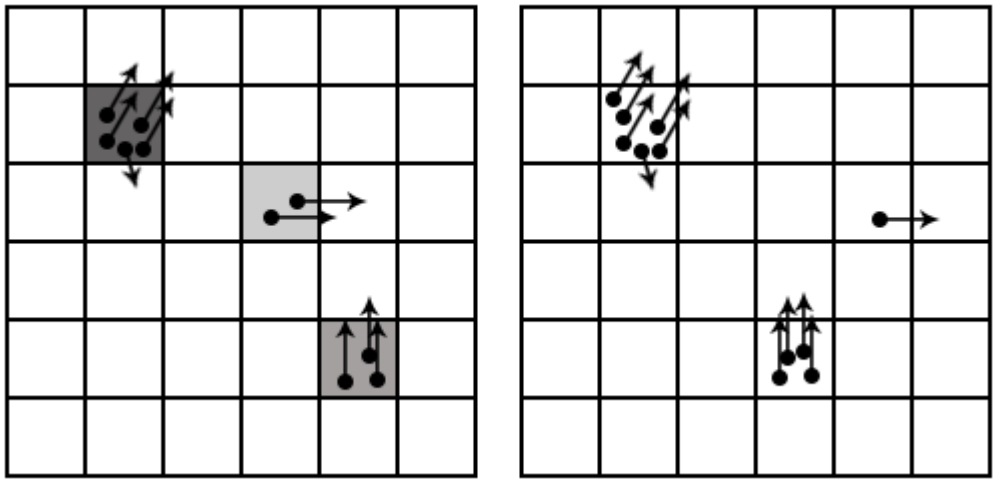


Figura 39. Ponderare și reeșantionare particule în celulele hărții de ocupare. Ponderea de ocupare a unei celule este codificată prin intensitatea celulei în imaginea din stânga. În dreapta se observă efectul actualizării: particulele sunt create sau eliminate.

În procesul ajustării și actualizării particulelor este utilizată harta binară obținută în urma măsurării scenei, iar în lucrarea [Danescu2011] autorii au folosit o hartă binară obținută din prelucrarea datelor măsurate cu un sistem de stereo-viziune. Incertitudinea poziției unui obstacol este dată de incertitudinea procesului de măsurare folosind stereo-viziunea. Poziția laterală a unui obstacol depinde de incertitudinea distanței față de obstacol. Pentru a folosi această abordare pe un sistem monocular, a fost adaptată metoda de calculare a incertitudinii distanței spre un obstacol folosind imaginea cu efectul de perspectivă eliminat (IPM) și întreaga soluție a fost publicată în [Dănescu2016a]. Procesul de estimare a distanței dintre vehicul și un obstacol din scenă este ilustrat în Figura 40.

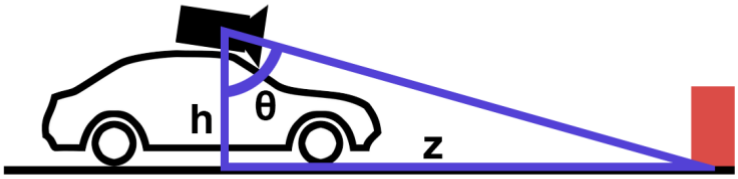


Figura 40. Măsurarea distanței spre obstacol în imagini IPM.

Unghiul de înclinare a camerei și înălțimea la care este montată față de sol *"h"* sunt cunoscute din procesul de calibrare. Corespondența dintre un punct în imaginea 2D și un punct 3D din scenă se face folosind proiecția perspectivă. Un pixel din imaginea IPM va corespunde unei linii din camera ce trece prin centrul optic formând unghiul *"θ"* cu linia

verticală (spre sol). Cunoscând *"h"* și unghiul *"θ"*, distanța *"z"* din Figura 40 se poate calcula folosind ecuația 55.

$$z = h\ tan\theta \tag{55}$$

Înălțimea camerei față de sol se poate considera fixă, astfel doar unghiul *"θ"* va determina o incertitudine în procesul de măsurare a scenei, deoarece vehiculul va circula pe suprafețe neregulate, uneori denivelate și va accelera și decelera (afectând unghiul de observare a obstacolelor din scenă). Abaterile mici ale suprafeței reale a drumului de la presupunerea inițială a drumurilor plate se pot modela ca fiind erori în unghiul *"θ"*. Totuși deviațiile semnificative, care nu sunt temporare, nu pot fi modelate în acest sistem și vor conduce la estimări și rezultate greșite. În primă fază, pentru a trata aceste situații am ales o valoare $\sigma\theta = 0.25°$ determinată experimental, iar apoi am folosit punctul de fugă pentru calcularea unghiului de înclinare și pentru corectarea deviațiilor (capitolul 3).

Relația dintre incertitudinea distanței *"z"* și *"θ"* este de fapt deviația standard a erorii unei măsurători *"z"* și se calculează astfel:

$$\sigma_z = \frac{dz}{d\theta}\sigma_\theta \tag{56}$$

Ecuația 56 se poate rescrie:

$$\sigma_z = h(1 + tan^2\theta)\sigma_\theta \tag{57}$$

Folosind ecuația 55, prin înlocuirea tangentei se poate obține ecuația finală pentru incertitudinea distanței "z" (ecuația 58).

$$\sigma_z = h\left(1 + \frac{z^2}{h^2}\right)\sigma_\theta + \sigma_{z\theta} \tag{58}$$

Valoarea σ_{z0} reprezintă o constantă cu rolul de a trata alte incertitudini sau erori care nu se pot modela. Din σ_z putem determina incertitudinea poziției laterale "x" (ecuația 59).

$$\sigma_x = \frac{x\sigma_z}{z} + \sigma_{x0} \tag{59}$$

În ecuațiile de mai sus, σ_z și σ_x modelează deviația standard a erorii la măsurare, exprimată în coordonatele *"Z"* și *"X"* din harta dinamică de ocupare, scalate la dimensiunea unei celule *D = 20cm*. Pentru fiecare celulă din harta dinamică de ocupare, funcția de densitate de probabilitate a unei celule de a fi ocupată în urma unei măsurători este calculată folosind funcția Gaussiană bivariată, care utilizează distanța pe coloane *"dx"* și distanța pe linii *"dz"* calculată între poziția în celulă și poziția obstacolului cel mai apropiat din harta binară de obstacole. Folosind aceste distanțe și valorile incertitudinilor din ecuațiile 58 și 59, probabilitatea unei celule de a fi ocupată este definită astfel:

$$p_{ocupat} = \frac{1}{2\pi\sigma_z\sigma_x} e^{-\frac{1}{2}\left(\left(\frac{d_z}{\sigma_z}\right)^2 + \left(\frac{d_x}{\sigma_x}\right)^2\right)} \tag{60}$$

În funcție de probabilitatea măsurătorii p_{ocupat}, particulele dintr-o celulă sunt fie multiplicate, fie distruse. Astfel, populația de particule este aliniată cu măsurătoarea scenei. Metodele hărții de ocupare dinamice a scenei pentru crearea particulelor noi, deplasarea, multiplicarea sau distrugerea lor, folosește datele din harta binara obținută în urma măsurătorii scenei (analiza imaginii IPM).

Deoarece în celulele individuale ale hărții de ocupare se află particule, ele pot fi grupate în funcție de asemănările lor, astfel încât se pot extrage cuboide din ele. Algoritmul de grupare ia în calcul proximitatea celulelor, vectorii lor de viteză și deplasarea lor pentru a extrage regiuni conectate care vor reprezenta un obiect. Două celule sunt grupate dacă: distanța dintre ele este de maxim 3 (poate fi o celulă goală între), diferența de orientare dintre vectorii de viteză este de mai puțin de 30°, și dacă ambele celule au probabilitatea de a fi ocupate de un obstacol de minim 0.5. În final, din aceste zone conectate este extrasă o formă dreptunghiulară orientată, iar din aceasta va fi construit cuboidul 3D (înălțimea cuboidului este setata fixă având 1,5 metri pentru toate obiectele). Algoritmul de extragere de cuboide din particule este același cu cel prezentat în lucrarea [Dănescu2016a].

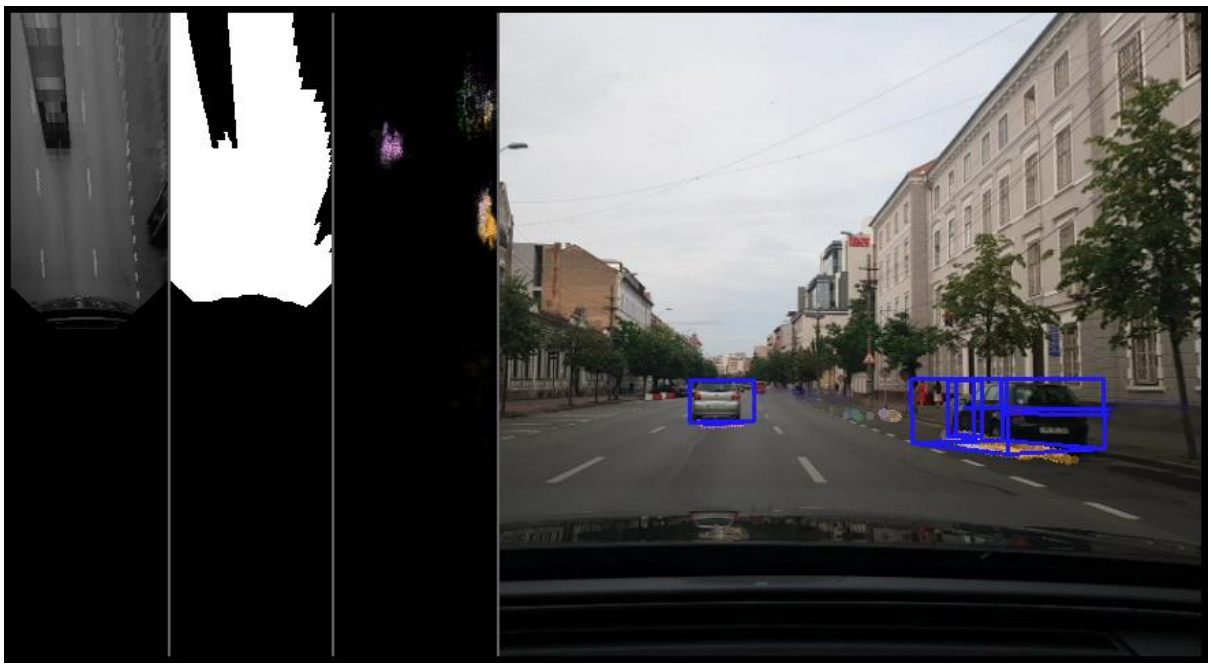


Figura 41. Framework-ul probabilistic de percepție a scenei.

Figura 41 ilustrează etapele de procesare din cadrul framework-ului probabilistic. În imaginea din stânga se pot observa etapele procesării și crearea hărții de ocupare dinamică: prima imagine reprezintă scena cu efectul de perspectivă eliminat (IPM), în a doua este ilustrată imaginea segmentată IPM (obținută din rețeaua neuronală convoluțională), a 3-a imagine ilustrează particulele și culoarea lor este codificată în funcție de vectorii de viteză (la fel ca în [Dănescu2011]: nuanța culorii reprezintă direcția de mers, saturația reprezintă magnitudinea vitezei, iar intensitatea codifică probabilitatea de ocupare). În imaginea din dreapta este prezentat rezultatul procesării și grupării particulelor și a celulelor în obstacole. Particulele sunt de asemenea proiectate și ilustrate în imaginea de perspectivă a scenei.

Pentru a evaluare acurateța sistemul de percepție, distanța până la obstacole este comparată cu distanța obținută dintr-un sistem bazat pe stereo-viziune. Rezultatul este ilustrat în tabelul figura următoare:

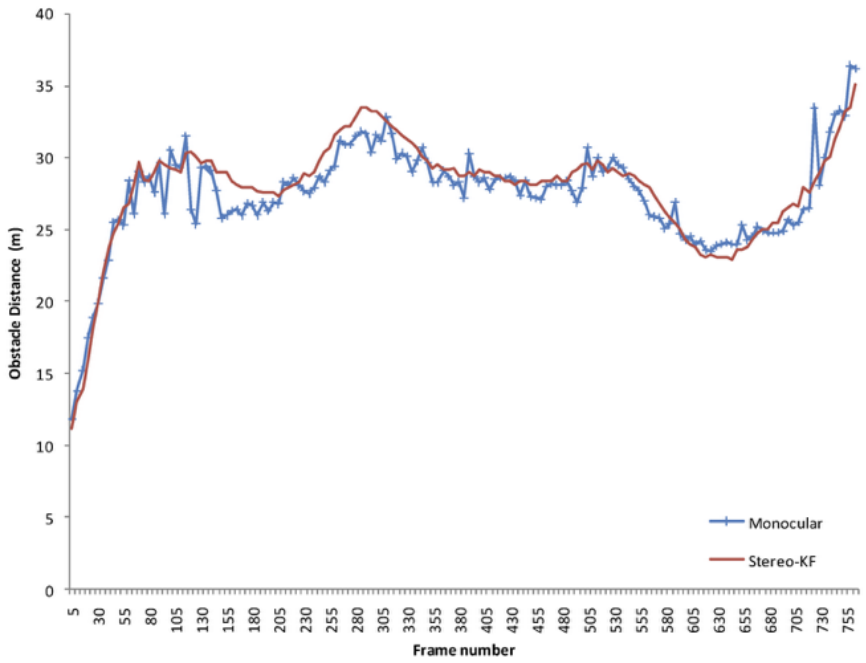


Figura 42. Comparare distanțe sistem propriu (monocular) vs. stereo-viziune. Axa verticală reprezintă distanța, iar cea orizontală reprezintă cadrul curent procesat.

În acest test este calculată distanța până la un singur vehicul aflat în față pe parcursul mai multor cadre (800, aproximativ 80 secunde), timp în care ambele vehicule accelerează sau decelează, modificându-și distanța dintre ele. Folosind sistemul bazat pe stereo-viziune ca și referință, eroarea MAE pentru măsurarea distanțelor este de 1.02m, iar RMSE este 1.33m. Unele figuri folosite în această secțiune au fost deja publicate în articolul [Danescu2016a].

2.13.5 Extensii și îmbunătățiri ale framework-ului probabilistic

Framework-ul probabilistic a fost extins pentru a putea urmări obiectele și când acestea ies din câmpul vizual al camerei foto. Astfel, în acest sistem imaginea IPM va avea o anumită dimensiune, iar harta de probabilitate poate fi diferită (în acest caz a fost setata lățimea egală cu cea a imaginii IPM și înălțimea egală cu dublul imaginii IPM). Pentru aceste modificări am setat o probabilitate ca o celulă să fie ocupată de 0.5 pentru toate celulele în afara câmpului vizual. În acest mod particulele care aparțin unor obiecte ce au ieșit cadrul imaginii sunt păstrate în harta de ocupare până vor “ieși” de tot din hartă sau vor fi distruse dacă nu sunt actualizate de o nouă măsurătoare.

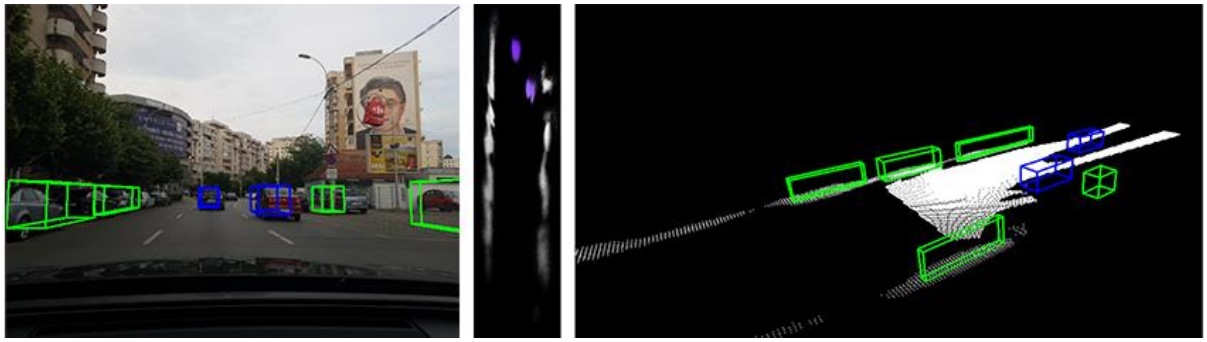


Figura 43. Se poate observa faptul că se mențin ipoteze despre obstacole (particule) și în spatele sau lateralul vehiculului propriu pentru obstacole care au fost deja urmărite, dar care au ieșit din zona vizibilă a imaginii.

În Figura 43 este ilustrată o captură de ecran dintr-o secvență pe o porțiune dreaptă de drum cu vehicule parcate pe fiecare parte și vehicule aflate în mișcare. Obstacolele statice extrase din filtrul de particule sunt ilustrate cu galben, iar cele dinamice sunt cu albastru. Se poate observa că se mențin ipoteze de obstacole și pentru cele ieșite din cadrul imaginii. Totodată, prima figură reprezintă imaginea de perspectivă cu cuboidele desenate, a doua imagine reprezintă particulele. În ultima figură am reproiectat scena dintr-o laterală, în care sunt ilustrate cuboidele, particulele și zona albă care este suprafața de drum segmentată de CNN.

a) Procesare vârfuri de histogramă

O altă îmbunătățire a framework-ului față de tehnicile pe care le-am publicat în [Danescu2016a], este pre-procesarea imaginii de măsurare. Pentru a îmbunătăți detecția de vehicule din zona segmentată (“free-space”) am folosit o tehnică având la bază unele idei din lucrarea [Bertozzi1998].

Obstacolele ating asfaltul doar cu roțile, iar acest lucru este amplificat în imaginile IPM, de unde rezultă o percepție falsă a distanței până la obstacole (în special la cele îndepărtate). Am rezolvat această problemă prin analiza histogramei radiale a fiecărui vehicul și prin unirea vârfurilor, adică umplerea zonei goale dintre roțile unui vehicul. Histograma radială este creată din raze având originea în punctul focal (Figura 44).

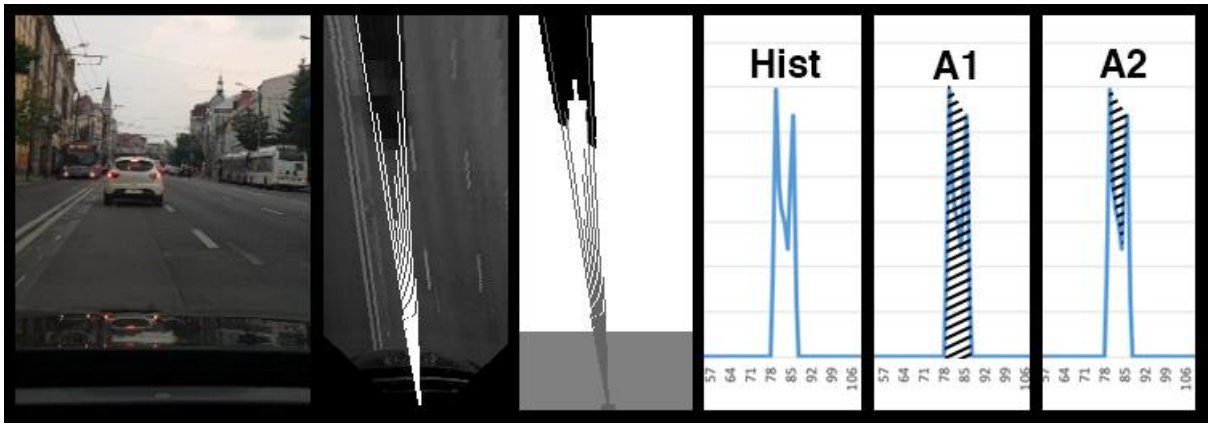


Figura 44. Prima imagine este cea de originală, a doua și a treia ilustrează histograma radială desenată peste imaginea IPM, respectiv imaginea IPM segmentată, iar ultimele trei figuri ilustrează: graficul histogramei, aria zonei A1 și aria zonei A2 (pe axa orizontală se observă și unghiul de scanare exprimat în grade).

Vârfurile din histogramă sunt identificate prin calcularea maximelor locale și apoi sunt procesate pentru a identifica perechi de linii care descriu marginile unui obiect. Pentru acest calcul am utilizat tehnica descrisă în [Bertozzi1998] și am calculat raportul dintre aria zonei *A1* din histogramă și aria valorilor din zona *A2* din histogramă. A1 reprezintă aria întregii zone dintre cele două vârfuri, în timp ce A2 reprezintă aria zonei "libere" dintre vârfuri (practic este A1 minus valorile din histograma între cele două vârfuri).

$$R = \frac{A1}{A2} \tag{61}$$

Ecuația 61 reprezintă calculul raportului pentru determinarea vârfurilor care vor fi conectate. Având aceste perechi de linii care vor corespunde unui obiect, următor pas a fost umplerea zonei dintre vârfuri. Această operație are rolul de a îmbunătăți semnificativ procesul de măsurare din filtrul de particule și al hărții de ocupare, deoarece este importantă determinarea zonei de contact și a distanței de la vehiculul propriu la restul de vehicule din scenă observată. Pentru umplere, este nevoie de găsirea celor 4 puncte ale poligonului convex: 2 puncte din linia din stânga a mărginii vehiculului și 2 puncte ale liniei din dreapta. Perechea de puncte ale unei linii este dată de punctul de contact din imagine (tranziția de la zona de "free space" la obiect) a celui mai înalt punct din vârful histogramei și punctul cel mai îndepărtat de pe linie spre extremitatea imaginii. Figura 45 reprezintă un exemplu unde aceste puncte sunt desenate sub forma unor cercuri gri.

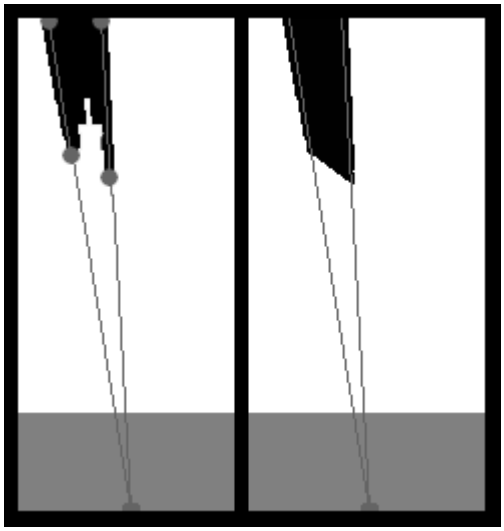


Figura 45. Punctele de contact pentru determinarea poligonului convex (stânga) și efectul umplerii poligonului (dreapta).

Efectul aplicării acestor operații este ilustrat în figura următoare (Figura 46), unde se poate observa o îmbunătățire semnificativă a modului în care sunt generate particulele în harta de ocupare (ele vor forma un obiect asemănător cu cel real).

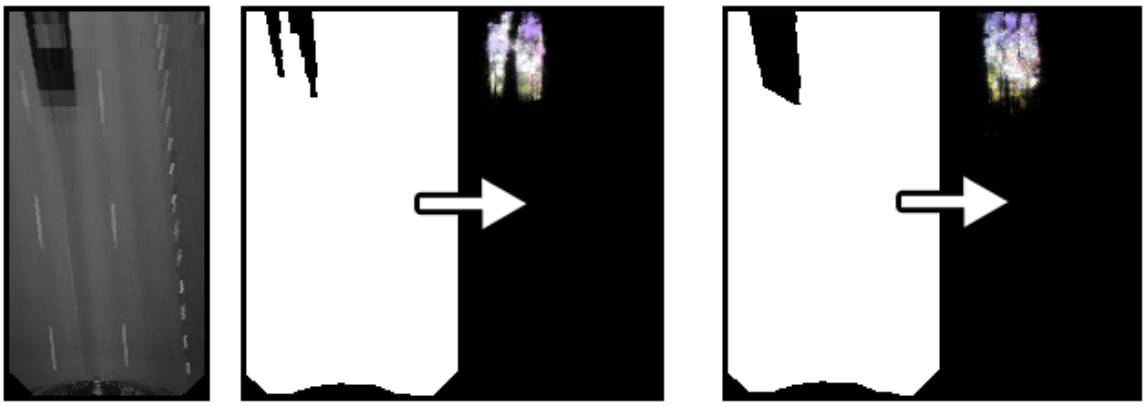


Figura 46. Îmbunătățirea hărții dinamice de ocupare. Perechi de imagini cu particulele generate în funcție de imaginea de măsurătoare (segmentată) utilizată.

Harta de măsurare pentru filtrul de particule este semnificativ îmbunătățită, iar procesul de creare a particulelor va funcția mai eficient și robust. Așadar, principalele dezavantaje ale unui sistem monocular pot fi soluționate și adresate prin utilizarea rețelelor neuronale convoluționale, dar și printr-o pre-procesare eficientă a datelor de intrare.

b) Rafinare parametri cameră

Parametri extrinseci referitori la unghiul de înclinare și cel de girație se pot corecta și rafina în timpul procesării. Pentru fiecare imagine dintr-o secvență se generează un număr

total de *N* imagini cu efectul de perspectivă eliminat (IPM), fiecare având un unghi de înclinare diferit. În mod similar se procedează și pentru a genera *N* imagini IPM având *N* unghiuri de girație diferite. Acest pas se poate aplica după ce se face o auto-calibrare a camerei (fie prin folosirea tehnicilor descrise în secțiunea 3.10 sau în 3.12.

În sistemul de percepție monocular am generat *N = 10* imagini IPM color ale scenei folosind variații de 0.1 grade în jurul valorii unghiului de înclinare calculat folosind filtrul Kalman extins din secțiunea 3.12.2. În următorul pas, cele 10 imagini IPM generate sunt binarizate cu un prag fix și apoi sunt detectate obstacolele din scenă folosind algoritmul descris în secțiunea 3.6. Aceste 10 imagini IPM binare reprezintă posibile hărți de ocupare, care sunt comparate cu harta de ocupare dinamică a filtrului de particule (generată din imaginea segmentată din CNN folosind algoritmul din secțiunea 3.6) și se calculează procentului de pixeli care se suprapun. Imaginea cu un procent mai mare de un prag fix reprezintă o potrivire mai bună a datelor asupra scenei urmărite, ceea ce înseamnă că se poate ajusta unghiul de înclinare pentru a mări precizia și robustețea sistemului.

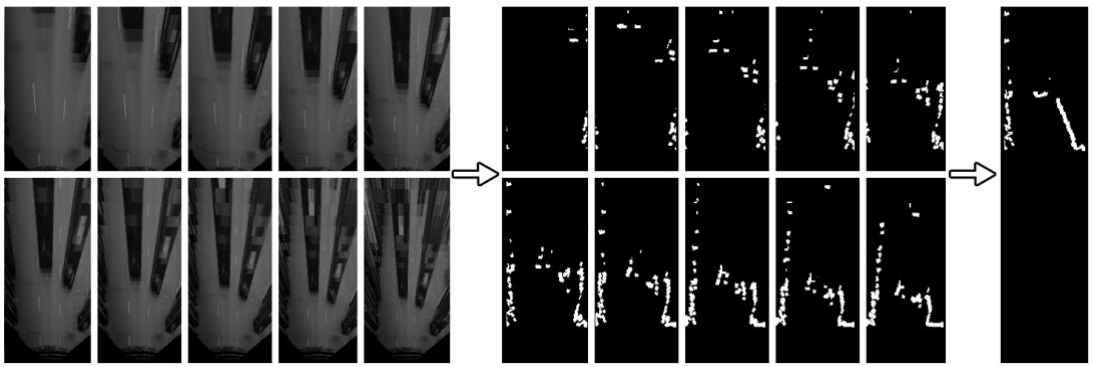


Figura 47. În stânga sunt cele 10 imagini IPM având unghiuri de pitch diferite, în centru sunt ilustrate aceleași imagini procesate care apoi sunt comparate cu harta curentă de ocupare a filtrului de particule (ilustrată în dreapta).

Totodată am generat și zece imagini IPM cu unghi de girație diferit la o variație tot de 0.1 grade și aplicat aceeași abordare. Un exemplu de zece imagini IPM și zece hărți de ocupare calculate din ele este ilustrat în Figura 47. Harta de ocupare din filtrul de particule este folosită pentru validare și ajustare a parametrilor deoarece va estima cât mai apropiat deplasarea vehiculelor în scenă și va fi mai puțin influențată de trepidații și mișcări ale camerei în timpul condusului.

2.13.6 Informații 3D din imaginea 2D

Pentru a implementa o percepție și mai eficientă, am ales să integrez și o rețea neuronală pentru a determina informații 3D ale obiectelor urmărite din imaginile 2D ale scenei. Astfel, ultimul pas al acestui sistem monocular de percepție constă în utilizarea unui CNN antrenat pentru a găsi rotația locală a unui obiect și dimensiunile aproximative ale obiectului exprimată direct în metri. Această idee a fost inițial prezentată în lucrarea [Chen2016c], unde autorii au antrenat pe baza de date KITTI [Geiger2012] și au avut rezultate promițătoare.

Estimarea dimensiunilor 3D și a orientării unui obiect este realizată prin extinderea unei rețele de detecție a obiectelor care oferă dreptunghiuri de încadrare în interiorul cărora se află obiectele detectate. Principala constrângere este dată de faptul că obiectul tridimensional din scenă se află în interiorul dreptunghiului 2D din imagine, astfel că proiecția 3D a acestuia se va afla în interiorul aceluiași dreptunghi (chenar) de încadrare obținut din detectorul de obiecte. Antrenarea unei rețele (SSD MobileNet) pentru detecția de obiecte este prezentată pe larg în secțiunea 3.12.1. Regresia unghiului de orientare este făcută folosind funcția de cost "smooth L1". O altă abordare pentru determinarea dimensiunii și orientării vehiculelor este lucrarea [Mousavian2017]. Procesul de regresie a dimensiunilor vehiculelor este facilitat de faptul că variația dimensiunilor nu este foarte mare (de exemplu autobuzele sunt aproximativ de aceeași dimensiune, camioanele de asemenea, etc.), iar acestea nu se schimbă pe măsură ce orientarea lor este diferită. Din acest motiv autorii au antrenat direct dimensiunile și nu vectorul de translație al obiectelor. În [Mousavian2017] orientarea obiectelor este determinată prin discretizarea unghiului de orientare și împărțirea lui în intervale ("bins"), fiecare predicție având și un grad de probabilitate p_i ca orientarea obiectului să se afle în intervalul *i*.

Am ales să utilizez o rețea similară, cu mici modificări ale funcției de cost prezentată în articol. Rețeaua neuronală convoluțională are primele 5 straturi la fel cu cele ale rețelei VGG16 [Simonyan2014]. Am folosit tehnica de învățare prin transfer și am inițializat ponderile primelor 5 straturi cu cele are rețelei VGG16 antrenată pentru clasificarea obiectelor din imagini. Ieșirea ultimei convoluții este apoi folosită pentru regresia dimensiunilor, orientării și a unei probabilități a orientării. Structura rețelei este următoarea:

- primul strat este format din două convoluții (C1) cu 64 filtre cu nucleu de 3 x 3 urmate de operația de agregare cu maxim ("max pooling")
- al doilea strat are două convoluții (C2) având 128 filtre de dimensiune 3 x 3 urmate de "max pooling"
- stratul al treilea conține trei convoluții (C3) cu 256 filtre de 3 x 3 și apoi operația de agregare "max pooling"
- al patrulea strat are trei convoluții (C4) cu 512 filtre având kernel de 3 x 3 și urmate de agregare "max pooling"
- ultimul strat de trei convoluții (C5) este tot cu 512 filtre cu nucleu de 3 x 3
- dimensiuni: strat complet conectat ("fully connected") FC1 de dimensiune 512 obținut din stratul C5, urmat de o activare ReLU și de operația de "dropout" (setată cu probabilitate de 0.5)
- dimensiuni: strat complet conectat (FC2) de dimensiune 3 (pentru informațiile despre dimensiunea vehiculelor: lățime, lungime, înălțime); funcția de cost MSE este folosită pentru calcularea erorii la învățare între dimensiunile din baza de date ("label") și cele din stratul FC2 ("predicție")
- orientare: strat complet conectat (FC3) de dimensiune 256 obținut din stratul C5, urmat de o activare ReLU și de o operație de "dropout" (cu probabilitatea 0.5)
- orientare: strat complet conectat (FC4) de dimensiune 2 * numărul de intervale *"BIN"* (în acest caz *BIN = 2*), urmat de o normalizare a orientării folosind L2; *BIN* este înmulțit cu 2 deoarece se va face predicția pentru valoarea funcției sinus a unghiului de orientare și pentru cosinus; rezultatul FC4 normalizat este folosit împreună cu funcția de cost L2 pentru regresia corecției unghiului de orientare
- probabilitate: strat complet conectat (FC5) de dimensiune 256 obținut din stratul C5, urmat de o activare ReLU și de operația de "dropout" (cu probabilitate de 0.5)

- probabilitate: strat complet conectat (FC6) de dimensiune *BIN*, având funcția de cost “cross entropy” pentru regresia probabilității unui unghi de face parte dintr-un interval sau altul

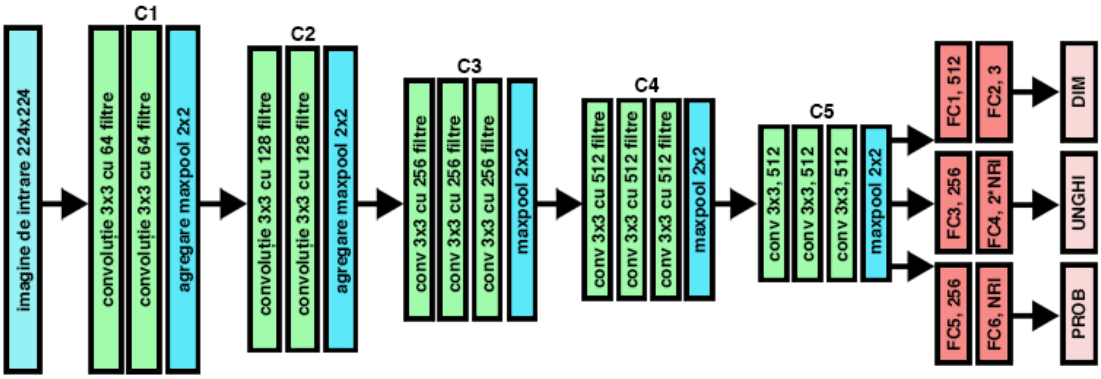


Figura 48. Arhitectura rețelei: primele 5 straturi din VGG16 urmate de mini-rețeaua de calculare a dimensiunilor, a unghiului de orientare și a probabilității unghiului de orientare.

Predicția unui unghi este mai robustă dacă este parametrizat prin sinus și cosinus, astfel se va face predicția a două valori care sunt în intervalul [-1, 1]. Totodată, după cum a fost prezentat inițial în lucrarea [Mousavian2017], unghiul de orientare este cuantificat (discretizat) și apoi împărțit în intervale (“bins”), iar rețeaua va face predicția asupra unei valori de corecție a rotației care trebuie aplicată orientării centrale dintr-un interval. Ieșirea stratului complet conectat “FC6” prezice probabilitatea de a fi într-un interval sau altul. Am folosit un total de 2 intervale (“bins”), iar funcția de cost pentru regresia orientării este L2, unde se calculează de fapt diferența între cosinus (valorile reale minus cele prezise) și diferența între sinus (valorile reale minus cele prezise). Funcția de cost a orientării va calcula în primul pas valoarea de mijloc a intervalului (“ancora”) din care face parte unghiul real (“ground truth” - GT) și apoi va calcula diferența între unghiul real și predicție (folosind “L2”), practic se va calcula corecția care trebuie aplicată valorii de sinus și cosinus unghiului prezis din interval pentru a obține valoarea de sinus și cosinus ale unghiul real (GT). Clasificarea pentru găsirea intervalului corect al orientării este realizată prin funcția de cost “cross entropy” și apoi prin aplicarea operației de “softmax” se obține distribuția de probabilitate. Se alege probabilitatea cea mai mare și apoi se recalculează unghiul de orientare. Funcția de cost pentru regresia dimensiunilor este “mean squared error” (MSE). Antrenarea rețelei am efectuat-o într-un total de 200 epoci iar dimensiunea lotului este 8 și rata de învățare 0.0001. Funcția de cost finală este compusă prin ponderarea celor 3 funcții de cost. În articolul [Mousavian2017] este menționată ponderarea funcțiilor de cost, dar nu sunt date ponderile exacte folosite, iar după mai multe teste și experimente am ales valorile prezentate în ecuația 62.

$$loss = 6 * loss_{orientare} + 3 * loss_{dimensiune} + 1 * loss_{probabilitate} \qquad (62)$$

Unghiul local de rotație (girație) al unui obiect în sistemul de coordonate al camerei este ilustrat în Figura 49. Antrenarea rețelei a fost făcută pe baza de date KITTI care oferă aceste orientări ale vehiculelor din scenă. Unghiurile de orientare din KITTI sunt în intervalul [-180°, 180°], de aceea primul pas înaintea antrenării constă în ajustarea unghiurilor prin mutarea lor în intervalul [0°, 360°]. În Figura 49 în stânga se pot observa exemple de unghiuri de orientare locală a vehiculelor în sistemul de coordonate al camerei: un vehicul văzut din spate va avea unghiul de orientare -90°, iar daca vine spre cameră și este privit din față va avea unghiul +90°. În lucrarea [Mousavian2017] autorii au implementat și calcularea unghiului dintre raza proiectată din cameră până în centrul obiectului în raport cu axa longitudinală proprie a obiectului (*θcameră* - cunoscând parametri intrinseci ai camerei acest unghi se poate determina ușor), dar în cadrul sistemului de percepție monocular prezentat în această teză acest unghi nu este așa de relevant, deoarece urmărirea obstacolelor se face într-o hartă 2D cu efectul de perspectivă eliminat care este aliniată la sistemul de coordonate al camerei.

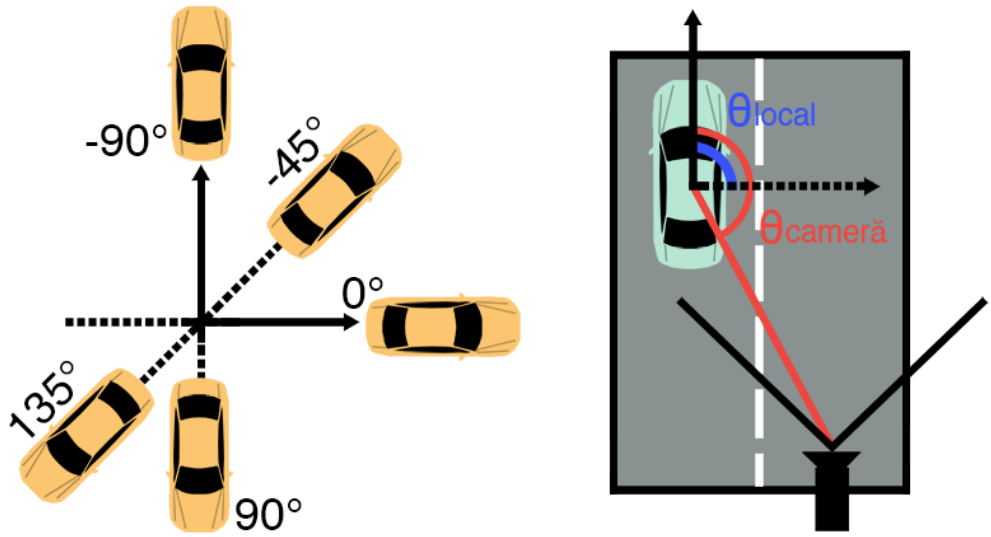


Figura 49. Orientarea locală a unui vehicul (θ local) în sistemul de coordonate al camerei și orientarea în raport cu raza ce pornește din centrul camerei până în centrul obstacolului (θ cameră).

Rezultatul antrenării rețelei după 200 de epoci este ilustrat în Figura 50.

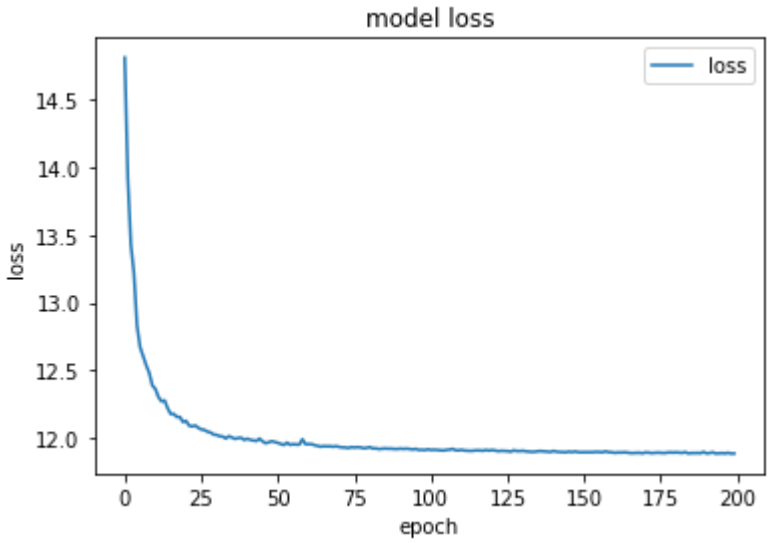


Figura 50. Funcția de cost scade și este stabilă.

Predicția pe un singur chenar de încadrare durează în medie 0.0067 ms. O comparație între 100 de unghiuri de orientare prezise de rețea și cele din setul de validare este ilustrată în Figura 51.

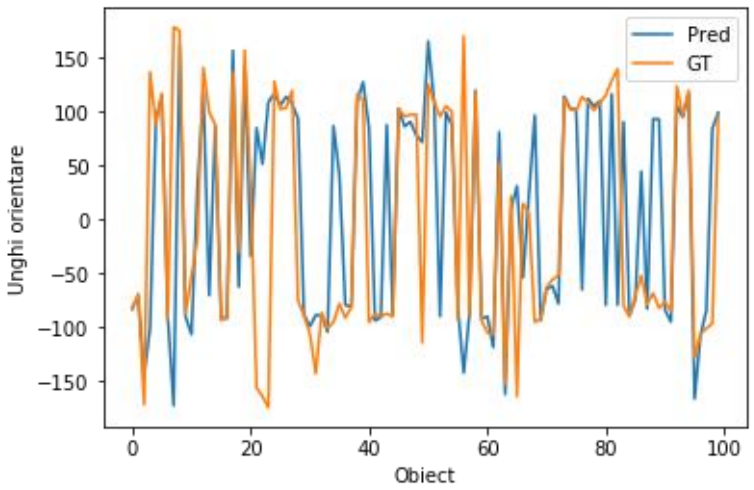


Figura 51. Comparație între orientările prezise de rețea (“pred”) și cele din baza de date de validare (“GT”). Unghiurile sunt exprimate în grade.

În Figura 51 se poate observa faptul că la unele predicții diferența este foarte mare (>160°), ceea ce înseamnă o confuzie a rețelei asupra orientării unui obiect: dacă este privit din spate ar putea orientarea -90°, iar același obiect văzut din față va avea orientarea +90°. Ținând cont de acest aspect, dacă am schimbat semnul unghiurilor unde diferența între predicție și valoarea reală (“ground truth” - GT) este mai mare de 150°, am obținut o medie a distanței angulare de 23.17°, față de o medie de 39.89° dacă nu am aplicat nici o ajustare a unghiurilor. Pentru calcularea mediei am folosit distanța angulară dintre 2 unghiuri care este calculată în modul următor:

$$dist = cos^{-1}(\cos(\alpha_1 - \alpha_2)) \quad (63)$$

În ecuația 63 unghiul α_1 este predicția și α_2 este valoarea reală (GT), iar în Tabel 8 este afișată eroarea MAE și RMSE pe setul de validare pentru predicția orientării.

Tabel 8. Evaluare predicție unghi orientare pe setul de validare.

MAE	RMSE
11.36°	21.99°

Aceste valori au fost obținute pe setul de validare din baza de date care conține un total de 1494 imagini și perechi de fișiere text care conțin chenare de încadrare pentru obiecte și informații despre orientare și dimensiuni, în timp ce setul de antrenare este format din 5987 imagini și fișiere text cu informații adiționale (același format ca și cel de validare).

Datorită utilizării a unei camere diferite, cu parametri și distanța focală diferită, rezultatul predicției din această rețea neuronală pentru dimensiunea obiectelor este în general cu un grad scăzut al preciziei. În schimb, rețeaua este foarte utilă pentru determinarea orientării obiectelor și produce rezultate apropiate de cele reale. Figura 52 prezintă rezultatele detecției de vehicule folosind rețeaua SSD MobileNet, iar în Figura 53 este afișată predicția orientării și a dimensiunii pentru fiecare vehicul.

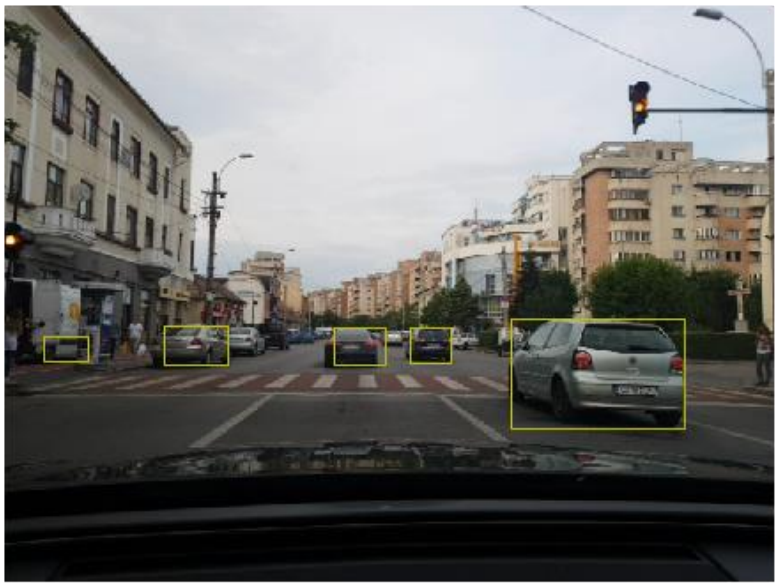


Figura 52. Imaginea de intrare având obstacole detectate și marcate prin chenare de încadrare obținute din rețeaua SSD MobileNet.

Figura 53. Rezultatul predicției rețelei de regresie a orientării și dimensiunii vehiculelor.

Dimensiunea și orientarea locală a unui vehicul este integrată în filtrul de particule, mai specific în algoritmul de grupare al particulelor care extrage cuboide din particulele din celule asemănătoare. Corespondența dintre vehiculele detectate de CNN și cele urmărite de filtrul de particule se face prin calcularea coeficientului Sorensen Dice (IoU). Am calculat IoU între latura din față sau cea din spate a cuboidului și chenarele de încadrare din rețeaua neuronală convoluțională SSD MobileNet. Situațiile cu un scor IoU mai mare de 0.5 sunt considerate valide și se modifică dimensiunile cuboidului și orientarea cu cele din rețeaua artificială din acest capitol. Astfel se obține fuziunea datelor dintre rețele convoluționale neuronale și urmărirea prin filtrul de particule.

Rezultatul fuziunii datelor este ilustrat în Figura 54 unde este afișată o vedere de sus a scenei în care este afișat rezultatul extracției de cuboide direct din filtrul de particule în paralel cu varianta ajustată și îmbunătățită.

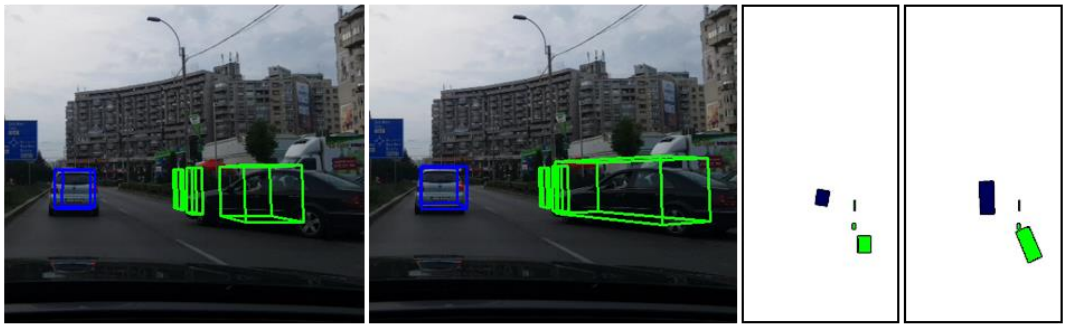


Figura 54. Fuziunea datelor folosind rețeaua CNN pentru orientarea obiectelor și a dimensiunilor. Prima imagine reprezintă cuboidele extrase din filtrul de particule, în centru sunt ilustrate cuboidele cu orientarea și lungimea lor ajustată din CNN. Ultima imagine reprezintă o vedere de sus a scenei cu aceleași obiecte (normale din filtrul de particule vs. ajustate din CNN).

Această secțiune a prezentat o abordare de găsire a unghiului de orientare local al unui vehicul dintr-o imagine 2D, abordare care nu este originală ci este o versiune similară a celei prezentate în [Mousavian2017]. Am încercat de asemenea și regresia directă a unghiului de orientare folosind funcția de cost L1, ca în lucrarea [Chen2016c], dar rezultatele au fost nesatisfăcătoare. Modul de folosire și integrare cu filtrul de particule este propriu și original și reprezintă contribuția principală din această secțiune. Scopul sistemului de percepție este de a utiliza atât viziune artificială prin procesare de imagini, cât și folosirea rețelelor convoluționale neuronale și integrarea și fuziunea acestor module de procesare.

2.13.7 Arhitectura sistemului

Arhitectura sistemului de percepție monocular este ilustrată în Figura 55. Sistemul constă într-o componentă de achiziție capabilă să preia date din traficul rutier și o componentă de procesare, care va prelucra datele. Achiziția este implementată momentan folosind dispozitive mobile, dar se poate extinde în așa fel încât imaginile noi să fie capturate și de alte camere foto împreună cu senzorii externi. Componenta de procesare este separată, implementată pe un sistem desktop sau laptop. Totuși, o parte din percepție se poate efectua direct pe dispozitivul mobil, după cum am prezentat în lucrarea [Danescu2016a] (unde se face o urmărire bazată pe filtre de particule cu o hartă de ocupare generată din segmentarea cu un prag fix a imaginii). Comunicarea între sistemul de achiziție (dispozitiv mobil) și modulul de procesare se poate face fie printr-o conexiune fizică cu un cablu de tip USB sau chiar prin WiFi Direct (această funcționalitate nefiind încă implementată și nu este disponibilă pentru toate dispozitivele mobile).

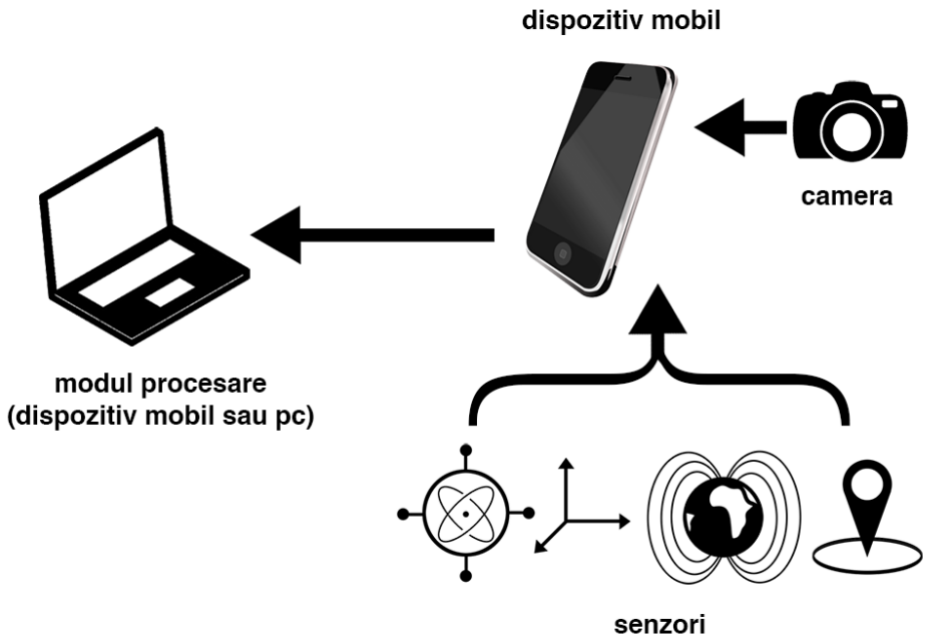


Figura 55. Sistemul de percepție monocular a traficului rutier.

Sistemul conține două componente de procesare principale: un modul scris în limbajul C++ și un modul scris în Python. Componenta scrisă în Python este pentru a folosi rețelele neuronale antrenate, în timp ce modulul C++ este responsabil pentru partea de procesare a datelor: urmărirea obstacolelor folosind filtrul de particule (generarea și predicția hărții de ocupare dinamică și actualizarea măsurătorii folosind datele din rețeaua neuronală), afișarea rezultatelor intermediare de procesare, dar și afișarea procesării și a rezultatelor finale.

Comunicarea între cele două module este implementată printr-un server local folosind socket-uri pentru a transmite datele. Serverul este realizat în Python și are rolul de a procesa cererile primite de la client, modulul scris în C++. Imaginile sunt procesate offline, pe un sistem desktop și sunt stocate pe disc. Modulul principal, cel de C++ va citi pe rând fișierele dintr-o anumită secvență. Indexul imaginii curente procesate va fi trimis prin socket-uri către modulul Python. Astfel, modulul C++ (clientul) va trimite cereri către modulul Python (server) pentru a segmenta semantic o anumită imagine folosind rețeaua convoluțională de segmentare, dar și pentru a detecta obstacole folosind o altă rețea neuronală. De asemenea, predicțiile de obstacole din a doua rețea sunt folosite ca și intrare pentru o a treia rețea neuronală care va oferi predicții despre dimensiunile obstacolelor (exprimată în metri) și orientarea locală a lor (exprimată în radiani). Rezultatele sunt transmise înapoi spre modulul C++ după cum urmează: rezultatul segmentării, adică harta nouă de segmentare este stocată pe disc iar modulul C++ așteaptă până când serverul scrie noul fișier și apoi îl va citi pentru a-l folosi în crearea hărții dinamice de ocupare; rezultatele din rețeaua de predicție a obstacolelor din scenă este transmisă prin socket-uri înapoi în C++ și stocată ca o listă de tip float conținând coordonatele dreptunghiurilor care încadrează obstacolele; informațiile din a treia rețea, despre orientarea și dimensiunea fiecărui obstacol este transmisă tot prin intermediul socket-urilor sub forma unei liste de tip float. Procesul de comunicare între module este ilustrat în Figura 56. Din testele realizate, transmiterea datelor folosind un hard disk nu introduce întârzieri semnificative în fluxul de procesare, mai ales dacă sunt folosite unități de tip SSD

pentru stocarea datelor. De altfel, am testat și soluțiile existente de încărcare de CNN și predicție direct în C++ cu software-ul OpenCV, dar nu au fost îmbunătățiri la timpii de procesare. Predicția și scrierea pe SSD prin server-ul Python și apoi citirea imaginii segmentate în C++ durează aproximativ 40 ms per imagine, în timp ce predicția direct în C++ durează undeva la 140 ms, deoarece soluțiile existente nu sunt foarte bine optimizate să execute predicția folosind plăcile grafice. Pe viitor aceste probleme vor fi rezolvate, iar sistemul va fi adaptat și actualizat.

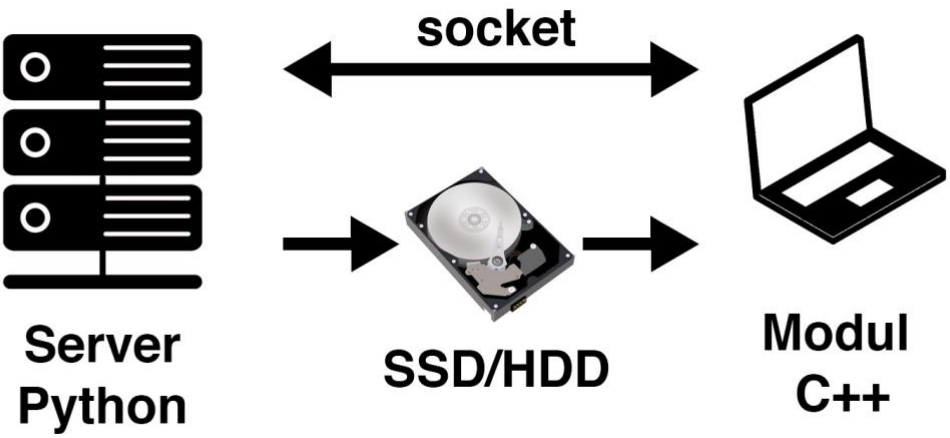


Figura 56. Procesul de transmitere a datelor între client (modul C++) și server (modul Python). Rezultatul segmentării semantice este salvat pe disc (SSD sau HDD) și apoi citit în modulul C++.

Cererile din server sunt procesate secvențial, pe rând pentru a nu "sări" peste cadre. Rezultatele parțiale, de exemplu harta de măsurare a scenei, sunt afișate în modulul C++ folosind software-ul OpenCV.

În Python este creat un server de tip TCP local pe un port definit, în acest caz am ales: "localhost:9998". Este necesară definirea unui "handler" unde sunt definite funcțiile și metodele serverului. Astfel, am folosit o clasă unde sunt definite diferite metode pentru a încărca cele trei rețele neuronale și ponderile lor, metoda de procesare a unei cereri, dar și o metodă pentru a șterge modelele încărcate și a opri serverul. Concret, metodele sunt următoarele:

- *handle()*: metoda principală de unde se verifică dacă CNN-urile au fost încărcate sau nu și de unde se creează un thread nou pentru a procesa cererea
- *processRequest()*: metoda de procesare a unei cereri, unde sunt citiți biții care conțin datele despre dimensiunea imaginii și locația de pe disk, iar apoi sunt apelate cele trei metode de inferență pentru fiecare CNN individual: doInference() pentru a genera predicții despre obstacole, *doOrientationInference()* pentru predicțiile asupra orientarea și dimensiunea obstacolelor detectate cu prima rețea și *doSegmentationInference()* care va genera segmentarea semantică și va salva pe disc imaginea nouă; după inferență folosind cele trei CNN-uri este construit un șir ("string") conținând un rezultatele structurate ca un document de tip JSON

- *loadModel()*: metoda de încărcare a rețelelor neuronale convoluționale, apelată din *handle()*
- *clearModel()*: metoda de închide sesiunile existente și de a reseta modelele CNN încărcate în memorie

Imaginile rezultate din segmentarea semantică sunt salvate pe disc folosind același nume cu cel al imaginii sursă, dar cu sufixul "_mask" adăugat la final. Citirea acestei imagini în modulul C++ este realizată doar după ce server-ul a trimis un semnal de confirmare ("acknowlegde"). O verificare suplimentară se face și prin verificarea existenței pe disc a fișierului nou cu sufixul definit.

Modulul C++ comunică prin socket-uri TCP cu serverul, iar pentru a implementa această conexiune am folosit software-ul QT [Qt], mai specific pachetul "*QTCPSocket*". Se creează un thread nou prin care se stabilește conexiunea la adresa "localhost" pe portul setat anterior (9998). Prin execuția normală a modulului C++ pe desktop, sunt parcurse și procesate pe rând imaginile dintr-o secvență din traficul rutier. Așadar, locația pe disc a fiecărei imagini, dar și dimensiunea acesteia sunt transmite prin socket spre server pentru a fi procesată. După transmiterea acestor informații, modulul C++ așteaptă până primește date pe socket. Citirea este realizată folosind o structură de tip JSON pentru a facilita mai ușor organizarea informațiilor despre detecții: serializarea, respectiv deserializarea datelor. Pe socket se transmite sub formă de "string" un șir de obiecte JSON ("JSONArray"), unde fiecare obiect ("JSONObject") reprezintă de fapt coordonatele în imagine a chenarului de încadrare a unui obstacol (coordonatele în imagine ale punctului din stânga sus și ale punctului din dreapta jos). În dată ce fișierul JSON este citit de pe socket, execuția normală a programului este reluată și datele pot fi procesate în C++.

După terminarea inferenței folosind serverul local, modulul C++ va genera procesarea acestor rezultate și va crea harta de ocupare dinamică folosind filtrul de particule. Tot aici sunt citite de pe disc și informațiile din senzori capturate în timpul procesului de achiziție a datelor. Se citesc astfel datele din accelerometru, giroscop, GPS și viteza. Acestea sunt folosite în harta de ocupare dinamică pentru a crea particule dinamice cu viteză ajustată în funcție de viteza de deplasare a vehiculului, dar și pentru a crea imaginea cu efectul de perspectivă eliminat (unde sunt folosite informațiile despre distanța focală și rata de rotație).

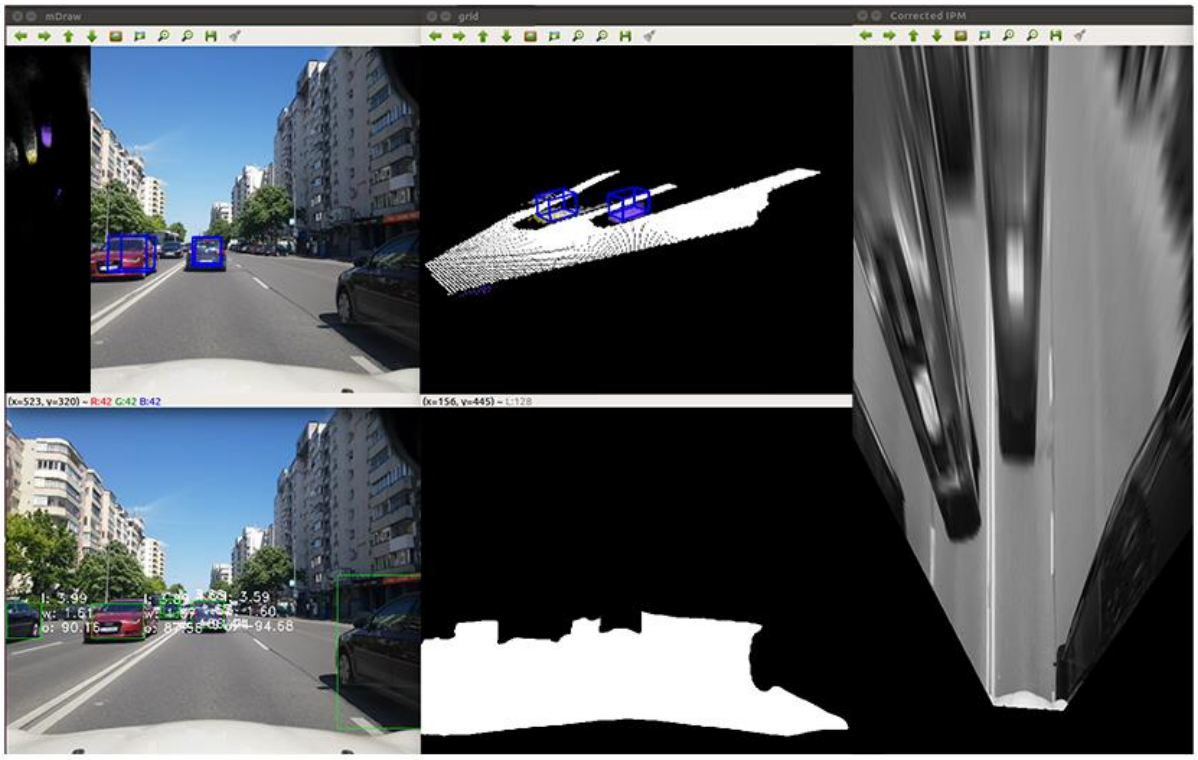


Figura 57. Modulul de procesare scris în C++ pe desktop unde se poate observa: imaginea color pe care sunt desenate cuboidele extrase din particule (ilustrate în zona din stânga), imaginea scenei cu cuboide și zona de drum “driveable” reproiectată (centru sus), rezultatul detecției de vehicule din CNN (stânga jos), imaginea segmentată de CNN (centru jos), iar în dreapta este imaginea IPM.

Întreg modulul de procesare a fost dezvoltat pentru a putea testa și implementa diferite abordări într-un mediu offline pe un sistem desktop, dar folosind un ”webcam” sau orice cameră video atașată la un laptop acest sistem poate fi portabil și utilizat într-un vehicul de dezvoltare. Algoritmii testați se pot implementa ulterior și pe platforme “embedded” sau chiar pe dispozitive mobile inteligente care să fie integrate direct într-un vehicul, acesta fiind obiectivul principal pentru dezvoltarea și continuarea acestui proiect de cercetare.

2.14 Concluzii

În cadrul acestui capitol am descris rețele neuronale artificiale și utilizarea lor în domeniul viziunii artificiale prin implementarea unor rețele neuronale convoluționale. Am descris principalele arhitecturi de rețele artificiale, am scris o comparație între clasificare și regresie și apoi o descriere a funcțiilor de activare. Totodată, am prezentat diferite tehnici de inițializare și actualizare a ponderilor unei rețele neuronale, precum și funcțiile de cost potrivite pentru clasificare, regresie sau segmentare semantică. Acest capitol descrie și segmentarea semantică folosind rețele neuronale și prezintă o comparație a celor mai bune arhitecturi la momentul scrierii tezei. Deoarece principalul obiectiv este de a crea un sistem de percepție monocular, am descris de asemenea și alte implementări existente folosind o singură cameră, în special cele existente deja pe dispozitive mobile. Tot în acest capitol este o descriere a

tehnicilor de a estima adâncimea (distanțele) în imagini monoculare. Această sarcină se poate face în două modalități: folosind rețele neuronale convoluționale sau prin metode clasice, bazate pe geometrie direct în imaginea cu efectul de perspectivă eliminat. Cea mai importantă realizare este descrisă în secțiunea 2.13 în care am descris sistemul de percepție. Prima dată am prezentat bazele de date existente care sunt folosite în antrenarea unor rețele artificiale și pentru testarea algoritmilor, iar apoi am descris sistemul de achiziție propriu. Având datele din trafic, următorul pas în implementarea percepției a fost dezvoltarea unei rețele convoluționale pentru segmentarea semantică, mai specific pentru a detecta zona de drum dintr-o imagine. Am prezentat arhitectura U-NET modificată pentru segmentare și apoi rezultate pe datele achiziționate de sistemul propriu. Din imaginea de drum segmentată am creat o hartă de ocupare care este folosită pentru generarea de particule, iar acestea sunt urmărite folosind filtrul de particule. Am descris modul de creare a particulelor, ponderarea și reeșantionarea lor. Am prezentat anumite extensii și îmbunătățiri prin procesarea imaginilor și apoi prin rafinarea parametrilor camerei, iar în final am prezentat o implementare a unei rețele neuronale convoluționale pentru a determina dimensiunea și orientarea obiectelor detectate din scenă. Modul în care este folosită împreună cu algoritmul de urmărire (”tracking”) este unic și reprezintă o contribuție originală. În final este descrisă întreaga arhitectură a sistemului de percepție care are la bază module de procesare scrise în limbajul C++ (pentru partea de procesare a imaginilor și de urmărire) și în limbajul Python (pentru predicția folosind CNN).

2.14.1 Contribuții

Principalele contribuții originale descrise în acest capitol sunt: crearea unui sistem propriu de achiziție de date folosind senzorii disponibili pe un dispozitiv mobil inteligent (camera foto, senzori de poziție prin satelit, accelerometru, giroscop, etc.) și apoi crearea unor baze de date proprii și implementarea unei modalități de a vizualiza aceste date pe sisteme desktop.

Am realizat un sistem de percepție bazat pe o singură cameră, combinând rețele neuronale convoluționale pentru a identifica obiecte din scenă și apoi am folosit un filtru de particule pentru a le urmări în timp. Primul pas constă în implementarea unei rețele artificiale pentru segmentarea semantică, având la bază arhitectura rețelei U-NET.

Am realizat o adaptare originală a filtrului de particule bazat pe o hartă de ocupare dinamică din [Danescu2011]. Abordarea descrisă în cadrul acestei teze folosește o hartă de ocupare extinsă, capabilă să mențină ipoteze despre posibile obstacole și în zona din spatele vehiculului propriu prin dublarea dimensiunii hărții de ocupare folosite. De asemenea, filtrul de particule folosit este adaptat pentru a folosi imagini monoculare.

Totodată, am rafinat procesul de creare a hărții binare de măsurare ce are la bază detecția de obstacole. Am analizat histogramele radiale corespunzătoare fiecărui vehicul din imaginea IPM segmentată și am eliminat zona liberă dintre roțile unui vehicul și asfalt, ceea ce are ca efect o îmbunătățire semnificativă în procesul de creare a particulelor. De asemenea, unghiurile de înclinare și de girație ale sistemului de percepție sunt ajustate în fiecare cadru prin generarea a zece imagini IPM, segmentarea lor folosind un prag adaptiv (secțiunea 2.12.3) și generarea a zece hărți de ocupare candidat, care sunt comparate cu harta de ocupare curentă din filtrul de particule (generată din imaginea segmentată de rețeaua convoluțională). De altfel, am implementat o versiune modificată a unei rețele existente pentru

a oferi predicții asupra dimensiunii obiectelor detectate în scenă, dar și asupra orientării lor. Aceste informații sunt fuzionate cu cele din filtrul de particule pentru a îmbunătăți robustețea și precizia sistemului și reprezintă o contribuție originală și importantă pentru sistemul de percepție monocular.

2.14.2 Publicații

Din tehnicile prezentate în acest capitol au rezultat următoarele publicații:

1. Razvan Itu, Radu Danescu, "An Efficient Obstacle Awareness Application for Android Mobile Devices", International Conference on Intelligent Computer Communication and Processing, pp. 157-163, 2014.

2. Radu Danescu, Razvan Itu, Andra Petrovai, "Sensing the Driving Environment with Smart Mobile Devices", IEEE International Conference on Intelligent Computer Communication and Processing, pp. 271-278, 2015.

3. Radu Danescu, Razvan Itu, Andra Petrovai, "Generic Dynamic Environment Perception Using Smart Mobile Devices", Sensors, Vol. 16, No. 10, Art. No. 1721, 2016.

4. Radu Danescu, Andra Petrovai, Razvan Itu, Sergiu Nedevschi, "Generic Obstacle Detection for Mobile Devices Using a Dynamic Intermediate Representation", Advances in Intelligent Systems and Computing, vol. 427, pp. 629-639, 2016.

3. Calibrare cameră și modelul camerei

3.1 Modelul camerei și fundamentare teoretică

Înainte de procesul de calibrare al camerei, este importantă o definire a modelului camerei folosit. Camerele foto sau video digitale, cât și ce clasice au la bază modelul de *“camera obscura”*, unde lumina pătrunde printr-o deschidere (apertură) mică într-o cameră sau cutie unde creează o imagine răsturnată a scenei. Folosirea unor lentile a dus la formarea unor imagini de calitate superioară și mai clare. Prin modul în care se formează imaginea 2D a scenei 3D sunt pierdute informații geometrice relevante: de exemplu distanța până la obiectele din scenă, liniile paralele din scenă se vor intersecta în imagini și ar putea apărea și anumite distorsiuni cauzate de lentilele folosite din cameră.

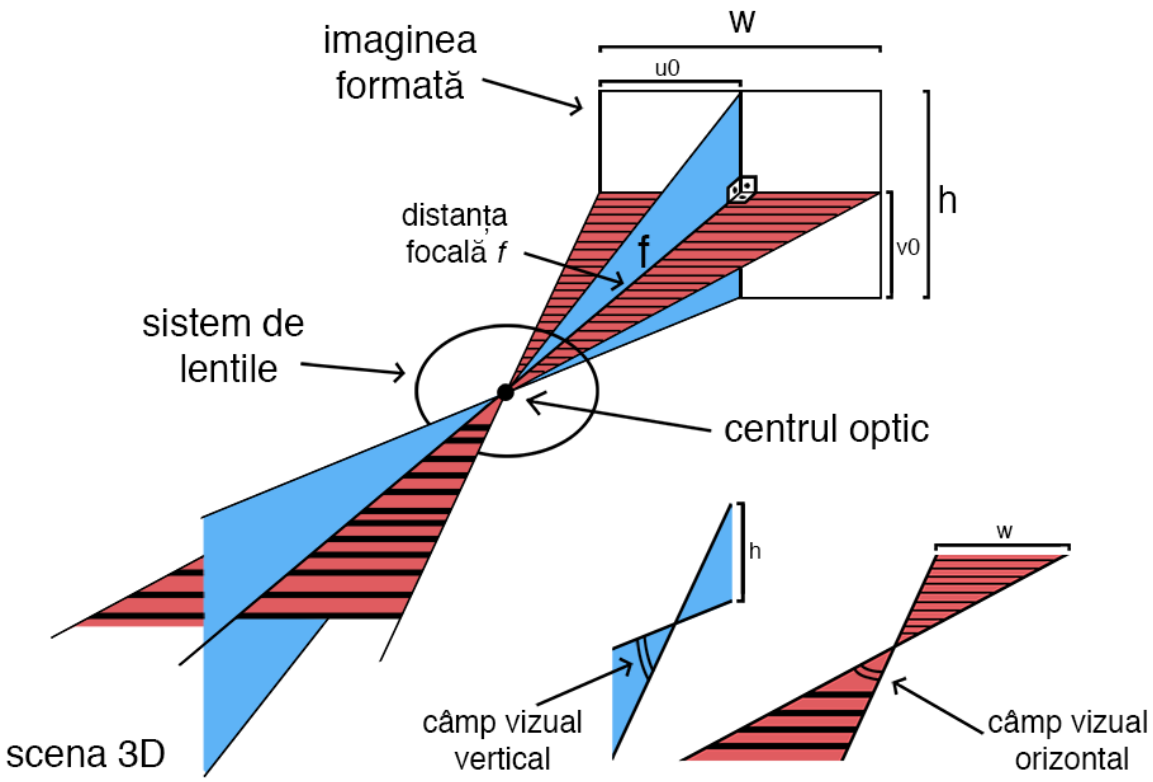


Figura 58. Modul de formare a imaginii.

În Figura 58 este prezentat modul de formare al imaginii folosind un model de lentile subțiri (“thin lens model”) bazat pe modelul *“camera obscura”*. Este de asemenea ilustrată și distanța focală, câmpul vizual orizontal și cel vertical, dar și centrul optic al camerei. Punctul principal (u0, v0) este reprezentat în imagine de raza care trece prin centrul optic și este perpendiculară pe planul imaginii.

Modelul perspectivă (“pin-hole model”) reprezintă o abstractizare a modelului cu lentile subțiri în care lentila/apertura este aproximată cu un punct (centrul de proiecție) și unde fiecare

punct din scenă este focalizat. Din camera “trece” o câte rază spre fiecare punct din scenă, iar imaginea din fața camerei la distanță focală *f* reprezintă planul imaginii obținut în modelul “pin-hole”. Totodată, imaginea este reprezentată normal și nu răsturnată ca în cazul modelului “camera obscură”. Figura 59 ilustrează modelul perspectivă al camerei în raport cu sistemul de coordonate al lumii, în contextul unei imagini obținute în traficul rutier (este ilustrată și o suprafață de drum cu linii de marcaje).

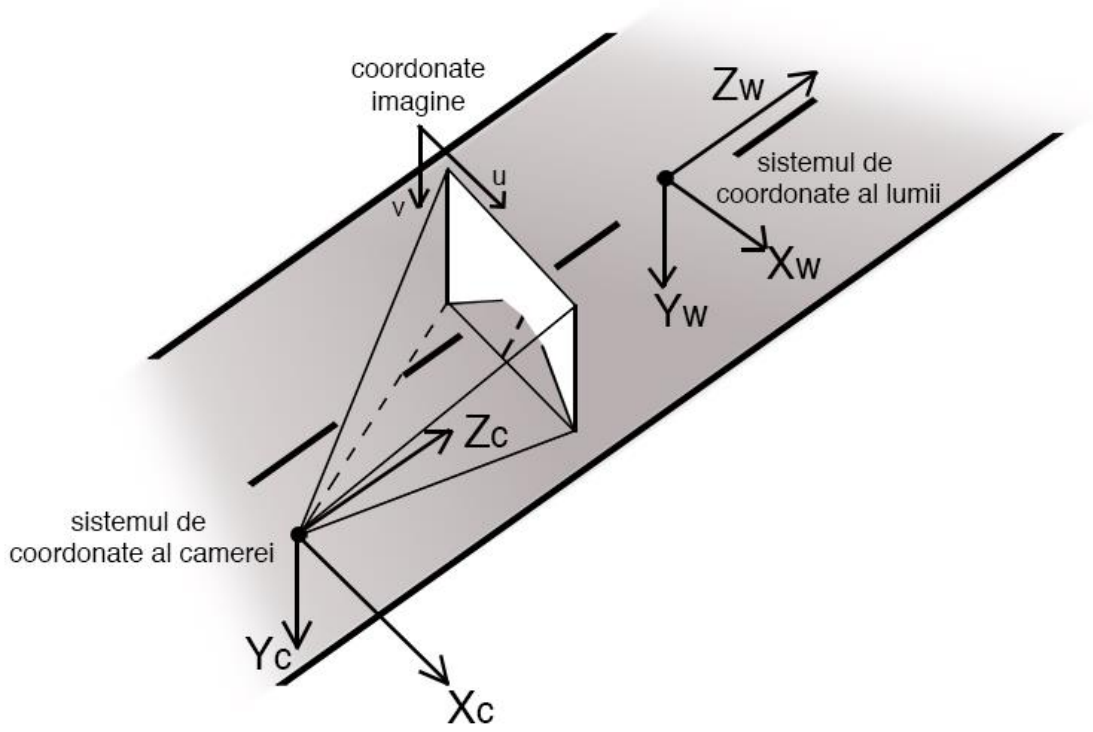


Figura 59. Modelul perspectivă cu sistemul de coordonate al camerei și al lumii, dar și planul imaginii.

Modelul perspectiva este cel folosit în domeniul viziunii artificiale. Având un model geometric definit, proiecția punctelor din scenă 3D în imagine este dată de matricea de proiecție *“P”* (ecuația 64) compusă din matricea intrinsecă a camerei *“A”* și matricea extrinsecă, formată din matricea de rotație *(R)* și vectorul de translație *(T)*.

$$P = A[R \mid T] \tag{64}$$

Matricea *“P”* va oferi corespondența între un punct 3D din sistemul de coordonate al lumii și un punct 2D în planul imaginii. În continuare voi prezenta modul de calculare și determinare a parametrilor intrinseci și extrinseci.

3.2 Parametri intrinseci

Matricea intrinsecă este compusă din parametri camerei: distanță focală (f_x și f_y) și punctele principale (c_x și c_y) pe ambele axe. Coeficienții de distorsiune radială și tangențială ai obiectivului reprezintă parametri non-liniari ai camerei care nu fac parte din matricea intrinsecă și sunt de multe ori aproximați prin procesele de calibrare statică într-un mediu controlat (de obicei laborator). Ecuația 65 reprezintă matricea intrinsecă a camerei.

$$A = \begin{pmatrix} f & 0 & u_0 \\ 0 & f & v_0 \\ 0 & 0 & 1 \end{pmatrix} \quad (65)$$

Calcularea distanței focale și a punctului principal reprezintă primii pași necesari calibrării, deoarece acești parametri formează matricea camerei. Formula de calcul este prezentată în ecuația 66 (unde θ este câmpul vizual orizontal).

$$f = \frac{w}{2tan(\frac{\theta}{2})} \quad (66)$$

În mod analog, se poate calcula distanța focală și cunoscând înălțimea imaginii și câmpul vizual vertical. De altfel, distanța focală exprimată în pixeli se poate deduce și cunoscând dimensiunea fizică a senzorului de imagine în milimetri, distanța focală în milimetri și lățimea imaginii exprimată în pixeli. Formula de calcul este prezentată în ecuația 61.

$$F = \left(\frac{focal_{mm}}{senzor\_width_{mm}}\right) * image\_width_{px} \quad (67)$$

Cunoscând câmpul vizual vertical se poate determina câmpul vizual orizontal, iar pentru a lua în considerare raportul dimensiunilor (“aspect ratio”) formula de calcul este:

$$\theta h = 2\text{atan}\left(aspect * \tan\left(\frac{\theta v}{2}\right)\right) \quad (68)$$

În ecuația 68 “θh” este câmpul vizual orizontal, “θv” este câmpul vizual vertical, iar “aspect” este raportul de aspect al imaginii. Ecuația 69 reprezintă formula de calcul a câmpului vizual vertical în care se ține cont și de nivelul de zoom al imaginii.

$$\theta h = 2\text{atan}\left(\frac{100 * \tan\left(\frac{\theta h}{2}\right)}{zoom}\right) \quad (69)$$

Folosind ecuația 66 se poate recalcula distanța focală ajustată în funcție de nivelul de zoom și raportul de aspect prin înlocuirea θ cu θh (calculat din ecuațiile 68-69).

Restul de parametri intrinseci ai camerei, adică punctul principal pe axa orizontală c_x și punctul principal pe axa verticală c_y sunt de multe ori aproximați ca fiind în centrul imaginii (lățime / 2 și înălțime / 2).

3.3 Parametrii extrinseci

Așadar, având parametrii intrinseci următorul pas constă în determinarea parametrilor extrinseci ai camerei. Matricea extrinsecă exprimă de fapt poziția și orientarea camerei în raport cu coordonatele lumii. Vectorul de translație conține distanță exprimată în metri (sau cm) a camerei față de originea sistemului de coordonate al lumii pe toate cele 3 axe. În sisteme de asistență a conducătorilor auto, unde camera este montată în parbrizul vehiculului, un astfel de vector ar putea fi: [W H L], unde *"W"* este deplasamentul pe axa X, *"H"* reprezintă înălțimea față de sol la care este montată camera (axa Y), iar *"L"* reprezintă deplasamentul pe axa Z (distanța de la cameră până în bară fata a vehiculului unde astfel de sisteme sunt centrate de obicei). Vectorul final de translație este prezentat în ecuația 70.

$$T = (W\ H\ L)^T \tag{70}$$

Sistemul de coordonate al camerei (Figura 60) este bazat pe regula mâinii stângi ("left hand rule") unde originea este amplasată în botul vehiculului și aliniată cu planul drumului.

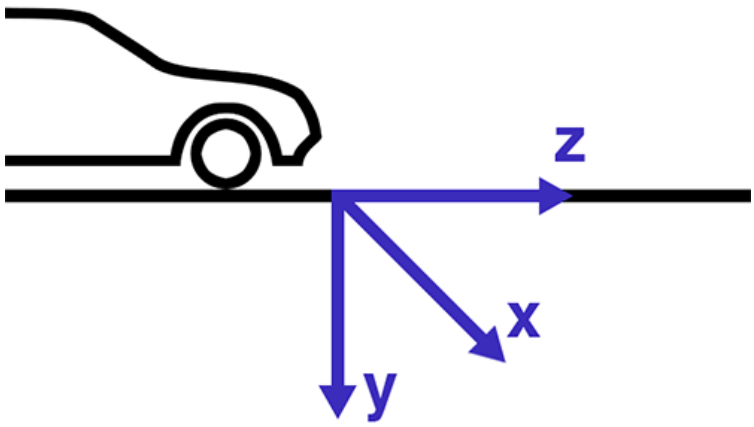


Figura 60. Sistemul de coordonate al camerei aliniat pe suprafața drumului și centrat în fața vehiculului.

Matricea de rotație este calculată prin înmulțirea celor trei matrici din ecuațiile 71-73 care descriu rotația camerei de-a lungul celor trei axe ale sistemului de coordonate 3D.

$$Rx(\theta) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos\ \theta & -sin\ \theta \\ 0 & sin\ \theta & cos\ \theta \end{pmatrix} \tag{71}$$

$$Ry(\psi) = \begin{pmatrix} cos\ \psi & 0 & sin\ \psi \\ 0 & 1 & 0 \\ -sin\ \psi & 0 & cos\ \psi \end{pmatrix} \tag{72}$$

$$Rz(\gamma) = \begin{pmatrix} cos\,\gamma & -sin\,\gamma & 0 \\ sin\,\gamma & cos\,\gamma & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (73)$$

$$R = Rx * Ry * Rz \quad (74)$$

Matricea de rotație finală (ecuația 74) reprezintă orientarea camerei în raport cu lumea (sistemul de coordonate al lumii). Cele trei unghiuri din fiecare matrice R_x, R_y și R_z sunt de fapt unghiurile de rotație în jurul fiecărei axe. Ele reprezintă unghiurile folosite și în aviație: *"pitch"* - unghiul de înclinare (ecuația 71) ce reprezintă rotația în jurul unei axei transversale X (perpendiculare), *"yaw"* - unghiul de girație (ecuația 72) care este de fapt rotația în jurul axei verticale Y și *"roll"* - unghiul de rotire (ecuația 73) care este determinat de rotația în jurul axei longitudinale Z (orizontale). În general în sistemele de conducere autonomă camerele sunt montate având un unghi de *"yaw"* și *"roll"* cât mai apropiate de 0, prin alinierea camerei cu axa longitudinală și transversală a vehiculului.

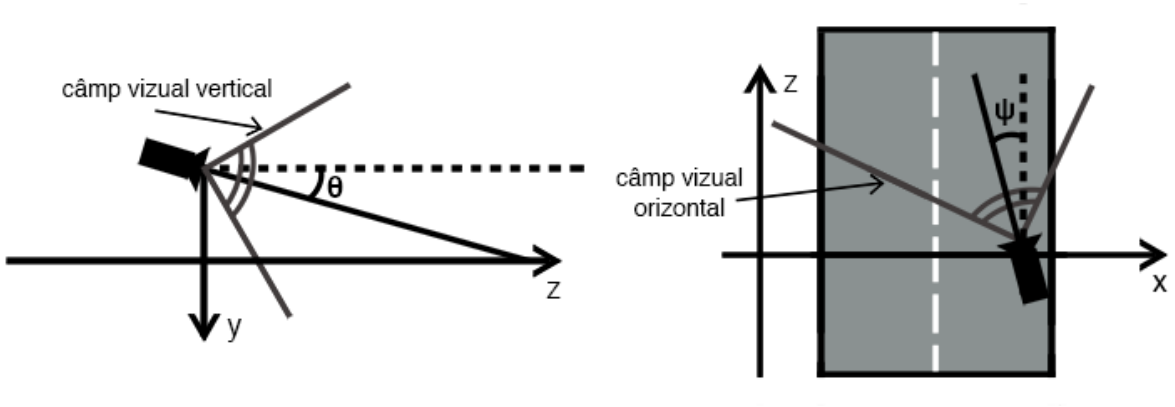


Figura 61. Ilustrare unghi de înclinare θ (stânga) și unghiul de girație ψ (dreapta).

Cunoscând câmpul vizual vertical și cel orizontal, dar și coordonatele punctului de fugă din imagine se poate calcula unghiul de înclinare cât și cel de girație. Formulele de calcul sunt prezentate în ecuațiile 75-76 și preluate din lucrarea [Nieto2007], unde *w* și *h* reprezintă lățimea și respectiv înălțimea imaginii, *VPx* și *VPy* sunt coordonatele în pixeli din imagine ale punctului de fugă, iar *αV* și *αH* reprezintă cele două deschideri angulare (câmpuri vizuale) ale camerei.

$$\theta = \arctan\left[tan\alpha_v\left(1 - \frac{2VP_y}{W}\right)\right] \quad (75)$$

$$\psi = \arctan\left[tan\alpha_h\left(\frac{2VP_x}{H} - 1\right)\right] \quad (76)$$

Având matricea intrinsecă și cea extrinsecă putem calcula matricea de proiecție P (ecuația 64), iar apoi proiecția unui punct 3D de coordonate (X_w, Y_w, Z_w) în spațiul imaginii (u, v) se poate face folosind ecuațiile 77-79.

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} x_W \\ y_W \\ w \end{bmatrix} = P * \begin{bmatrix} X_W \\ Y_W \\ Z_W \\ 1 \end{bmatrix} \quad (77)$$

$$u = \frac{x_w}{w} \quad (78)$$

$$v = \frac{y_w}{w} \quad (79)$$

Prin folosirea ecuației de mai sus și a matricei de proiecție ”P” se poate elimina efectul de perspectivă din imagini, iar algoritmul este descris in secțiunea 3.6. De altfel, proiecția unui punct 2D din imagine în spațiul 3D pe planul drumului (unde Y=0) se face folosind ecuațiile 80-88.

$$a = P_{00} - P_{20} \times u \quad (80)$$

$$b = P_{02} - P_{22} \times u \quad (81)$$

$$c = u \times P_{23} - P_{03} \quad (82)$$

$$d = P_{10} - P_{20} \times v \quad (83)$$

$$e = P_{12} - P_{22} \times v \quad (84)$$

$$f = v \times P_{23} - P_{13} \quad (85)$$

$$x = \frac{(c - b \times z)}{a} \quad (86)$$

$$y = 0 \quad (87)$$

$$z = \frac{(d \times c - f \times a)}{(d \times b - e \times a)} \quad (88)$$

În ecuațiile de mai sus se face rezolvarea ecuației de proiecție cunoscând coordonatele de imagine u, v și presupunând că *y = 0*.

3.4 Stadiul actual al tehnicilor de calibrare a camerei

Calibrarea camerelor este necesară pentru a putea obține informații 3D din lumea reală. Procesul de calibrare a camerelor se referă la determinarea parametrilor interni și geometrici (parametri intrinseci) și a caracteristicilor despre poziția și orientarea camerei în relație cu scena din lumea reală (parametri extrinseci). Parametri intrinseci sunt reprezentați de distanța focală, centrul optic și coeficienții de distorsiune radială și tangențială (ai lentilelor obiectivului). Parametri extrinseci vor fi exprimați folosind un vector de translație și un vector de rotație. Acești parametri sunt utilizați pentru a elimina distorsiuni din imagini și pentru a determina matricele de transformare din o perspectivă în alta sau pentru a deduce sau calcula informații 3D din coordonatele imaginii sau invers (de exemplu determinarea coordonatelor în imagini 2D a punctelor 3D din lumea reală). O mare parte a cercetării științifice s-a concentrat pe determinarea unor metode de calibrare automata si cat mai eficienta. În ultimii s-au dezvoltat multiple metode care se pot împărți în două categorii mari: metode de calibrare bazate pe fotogrammetrie și metode bazate pe auto-calibrare. Prima variantă constă în

observarea unui șablon sau obiect a cărui poziție în scenă este cunoscută (folosirea unor puncte de calibrare cunoscute, numite și puncte de control). A doua metodă de calibrare nu utilizează informații despre obiecte și poziția lor 3D, ci presupune mutarea camerei în scena statică (obiectele sunt nemișcate, doar camera se va repoziționa) și presupune de asemenea cunoașterea corespondențelor între punctele de calibrare din pozițiile diferite. Fotogrammetria a fost folosită inițial începând cu primul război mondial pentru cartografiere (topografie aeriană). Primele calibrări s-au făcut în Canada în anii 1920 pentru a determina punctul principal, adică zona unde distorsiunea radială cauzată de lentile este simetrică. Până în anii 1950 tehnicile erau asemănătoare cu ideile prezentate în Canada. Prin modificarea distanței focale s-a remarcat faptul că se poate reduce eroarea cauzată de distorsiunile radiale [Clarke1998]. Distanța focală "calibrată" înseamnă că distorsiunile radiale sunt echilibrate: părți egale de distorsiuni negative și pozitive. Totodată la sfârșitul anilor 1940 s-a observat și faptul că pot exista distorsiuni tangențiale, care sunt cauzate de alinierea defectuoasă a componentelor din interiorul lentilelor sau sticla de proastă calitate în lentile.

Principalele variante de calibrare folosind fotogrammetria presupun calcularea matricei de transformare a proiecției [AdbelAziz1971], [Strat1984] sau [Ganapaphy1984], optimizarea neliniară a modelelor de imagini complexe [Faig1975], [Gennery1979], metoda celor două planuri [Yakimovsky1978], [Martins1981] sau calibrarea în doi pași propusă de Roger Tsai [Tsai1986]. Un mare efort se concentrează și pe metodele de auto-calibrare [Hartley1997], [Enciso1997], iar o primă analiză a acestor tehnici a fost prezentată în [Clarke1998]. În continuare voi prezenta o lucrare de referință pentru calibrare, pe care am folosit-o pentru a compara rezultatele obținute folosind metodele prezentate în această teză.

Una dintre cele mai populare lucrări este [Zhang2000] care combină elemente atât din fotogrammetrie (observația unui model cunoscut), cât și din auto-calibrare (mutarea camerei în scena statică). Algoritmul de calibrare din lucrare este implementat atât în software-ul OpenCV [OpenCV], cât și în Matlab [Matlab2019]. Metoda lui Zhang se bazează pe minim 3 observații ale unui model cunoscut ("pattern") pe o suprafață plană din scenă 3D (unde în general se axa Z = 0), de obicei o tablă de șah la se cunosc numărul de pătrate negre dar și dimensiunea lor în milimetri. Colțurile pătratelor care se ating din tablă de șah sunt identificate în spațiul imaginii definit ca *[u, v, 1]* și vor forma puncte de control pe planul lumii definit ca fiind: *W = [X, Y, 0]*. Relația de corespondența dintre cele două planuri este definită de matricea de omografie *H* ("homography matrix") și exprimată în ecuația 89.

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = H \times \begin{bmatrix} X \\ Y \\ 0 \end{bmatrix} = A \times [\, r_1 \quad r_2 \quad t\,] \times \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \tag{89}$$

Ecuația 90 reprezintă calculul matricei *H*, determinată de matricea camerei înmulțită cu un parametru de scalare λ.

$$H = [\, h_1 \quad h_2 \quad h_3\,] = \lambda A\, [\, r_1 \quad r_2 \quad t\,] \tag{90}$$

Din faptul că r_1 și r_2 sunt ortogonale, se obțin următoarele două relații între h_1 și h_2:

$$h_1^t A^{-t} A^{-1} h_2 = 0 \tag{91}$$

$$h_1^t A^{-t} A^{-1} h_1 = h_2^t A^{-t} A^{-1} h_2 \quad (92)$$

Din perechile de puncte de calibrare identificate în imagine și punctele din plan, se estimează câte o matrice H_i pentru fiecare imagine *"i"* a scenei folosind metoda DLT ("direct linear transformation") descrisă în lucrarea [AdbelAziz1971]. În următorul pas, prin ignorarea coeficienților de distorsiune se pot determina parametri intrinseci ai camerei (matricea A_i) pentru fiecare matrice H_i. Cunoscând matricea A_i parametri extrinseci R_i și t_i se pot estima folosind ecuațiile 93-96.

$$r_1 = \lambda A^{-1} h_1 \quad (93)$$
$$r_2 = \lambda A^{-1} h_2 \quad (94)$$
$$r_3 = r_1 \times r_2 \quad (95)$$
$$t = \lambda A^{-1} h_3 \quad (96)$$

Parametrul de scalare lambda (λ) este determinat în modul următor (ecuația 97):

$$\lambda = \frac{1}{\|A^{-1} h_1\|} = \frac{1}{\|A^{-1} h_2\|} \quad (97)$$

Coeficienții de distorsiune radială k_1 și k_2 sunt estimați folosind minimizarea "linear least squares", iar în ultimul pas sunt rafinați toți parametri A_i, k_{1i}, k_{2i} pentru toate observațiile (imaginile). Se calculează eroarea proiecției totale pentru corespondențele de puncte și se încearcă minimizarea acestei erori. Această rafinare reprezintă o problemă de optimizare non-liniară și se rezolvă folosind tehnica Levenberg-Marquart [More1977].

O bună parte din aceste lucrări științifice se axează pe determinarea simultană a parametrilor intrinseci și extrinseci ai camerelor folosite. În continuare sunt prezentate diferite tehnici pentru a determina în mod semi-automat sau complet automat a unor parametri de calibrare ai camerei folosind fie viziune artificială sau inteligență artificială bazată pe rețele neuronale.

Calibrarea automată a camerei reprezintă un factor crucial pentru a obține sisteme precise și robuste bazate pe viziune artificială cu scopul de a asista conducătorii autovehiculelor. Sistemele de percepție care măsoară scena vor necesita o corelație între scenă 3D a lumii și imaginea 2D. Aceste corespondențe sunt ușor de calculat în sisteme bazate pe mai multe camere (ex: stereo-viziune), dar soluțiile bazate pe o singură cameră, monoculare sunt mai ușor de folosit și de implementat în practică. Principalul dezavantaj al unei soluții monoculare este lipsa informației despre adâncime din scenă, informație prezentă în sisteme cu două sau mai multe camere. Pentru a depăși aceste limitări, sistemele monoculare se bazează în general pe constrângeri geometrice impuse scenei: de exemplu presupunerea că drumul este plat, că obiectele sunt de dimensiuni standard, etc., dar aceste sisteme tot vor avea nevoie de o calibrare corectă. În mod tradițional, întreg procesul de calibrare este unul laborios efectuat într-un mediu foarte bine controlat (de obicei un laborator), care presupune măsurarea unor obiecte plasate manual în scena observată. Dacă ne referim la plasarea unui dispozitiv mobil pe un parbriz al unui vehicul, aceste constrângeri de calibrare nu se pot îndeplini, de aceea este util un proces de calibrare asistat sau chiar complet automatizat.

Calibrarea camerelor reprezintă o zonă de interes în cercetare, în contextul sistemelor de asistență a conducătorilor auto. Un prim pas în rezolvarea problemei calibrării este dat de găsirea punctului din imagine unde liniile paralele din scena 3D a lumii se intersectează, aceste punct fiind numit și punct de fugă ("vanishing point"). Acesta este util pentru determinarea parametrilor extrinseci ai sistemului camerei. În [Caprile1990] autorii au prezentat idei similare pentru a determina orientarea camerei încă din anii 1990, unde au folosit un sistem bazat pe doua camere. În primul pas parametri intrinseci ai fiecărei camere sunt determinați dintr-o imagine a unui cub (folosit pentru a determina punctele de fugă). În al doilea pas sunt determinați parametri extrinseci dintr-o pereche de imagini a unui șablon de calibrare de dimensiuni cunoscute. O altă lucrare folosind punctele de fugă pentru calibrare este [Wildenauer2012] care folosește algoritmul RANSAC pentru clasificarea segmentelor de linii care sunt folosite ca ipoteze ale punctelor de fugă.

Un alt caz unde este necesară o calibrare a unei singure camere video este cel în care se monitorizează traficului rutier. Camera este amplasată într-o poziție fixă, iar calibrarea se poate face prin amplasarea unui șablon pe șosea [Masoud2004], dar se poate face și din analiza trăsăturilor din imagini. Lucrarea [Wang2007] reprezintă un mod de calibrare bazat pe identificarea marcajelor rutiere (a liniilor de separare a benzilor de circulație) din care se extrag segmente. Din intersecția segmentelor se identifică punctul de fugă care este utilizat pentru calcularea informațiilor despre cameră. Totuși, este necesară informația despre lățimea benzii de circulație și înălțimea camerei față de sol sau lungimea unui marcaj detectat în imagine.

Folosirea algoritmilor de flux optic a fost de asemenea utilizată pentru calibrarea camerelor, în special în contextul vehiculelor inteligente. Algoritmii de flux optic ("optical flow") oferă informații despre viteza proprie a vehiculului, dar și a celorlalte obiecte din scenă prin asignarea unui vector de viteză pentru trăsăturile relevante din imagine. În general algoritmii de flux optic au la bază detecția trăsăturilor importante (prin algoritmii SIFT, SURF, etc.) și găsirea acelorași trăsături în cadrele următoare. Pentru calibrarea camerelor în vehicule autonome se face presupunerea că vehiculul se deplasează pe o suprafață plată, astfel încât segmentele definite de fluxul optic sunt proiecții ale unor linii paralele în sistemul de coordonate al lumii, iar aceste linii paralele se intersectează în spațiul imaginii în punctul de fugă. Dacă deplasarea vehiculului se face drept, iar camera este orientată înainte înspre direcția de mers, atunci punctul de fugă va defini și locația liniei de orizont din imagine. Aceste informații sunt folosite pentru determinarea orientării camerei. Un sistem de calibrare folosind aceste idei a fost prezentat în [Tan2006], unde camera este montată în lateral și este utilizată pentru a analiza unghiul mort ("blind spot"). Fluxul optic este util și pentru a determina timpul până la o coliziune cu un obstacol ("time to collision") și este folosit în sisteme de asistență a șoferilor pentru evitarea accidentelor.

Sistemele de calibrare monoculară pot dispune și de utilizarea unor senzori adiționali montați pe vehicul, cum ar fi: senzori bazați pe laser sau radar. Calibrarea camerelor folosind și senzori de tip LIDAR a fost folosită atât în comunitatea științifică, cât și în producție de către producătorii auto. Lucrările existente urmează în general aceiași pași: găsirea corelării dintre punctele 3D LIDAR și trăsăturile sau muchiile imaginilor 2D de la camera monoculară. Punctele obținute din LIDAR sunt aliniate cu punctele imaginii din camera utilizând algoritmi de potrivire a conturului. Muchiile obiectelor din scenă 3D LIDAR sunt proiectate în imagine. Calibrarea este efectuată prin ajustarea parametrilor extrinseci ai camerei până când aceste

puncte 3D proiectate în imagine sunt aliniate la trăsăturile 2D detectate în imaginea camerei. Astfel de abordări au fost prezentate în [Bileschi2009] și [Levinson2013]. Procedura de calibrare trebuie să combine într-un final toate informațiile senzoriale într-un sistem de referință comun. Lucrarea [Debattisti2013] prezintă o soluție de calibrare între cameră și senzori bazați pe laser unde se folosește un triunghi pe post de șablon de calibrare pentru a extrage corespondențe dintre senzori. În [Pereira2016] autorii oferă o soluție de calibrare automată între camere și senzori LIDAR prin folosirea unei mingi pe post de șablon de calibrare. Fiecare senzor din vehiculul autonom va detecta mingea și apoi centrul ei pentru a determina relațiile dintre senzori, iar pentru o calibrare completă va fi nevoie de cel puțin 3 cadre în care mingea să fie amplasată diferit. Calibrarea folosind o minge a mai fost prezentată și în lucrarea [Wahab2011], unde autorii au amplasat mingea la distanțe de câte 10 cm și au determinat lățimea ei (în pixeli) în imagini. Apoi au extras relația dintre lățime și distanță pentru robotul mobil, dar având erori de până la 15% la estimarea distanțelor. În literatură au mai fost prezentate abordări de calibrare a camerelor monoculare și prin folosirea unui proiector ce iluminează scena observată cu un model cunoscut de lumini, tehnică fiind cunoscută sub denumirea de “lumină structurată”. Astfel de abordări sunt prezentate în [Moreno2012] sau în [Zhang2006]. Cu toate acestea, utilizarea senzorilor externi reprezintă un factor de cost suplimentar și reduce mobilitatea și portabilitatea sistemului de percepție a traficului rutier. De asemenea, senzorul LIDAR necesită o calibrare cu propria sa metodologie și constrângeri.

Calibrarea folosind rețele neuronale a fost propusă încă din anii 1990. În [Lynch1991] autorii au realizat calibrarea a unui sistem compus din două camere folosind o rețea neuronală și algoritmul “backpropagation” pentru găsirea corespondenței între cele două camere și sistemul de coordonate al lumii. Maparea neliniară între sistemele de coordonate ale camerelor și a lumii este învață de rețeaua neuronală. În lucrarea [Ahmed1999] este prezentată o soluție ce folosește o rețea cu mai multe niveluri de tip “feed forward” (perceptron multi-strat). Având corespondența între puncte 3D și 2D, rolul rețelei este de a genera parametri camerei prin regresie. Aceste soluții sunt foarte eficiente ca și timp de execuție după ce au fost antrenate rețelele, dar au dezavantajul ca necesită informații “a priori” despre punctele din scenă și cele din imagine. Pregătirea datelor de intrare necesită timp și este un proces minuțios.

3.5 Calculare automată distanță focală

Calcularea distanței focale și a punctului principal reprezintă primii pași necesari calibrării, deoarece acești parametri formează matricea camerei. Am utilizat dispozitive mobile bazate pe Android pentru percepție. Pentru a determina distanța focală exprimată în pixeli este necesară cunoașterea câmpului vizual orizontal sau vertical. Sistemul de operare Android oferă o interfață de acces (API - Application Programming Inferface) a parametrilor intrinseci, astfel se pot obține valorile câmpului vizual orizontal dar și cel vertical.

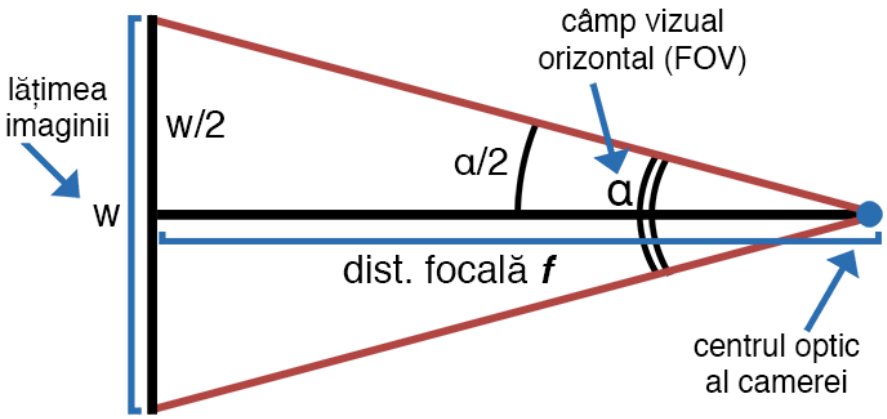


Figura 62. Determinarea distanței focale cunoscând dimensiunea imaginii și câmpul vizual.

Figura 62 ilustrează modul de calcul al distanței focale, bazat pe lățimea imaginii și câmpului orizontal de vizualizare. Formula de calcul este reprezentată în ecuația 66 (secțiunea 3.2).

Distanța focală calculată prin folosirea informațiilor din API-ul Android a fost comparată cu distanța focală calibrată manual. Procesul manual de calibrare este laborios și constă în amplasarea dispozitivului într-un punct fix și fotografierea unui model cunoscut (în acest caz o tablă de șah de dimensiuni 9x6) la orientări și distanțe diferite și apoi copierea pozelor pe un sistem desktop și calcularea parametrilor folosind software-ul OpenCV. Figura 63 reprezintă acest proces manual de calibrare. Calibrarea se poate face și în mod analog: prin amplasarea fixă a modelului de calibrare și mutarea telefonului la orientări și distanțe diferite.

Figura 63. Calibrare manuală folosind un șablon cunoscut (tabla de șah 9x6): se fac mai multe poze în care telefonul este menținut în poziție fixă, iar tabla de șah este rearanjată în scenă la orientări diferite.

O comparație a rezultatelor calibrării preluat dintr-o lucrare anterioară [Danescu2016a] și extinsă cu dispozitive mobile mai actuale este prezentată în Tabel 9.

Tabel 9. Distanțe focale (în pixeli) calculate vs. calibrate.

	Samsung Galaxy S5 (2014)	**Samsung Galaxy S8 Plus (2017)**	**Samsung Galaxy A8 (2018)**	**Sony Xperia Z1 (2013)**	**Motorola Moto G (2013)**	**HTC One Mini 2 (2014)**
Dist. focală calculată Android	697	502	503	484	598	530
Dist. focală calibrată	686	504	504	502	617	520
Procent eroare orizontală	1.71%	0.31%	0.15%	2.81%	2.96%	1.56%
Procent eroare verticală	2.29%	0.41%	0.20%	3.75%	3.95%	2.08%

Din Tabel 9 se poate observa că rezultatele, deși nu sunt identice, se află într-o marjă de eroare de sub 4% raportat la dimensiunea verticală a imaginii și sub 3% raportat la înălțimea imaginii și se pot folosi în sisteme monoculare bazate pe dispozitive mobile. Dispozitivele de ultimă generație (Samsung S8 Plus din 2017 și Samsung A8 din 2018) au eroarea sub 0.4% datorită avansului tehnologic (lentilele prezintă distorsiuni minimale).

Având distanța focală, se poate considera punctul principal ca fiind în centrul imaginii (lățime / 2, înălțime / 2), iar coeficienții de distorsiune sunt ignorați. Folosind această abordare se poate calcula dinamic matricea intrinsecă a camerei caracteristică fiecărui dispozitiv mobil Android specifică pentru dimensiunea imaginii și a nivelului de zoom ales. În cazul sistemelor unde se folosesc alte tipuri de camere video, distanța focală este dată de producător și este setată static, manual în algoritm.

Așadar, problema principală a calibrării este dată de parametri externi, de cunoașterea matricei extrinsecă a camerei care este compusă din: vectorul de translație și matricea de rotație a camerei. În acest capitol voi prezenta abordări diferite pentru a estima și acești parametri într-un mod dinamic și automat.

3.6 Eliminare efect de perspectivă din imagini

Imaginile achiziționate cu o cameră vor avea un efect de perspectivă. Mai specific, în imaginile din traficul rutier liniile de marcaje se vor intersecta într-un punct la orizont (în punctul

de fugă) deși ele sunt dispuse paralel. Obstacolele vor apărea mai mari la o distanță mai mică de cameră și viceversa: vor avea o dimensiune mai mică în imagine dacă sunt la o distanță mai mare față de cameră. Presupunând că suprafața drumului este plată și că este aliniată cu sistemul de coordonate al camerei, se poate transforma imaginea de perspectivă cu una în care efectul de perspectivă este eliminat. În literatura de specialitate această imagine este denumită IPM ("Inverse Perspective Mapping") sau "bird's eye view" sugerând o vedere periferică de sus a scenei. Imaginile cu efectul de perspectivă eliminat sunt folosite în contextul unui vehicul autonom pentru următoarele sarcini: detecția de benzi de circulație [Bertozzi1998] sau a marcajelor rutiere [Chira2010], detecția de obstacole [Danescu2011], [Danescu2016a], pentru calcularea fluxului optic [Mallot1991] sau pentru analiza curburii drumului [Pomerleau1995] pentru determinarea unghiului de virare a unui vehicul. Figura 64 ilustrează un exemplu de imagine IPM din traficul rutier.

Figura 64. Imagine sursă (strângă) și aceeași imagine cu efectul de perspectivă eliminat (dreaptă).

Fiecare punct din imaginea IPM are un corespondent direct în sistemul de coordonate al lumii, pe planul drumului *XOZ* (Figura 60), unde coordonata pe înălțime Y este 0. Prin urmare, pentru obstacolele detectate în imaginea IPM se poate calcula poziția în sistemul de coordonate 3D al lumii. Obstacolele în imaginile IPM vor avea un aspect distorsionat deoarece ele nu aparțin suprafeței drumului și nu sunt conforme geometriei drumului plat. De aceea doar punctul de contact dintre obstacol și sosea este relevant în cazul unui algoritm de detecție a obstacolelor. Poziția acestui punct de contact poate fi corelată direct cu poziția obstacolului din lumea reală.

Algoritmul de calculare a imaginii cu efectul de perspectivă eliminat este următorul:

Algoritm calculare IPM
Intrare: imaginea sursă I
Ieșire: imaginea IPM I_T
Pentru fiecare pixel de coordonate (u_T, v_T) *din* I_T
    $x_W = k\ u_T + x_{min}$
    $z_W = j\ v_T + z_{min}$
    $y_W = 0$
    $(u, v) = Proiecție3Dla2D(x_W, y_W, z_W)$
    $I_T(u_T, v_T) = I(u, v)$
Final buclă

În algoritmul de sus, constantele *k, j,* x_{min} și z_{min} au fost alese în așa fel încât să reprezinte doar porțiunea cea mai relevantă a zonei de drum în imaginea IPM generată. În sistemul propriu de percepție am definit zona vizibilă ca fiind de dimensiune 24 metri lățime pe axa X și 50 metri lungime pe axa Z, ceea ce înseamnă că parametrul x_{min} este -12000, x_{max} este +12000, iar z_{min} este 0 și z_{max} este +50000 (valorile exprimate sunt în milimetri). În algoritmul de mai sus mărimea pixelului pe axa X (*"k"*) este definită de ecuația 98, în timp ce dimensiunea pe axa Z (*"j"*) este dată de ecuația 99.

$$k = \frac{x_{max} - x_{min}}{IPM\_width} \tag{98}$$

$$j = \frac{z_{max} - z_{min}}{IPM\_height} \tag{99}$$

Funcția de proiecție apelată folosește matricea de proiecție formată din parametri intrinseci și extrinseci ai camerei pentru a calcula poziția în planul imaginii a unui punct 3D din sistemul de coordonate al lumii (ecuația 77 din secțiunea 3.3).

O variantă mai eficientă ar fi cea în care se folosește matricea de omografie *H* pentru calcularea coordonatelor noi ale punctelor din imaginea sursă în imaginea IPM. Pentru a determina matricea *H*, mai întâi se calculează matricea de scalare *S* (ecuația 100):

$$S = \begin{bmatrix} k & 0 & x_{min} \\ 0 & j & z_{min} \\ 0 & 0 & 1 \end{bmatrix} \tag{100}$$

Matricea de omografie este calculată astfel:

$$H = P * S \tag{101}$$

Transformarea punctelor se va face folosind următoarea ecuație:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{102}$$

Algoritmul de calculare eficient folosind matricea de omografie H este descris mai jos:

Algoritm calculare IPM eficient

Intrare: imaginea sursă I
Ieșire: imaginea IPM I_T
Pentru fiecare pixel de coordonate (u_T, v_T) *din* I_T
    $x = h_{00} * u_T + h_{01} * v_T + h_{02}$
    $y = h_{10} * u_T + h_{11} * v_T + h_{12}$
    $s = h_{20} * u_T + h_{21} * v_T + h_{22}$
    $u_p = x / s$
    $v_p = y / s$
    $I_T(u_T, v_T) = I(u_p, v_p)$
Final buclă

Restricțiile impuse de folosirea unui model plat de drum pot fi eliminate prin algoritmi de urmărire capabili să modeleze erorile cauzate de aceste presupuneri și limitări ale geometriei drumului care nu sunt conforme cu realitatea în anumite situații.

3.7 Interfață intuitivă pentru asistarea calibrării manuale a parametrilor extrinseci

Calibrarea parametrilor extrinseci se poate face într-un mod manual, după cum am prezentat în lucrarea [Danescu2016a]. Având la bază niște presupuneri se poate simplifica procesul de calibrare astfel încât să se poată efectua de către utilizatorii de rând ai aplicațiilor. Un prim pas constă în alinierea dispozitivului mobil în centrul parbrizului din vehicul. Folosind aplicația dezvoltată, utilizatorii au posibilitatea de a introduce manual înălțimea camerei față de sol (*"H"* din matricea intrinsecă) și distanța dintre cameră și bara față a vehiculului (*"L"* din matricea intrinsecă). De asemenea, utilizatorul este nevoit să introducă manual unghiul de înclinare θ ("pitch"). Pentru a obține rezultate robuste, se proiectează linia orizontului rezultată din unghiul introdus, iar utilizatorul va trebui doar să alinieze linia virtuală cu linia reală a orizontului pentru a obține unghiul corect de înclinare. În Figura 65 se poate observa interfața de calibrare manuală. Totodată, sistemul generează imaginea IPM automat în funcție de parametri introduși. Această imagine IPM este suprapusă peste o rețea de linii orizontale și verticale. Distanța între linii (în pixeli) este cunoscută și poate fi o metrică bună pentru ajustarea celor 2 parametri: "H" și "L". Dacă benzile de circulație sunt vizibile, un indicator pentru o calibrare reușită este dată de faptul că liniile benzii vor deveni paralele în imaginea IPM.

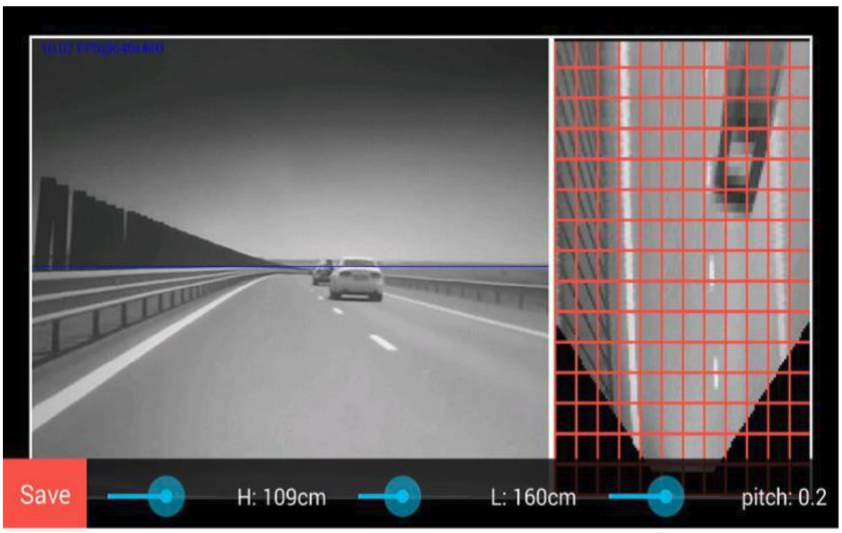


Figura 65. Interfața de calibrare manuală.

Această interfață utilizator poate fi oricând extinsă pentru a include și calibrarea manuală a unghiurilor de girație ("yaw") și rotire ("roll"). Am realizat sistemul de calibrare și

pentru sisteme desktop care include toate unghiurile de calibrare, precum și posibilitatea de a controla restul de parametri intrinseci ai camerei.

3.8 Auto-calibrare parametri vector de translație - modul offline de procesare a datelor

O metodă pentru auto-calibrarea sistemelor monoculare se poate baza pe prin utilizarea unei corespondențe dintre dimensiunea fixă unui obiect din scenă (sistemul de coordonate al lumii) și dimensiunea acestuia în pixeli (în spațiul imaginii). În literatura de specialitate există lucrări și sisteme unde auto-calibrarea este realizată doar pentru sisteme unde camera este într-o poziție fixă, având o scenă fixă și doar vehiculele sunt în mișcare [Wang2007]. Această corespondență este dată de o serie de ecuații cu caracter liniar.

Pentru a obține un sistem capabil de auto-calibrare, am ales ca obiect de referință dimensiunea unei benzi de circulație. Astfel folosind algoritmi de viziune artificială am detectat banda curentă de circulație și am obținut dimensiunea (lățimea) ei exprimată în pixeli în imaginea cu perspectiva eliminată. Din experimentele efectuate, se observă o corespondență între informațiile despre obiect (lățimea benzii) și înălțimea camerei ("H" din vectorul de translație).

Algoritmul de alegere a unei valori optime pentru înălțimea camerei ("H") în funcție de lățimea benzii de circulație ("LW") din imaginea IPM este prezentat mai jos, iar diferite tehnici de determinare efectivă a lățimii benzii de circulație într-o imagine sunt prezentate tot în această secțiune la final. Primul pas al algoritmului constă în determinarea lățimii de benzii curente exprimată în pixeli din imaginea IPM. Considerând o dimensiune standard a benzii de circulație de 3.5 metri liniari în toată secvența, dimensiunea *LW* optimă va fi echivalentul acesteia exprimat în pixeli, calculată din constantele z_{min}, z_{max} și x_{min}, x_{max} alese pentru scenă. Dacă *LW* curent este mai mic decât *LW* optim atunci *H* se va decrementa, în caz contrar se va incrementa. Următorul pas constă în determinarea noii lățimi de bandă din imaginea IPM cu noua matrice de proiecție având noul *H* setat. Din această măsurătoare a lățimii benzii se poate calcula folosind o ecuație liniară noul *H* optim, astfel încât acesta să corespundă unui *LW* optim, într-o marjă de eroare de +/- 5% și se trece la procesarea imaginii următoare din secvență. Totuși, dacă noul *LW* nu corespunde valorii așteptate cu tot cu marja de eroare, înseamnă că măsurătoarea benzii de circulație nu este bună și se apelează recursiv funcția pentru a încerca din nou, prin generarea altor valori pentru *H*. Algoritmul menține și informația despre numărul de iterații recursive efectuate pe o imagine, iar ca măsură de siguranță se va trece automat la următoarea imagine după patru încercări recursive eșuate. Această măsură este utilă pentru imaginile în care nu se poate detecta deloc lățimea unei benzi de circulație (nu există marcaje vizibile).

Pseudocod: Algoritm calibrare înălțime cameră:

Pentru *fiecare imagine IPM din secvență (I_T, nr_iterații = 0):*

    Dacă *nr_iterații = 1:*

        H = H _inițial

LW_inițial = calculareLățimeBandă(I_T)
LW = LW_inițial
***Dacă** nr_iterații = 4:*
procesează următoarea imagine din secvență (I_{T+1}, nr_iterații = 0)
Altfel:
LW = calculareLățimeBandă(I_T)

***Dacă** LW < LW_optim:*
H = decrementare înălțime cameră
Altfel:
H = incrementare înălțime cameră

recalculare matrice de proiecție a camerei
recalculare imagine IPM I_T
LW = calculareLățimeBandă(I_T)
newH = calculare înălțime cameră din ecuația liniară determinată de H_inițial, LW_inițial și LW
***Dacă** LW >= LW_optim * 0.95 && LW <= LW_optim * 1.05:*
H = newH
salvare newH într-o listă
procesează următoarea imagine din secvență (I_{T+1}, nr_iterații = 0)
Altfel:
reprocesează imaginea din secvență (apel recursiv) (I_T, nr_iterații++)

În cazul rulării auto-calibrării pe o secvență compusă din mai multe imagini se pot aduce anumite îmbunătățiri ale rezultatelor. Un prim pas ar fi salvarea tuturor valorilor pentru "H" obținute pe o secvență și apoi folosirea unui filtru median pe acest șir. Acest pas a adus îmbunătățiri semnificative pe majoritatea secvențelor folosite. O altă posibilă îmbunătățire constă în salvarea și a tuturor valorilor pentru lățimea benzii ("LW"). Există unele cadre unde nu se poate detecta o bandă vizibilă, iar alegerea unui *"H"* optim se poate face doar dintr-o listă unde valorile de *LW = 0* sunt eliminate.

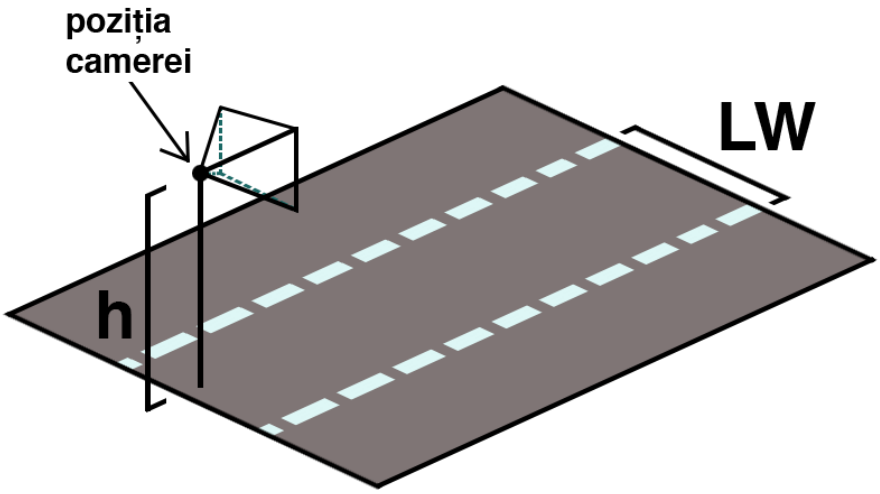


Figura 66. Ilustrare sistem cameră și lățime bandă.

Lățimea benzii curente se poate determina în 2 moduri: în imaginea originală (perspectivă) sau în imaginea cu efectul de perspectivă eliminat (“IPM”), iar principalele abordări au la bază identificarea marcajelor rutiere care separă benzile.

În imaginea perspectivă, cunoscând punctul de fugă (secțiunea 3.9) se poate determina banda curentă de circulație prin alegerea liniilor dominante care se suprapun pe marcajele care au intersecția în punctul de fugă. Liniile dintr-o imagine pot fi determinate folosind algoritmul Hough [Hough1962], [Duda1972], care oferă o listă de linii sub formă de coordonate polare (ρ, θ). Alegerea liniilor care determină banda curentă se face folosind două liste de linii filtrate: liniile din stânga (Ls) care aparțin unui interval angular $\theta \in [20°\text{-}80°]$ și liniile din dreapta (Ld) care au orientarea $\theta \in [110°\text{-}170°]$. Următor pas constă în calcularea intersecției liniilor din cele două liste (Ls și Ld) și se păstrează doar liniile care au punctul de intersecție în proximitatea punctului de fugă. Se alege o valoare mediană din cele două liste, iar apoi se aleg 2 puncte de pe fiecare linie mediană din Ls și Ld, aliniate pe axa orizontală sub punctul de fugă. Aceste două puncte sunt proiectate în spațiul 3D și apoi în planul 2D al imaginii IPM (folosind ecuațiile din secțiunea 3.3) pentru a determina lățimea benzii de circulație. Această metodă presupunerea cunoașterii punctului de fugă, iar soluții pentru determinarea acestui punct sunt prezentate în secțiunea 3.9.

A doua metodă de calcul folosește imaginea cu efectul de perspectivă eliminat (IPM) pentru detecția liniilor unei benzi de circulație. Proprietatea cea mai importantă a marcajelor este că acestea au o intensitate diferită față de asfalt. Astfel, marcajele se pot detecta prin identificarea tranzițiilor de intensități, denumite în literatura de specialitate și întuneric-lumină-întuneric (DLD - “dark light dark”), datorită tranziției dintre intensități închise, deschise și apoi din nou închise. În literatura de specialitate au fost propuse și implementate diferite filtre pentru detecția acestor tranziții. O lucrare de referință este prezentată în [Broggi1995]. Algoritmul din lucrare se bazează pe faptul că intensitatea pixelului care aparține unui marcaj este mai mare decât valoarea pixelilor din stânga și din dreapta, care aparțin asfaltului. Rezultatul aplicării unui astfel de filtru este ilustrat în figura de mai jos:

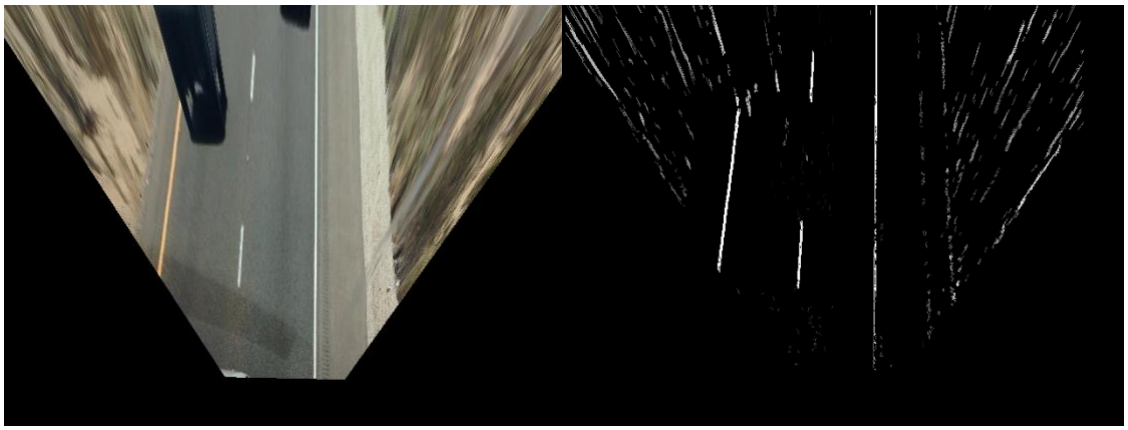


Figura 67. Imagine sursă și imagine rezultată în urma aplicării filtrului DLD.

Pentru a obține o astfel de imagine binară se creează o hartă de voturi, unde fiecare pixel va primi o valoare în funcție de valoarea intensității sale și a intensității pixelilor din stânga și din dreapta. Pentru buna funcționare a acestor filtre de tip DLD este necesară definirea unei valori standard de lățime a marcajului, iar valoarea este dată în număr de pixeli. Având o imagine IPM cu filtrul DLD aplicat, se poate aplica transformata Hough sau orice altă metodă pentru găsirea liniilor verticale din centrul imaginii care vor forma banda curentă. Distanța dintre ele va reprezenta lățimea benzii curente.

Umbrele de pe carosabil ar putea produce dificultăți în procesul de filtrare de marcaje din imagini, dar această problemă a fost abordată și au fost propuse soluții în literatura de specialitate. O variantă de rezolvare este dată de folosirea unui altfel de spațiu de culoare care este invariant la umbre. Acest spațiu de culoare este denumit log-chromaticity, iar o abordare este prezentată în [Alvarez2007]. Dezavantajul este dat de timpul computațional ridicat, deoarece este necesară o calculare a unor logaritmi compuși din valoarea intensității pixelului pe anumite canale de culoare, dar și de necesitatea unei camere foto calibrate.

Cele două variante de calcul a benzilor de circulație oferă două valori apropiate și pot fi folosite împreună pentru determinarea unui “H” optim. În concluzie, prin folosirea informațiilor din lumea reală se poate efectua calibrarea automată a parametrilor extrinseci a sistemelor compuse din o singură cameră, unde în general calibrarea se realizează în medii controlate (de exemplu în laboratoare sau scene statice).

3.9 Calculare punct de fugă folosind viziune artificială

În imaginile din traficul rutier în scene urbane sau pe autostradă, se poate observa prezență a unui singur punct de fugă (“vanishing point” - VP). Acesta este punctul din imaginea perspectiva în care liniile paralele din scenă se intersectează și se poate folosi pentru a extrage informațiile despre parametri extrinseci ai camerei, mai specific unghiurile de înclinare și girație.

Metodele convenționale de găsire a punctului de fugă se folosesc de proprietățile geometrice [Kong2009], [Moghadam2012] sau de informațiile despre textură [Bui2013a], [Bui2013b] ale trăsăturilor relevante din scenă, cum ar fi: liniile benzilor de circulație [Wang2004] sau muchiile trotuarelor. Aceste abordări extrag trăsăturile relevante, după care aplică o schemă de votare și în cele din urmă extrag candidați pentru puncte de fugă, astfel utilizând spațiul imaginii. Totuși, în literatura de specialitate au fost prezentate abordări care folosesc și sfera de tip Gaussian [Magee1984], [Kluger2018], unde liniile paralele din scena observată sunt proiectate pe o sferă Gaussiană unde sunt procesate sub formă de elipse sau cercuri. În [Bazin2012] autorii au prezentat o soluție bazată pe algoritmul RANSAC pentru determinarea punctului de fugă. Majoritatea metodelor sunt bazate pe folosirea unor algoritmi de urmărire: filtre Kalman [Suttorp2006] sau scheme de votare [Kong2009]. În lucrarea [Kong2009] autorii propun o metodă bazată strict pe procesare de imagini, fără urmărire, folosind filtre Gabor pentru analiza texturii. Aceste filtre sunt configurate pentru a răspunde cât mai bine pentru imaginile capturate în traficul rutier. Filtrele sunt alese la diferite unghiuri

și oferă un răspuns mare pentru pixelii cu aceeași orientare ca și filtrul. Aplicarea mai multor filtre, cu diferite unghiuri și diferite mărimi adaugă o complexitate în plus și crește timpul de procesare al sistemului. Folosirea filtrelor Gabor pentru detecția punctului de fugă a fost publicată și în [Russmusen2004]. Lucrarea [Wu2016] prezintă o abordare diferită în care se extrag liniile dintr-o imagine folosind algoritmul LSD ("line segment detector") [Grompone2010]. În pasul următor aceste linii sunt filtrate (se elimină cele cu orientări eronate, cum sunt cele orizontale) și apoi sunt folosite pentru a crea o hartă de voturi pentru punctul de fugă. Schema de votare folosește o ponderare a lungimii liniilor, a orientării liniilor și ia în calcul și orientarea pixelilor vecini ai liniilor (și aceștia având o pondere). În cadrul acestui capitol voi prezenta o abordare clasică, dar și una în care am folosit rețele neuronale convoluționale pentru determinarea coordonatelor punctului de fugă. În secțiunea 3.9 voi prezenta soluția care utilizează o imagine ca și intrare, iar rețeaua neuronală va oferi predicția punctului de fugă sub forma coordonatelor de pixeli x și y în imagine. Această abordare folosește o bază de date proprie, iar toată soluția a fost publicată în [Itu2017].

Am propus folosirea unei abordări noi, bazată direct pe gradientul și magnitudinea unui pixel. Gradientul unui pixel (punct) al imaginii reprezintă un vector care oferă direcția de variație a intensității imaginii în acel punct. Magnitudinea sau amplitudinea acestui vector va oferi o măsură a vitezei cu care este variată această intensitate. Astfel, punctele din imagine care au o anumită direcție vor forma un unghi care poate fi utilizat pentru a detecta punctul de fugă. Abordarea propusă de mine utilizează gradientul și amplitudinea gradientului pentru a crea o hartă de voturi a pixelilor care au orientarea spre un punct de fugă. Sunt analizate și calculate magnitudini și orientări doar pentru pixelii care reprezintă muchii. Punctele de muchii sunt obținute folosind algoritmul Canny [Canny1986].

Determinarea unei zone de interes ("region of interest" - ROI) este primul pas în calcularea cât mai eficientă a unui punct de fugă. În general zona care trebuie procesată este doar cea formată din drum: între linia orizontului și botul mașinii (dacă este vizibil în imagini). Astfel, prin folosirea unei imagini segmentate care conține doar zona de drum am calculat dinamic zona de interes într-o secvență. Procesul constă în proiectarea pixelilor drumului pe orizontală, crearea unei histograme și găsirea primului punct de drum (care va da coordonata pe axa y a liniei orizontului) și al ultimului punct de drum (care va reprezenta coordonata pe axa y a botului mașinii) din imaginea reproiectată (din histogramă). Obținerea imaginii de drum segmentată este prezentată pe larg în secțiunea 2.11.

Figura 68. Extragerea zonei de interes folosind proiecția pe orizontală (dreapta) a imaginii de drum segmentată (centru). În stânga este imaginea color a scenei.

Figura 68 reprezintă procesul de extragere a zonei de interes unde în imaginea din dreapta este ilustrată proiecția orizontală și sunt ilustrate și cele două linii care determină partea superioară și inferioară a zonei care va fi procesată. Prima figură este imaginea originală, iar a doua reprezintă imaginea segmentată a scenei. Am luat în calcul și situația în care sunt multe vehicule în fața mașinii iar imaginea segmentată de drum nu va conține pixeli în proximitatea liniei de orizont. În acest caz, am ales limita superioară a zonei de interes ca fiind la o distanță de 30% din înălțimea imaginii (originea fiind punctul din colțul stânga jos al imaginii).

În zona de interes, pentru a elimina zgomotul din imagine și pentru a obține rezultate cât mai solide, imaginea sursă este pre-procesată prin aplicarea unui filtru de tip Gaussian cu un nucleu de dimensiune 5. În pasul următor se vor identifica muchiile din zona de interes folosind algoritmul Canny, iar după aceea se poate calcula orientarea și magnitudinea fiecărui punct de muchie. Punctele de pe marcajele rutiere și liniile separatoare de benzi au proprietatea de a converge spre punctul de fugă, iar aceste puncte vor avea orientarea (direcția) gradientului cuprinsă într-un interval angular de (20°, 80°), (-170°, -110°) sau (110°, 170°), (-20°, -80°), după cum am ilustrat în Figura 69.

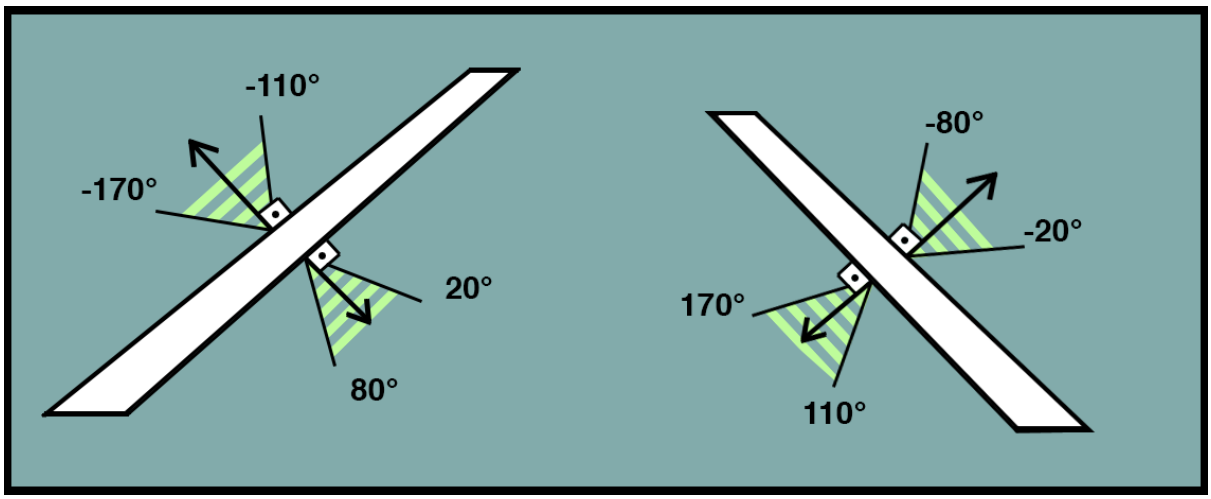


Figura 69. Gradientul punctelor de pe marcaje și intervalele angulare folosite pentru filtrare.

Totuși, determinarea punctelor de muchii folosind algoritmul Canny și apoi calcularea orientării gradienților reprezintă o operație ineficientă din punct de vedere computațional, chiar dacă pot fi folosite și tabele de căutare (“look-up tables” - LUT). De aceea am realizat și implementat o soluție mai eficientă, prin eliminarea necesității detecției muchiilor și prin înlocuirea pasului de calculare a orientării gradienților și filtrarea lor în funcție de intervalul angular.

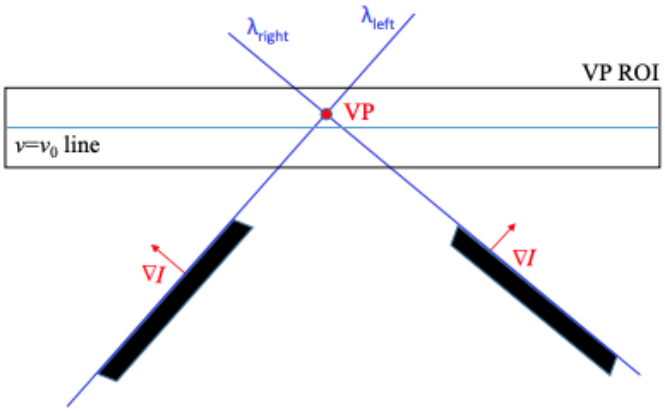


Figura 70. Punctul de fugă (VP) este dat de intersecția liniilor care trec prin punctele de pe marcajele rutiere. Direcția acestor linii este perpendiculară cu orientarea gradientului.

Astfel, pentru fiecare punct (pixel) din zona de interes am calculat gradientul și magnitudinea gradientului. Direcția linei posibile care trece prin acest punct este dată de direcția perpendiculară pe orientarea gradientului (Figura 70). Votul într-o matrice de acumulare 2D se face de-a lungul acestei linii de fiecare dată când linia trece prin zona de interes deja definită. Votul se face proporțional cu magnitudinea gradientului punctului, astfel încât pixelii de tranziție (ai marcajelor benzii de circulație) vor avea o influență mai mare. Pentru punctele din stânga axei centrale a imaginii voturile sunt înregistrate într-o matrice *vpStânga*(x, y), iar pentru punctele din dreapta axei centrale a imaginii, voturile sunt salvate într-o matrice *vpDreapta*(x, y). Deoarece punctul de fugă este intersecția liniilor din toate direcțiile, matricea votului final (*vpFinal*) este calculată ca produsul dintre cele două matrici (*vpStânga* și *vpDreapta*). Pașii algoritmului sunt descriși mai jos:

Algoritm votare punct fugă
Intrare: imaginea de sursă I
Ieșire: harta de voturi vpFinal
*y_end = ROI_end sau 0.3 * height*
***Pentru** fiecare pixel de coordonate (i, j) din I:*
    dx = calcul gradient pe axa x
    dy = calcul gradient pe axa y
    *mag = sqrt(dx * dx + dy * dy)*
    ***Dacă** m > prag*
        ***Pentru** y = (height, y_end):*
            *x = (y - i) * (-dy)/dx + j*
            ***Dacă** (-dy) / dx < 0*
                vpStânga(x, y) += mag
            Altfel
                vpDreapta(x, y) += mag
        Final buclă
Final buclă
*vpFinal = vpStânga * vpDreapta*

După acest pas de procesare se obține o imagine a hărții de voturi în care punctul de fugă este reprezentat de zona cu cele mai multe voturi, ceea ce vizual va reprezenta o zonă

în imagine cu intensitatea luminoasă foarte pronunțată. În Figura 71 se pot observa hărțile de voturi și imaginea sursă folosită. În acest algoritm imaginea de intrare este de tip ”grayscale”, iar singura pre-procesare necesară este aplicarea unui filtru Gaussian pentru atenuarea zgomotelor din imagine.

Figura 71. Rândul de sus: prima imagine conține voturile din lista *vpStânga*, iar a doua imagine voturile din *vpDreapta*. Rândul de jos: în stânga este harta de voturi finală prin înmulțirea primelor două imagini (*vpFinal*), iar în dreapta este imaginea sursă de intrare.

Punctul de fugă este dat de punctul din *vpFinal* cu cele mai multe voturi. Am implementat un mecanism de fereastră glisantă (“sliding window”) pentru a calcula suma intensităților dintr-o fereastră și am determinat fereastra cu cea mai mare sumă (f_{MAX}). Următorul pas constă în alegerea centrului ferestrei f_{MAX} ca și punct final de fugă. Totuși, am implementat și un pas de validare, care este făcut prin calcularea unei valori medii a ferestrelor din jurul f_{MAX}. Analizând imaginea cu harta de voturi am remarcat că un punct bun de fugă este dat doar de mijlocul unei ferestre în care avem valoarea maximă de voturi, în timp ce suma intensității pixelilor din ferestrele diagonale vecine au un raport mai mic de 0.6 față de suma intensității ferestrei punctului de fugă ales (f_{MAX}). Această valoare de prag a raportului a fost determinată experimental și este ajustabilă în funcție de secvențe și de anumite condiții de drum sau iluminare a scenei. Pentru a obține timpi de procesare cât mai eficienți, în calcularea sumei ferestrelor din jurul punctului candidat am folosit o tehnică numită imagine integrală (“integral image” sau “summed area table”). Aceste imagini reprezintă sub formă de tabel (matrice) suma valorii pixelilor astfel: într-un punct (x, y) al imaginii de tip “integral”, vom avea

suma intensității pixelilor din stânga și deasupra punctului (x, y). Valoarea imaginii integrale într-un punct *I(x, y)* se poate determina cu ecuația 103, care poate fi folosită pentru a calcula toată imaginea integrală printr-o singură parcurgere a imaginii sursă. În ecuația 103, *i(x, y)* este valoarea intensității din imaginea sursă.

$$I(x,y) = i(x,y) \; + \; I(x,y-1) + I(x-1,y) - I(x-1,y-1) \qquad (103)$$

Acest tip de reprezentare este utilă pentru calcularea eficientă a sumei pixelilor într-o fereastră dacă se cunosc coordonatele ferestrei. Modul de calcul este prezentat în Figura 72 și ecuația 104.

$$S \; = \; D - B - C - A \qquad (104)$$

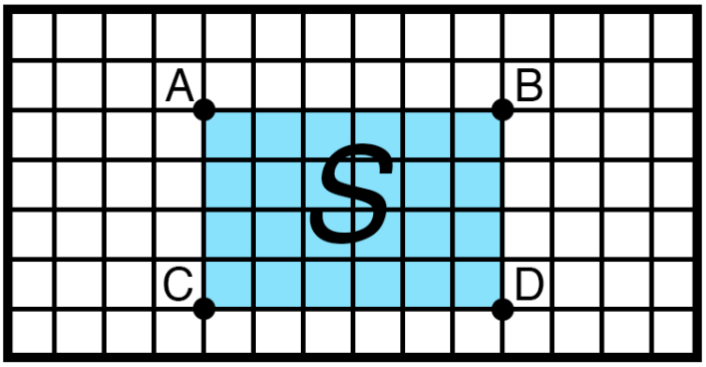


Figura 72. Suma intensității pixelilor dintr-un dreptunghi definit de coordonatele punctelor A, B, C, D într-o imagine integrală.

Figura 73. Stânga: ferestrele de votare alese pentru determinarea unui maxim. Dreapta: rezultatul final al algoritmului cu punctul de fugă calculat.

Dimensiunea ferestrei de căutare a maximului este calculată proporțional în funcție de dimensiunile imaginii sursă și a fost setată să fie 55% din înălțimea imaginii sursă. Avantajul algoritmului propus este dat de viteza redusă de procesare obținută (30-40ms pentru imagini de intrare 640x480 și cu o zonă de interes de votare de aproximativ 200x480, fără paralelizare

sau multi-threading) și de faptul că nu este necesară cunoașterea unor informații despre scenă sau lume. Abordarea este complet independentă de camera foto folosită pentru capturarea imaginilor, de aceea se poate folosi ca un prim pas de analiză a secvențelor video pentru a obține informații despre drum și scena din trafic.

3.10 Calculare parametri extrinseci folosind punctul de fugă calculat din rețele neuronale convoluționale

3.10.1 Introducere și baza de date pentru detecția punctului de fugă

Metoda prezentată în secțiunea anterioară a fost folosită pentru a genera o bază de date de imagini și de puncte de fugă. Rezultatele au fost verificate și ajustate manual unde algoritmul nu a dat rezultate bune. Algoritmul de detecție a punctului de fugă a fost executat pe imagini din traficul rutier obținute din OpenStreetCam [OpenStreetCam], iar fiecare imagine din baza de date creată are asociată un fișier text care conține numele imaginii (pentru referință) și coordonatele x și y ale punctului de fugă. Baza de date a fost împărțită în două categorii în funcție de unde au fost capturate datele: scenarii urbane (oraș) și scenarii de autostradă. Imaginile sunt de format 4:3 și dimensiune standard VGA: 640 x 480 pixeli. Baza de date inițială conține așadar 2828 imagini, din care 2233 scene de autostradă și 595 cadre din medii urbane.

Determinarea unui punct de fugă precis va oferi unghiuri de înclinare și girație bune ale camerei, care pot fi utilizate pentru generarea matricei de proiecție. Spre deosebire de metodele clasice, în lucrarea [Itu2017] am prezentat un sistem nou de detecție folosind rețele neuronale. În continuare este descrisă toată soluția folosind CNN.

Pentru a îmbunătăți performanța atât la antrenarea rețelei cât și la predicție am ales eliminarea zonei superioare a imaginilor și redimensionarea la o dimensiune finală de 160 x 48 pixeli. Partea de sus a imaginilor conține în general zone de cer, clădiri sau copaci, adică informații care nu sunt relevante pentru procesul de antrenare. Deoarece am șters zona de superioară a imaginilor inițiale folosind un prag fix, au rezultat imagini în care punctul de fugă a fost de asemenea eliminat, iar aceste imagini au fost scoase din baza de date de antrenare. Astfel, a rezultat o bază de date cu un total de 2090 imagini cu scene de autostradă și 594 imagini din orașe, dimensiunea totală fiind de 2684 imagini de antrenare. Aceste date sunt apoi împărțite în 90% date de antrenare și 10% date de testare.

O proprietate a rețelelor neuronale convoluționale este că ele vor produce rezultate bune dacă în procesul de antrenare se folosesc baze de date de dimensiuni mari cu multe informații. Augmentarea datelor reprezintă o modalitate accesibilă de a genera mai multe date de antrenare și se poate face prin mai multe variante, cum ar fi: translatarea imaginilor, scalare și oglindire aleatorie sau prin ajustarea intensității sau a contrastului imaginilor. În această abordare, prin augmentarea datelor am reușit dublarea dimensiunii inițiale prin simpla oglindire orizontală a imaginilor și ajustarea punctelor de fugă, ajungând la 4830 imagini de antrenare și 269 imagini de test. Datele de antrenare au fost împărțite din nou în două: 90%

au rămas date de antrenare și 10% au fost folosite ca și date de validare în momentul antrenării rețelei. Distribuția tuturor punctelor de fugă din baza de date augmentată este ilustrată în Figura 74.

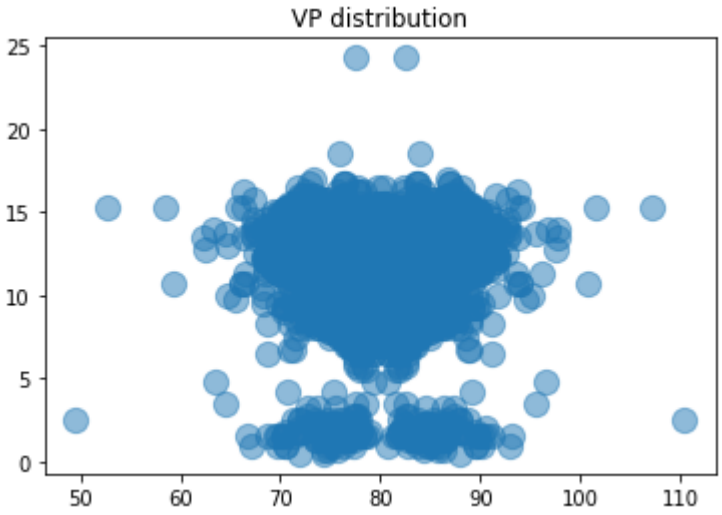


Figura 74. Distribuția punctelor de fugă din baza de date augmentată.

În Figura 74 e poate observa faptul că punctele de fugă sunt situate predominant în zonele din centrul imaginilor de intrare, iar în funcție de înclinația camerei ele pot fi prezente și în partea din centru jos. Punctele laterale și extreme sunt în general punctele de fugă obținute în timpul efectuării virajelor pe drumuri curbate (de exemplu intrări pe autostradă sau ieșiri).

3.10.2 Sistemul de detecție a punctului de fugă folosind rețele neuronale convoluționale

În Figura 75 este ilustrat sistemul propus pentru antrenarea punctului de fugă bazat pe rețele neuronale convoluționale.

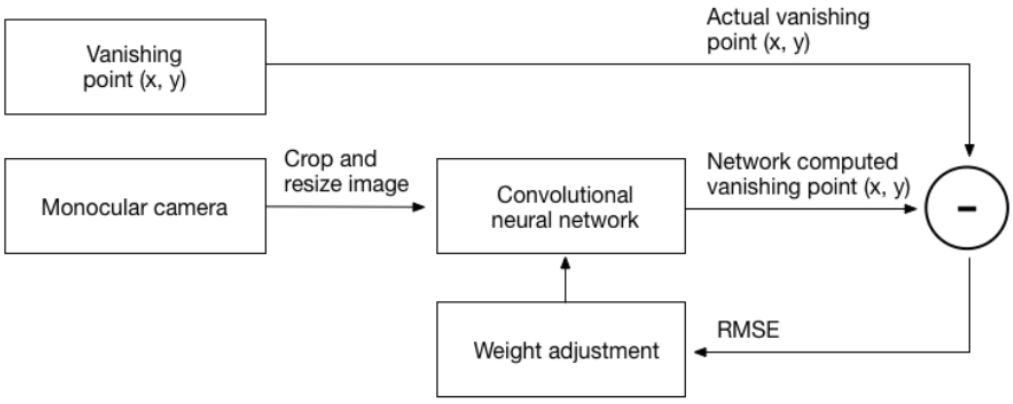


Figura 75. Sistemul de detecție a punctului de fugă bazat pe rețele neuronale.

Acest model a fost inspirat dintr-o lucrare publicată de Nvidia [Bojarski2016], unde folosind rețele neuronale se face predicție de unghi de virare a unui vehicul folosind perechi de imagini și unghiul de virare a roților ca și date de antrenare. Am ales folosirea unei rețele convoluționale neuronale pentru a procesa imaginile redimensionate, achiziționate inițial cu un sistem de camera monocular. Următorul pas este de a obține o predicție a punctului de fugă și compararea acestuia cu punctul de fugă original ("ground truth") din baza de date.

Modelul rețelei neuronale convoluționale este prezentat în Figura 76. Aceste tipuri de rețele au devenit populare în anii recenți datorită unei competiții numite ImageNet, unde în 2012 s-au obținut îmbunătățiri semnificative la clasificarea imaginilor. Unii pași tradiționali cum ar fi: pre-procesarea imaginii, selecția trăsăturilor relevante și extragerea lor, iar apoi clasificarea acestor trăsături, sunt eliminați sau înlocuiți complet de rețele neuronale convoluționale.

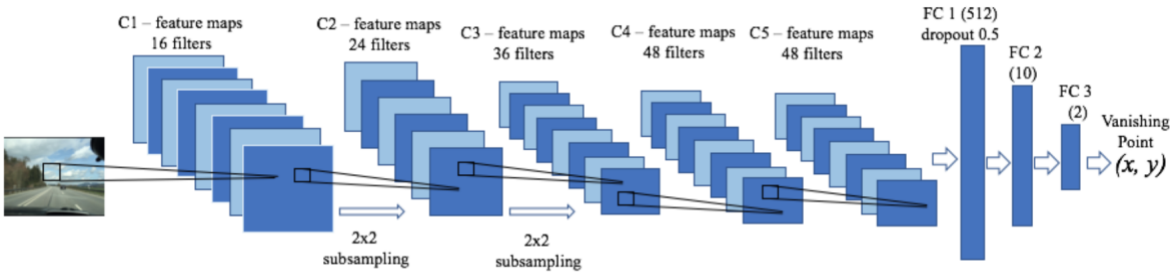


Figura 76. Modelul rețelei neuronale convoluționale.

Structura rețelei este la fel cu cea din lucrarea [Krizhevsky2012], în ideea în care am utilizat tot 5 niveluri convoluționale urmate de 3 straturi complet conectate, dar numărul de nuclee de convoluție este diferit. Prin comparație, rețeaua din lucrarea [Bojarski2016] folosește tot 5 niveluri de convoluție, dar 4 straturi complet conectate. Astfel, în implementarea mea datele de intrare sunt normalizate și apoi trecute prin 5 niveluri convoluționale (C1-C5) cu un număr diferit de filtre (nuclee de convoluție) cu dimensiune variabilă. Ultimele niveluri ale rețelei sunt reprezentate de FC1-FC3 care sunt denumite niveluri complet conectate ("fully-connected"), echivalentul unor rețele neuronale clasice unde în funcție de neuronii de intrare se activează o anumită ieșire. De altfel, ultimul FC3 cu cele 2 valori ale neuronilor reprezintă valorile coordonatelor x și y ale punctului de fugă prezis. Straturile din rețeaua propusă sunt organizate astfel:

- primul strat reprezintă normalizarea lotului de date (B)
- al doilea strat este de convoluție (C1) cu 16 filtre în total cu nucleu (kernel) de dimensiune 3 x 3
- al treilea strat este o convoluție (C2) cu 24 filtre cu kernel de dimensiune 3 x 3
- al patrulea strat este tot o convoluție (C3) cu 36 filtre cu kernel de dimensiune 3 x 3
- al cincilea strat este o convoluție (C4) cu 48 filtre cu kernel de dimensiune 3 x 3
- al șaselea strat reprezintă încă o convoluție (C5) cu 48 filtre și kernel de dimensiune 3 x 3

- al șaptelea strat este primul de tip “fully-connected” (FC1) care conține 512 neuroni, urmat de operația de “dropout” (setată cu probabilitate de 0.5) și cu ReLU ca și funcție de activare

- al optulea strat este încă unul de tip “fully connected” (FC2) cu 10 neuroni și cu funcția de activare ReLU

- ultimul strat al rețelei reprezintă un strat de tip “fully connected” (FC3) cu 2 neuroni care vor fi de fapt predicțiile rețelei

3.10.3 Antrenarea rețelei

Operația de normalizare a lotului de date a fost folosită pentru a obține o antrenare mai rapidă și pentru a crește precizia predicțiilor. Cei 2 neuroni din ultimul strat reprezintă de fapt valorile numerice ale coordonatei x și y ale punctului de fugă. Straturile convoluționale conțin filtre cu ponderi ajustate în așa fel încât ele vor funcționa similar cu algoritmi de detecție a liniilor. Răspunsul obținut din primul nivel convoluțional al rețelei (cu 16 filtre cu kernel de dimensiune 3 x 3) este ilustrat în Figura 77.

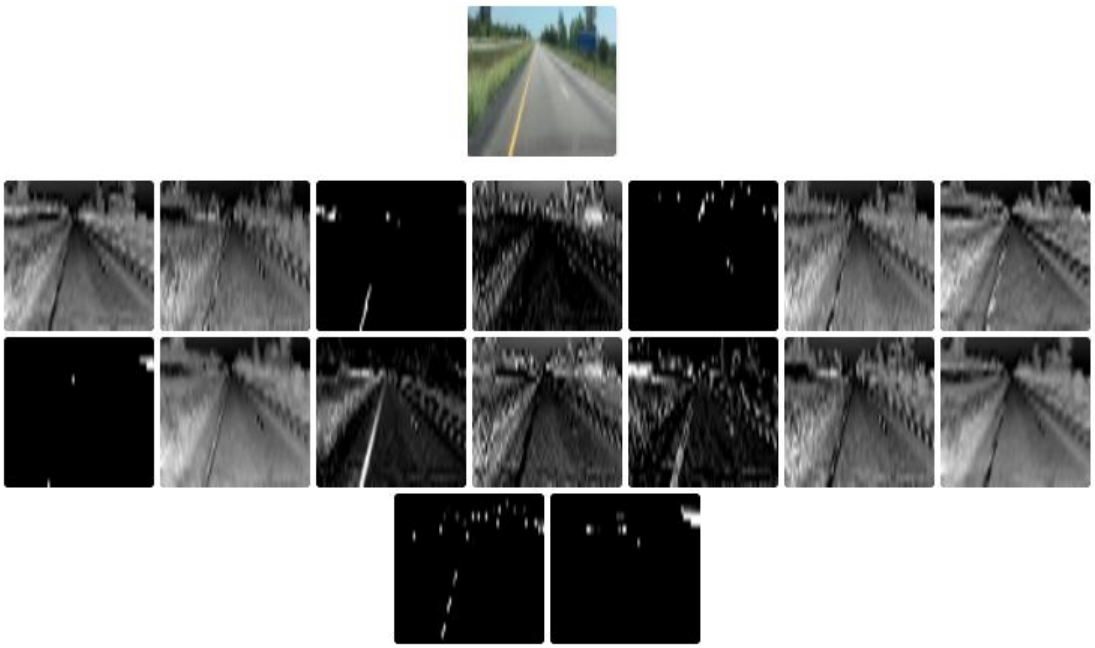


Figura 77. Răspunsul primului nivel convoluțional (C1).

Am antrenat rețeaua pentru un total de 20 de epoci, folosind o dimensiune a lotului de 256 imagini. Evoluția erorii din funcția de minimizare este prezentată în Figura 78, unde se poate observa faptul că după 5 epoci funcția de minimizare a erorii converge, iar antrenarea poate fi oprită mai repede.

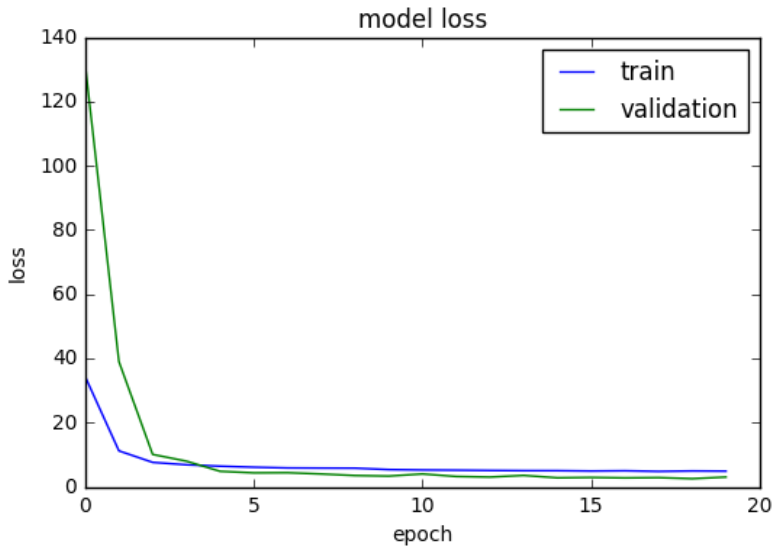


Figura 78. Funcția de minimizare a erorii, evoluție în timpul antrenării.

Rețeaua neuronală a fost antrenată folosind funcția de minimizare a erorii “RMSE” (“root mean squared error”) care în cazul de față reprezintă distanță în pixeli dintre punctul prezis de rețea și punctul de fugă real din baza de date. Valorile ponderilor din straturile convoluționale sunt ajustate automat în procesul de minimizare a erorii RMSE prin folosirea algoritmului “gradient descent”. Antrenarea rețelei s-a realizat pe computere folosind plăci video (GPU), unde antrenarea unei epoci durează aproximativ 2 secunde, în timp ce pe un sistem folosind exclusiv procesorul (CPU) procesul durează aproximativ 20 de secunde. Așadar, folosirea unei plăci video reduce timpii de antrenare, dar sistemul este capabil de a oferi predicții în timp real și folosind doar un CPU.

După antrenare, rețeaua poate oferi predicții de VP pe imagini noi care nu au fost văzute în procesul de antrenare și validare. Un astfel de rezultat este ilustrat în Figura 79.

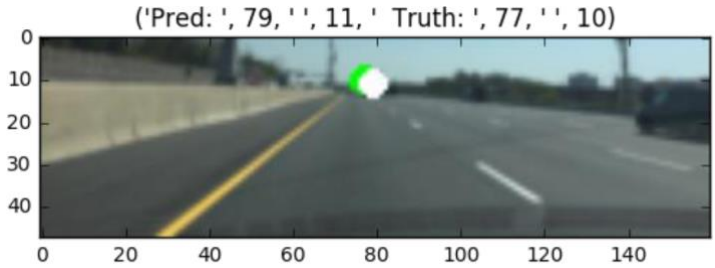


Figura 79. Exemplu de predicție de punct de fugă. Punctul verde reprezintă valoarea reală, iar punctul alb reprezintă predicția.

3.10.4 Calculare unghi înclinare și girație din punctul de fugă

Revenind la partea de calibrare automată, putem calcula unghiurile de înclinare (“pitch”) și girație (“yaw”) folosind coordonatele punctului de fugă și ecuațiile 105-107.

$$pitch = original\ pitch + \Delta pitch \quad (105)$$

În ecuația 88 *Δpitch* este obținut astfel:

$$\Delta pitch = atan\left(\frac{\Delta y}{focal\ length}\right) \quad (106)$$

$$\Delta y = original\ vp.y - vp.y \quad (107)$$

Calcularea diferenței pentru ajustarea unghiului de înclinare ("pitch") folosind coordonata y a punctului de fugă original (inițial) și cel calculat este reprezentată în ecuațiile 89-90.

Prin utilizarea coordonatei x a punctului de fugă, putem ajusta formulele și calcula și unghiul de girație. Valorile originale (de exemplu *"original pitch"* din ecuația 105) pentru cele două unghiuri (cel de înclinare și cel de girație) sunt cele inițiale care pot fi calibrate sau dacă nu există informații despre poziția inițială se pot presupune a fi egale cu 0. Diferențele de unghiuri de înclinare și girație calculate din diferențele de VP vor deveni valorile efective ale unghiurilor camerei. Valorile pentru coordonatele x și y ale punctului de fugă original (*"original vp.y"* din ecuația 107) sunt calculate prin proiecția unui punct 3D situat la distanță îndepărtată pe axa Z în planul 2D al imaginii folosind matricea de proiecție cu unghiul de înclinare și de girație original. De asemenea, în ecuația 106 distanța focală este exprimată în pixeli și modul de calculare a fost prezentat în secțiunea 3.5.

Unghiul de rotire ("roll") nu afectează punctul de fugă și nu poate fi calculat din coordonatele punctului de fugă. În lipsa unor informații adiționale sau alți senzori, acest unghi a fost presupus a fi egal cu 0.

Unghiurile de înclinare și girație sunt apoi folosite pentru a calcula matricea extrinsecă a camerei, care este apoi folosită pentru determinarea matricei de proiecție care oferă transformarea punctelor 3D din scenă în puncte 2D din spațiul imaginii. Aceasta matrice de proiecție se poate folosi și pentru a obține imagini cu efectul de perspectivă eliminat, după cum am prezentat în capitolul 3.6.

3.10.5 Testare și validare rezultate

Pentru validarea metodei de calcul a VP, am utilizat setul de date de validare și metrică de calcul prezentată anterior: RMSE, dar și o metrică asemănătoare numită "NormDist" care reprezintă de fapt valoarea RMSE împărțită la diagonala imaginii, mai specific: RMSE normalizat. Formula de calcul este prezentată în ecuația 108.

$$NormDist = \frac{||VPP - VPG||}{diag} \quad (108)$$

Această metrică a mai fost utilizată în literatura de specialitate în [Moghadam2012] sau [Wu2016]. O analiză asupra setului de date de oraș și autostrada este prezentată în Tabel 10.

Tabel 10. Calcularea erorii de predicție pe setul de date de validare.

Metrică eroare	Oraș	Autostradă
NormDist	0.05072	0.02612
RMSE (pixeli)	8.47350	4.36344

Am calculat de asemenea și eroarea NormDist pe fiecare imagine din baza de date de validare, iar rezultatele sunt prezentate în Figura 80.

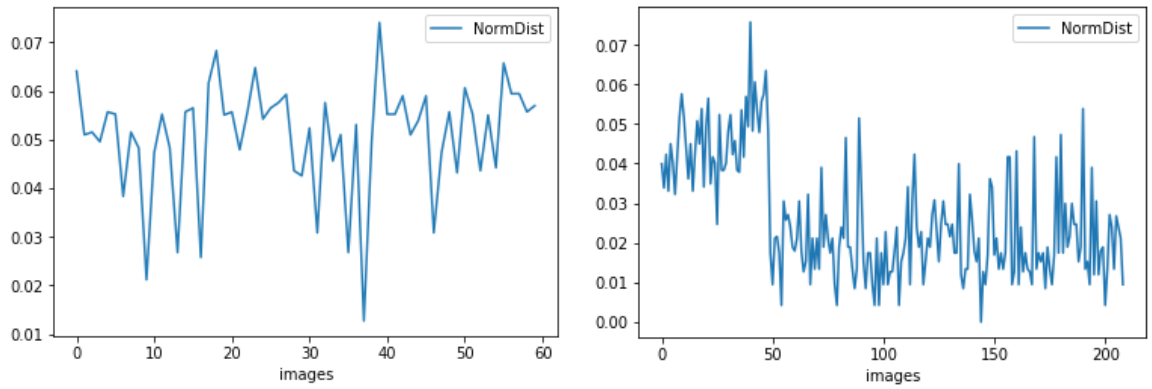


Figura 80. NormDist calculat pe baza de date de validare: 59 imagini din oraș (sus) și 209 imagini de pe autostradă (jos).

Tabel 11 reprezintă o analiză a metricii RMSE, calculată pe toată baza de date. Am calculat numărul de puncte cu eroare mai mică decât 2, cu valori cuprinse între 2 și 3, între 3 și 5, între 5 și 10 și valori mai mari decât 10. Tabelul reprezintă rezultatele sub formă de procent din totalul numărului de imagini din baza de date.

Tabel 11. Analiza detecției punctului de fugă pe cele două baze de date: oraș și autostradă.

Oraș (594 imagini)				**Autostradă (2090 imagini)**			
interval	**număr**	**interval**	**procent**	**interval**	**număr**	**interval**	**procent**
< 2	68	< 2	11.4 %	< 2	1122	< 2	53.6 %
[2 - 3)	65	< 3	22.3 %	[2 - 3)	487	< 3	76.9 %
[3 - 5)	215	< 5	58.5 %	[3 - 5)	340	< 5	93.2 %
[5 - 10)	238	**< 10**	**98.6 %**	[5 - 10)	139	**< 10**	**99.9 %**
> 10	8	> 10	1.34 %	> 10	2	> 10	0.09 %

Din Tabel 11 se poate observa în mod clar că setul de date cu scene din mediul urban are o incertitudine mai mare, în timp ce predicțiile din setul de pe autostradă sunt mai precise: peste 76% din predicții au valoarea RMSE mai mică decât 3, iar 93% au RMSE sub 5. Incertitudinea mai mare din setul de date din oraș este cauzată de prezența unui număr mai mare de vehicule și de alte obstacole din fața vehiculului, ceea ce afectează vizibilitatea suprafeței drumului și a marcajelor rutiere.

Uneori, punctul de fugă prezis de CNN este mai bun decât cel obținut prin metode clasice, după cum se poate vedea în Figura 81. Acest lucru este cauzat de prezența obstacolelor suplimentare din fața vehiculului care pot influența rezultatele algoritmilor tradiționali.

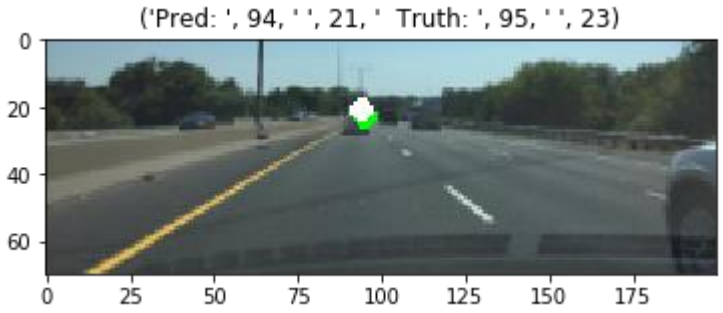


Figura 81. Exemplu în care punctul de fugă calculat de CNN (cercul alb) este mai bun decât cel obținut prin algoritmi tradiționali (cercul verde).

În Figura 82 sunt prezentate mai multe variante de predicții ale rețelei neuronale. Spre deosebire de exemplul precedent, uneori rezultatul rețelei nu este corect (de exemplu imaginea din mijloc din ultima linie de pe coloana a doua).

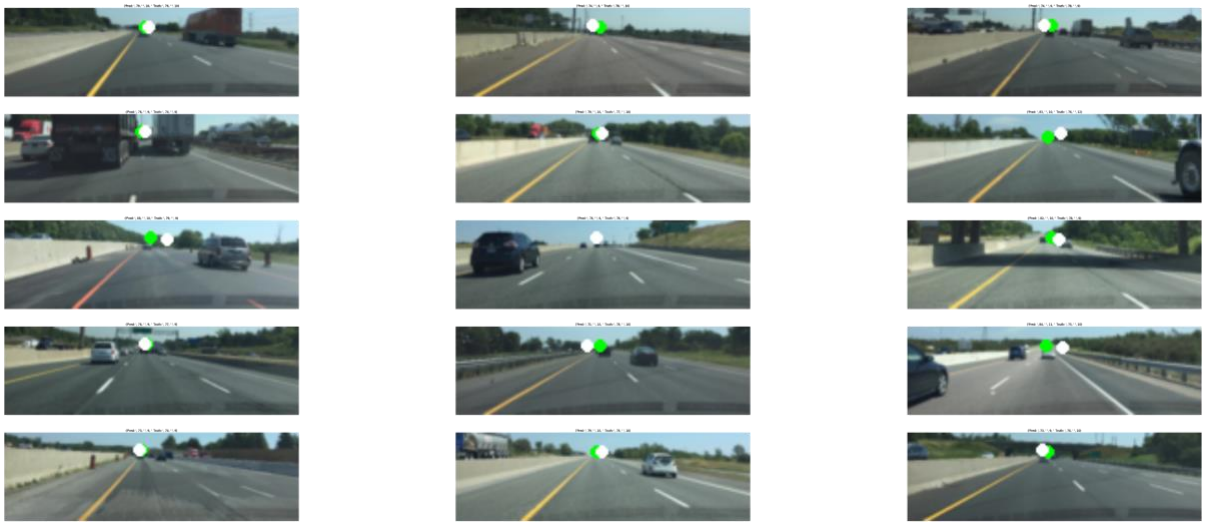


Figura 82. Exemple între punctul de fugă din predicția rețelei (cercul alb) și punctul de fugă inițial (cercul verde).

Timpul de procesare pentru a genera o predicție pentru un punct de fugă este aproximativ 1.7 milisecunde pe imagine folosind un sistem desktop cu procesor Intel i7 6700K și o placă video Nvidia GTX 1080 Ti. Am testat pe aceleași date pe un laptop cu procesor Intel i7 3615QM, iar timpul necesar unei predicții a fost aproximativ 100 milisecunde. Din testele efectuate se poate trage concluzia că un astfel de sistem poate oferi rezultate necesare funcționarii în timp real dacă se utilizează un procesor modern și capabil. Algoritmul prezentat în secțiunea 3.9 bazat pe gradienții din imagine necesită un timp de procesare de aproximativ 30-40ms pe un sistem i7 6700K, ceea ce ar justifica folosirea unei abordări bazate pe CNN mai ales dacă sistemul dispune de o unitate GPU.

Metoda pentru folosirea unei rețele convoluționale neuronale pentru detecția punctului de fugă a fost diseminată în lucrarea [Itu2017]. Totuși, în urma publicării lucrării am ales să extind această abordare și inclusiv să folosesc rețeaua direct pe dispozitive mobile inteligente, de aceea am ales să implementez pe un telefon mobil Android.

3.11 Determinare punct de fugă folosind două rețele convoluționale

Versiunea nouă a sistemului de predicție a punctului de fugă folosește două rețele neuronale. O rețea este folosită pentru a oferi predicții de-a lungul axei x (coordonatele x din imagine), iar a doua rețea oferă predicții pentru axa y. Prin folosirea acestei abordări am reușit să îmbunătățesc acuratețea sistemului. De asemenea cele două rețele se pot exporta și ocupă mai puțin spațiu pe disk decât o singură rețea mare, ceea ce facilitează portabilitatea și implementarea pe un dispozitiv mobil. Antrenarea rețelelor a rămas la fel, pe sisteme de tip desktop, dar predicția se poate face și pe dispozitive mobile Android folosind software-ul TensorFlow în variantă pentru mobile.

3.11.1 Augmentare bază de date

Datele inițiale de antrenare și de testare erau compuse din 2544 de imagini de intrare, fiecare având și un fișier text asociat care conține coordonatele punctului de fugă. Aceste date erau apoi împărțite în 90% date de antrenare și 10% date de testare. În această abordare, am reușit o extindere a bazei de date la peste 110.000 imagini prin folosirea mai agresivă a tehnicilor de augmentare. O primă îmbunătățire o reprezintă folosirea unei ferestre glisante si decuparea imaginilor în funcție de rețeaua folosită pentru antrenare. De exemplu, pentru o generalizare cât mai bună pentru CNN-ul care prezice coordonatele de-a lungul axei X, am ales să folosesc o fereastră glisantă de-a lungul axei X, astfel încât am obținut peste 20 de imagini dintr-o singură imagine sursă. Am folosit un algoritm simplu de tip fereastră glisantă (“sliding window”) pentru generarea imaginilor noi. Figura 83 ilustrează rezultatele augmentării unei imagini. Aceeași tehnică am folosit-o și pentru a genera imagini de-a lungul axei y, pentru a doua rețea care prezice coordonatele Y ale punctului de fugă. Acest număr de imagini generate se poate seta dinamic și am ales un număr maxim de 20. Aceste imagini sunt apoi salvate pe disk și încărcate în loturi mai mici pentru a nu umple memoria RAM în timpul antrenării.

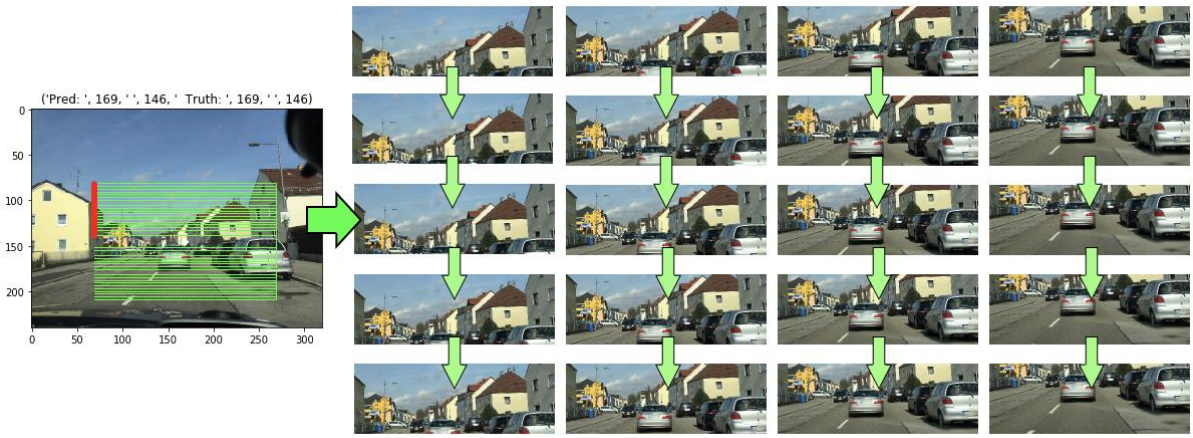


Figura 83. Tehnica “sliding window” pentru generarea noilor imagini de-a lungul axei X.

Prin acest proces de generare a noilor imagini pentru cele două axe ale sistemului de coordonate vor rezulta baze de date mult mai bine echilibrate, ceea ce va ajuta în procesul de antrenare. Figura 84 reprezintă histograma valorilor coordonatelor X ale punctului de fugă din baza de date augmentată. În imagini de dimensiune 200 x 70 pixeli se poate observa distribuția aproape uniformă a punctelor de-a lungul înălțimii imaginii, cu o pondere mai mare în intervalul [20, 30] pe coordonata Y și [80, 120] pe coordonata X unde sunt predominante punctele de fugă în imagini de asemenea dimensiuni și din această bază de date.

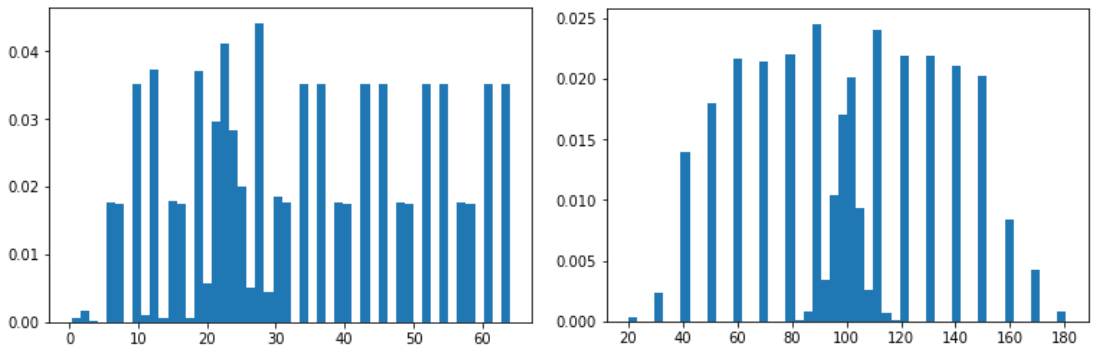


Figura 84. Histograma punctelor de fugă de pe coordonata y (stânga) și coordonată x (dreapta).

Imaginile de intrare pentru rețele sunt pe trei canale, în spațiul de culoare RGB. Astfel, o altă tehnică de a obține o predicție mai bună este de a introduce câte o ocluzie aleatoare în imagini. Acesta este adăugată pe toate cele trei canale, mai precis se generează valori aleatorii pentru intensitatea pixelilor care sunt adăugați într-o zonă aleatoare în imagine într-un dreptunghi de dimensiuni aleatorii, dar nu mai mare decât 20% din suprafața totală a imaginii. Figura 85 reprezintă un exemplu de ocluzie aleatoare în două imagini din baza de date.

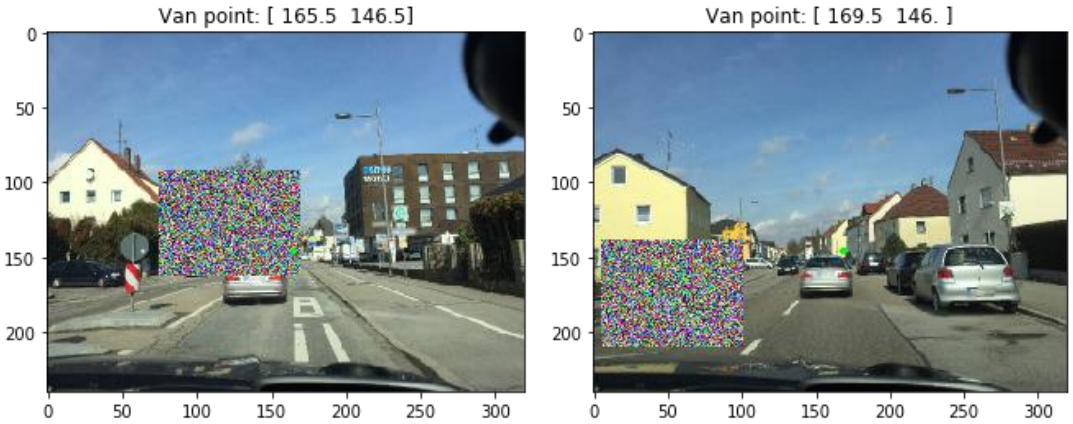


Figura 85. Exemplu de augmentare cu dreptunghiuri care conțin ocluzii adăugate aleator în imagini, în zone aleatoare.

Un total de cinci imagini cu zgomot sunt generate dintr-o imagine sursă. O altă variantă de augmentare este oglindirea imaginilor pe orizontală, astfel încât se poate obține ușor o dublare întregii baze de date.

3.11.2 Arhitectura celor două rețele convoluționale neuronale și antrenarea lor

Arhitectura și structura nivelurilor celor 2 rețele este la fel cu cea din capitolul 3.10, cu excepția ultimului strat “fully-connected” care conține un singur neuron. Prima rețea a fost

antrenată pentru a oferi predicții de coordonate X ale punctului de fugă, în timp ce a doua rețea oferă predicții ale coordonatei pe axa Y. Dimensiunea imaginilor de intrare este mărită la 200 x 70 pixeli. Fiecare rețea este antrenată pentru un total de 100 epoci și folosind o dimensiune a lotului de 256. Rata de învățare inițială este de 0.0001 și este ajustată automat în timpul antrenării datorită folosirii optimizatorului Adam.

3.11.3 Rezultate

Tabel 12 prezintă o comparație între erorile NormDist între cele două variante de detecție de punct de fugă: soluția inițială (V1 - capitolul 3.10) care oferă predicții asupra ambelor coordonate simultan și soluția cu două rețele neuronale diferite (V2 - din această secțiune), antrenată pe mai multe date augmentate. Comparația a fost făcută pe aceeași bază de date de validare pentru scenariul de oraș și autostradă.

Tabel 12. Comparație între cele două variante.

Metrică eroare	**Oraș (V1)**	**Oraș (V2)**	**Autostradă (V1)**	**Autostradă (V2)**
NormDist	0.05072	**0.029725**	0.02612	**0.012905**

Din Tabel 12 se poate observa o îmbunătățire clară a soluției cu două rețele neuronale, iar din testele efectuate diferența cea mai semnificativă este asupra prezicerii punctului de coordonată y, corespunzător liniei de orizont din care poate obține unghiul de înclinare. Analizând separat rezultatul fiecărei rețea și calculând erorile doar în raport cu coordonata corespunzătoare din baza de date inițială obținem valorile din Tabel 13.

Tabel 13. Analiza individuală a celor 2 rețele.

Metrică eroare	**Rețea Y**	**Rețea X**
RMSE pixeli (Oraș)	1.36666	8.65000
RMSE pixeli (Autostradă)	1.25339	3.39819
NormDist (Oraș)	0.00644	0.04082
NormDist (Autostradă)	0.00591	0.01603

Din Tabel 13 se poate observa faptul că eroarea este mai mică pe axa y (înălțimea imaginii) decât pe lățime (axa x), ceea ce înseamnă un unghi de înclinare (“pitch”) al camerei mai precis decât cel de girație (”yaw”).

Eroarea NormDist pentru ambele axe calculată fiecare imagine din setul de date de validare este ilustrată în Figura 86.

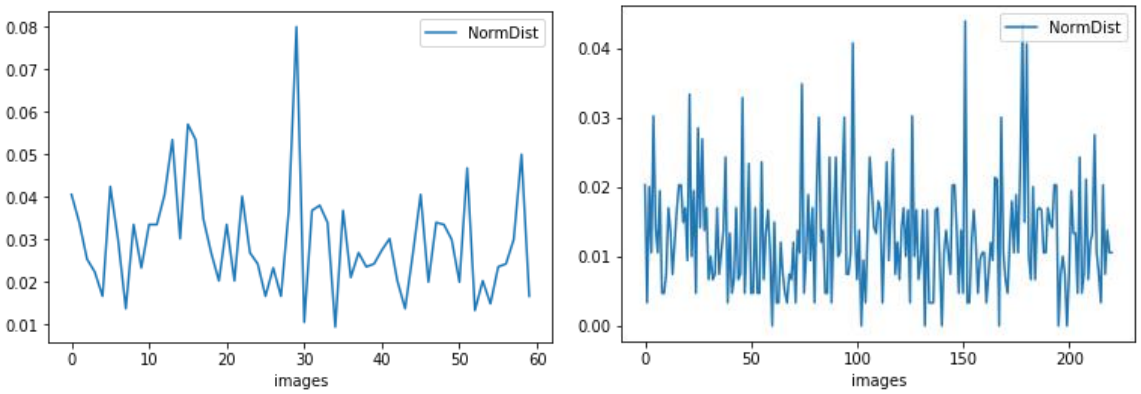


Figura 86. NormDist calculat pe baza de date de validare: imaginile din oraș (stânga) și imagini de pe autostradă (dreapta).

3.11.4 Implementare pe dispozitive mobile Android

Sistemul acesta permite implementarea unei soluții de calibrare automată folosind punctul de fugă, direct pe dispozitive mobile inteligente. Având punctul de fugă putem genera imagini ale scenei în care efectul de perspectivă este eliminat direct pe dispozitive mobile.

În Figura 87 sunt ilustrate două exemple de predicții a punctului de fugă obținut din cele două rețele neuronale unde predicțiile sunt ilustrate cu un cerc alb, în timp ce punctul de fugă inițial este cu verde.

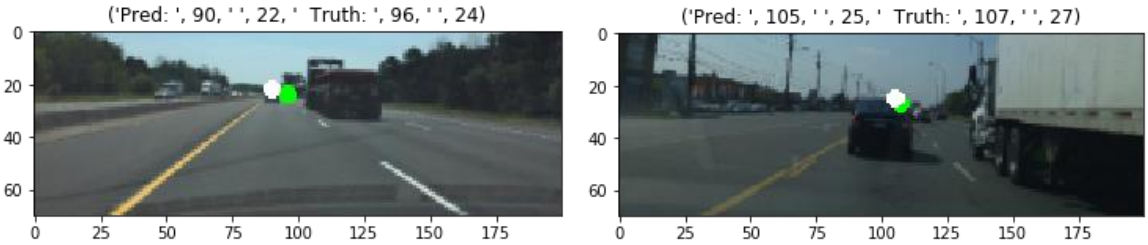


Figura 87. Predicțiile sistemului îmbunătățit cercul alb și punctul de fugă inițial (cercul verde).

Figura 88 prezintă modul de lucru pentru testare și dezvoltare al algoritmului într-un mediu controlat, unde dispozitivul mobil este amplasat într-un punct fix și orientat spre un ecran unde rulează o secvență de test.

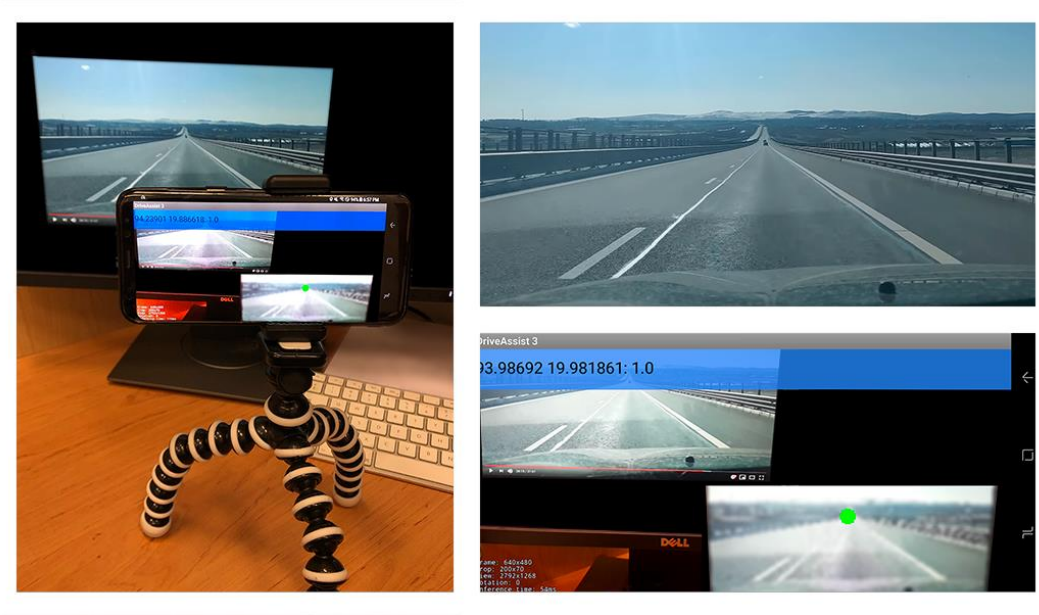


Figura 88. Calculare punct de fugă direct pe dispozitivul mobil. În dreapta sus este imaginea sursă, iar în dreapta jos este o captură de ecran de pe dispozitivul mobil.

Baza de date inițială pentru detecția punctului de fugă a fost publicată în [Itu2017] și a fost deja utilizată și de alți cercetători [Sistu2019] și [Yoon2019].

3.12 Calibrare automată din traiectoria vehiculelor

Majoritatea tehnicilor de calibrare automată a camerelor folosesc trăsături obținute din marcajele de pe drum (de exemplu marcajele benzilor de circulație). În anumite situații, aceste marcaje ar putea fi obstrucționate de prezența altor vehicule din scenă sau ele pot fi acoperite, murdare sau chiar inexistente pe anumite porțiuni de drum. În acest capitol voi prezenta o abordare care nu se bazează pe marcaje, ci folosește exclusiv vehicule. Ideea principală este de a folosi o rețea convoluțională neuronală pentru a detecta vehicule în scenă. Această rețea nu necesită o calibrare și va genera predicții pentru vehicule, sub forma unor chenare de încadrare (“bounding box”) în imagine. Folosind aceste informații și anumite constrângeri geometrice, se poate obține o calibrare a unghiurilor de rotație a camerei, dar și a înălțimii camerei față de sol.

Soluția de calibrare propusă nu necesită nici o acțiune din partea utilizatorului aplicației, în afară de a conduce vehiculul în trafic. Există doar o singură constrângere, aceea de a avea un număr suficient de vehicule observate în scenă pentru a putea analiza mai bine informațiile și de a putea reduce erorile la măsurare sau detecție a vehiculelor.

3.12.1 Detecția vehiculelor folosind rețele neuronale convoluționale

Principalele componente utilizate în acest proces de calibrare sunt marginile obiectelor detectate. Pentru a le extrage, este nevoie de un detector suficient de rapid încât să ofere rezultate în timp real pe resursele limitate ale dispozitivelor mobile (Figura 89) și să nu necesite o calibrare, ceea ce înseamnă că va oferi detecții ale obiectelor indiferent de dimensiunea, orientarea și poziția lor în imagini.

Figura 89. Dispozitivul mobil este poziționat în parbriz și detectează doar vehicule.

Pentru detecție, am ales o implementare bazată pe rețele neuronale convoluționale, antrenată în prealabil pe un sistem desktop folosind perechi de imagini și obstacole, exprimate ca liste de coordonate ale chenarelor dreptunghiulare care le înconjoară. Am folosit colecția de algoritmi pentru detecție de obstacole din TensorFlow [Tensor], iar pentru a simplifica întreg procesul de detecție am ales să re-antrenez detectorul “single shot multibox detector” (SSD) [Wei2016] oferind predicții dintr-un singur pas. Pentru extragerea de trăsături relevante detectorul SSD va folosi rețeaua MobileNet [Howard2017]. Cu alte cuvinte, rețeaua MobileNet va extrage prin convoluții repetate trăsăturile relevante, iar rețeaua SSD este folosită pentru clasificare și oferă predicții pentru chenare de încadrare. Arhitectura MobileNet se pretează foarte bine pentru dispozitive mobile, fiind dezvoltată inițial de Google exact pentru acest scop: funcționarea optimă pe resurse hardware limitate.

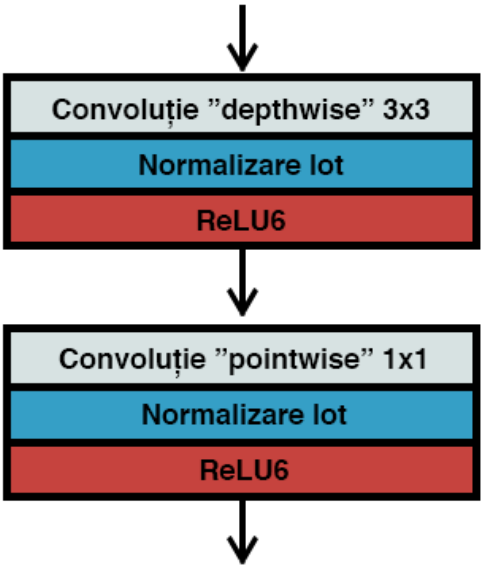


Figura 90. Un nivel cu o convoluție "depthwise" urmată de o convoluție 1 x 1 în rețeaua MobileNet.

Arhitectura rețelei MobileNet are un prim nivel de convoluție, în timp ce restul nivelurilor (13 în total) sunt create folosind convoluții separabile pe adâncime. O convoluție normală este împărțită (factorizată) într-o convoluție pe adâncime (cu un nucleu 3 x 3) și o convoluție 1 x 1 (numită și "pointwise"). În MobileNet, un singur filtru este aplicat inițial fiecărui canal de intrare de către convoluția pe adâncime ("depthwise convolution"), în timp ce convoluția 1 x 1 va combina rezultatele (Figura 90). Timpul redus de execuție este realizat de această factorizare. Între nivelurile rețelei nu este operația de agregare ("pooling"), în schimb unele niveluri conțin un pas ("stride") de 2 la operațiile de convoluție pentru a reduce dimensiunile spațiale ale datelor de intrare. Fiecare strat al rețelei este urmat de o normalizare a lotului și se folosește funcția de activare ReLU6 pentru neliniaritate. Nivelul final nu folosește ReLU6 și este conectat la o operație "softmax" pentru clasificarea finală. Funcția de activare ReLU6 este o variație a ReLU care previne activările să aibă valori prea mari:

$$y = min((max(0, x), 6) \quad (109)$$

ReLU6 funcționează mai eficient și robust mai ales pe dispozitive mobile. Datorită folosirii convoluțiilor separabile pe adâncime, rețeaua MobileNet obține aproximativ aceeași acuratețe cu rețele convoluționale tradiționale, dar cu o reducere a timpului de execuție de până la 10 ori.

Am ales folosirea rețelei SSD MobileNet care a fost antrenată inițial pe baza de date "Common Objects in Context" (COCO) publicată în [Lin2014]. Această rețea a fost apoi reantrenată folosind baza de date de obstacole "KITTI Vision" [Geiger2012] și o bază de date de la Udacity [Udacity2019], unde am limitat clasele de obstacole să cuprindă doar autoturisme, deoarece doar acestea sunt utilizate în procesul de calibrare. Antrenarea este realizată folosind tehnică "gradient descent" și prin folosirea a două funcții de cost: una pentru detecție și încă una pentru clasificare. Funcția de cost "smooth L1" este utilizată pentru localizare și funcția de cost "cross entropy" ponderată este utilizată pentru clasificare.

În sistemul propus, am folosit imagini de intrare de dimensiune fixă 300 x 300 pixeli. În procesul de antrenare, software-ul TensorFlow pentru detecția de obstacole folosește ceea ce se numește “online hard-negative mining”. Practic în momentul antrenării se utilizează ca exemple negative pentru rețea acele chenare de încadrare (“bounding boxes”) care au valoarea cea mai mică pentru funcțiile de cost.

Figura 91. Rezultatul predicției de vehicule din rețeaua neuronală convoluțională.

Figura 91 ilustrează o captură de ecran direct de pe dispozitivul mobil, unde se pot observa detecțiile de vehicule într-o scenă. Se poate remarca performanța rețelei convoluționale neuronale chiar și în condiții de iluminare proastă (Figura 92), cu soarele în față (rezultând imagini întunecoase și cu un contrast foarte slab).

Figura 92. Detecția vehiculelor (captură de ecran direct de pe dispozitivul mobil).

Pe dispozitivul mobil, rețeaua neuronală generează în timp real predicții de vehicule în imagini. De asemenea, am folosit algoritmul de urmărire a obstacolelor din software-ul

TensorFlow pentru Android. Algoritmul de urmărire din TF utilizează trăsăturile FAST [Rosten2010] generate de fluxul optic calculat folosind metoda “pyramidal Lucas Kanade” [Lucas1981]. Mediana deplasării trăsăturilor este analizată în fiecare cadru, iar algoritmul de urmărire a contururilor va renunța la un contur când corelația cu detecția originală este sub un nivel de prag fix (determinat experimental). Conturul actual care este urmărit se poate actualiza dacă nouă detecție de contur (din rețeaua neuronală) are o suprapunere mare (un prag fix).

Folosirea algoritmului de urmărire are atât avantaje cât și dezavantaje. Rezultatele vor fi mai stabile, dar ar putea genera și date false din cauza inerției actualizării. Metodologia prezentă de calibrare poate funcționa atât cu algoritmul de urmărire, cât și fără. Încadrarea obiectelor nu trebuie să fie perfectă sau completă. Metodologia de calibrare necesită doar un set reprezentativ din punct de vedere statistic de detecții de vehicule, pentru diferite distanțe și pentru mai multe dimensiuni ale vehiculului.

3.12.2 Filtrul Kalman extins pentru estimarea parametrilor extrinseci ai camerei

Algoritmul de calibrare va estima înălțimea camerei față de sol și unghiurile de rotație a camerei: “pitch” și “yaw”. Primii parametri care pot fi estimați sunt: înălțimea camerei față de sol (h) și unghiul de înclinare θ. Parametri intrinseci ai camerei sunt cunoscuți: punctul principal este considerat centrul imaginii, iar distanța focală a camerei este citită din dispozitivul mobil (după cum a fost descris în capitolul 3.5).

Matricea intrinsecă este cea exprimată în ecuația 65 (capitolul 3.2) unde parametri *h* și *w* reprezintă înălțimea și respectiv lățimea imaginii exprimată în pixeli, iar distanța focală *f* este exprimată tot în pixeli. Poziția camerei în sistemul de coordonate al lumii este determinată de înălțimea față de sol (ecuația 70 din capitolul 3.3). Matricea de rotație între cameră și scenă (lume) depinde în acest pas doar de unghiul de înclinare (ecuația 71 din capitolul 3.3). Matricea de proiecție, care va proiecta un punct 3D (X, Y, Z) din sistemul de coordonate al lumii într-un punct al planului imaginii (u, v) este calculată folosind ecuația 77 din capitolul 3.3.

Având două linii în imagine: v_1 și v_2 și o lățime standard a vehiculului *L*, putem determina lățimea vehiculului în planul imaginii pe aceste două linii, presupunând că vehiculul este pe drum și este observat din spate, într-o poziție centrală în imagine. Acești parametri: liniile v_1, v_2 și dimensiunea în pixeli pentru lățimea unui vehicul *L* sunt presupuse a fi fixe, astfel încât lățimea unui vehicul în spațiul imaginii va depinde doar de parametri *h* și θ, care vor forma vectorul *X* (cel care trebuie estimat):

$$\mathbf{X} = \begin{pmatrix} h \\ \theta \end{pmatrix} \tag{110}$$

Algoritmul pentru estimarea lățimii unei mașini având liniile v_1 și v_2 este următorul:

1. Presupunând că vehiculul se află pe drum, într-o poziție centrală în imagine, extremele laterale ale vehiculului sunt definite de punctele (-*L*/2, 0, *Z*) și (*L*/2, 0, *Z*), *L* fiind lățimea mașinii iar *Z* fiind distanța față de cameră. Am ales 2 distanțe Z_1 și Z_2 și am generat patru puncte 3D: două în stânga (P1, P3) și două în dreapta axei Z (P2, P4) pe planul drumului.
2. Cele patru puncte sunt proiectate în planul imaginii folosind matricea de proiecție *P*. Punctele rezultate sunt următoarele: P1'($u_{L,1}$, $v_{L,1}$), P3'($u_{L,2}$, $v_{L,2}$), P2'($u_{R,1}$, $v_{R,1}$) și P4'($u_{R,2}$, $v_{R,2}$).
3. Cele două linii formate de punctele din zona stângă și punctele din zona din dreapta, se vor intersecta cu liniile orizontale definite de coordonatele pe linie: v_1 și v_2. Intersecția acestor puncte vor avea coordonatele pe coloană: $u_{L,1}$, $u_{L,2}$, $u_{R,1}$ și $u_{R,2}$.
4. Se calculează cele două lățimi: $w_1=u_{R,1}-u_{L,1}$ și $w_2=u_{R,2}-u_{L,2}$.

Figura 93 reprezintă vizual modul de calculare a lățimii unei mașini (algoritmul descris mai sus):

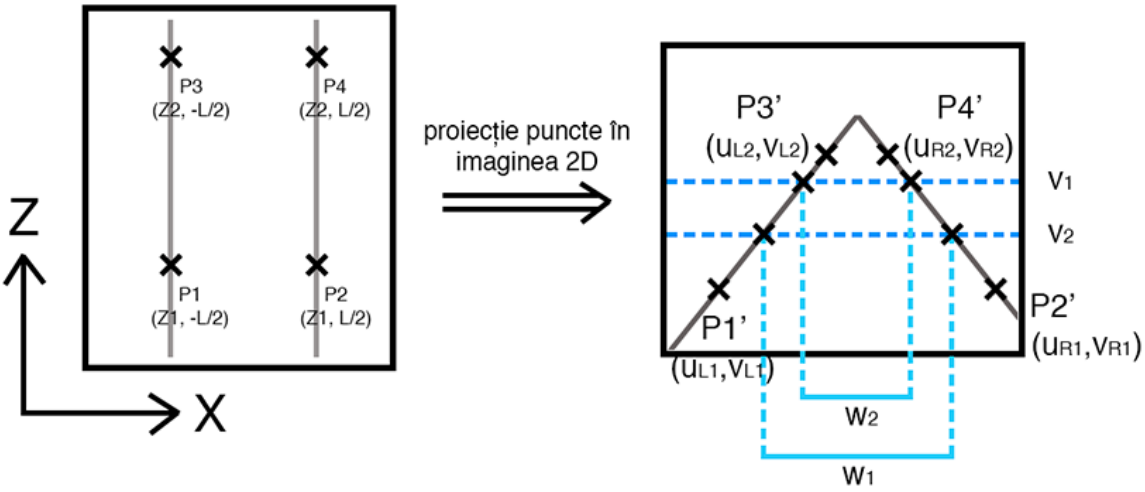


Figura 93. Determinarea celor două lățimi w_1 și w_2.

Astfel, se poate defini funcția *g*, de proiecție a lățimii:

$$g_{v1,v2,L}(X) = \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} \quad (111)$$

Ieșirea funcției *g* se poate interpreta ca fiind vectorul de măsurare *Z*, astfel încât problema de calibrare se poate formula ca și o problemă de estimare: cunoscând vectorul de măsurare *Z* și funcția de măsurare *g*, trebuie estimat parametrul necunoscut: vectorul *X*. Pentru estimarea acestui vector se pot folosi ecuațiile filtrului Kalman extins (EKF). Se execută mai multe iterații, începând de la o estimare inițială *X*, X_0, cu o matrice de covarianță inițială

P0, care să poată acoperi un interval mare pentru posibile valori de înălțime față de sol *h* și unghiuri de înclinare *θ*. Pentru fiecare iterație *k*, următorii pași vor fi executați:

1. Predicția vectorului de măsurare Z:

$$\mathrm{Z'}_k = \mathrm{g}_{v1,v2,L}(\mathrm{X_k}) \tag{112}$$

2. Calcularea matricei de măsurare (matricea Jacobiană a funcției *g*). Acest pas este implementat prin diferențiere numerică, prin variații mici ale înălțimii camerei și a unghiului de înclinare în jurul valorilor prezise de vectorul de stare X_k actual.

$$\mathrm{M}_k = \begin{pmatrix} \frac{\partial w_1}{\partial h} & \frac{\partial w_1}{\partial \theta} \\ \frac{\partial w_2}{\partial h} & \frac{\partial w_2}{\partial \theta} \end{pmatrix} \tag{113}$$

3. Calculare matrice de amplificare Kalman (”gain”):

$$\mathrm{K}_k = \mathrm{P}_k \mathrm{M}^T \left(\mathrm{M}_k \mathrm{P}_k {\mathrm{M}_k}^T + \mathrm{R}\right)^{-1} \tag{114}$$

În ecuația 114 *“R”* reprezintă matricea de covarianță a măsurătorii, o matrice diagonală care codifică incertitudinea măsurării lățimii vehiculului în spațiul imaginii (în pixeli).

4. Actualizarea vectorului de stare *X*. Folosind vectorul actual de măsurare *Z*, extras din datele de măsurare (proces descris în secțiunea următoare: 3.12.3), o actualizare a vectorului de stare *X* se calculează folosind ecuația 115.

$$\mathrm{X}_k = \mathrm{X}_{k-1} + \mathrm{K}_k(\mathrm{Z} - \mathrm{Z}'_k) \tag{115}$$

Pașii se execută de zece ori, iar de obicei după cinci iterații valorile pentru înălțimea camerei și unghiul de înclinare vor converge. Valorile inițiale setate pentru X_0 sunt: 1000 mm pentru înălțimea camerei și un unghi de înclinare de 0 grade. Deviația standard a înălțimii (în P_0) este 400 mm iar pentru unghiul de înclinare este de 3 grade. Deviația standard pentru lățimea unui vehicul (în pixeli) pentru matricea *R* este de 5 pixeli.

Filtrul Kalman extins conține un pas adițional: actualizarea matricei stării de covarianță *P*. Prin testare experimentală, am ales folosirea $P = P_0$ pentru toate iterațiile, deoarece nu afectează convergența rezultatelor.

3.12.3 Extragerea datelor de măsurare pentru calibrarea camerei

Conform secțiunii anterioare, pentru a calibra înălțimea camerei și a unghiului de înclinare, este nevoie doar de lățimea unui vehicul (exprimată în pixeli) în două cadre diferite. Pentru aceasta am folosit rețeaua convoluțională neuronală descrisă în secțiunea 3.10 care

va produce chenarele de încadrare pentru fiecare obstacol dintr-o imagine. Linia de jos a dreptunghiului este cea de care avem nevoie, deoarece arată punctul de contact al mașinii cu drumul, astfel încât se va potrivi cu ipoteza conform căreia coordonată de înălțime pe axa y este zero.

Totuși vehiculele detectate nu pot fi selectate pur și simplu, extrase cele două linii și apoi aplicat filtrul EKF, deoarece există mai multe motive pentru care această abordare va eșua:

- Vehiculele detectate nu sunt întotdeauna văzute din spate și prin urmare, lățimea lor percepută poate fi mai mare (de exemplu vehiculul roșu din Figura 91).

- Este posibil ca CNN-ul să nu genereze întotdeauna o detecție corectă pentru vehicul (de exemplu autoturismul din fața vehiculului din Figura 91), ceea ce înseamnă că este detectată o lățime incorectă.

- Vehiculul urmărit are o lățime necunoscută, din cauză că el ar putea fi prea îngust sau prea lat.

Din aceste motive, vectorul de măsurare *Z* va fi generat din o analiză statistică a dreptunghiurilor de încadrare a vehiculelor, utilizând secvențe de imagini dobândite în timpul conducerii timp de câteva minute. Figura 94 prezintă lățimile detectate, pentru diferite coordonate ale rândului unei imagini, pentru o secvență de aproximativ 7-8 minute din Cluj-Napoca.

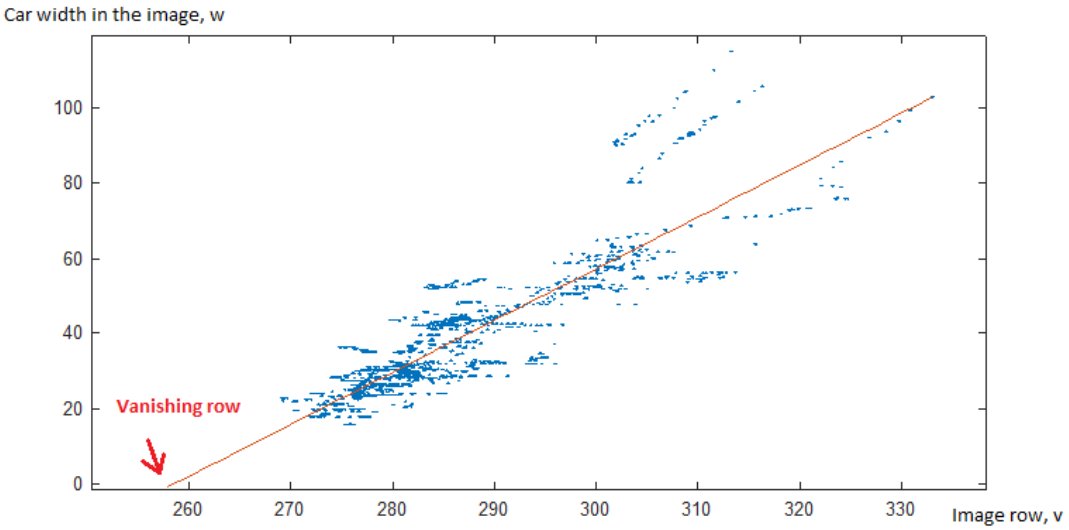


Figura 94. Relația liniară între lățimea unei mașini și coordonata de jos a chenarului de încadrare a vehiculului pe axa orizontală (coordonata linei).

În mod ideal, între lățimea unei mașini (“w”) și coordonata de jos pe axa verticală a chenarului de încadrare în imagine (*“v”*) există o relație liniară. Pentru a determina această linie se folosește metoda RANSAC [Fischler1981], mai ales deoarece se pretează mai bine în cazul punctelor care se abat (“outlier”). Linia rezultată este ilustrată în Figura 94 și va avea

valoarea lățimii unui vehicul *w=0* când coordonata ”v” corespunde cu cea a punctului de fugă în imagine (ceea ce înseamnă că lățimea unui vehicul este minimă pe linia orizontului).

O altă variantă de a obține această relație liniară este de a stoca aceste puncte direct într-o matrice bidimensională indexată astfel: pe coloane va fi coordonata lățimii unei mașini (“w” - determinată din diferența între cele două coordonate xmax - xmin), iar pe linii va fi coordonata rândului (“v” - coordonata ymax unde vehiculul are contact cu șoseaua). Găsirea liniei determinată de punctele din matrice se poate face mai eficient prin utilizarea unei transformate Hough modificată și având ca intrare harta de voturi (matricea bidimensională). Algoritmul de calculare a hărții de voturi este descris mai jos:

Algoritm calculare harta de voturi:

```
Intrare: lista de vehicule (chenare de încadrare)
Ieșire: harta de voturi (acumulator)
Pentru: fiecare vehicul din listă:
        int w = (xmax - xmin)
        int v = ymax
        accumulator(v, w)++
```

Găsirea liniei cu cele mai multe voturi din acumulator constă în alegerea perechii de *rho* și *theta* (coordonate polare de exprimare a unei linii) care are cele mai multe voturi din acumulator. Algoritmul de calculare a transformatei Hough este prezentat mai jos, unde *height* reprezintă înălțimea imaginii, iar *width* este lățimea imaginii de intrare (harta de voturi).

Algoritm calculare Hough:

```
Intrare: harta de voturi
Ieșire: linia cu cele mai multe voturi (rho, theta)
Pentru: i = (0, height):
        Pentru: j = (0, width):
                Dacă: acumulator(i, j) > 0:
                        Pentru: theta = (0, 360):
                                rho = j * cos(theta) + i * sin(theta)
                                hough(rho, theta) += acumulator(i, j)
rhoMax, thetaMax = calculare maxim (hough)
```

Rezultatul algoritmilor descriși mai sus este prezentat în Figura 95, unde în stânga este imaginea de intrare, în centru este linia extrasă folosind transformata Hough modificată, iar în dreapta este ilustrată harta de voturi (acumulatorul).

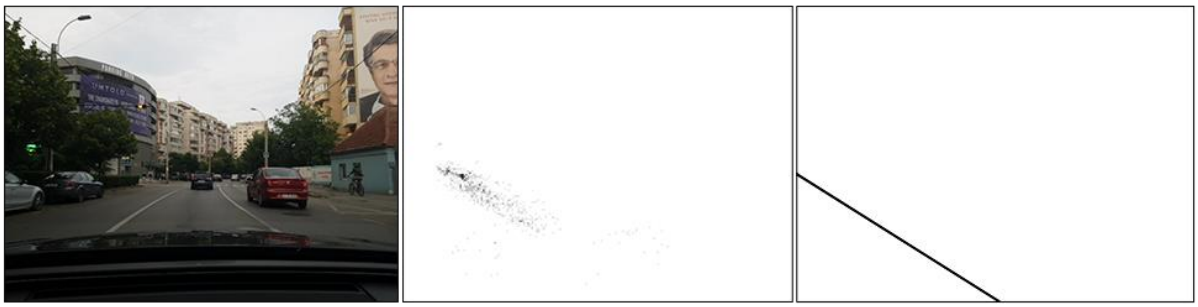


Figura 95. Ilustrare algoritm Hough modificat: în stânga este imaginea sursă, în centru este harta de voturi, iar în dreapta este linia maximă calculată folosind transformata Hough din harta de voturi.

Având dependența liniară calculată din datele achiziționate, se pot alege cele două linii verticale, corespunzătoare cu rândurile v_1 și v_2 folosite în algoritmul anterior. Pentru o imagine de dimensiune 640 x 480 pixeli, am ales v_1 = 310 și v_2 = 360 și o lățime medie a vehiculelor de 1750 mm. Cu aceste valori se poate executa algoritmul EKF. În Figura 96 este prezentat rezultatul rulării după 10 iterații pentru estimarea înălțimii camerei (în milimetri) și estimarea unghiului de înclinare (exprimat în grade). Înainte de calibrare, înălțimea camerei față de sol a fost măsurată ca fiind 1250 mm, în timp ce unghiul de înclinare (“pitch”) nu a fost măsurat. Totuși, precizia acestui unghi se poate valida prin observarea efectului său asupra matricei de proiecție.

În Figura 97 se pot observa efectele aplicării parametrilor de calibrare estimați din EKF. Din noii parametrii extrinseci am generat matricea de proiecție care poate fi utilizată pentru a proiecta un punct 3D situat în depărtare la o distanță mare în imagine și am calculat coordonata rândului său, care va corespunde cu linia orizontului (Figura 97 în stânga). Se poate calcula și noua matrice de proiecție pentru a genera o imagine cu efectul de perspectivă eliminat (IPM), după cum se poate observa în Figura 97 în dreapta. În imaginea IPM, coordonatele pixelilor sunt proporționale distanțele laterale și longitudinale (coordonatele 3D: X și Z), cu un factor de scalare de 50 mm pentru 1 pixel. Dacă înălțimea camerei și unghiul de înclinare sunt corecte, atunci imaginea IPM va afișa marcajele benzilor de circulație ca fiind paralele, iar distanța dintre ele ar trebui să corespundă cu lățimea unei benzi din scenă 3D (din lume). Lățimea benzii curente măsurată în imaginea IPM este de 65 pixeli, ceea ce corespunde la 3250 mm, iar lățimea benzii din scenă a fost măsurată ca fiind 3200 mm.

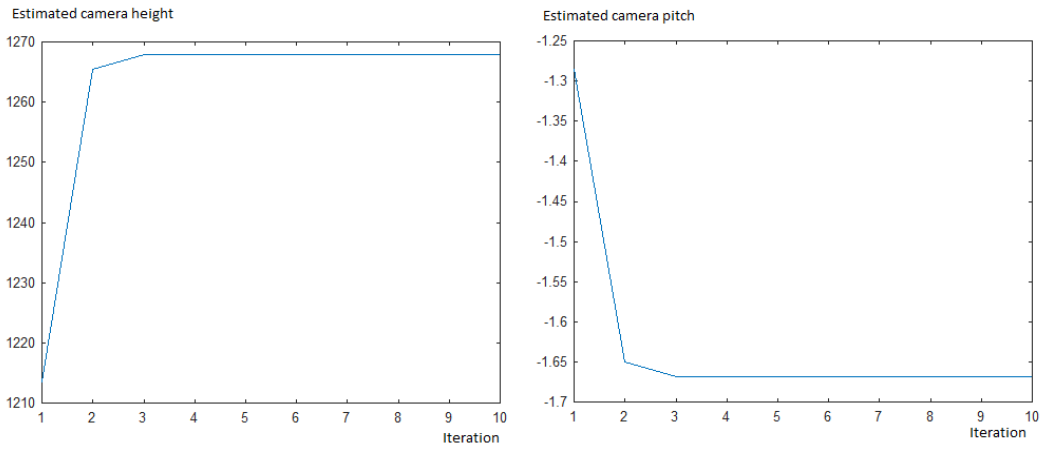


Figura 96. Estimarea înălțimii camerei și a unghiului de înclinare pe parcursul a zece iterații.

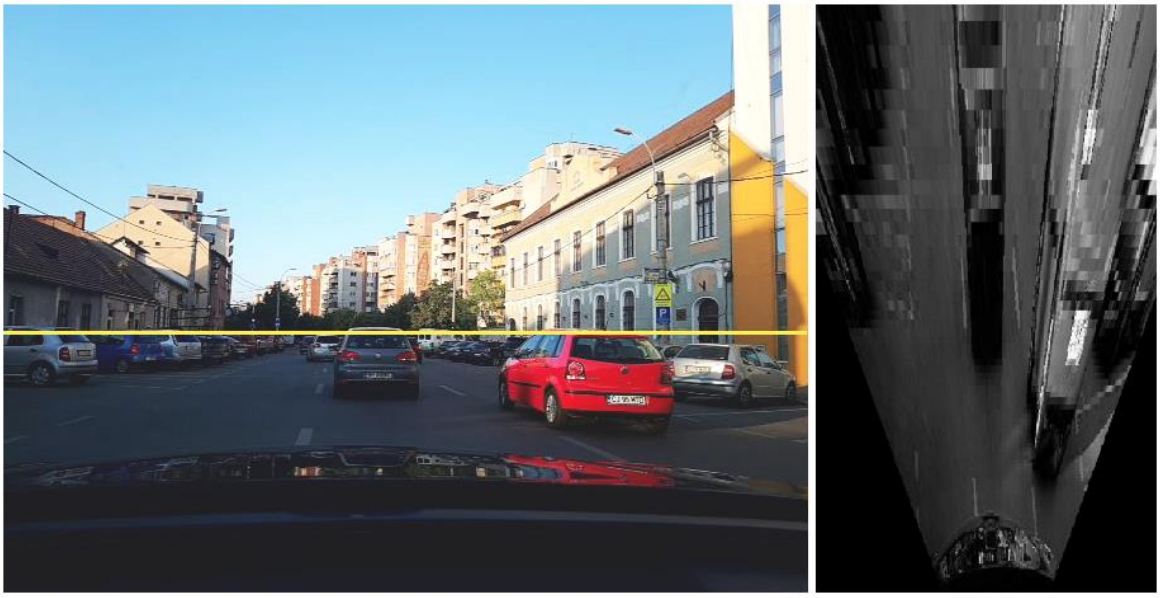


Figura 97. Stânga: linia orizontului rezultată din unghiul de înclinare (“pitch”), dreapta: imaginea IPM generată folosind matricea de proiecție estimată.

Din Figura 97, se poate observa că estimarea unghiului de înclinare a camerei și a înălțimii față de drum (sol) corespunde cu parametrii reali ai camerei. Totuși, se poate remarca și faptul că banda de circulație din Figura 97 nu este aliniată cu direcția de mișcare, ceea ce înseamnă că este nevoie de o estimare a unghiului de girație (“yaw”) în raport cu vehiculul propriu.

3.12.4 Estimarea unghiului de girație (“yaw”)

Pentru a estima unghiul de girație (orientarea camerei în raport cu axa longitudinală a vehiculului), este necesar să estimăm locația punctului de fugă în scenă. Acest punct reprezintă intersecția liniilor paralele (cum ar fi marcajele benzilor de circulație) în spațiul imaginii. În această abordare am utilizat obiectele detectate pentru a găsi această intersecție, în detrimentul folosirii liniilor de marcaje (abordare des întâlnită în lucrările existente în literatura de specialitate).

Dacă un obstacol este urmărit în timp și dacă se deplasează pe drum, pozițiile inferioare din stânga, respectiv dreapta ale dreptunghiului care le încadrează din spațiul imaginii vor forma linii sau curbe care se vor intersecta în punctul de fugă (Figura 98 stânga). Deși nu am folosit algoritmi de urmărire de obstacole, am salvat perechi de linii din coordonatele stânga și dreapta ale chenarelor de încadrare ale obstacolelor detectate în mijlocul imaginii și nu foarte îndepărtate, asemănător cum am prezentat deja în secțiunea 3.12.3. Am calculat intersecția liniilor din lista stângă și dreapta, iar aceste puncte sunt filtrate prin eliminarea celor care nu sunt în raza de 15 pixeli față de coordonata orizontală v_0 deja estimată a punctului de fugă. Punctele de intersecție rămase sunt sortate și apoi se alege

valoarea mediană pentru a forma coordonata verticală a punctului de fugă u_0 (Figura 98 dreapta).

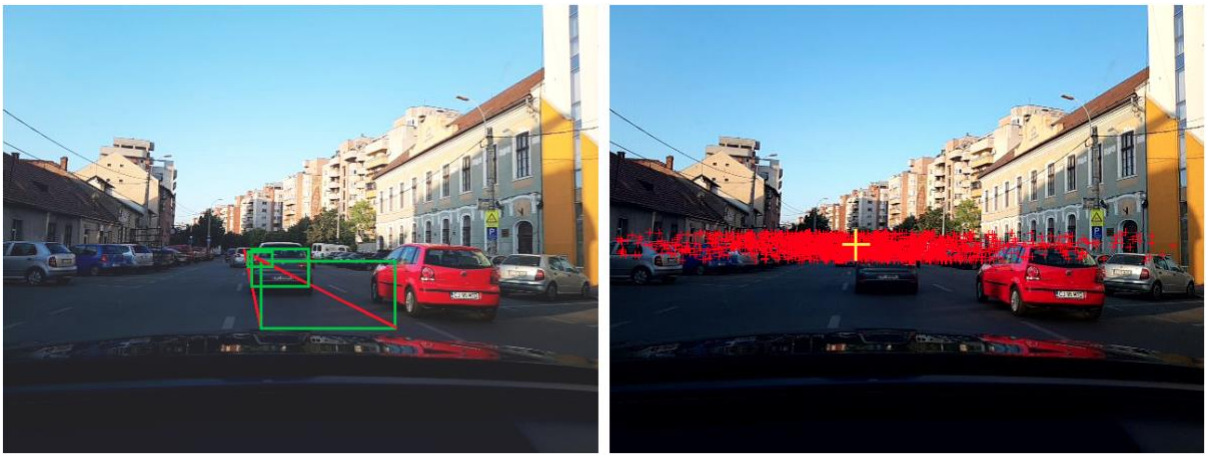


Figura 98. Estimarea unui punct de fugă candidat din traiectoria unui obstacol (stânga) și filtrarea candidaților prin aplicarea unui filtru median (dreapta).

Folosind ecuația 116 putem calcula unghiul de girație ("yaw") notat cu ψ, iar W este lățimea imaginii și f reprezintă distanța focală în pixeli.

$$\tan \Psi = \frac{u_0 - W/2}{f} \tag{116}$$

Cunoscând unghiul de girație se poate recalcula matricea de rotație folosind ecuația 117:

$$\mathbf{R}_{WC} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} \cos\Psi & 0 & \sin\Psi \\ 0 & 1 & 0 \\ -\sin\Psi & 0 & \cos\Psi \end{pmatrix} \tag{117}$$

Totodată, matricea de proiecte se poate recalcula folosind nouă matrice de rotație. Folosind matricea nouă de proiecție se poate genera imaginea cu efectul de perspectivă eliminat (IPM) în care drumul este aliniat cu axa vehiculului (Figura 99).

Figura 99. Comparație între imaginea IPM unde unghiul de rotație lateral este presupus a fi egal cu zero (stânga) și unghiul estimat corect (dreapta).

3.12.5 Rezultate și concluzii

Algoritmul de calibrare a fost testat pe mai multe secvențe din traficul rutier din Cluj-Napoca. Detecția obstacolelor funcționează la o frecvență de 10 Hz (10 cadre / secundă) pe un dispozitiv mobil Samsung Galaxy S8. Sistemul stochează detecțiile vehiculelor din scenă în memoria internă a dispozitivului mobil, iar utilizatorul va apela algoritmul de calibrare. Cele mai bune rezultate au fost obținute pe secvențe de trafic mai lungi, obținute prin conducerea timp de mai mult de 5 minute. Unghiul de înclinare și de giratie ("pitch" și "yaw") sunt estimate cu precizie în majoritatea imaginilor (erorile sunt mai mici de 0.1 grade). Unghiul de rotire ("roll") a fost doar simulat (imaginile fiind rotite folosind un unghi artificial), deoarece nu putem extrage o valoare reală pentru acest unghi, iar efectul asupra imaginii cu efectul de perspectivă eliminat (IPM) nu este clar întotdeauna (de exemplu drumul ar putea fi înclinat, nu doar camera foto). Totuși, sistemul a estimat unghiul artificial cu o eroare mai mică de 0.2 grade.

Estimarea înălțimii camerei este în general mai sensibilă, deoarece există situații în care sunt erori mai mari de 10 centimetri. Principala cauză pentru aceste erori este lipsa unui număr mare de vehicule detectate și că acestea nu variază foarte mult. Acest lucru se întâmplă în următoarele situații:

- o secvență în care este doar un vehicul în față, care este urmărit. Dacă este vehicul nu se încadrează în lățimea medie de 1.75 (dacă este prea lat sau prea îngust), atunci estimarea înălțimii va eșua.

- majoritatea vehiculelor detectate se află pe marginea drumului și vor fi încadrate incluzând vederea laterală a acestora.

Soluția pentru a depăși aceste limitări este de a colecta mai multe date, cu cât mai multe obstacole în fața vehiculului.

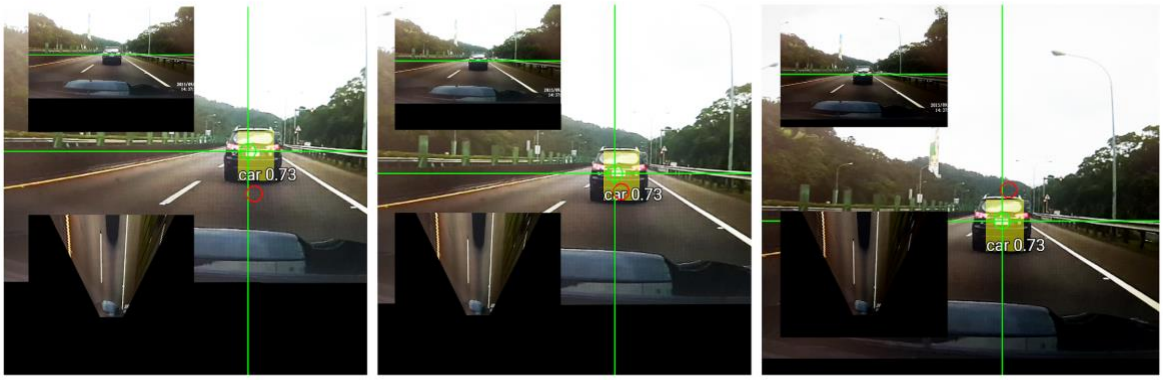


Figura 100. Sistemul de auto-calibrare și ajustarea matricei de proiecție pe dispozitive mobile.

În Figura 100 este prezentat un test al algoritmului de auto-calibrare pe dispozitivul mobil într-un mediu controlat. Cele trei imagini sunt capturate având un unghi de înclinare diferit al dispozitivului mobil. Se poate observa capacitatea sistemului de a ajusta parametri camerei în timp real și de a oferi o imagine IPM constantă, indiferent de înclinația telefonului. Intersecția liniilor verzi reprezintă punctul de fugă.

Figura 96 reprezintă primul test efectuat pentru validare, iar în Tabel 14 am prezentat rezultatele și pentru alte două teste proprii efectuate prin amplasarea camerei în vehicule diferite, și evaluare pe baza de date KITTI unde există informație doar despre înălțimea camerei. Valorile determinate pentru înălțimea camerei sunt apropiate de cele reale, iar pentru validarea unghiului de înclinare se poate face prin analiza imaginii IPM. Folosind unghiul de înclinare și înălțimea camerei se poate genera matricea de proiecție și imaginea cu efectul de perspectivă eliminat, în care liniile benzilor de circulație vor fi paralele daca valorile calculate sunt bune. Două exemple sunt ilustrate în Figura 101.

	Înălțime cameră calculată (”H”)	Înălțime cameră reală	Unghi înclinare calculat (”pitch”)
Test 1	1268	1250	-1.68°
Test 2	1234	1200	-3.8°
Test 3	1468	1480	2.97°
KITTI	1649	1650	-0.85°

Tabel 14. Rezultate evaluare calibrare automată.

Figura 101. Generare imagini IPM folosind calibrarea automată.

Algoritmul bazat pe rezultatele detecției de vehicule de la o rețea convoluțională neuronală este capabil să estimeze parametrii extrinseci ai unei camere monoculare în raport cu un cadru de referință bazat pe un vehicul standard. Sistemul funcționează precis și robust când secvențele sunt lungi și în diverse situații de trafic, când sunt suficiente date disponibile. Deoarece detecția vehiculelor influențează estimarea parametrilor, în viitor propun să îmbunătățesc acest aspect. Unele figuri folosite în această secțiune au fost deja publicate în articolul [Itu2018].

3.13 Concluzii

Acest capitol prezintă modelul camerei și tehnicile de calibrare pentru camere foto/video. În prima secțiune am descris modelul “thin lens” și apoi modelul perspectivă cu sistemul de coordonate al camerei și al lumii, unde este prezentat și planul imagini. Proiectarea punctelor din spațiul 3D în planul 2D al imaginii se face folosind matricea de proiecție, compusă din parametri intrinseci și extrinseci ai camerei. Parametri intrinseci sunt distanța focală și punctele principale, care vor forma matricea intrinsecă. Am prezentat modul de calculare al distanței focale cunoscând dimensiunea senzorului optic al camerei și dimensiunea imaginii sau din unghiul de vizualizare (deschiderea angulară) și dimensiunea imaginii. Parametri extrinseci ai camerei formează matricea extrinsecă și reprezintă de fapt poziția și orientarea camerei în scenă. Această matrice este formată din vectorul de translație și matricea de rotație. Am prezentat formulele de calcul pentru unghiurile de înclinare și de girație din coordonatele punctului de fugă. Secțiunea 3.4 prezintă un studiu actual al tehnicilor de calibrare existente la momentul scrierii tezei de doctorat. Restul de secțiuni din capitol descriu în detaliu implementări proprii pentru calibrarea automată a camerelor. În prima fază am prezentat rezultatele calculării automate a distanței focale direct pe dispozitive mobile, iar apoi am prezentat metode și algoritmi de eliminare a efectului de perspectivă din imagini (IPM). O primă realizare a fost crearea unei interfețe de calibrare manuală și de generare a imaginii IPM direct pe un dispozitiv mobil. O altă soluție are la bază procesarea “offline” a unei

secvențe și determinarea parametrilor camerei. Aceasta are la bază calcularea lățimii unei benzi și apoi calcularea înălțimii camerei față de sol. Ultimele secțiuni din capitol (3.9-3.12) prezintă soluții mai complexe pentru a calibra un sistem monocular. Mai întâi am descris un algoritm unic de calculare a punctului de fugă folosind procesarea imaginilor, de unde am reușit să creez o bază de date de imagini și puncte de fugă. Apoi, am folosit această bază de date pentru a antrena o rețea neuronală convoluțională pentru determinarea punctului de fugă dintr-o imagine din traficul rutier. Am extins această abordare pentru a folosi două rețele convoluționale, iar în ultima secțiune am prezentat o tehnică pentru a calibra camere direct din analiza traiectoriei vehiculelor și prin folosirea unui filtru Kalman extins pentru predicția parametrilor. În concluzie, acest capitol are la bază tehnici de determinare și predicție automată a parametrilor unui sistem monocular. Tehnicile sunt folosite în sisteme de roboți mobili sau de vehicule autonome, cu rolul de facilită dezvoltarea unor sisteme mai eficiente ca și cost pentru percepția mediului înconjurător.

3.13.1 Contribuții

Contribuțiile originale rezultate în urma soluțiilor propuse în acest capitol sunt: determinarea automată a distanțelor focale pe dispozitive mobile, crearea unui sistem de calibrare manual și intuitiv pe dispozitive mobile, unde utilizatorul poate ajusta parametri extrinseci ai camerei. O altă contribuție este crearea unui sistem de calibrare “offline” prin analiza imaginilor din trafic și determinarea lățimii unei benzi de circulație. Banda de circulație este folosită pentru a avea o corespondență între scena 3D a lumii și spațiul 2D al imaginii. Prin această metodă se determină înălțimea camerei față de sol (parte din vectorul de translație al camerei).

O contribuție principală din acest capitol se referă la elaborarea unui algoritm nou de calculare a punctului de fugă din imaginile din traficul rutier. Aceste imagini prezintă un efect de perspectivă în care liniile benzilor de circulație se vor intersecta în punctul de fugă. Algoritmul exploatează trăsăturile din imagini care au o anumită magnitudine și orientare și le folosește pentru a construi o hartă de voturi care este analizată și validată folosind ferestre glisante. Punctul cu cele mai multe voturi al unei ferestre validă va reprezenta un punct de fugă. Având această metodă am construit o bază de date de imagini și puncte de fugă, pe care am folosit-o pentru a antrena o rețea neuronală convoluțională cu scopul de a avea predicții din rețea asupra locației unui punct de fugă. Rețeaua are ca intrare imaginea scenei redimensionată, iar ieșirea va fi formată din cele 2 coordonate ale punctului de fugă. Rezultatele acestei abordări au fost foarte bune, iar apoi întreaga metodă a fost extinsă pentru a utiliza mai multe imagini de antrenare și două rețele neuronale convoluționale diferite: una pentru a prezice poziția pe coordonata orizontală, iar cealaltă pentru a prezice poziția punctului de fugă pe coordonata verticală. Am prezentat o comparație între metode, iar folosirea a două rețele a fost mai eficientă și a adus îmbunătățiri semnificative. Această soluție a fost implementată și pe dispozitive mobile.

Ultima secțiune constă în implementarea unei soluții de auto-calibrare folosind traiectoria vehiculelor. Detecția vehiculelor am realizat-o prin folosirea unei rețele convoluționale neuronale, mai specific rețeaua SSD și MobileNet care oferă un raport bun între precizia detecțiilor și viteza de execuție (timpul de predicție pe un cadru). Rezultatul rețelei va fi o listă de dreptunghiuri de încadrare corespunzătoare fiecărui vehicul din scenă. Aceste dreptunghiuri sunt urmărite de la un cadru la altul, iar pentru calibrare este necesar

doar un anumit număr de vehicule detectate. Din coordonatele dreptunghiului de încadrare sunt extrase lățimile obiectelor din scenă și coordonata lor pe axa verticală (axa y a imaginii). Următorul pas constă în determinarea relației liniare dintre lățimi și coordonata verticală. Inițial am calculat relația liniară folosind RANSAC, iar apoi prin utilizarea algoritmului Hough modificat. Algoritmul de estimare a înălțimii camerei și a unghiului de înclinare folosește un filtru Kalman extins, iar vectorul de măsurare Z este format din lățimea w_1 și w_2 a vehiculelor la coordonata orizontală (axa x) $v_1 = 310$ și $v_2 = 360$, determinate din relația liniară calculată în pasul precedent. Am ales 2 puncte în spațiul 3D care să reprezinte un vehicul teoretic cu o lățime fixă (standard) de 1750 mm aflat la o distanță de $Z_1 = 7000$ mm și încă un set de puncte 3D pentru un vehicul aflat la o distanță $Z_2 = 20000$ mm. Aceste 4 puncte care descriu 2 vehicule la 2 distanțe diferite sunt proiectate în spațiul 2D al imaginii, iar perechile de coordonate laterale vor forma două linii care se vor intersecta într-un punct de fugă. În următorul pas se vor calcula coordonatele punctelor la indexul $v_1 = 310$ și $v_2 = 360$. Practic, algoritmul EKF va încerca să facă o potrivire între aceste 4 puncte teoretice ale celor 2 vehicule și cele 4 puncte ale vehiculelor din vectorul de măsurare. Algoritmul va ajusta înălțimea camerei și unghiul de înclinare până când rezultatul este stabil și se ajunge la o potrivire cât mai bună (în general sunt de ajuns 10 iterații ale filtrului EKF). Aceste abordări au fost publicate la conferințe naționale și internaționale, iar baza de date a punctelor de fugă este disponibilă online.

3.13.2 Publicații

Următoarele publicații în conferințe internaționale au rezultat din metodele propuse în cadrul acestui capitol:

1. Razvan Itu, Diana Borza, Radu Danescu, "Automatic extrinsic camera parameters calibration using Convolutional Neural Networks", 2017 IEEE 13th International Conference on Intelligent Computer Communication and Processing (ICCP 2017), pp. 273-278.

2. Razvan Itu, Radu Danescu, "Machine Learning Based Automatic Extrinsic Calibration of an Onboard Monocular Camera for Driving Assistance Applications on Smart Mobile Devices", 2018 International Forum on Advanced Microsystems for Automotive Applications, pp. 16-28.

3. Radu Danescu, Razvan Itu, ”Camera Calibration for CNN based Generic Obstacle Detection”, 2019 Portuguese Conference on Artificial Intelligence (EPIA), acceptat iunie 2019.

4. Concluzii

În cadrul acestei cărți am prezentat un sistem de percepție și urmărire a obstacolelor din traficul rutier folosind o cameră monoculară. Cartea este structurată în patru capitole: primul conține o introducere în domeniu și necesitatea unor sisteme de asistență a șoferilor și obiectivele propuse, al doilea capitol prezintă metode de detecție și urmărire a obstacolelor și al treilea capitol descrie tehnicile de calibrare a camerei. Acest ultim capitol al cărții se referă la concluzii și dezvoltări ulterioare, urmate de bibliografie și anexe.

În primul capitol este descrisă o statistică sumară a accidentelor din ultimii ani, de unde rezultă o nevoie clară de a avea vehicule cât mai inteligente pe șosele. Am prezentat o scurtă istorie a vehiculelor inteligente: primele din anii 1960 care foloseau tehnici rudimentare pentru a-și menține banda, continuând cu cele de la sfârșitul anilor 1980 și anii 1990 care foloseau deja viziune artificială. În anii 2000 au fost salturi majore în acest domeniu facilitate în principal de competiția celor de la DARPA, iar după anii 2010 au fost deja lansate vehicule cu grad mare de automatizare pentru șoferi. Standardele internaționale descriu în total 5 grade de automatizare, iar senzorii folosiți pentru a obține datele din trafic sunt de asemenea prezentate în primul capitol. Obiectivele propuse, structura tezei și partea de mulțumiri sunt prezentate în secțiunile 1.2-1.4.

Capitolul 2 descrie metodele de detecție și de urmărire a obstacolelor. Am prezentat abordări pentru urmărirea obstacolelor folosind probabilistică, în care sunt menținute mai multe ipoteze despre poziția sau starea curentă a obstacolelor sau a vehiculelor, iar fiecare ipoteză are o probabilitate diferită. Am prezentat filtrul Bayes și apoi alte abordări bazate pe acesta: filtrul Kalman, filtrul Kalman extins și filtrul de particule (util pentru sisteme care nu se pot modela Gaussian). Secțiunile 2.5 și 2.6 reprezintă o introducere în inteligență artificială și rețele neuronale artificiale, care au la bază un model inspirat din neuronul biologic. Am prezentat și exemplificat operația de convoluție, iar în secțiunea 2.7 am descris diferența între clasificare și regresie și apoi în 2.8 am prezentat diferite tipuri de funcții de activare folosite pentru cele două tipuri de învățare, dar și pentru segmentare semantică. Secțiunea 2.9 conține un sumar al algoritmului de propagare a ponderilor și tehnica “gradient descent”, iar în secțiunea 2.10 sunt funcțiile de cost pentru rețele neuronale. Partea de segmentare semantică folosind rețele neuronale este descrisă în 2.11, în timp ce în secțiunea 2.12 am introdus tehnici pentru percepția monoculară. Prima parte (2.12.1) reprezintă un studiu al soluțiilor actuale bazate pe dispozitive mobile inteligente. În 2.12.2 am descris tehnicile folosite pentru estimarea adâncimii în imagini capturate folosind o singură cameră și apoi am prezentat o tehnică ce am folosit-o în alte lucrări pentru a detecta obstacole în imagini monoculare în care efectul de perspectivă este eliminat. Contribuțiile principale sunt descrise începând cu capitolul 2.13, unde am prezentat mai întâi bazele de date cu imagini din trafic (2.13.1) pe care le-am folosit pentru dezvoltarea un sistem complex de percepție a traficului rutier. Software-ul propriu pentru achiziția de imagini noi este descris în secțiunea 2.13.2. Următorul subcapitol (2.13.3) descrie pe larg o rețea neuronală pe care am folosit-o pentru segmentarea imaginilor din traficul rutier care să extragă doar zona de drum dintr-o imagine. Rezultatele obținute sunt foarte bune și pe baza de date proprie (care nu a fost folosită în timpul antrenării), iar pe seturile de validare ale bazelor de date existente rezultatele sunt bune chiar dacă metrica de evaluare este mai redusă decât a altor abordări similare. Analizând imaginile în

care rezultatele segmentării nu au fost bune am constat că există anumite erori în adnotările din bazele de date cunoscute, iar rețeaua dezvoltată de mine a oferit uneori predicții mai bune. Au fost și situații și opuse, în care predicția a fost eronată, iar toate acestea sunt prezentate și ilustrate în teză. În subcapitolul 2.13.4 am descris algoritmul de urmărire bazat pe filtre de particule și o hartă de ocupare dinamică. Această abordare nu este una originală și a mai fost publicată, dar modul în care am extins-o și cum am construit partea de măsurare este unică. Harta de măsurare este folosită pentru a genera particule noi, care sunt grupate în celule din care se pot extrage cuboide care reprezintă obstacolele detectate și urmărite în scenă. Această hartă de măsurare este de fapt o hartă de ocupare binară, construită din imaginea de drum segmentată de rețeaua neuronală convoluțională. Având măsurătorile în fiecare cadru, particulele pot fi create, deplasate sau distruse, astfel se pot urmări obstacolele. Incertitudinea măsurătorii este determinată de distanța până la obstacole care se este determinată în imaginea IPM. În ultimul pas, particulele sunt grupate în funcție de similaritățile lor (proximitate, vectorii de viteză) și apoi se extrag cuboide din ele. O altă contribuție față de lucrările publicate în trecut o reprezintă rafinarea hârții de ocupare, prin procesarea imaginii segmentate: am realizat un algoritm care să unească vârfurile din histogramă care aparțin unui obstacol. Efectul asupra creării particulelor este bine ilustrat și exemplificat în subcapitolul 2.13.5 (a). O altă contribuție constă în rafinarea parametrilor camerei prin generarea unor imagini IPM având unghiuri de înclinare diferite și compararea hărților de ocupare cu cea actuală din filtrul de particule. În secțiunea 2.13.6 am descris o modalitate de a estima orientarea locală a obiectelor și dimensiunile lor folosind o rețea neuronală convoluțională. Implementarea aceasta nu este unică, dar contribuția constă în integrarea cu sistemul de percepție bazat pe filtrul de particule. Astfel, orientarea și dimensiunea cuboidelor extrase din filtrul de particule sunt ajustate folosind CNN-ul. Alte detalii de implementare a tehnologiilor folosite și arhitectura sistemului sunt prezentate în secțiunea 2.13.7. Capitolul 2 se încheie cu concluzii și o listă de publicații rezultate.

În capitolul 3 am descris tehnici de calibrare a camerei. În primul subcapitol (2.1) am prezentat modelul camerei și apoi parametri intrinseci (2.2) și cei extrinseci (2.3). Principalul obiectiv în urma calibrării este de a cunoaște parametrii camerei pentru a calcula matricea de proiecție, care poate fi folosită pentru a genera imagini cu efectul de perspectivă eliminat, o vedere periferică (de sus) a scenei de drum. În secțiunea 3.4 am realizat un studiu actual al tehnicilor de calibrare a camerelor, iar în 3.5 am prezentat modalități de calculare automată a distanței focale. Secțiunea 3.6 conține algoritmii de calculare a imaginilor IPM, iar în 3.7 am prezentat o interfață intuitivă pentru asistarea calibrării manuale a parametrilor extrinseci. O altă contribuție este dată de implementarea unui algoritm de auto-calibrare a parametrilor vectorului de translație, prin analizarea “offline” unei secvențe cadru cu cadru. Înălțimea camerei față de sol este ajustată astfel încât lățimea benzii de circulație detectată în imaginea IPM să corespundă cu valoarea în pixeli corespunzătoare unei lățimi standard de 3.5 metri. Tot aici am prezentat pe scurt modul de determinare a unei benzi de circulație. În secțiunea 3.9 am descris tehnici pentru determinarea punctului de fugă din imaginile capturate în traficul rutier. Am prezentat abordări existente și apoi o tehnică proprie, bazată pe procesarea de imagini în care sunt analizate orientările și magnitudinile punctelor de muchii care aparțin marcajelor rutiere. Folosind o schemă de votare proprie am realizat o hartă de voturi din care este extras punctul de fugă final. Rezultatele au fost suficient de bune încât să creez o bază de date de imagini și puncte fuga, care apoi am folosit-o pentru a antrena o rețea neuronală convoluțională care să ofere predicții ale coordonatelor x și y din imagine a punctului de fugă (secțiunea 3.10). Metoda a fost publicată la o conferință internațională. Am folosit mai multe

tehnici de augmentare a bazei de date si apoi am extins aceasta abordare prin folosirea unei rețele antrenata pentru predicția coordonatei x, si aceleiași rețele antrenata pentru coordonata y a punctului de fugă, iar rezultatele față de varianta inițială sunt mai bune (secțiunea 3.11). Totodată, am implementat si testat aceasta soluție și pe dispozitive mobile Android (3.11.4). În secțiunea 3.12 am descris o soluție de auto-calibrare a unei camere. Din analiza traiectoriei unui vehicul am determinat înălțimea camerei față de sol și unghiul de înclinare folosind un filtru EKF, dar și punctul de fugă care este folosit pentru corecția imaginii IPM. Întreaga soluție a fost publicată la o conferință internațională și implementată atât pe sisteme desktop, cat și pe dispozitive mobile Android. Capitolul 3 se încheie cu partea de concluzii si contribuții originale și cu o listă de lucrări publicate.

Cartea are bază teza de doctorat susținută, care introduce contribuții în domeniul roboților și a vehiculelor autonome în care percepția este realizată folosind o singură cameră. Algoritmii de percepție au la baza algoritmi și tehnici de procesare de imagini, dar și algoritmi de inteligență artificială prin folosirea rețelelor neuronale convoluționale. Cartea prezintă o soluție completă de percepție a scenei de trafic rutier, începând de la achiziția imaginilor și până la procesarea și afișarea rezultatelor. Sistemul monocular este capabil să se calibreze singur după un anumit timp, iar datele procesate pot fi folosite și integrate în sisteme inteligente de asistate a șoferului, cum ar fi sisteme de prevenție a accidentelor care pot acționa în locul șoferului său pot oferi doar avertizări acustice sau sonore. Soluțiile prezentate în carte au fost implementate atât pe sisteme mobile (smartphone-uri sau tablete), cât și pe PC-uri (desktop sau laptop). Prin metodele descrise în această lucrare se poate aduce un beneficiu major prin îmbunătățirea siguranței în traficul rutier, având avantajul că astfel de sisteme se pot integra și pe vehicule existente la un cost redus (prin folosirea unei singure camere video pentru analiza scenei). În concluzie, pe viitor îmi doresc să implementez un sistem complet care să fie integrat într-un vehicul.

Bibliografie

[ASIRT2018] - Association For Safe International Road Travel, “Road Safety Facts - Annual Global Road Crash Statistics”, 2018, online la adresa: https://www.asirt.org/safe-travel/road-safety-facts/

[AdbelAziz1971] - Y. I. Adbel-Aziz, H. M. Karara, “Direct linear transformation into object space coordinates in close-range photogrammetry”, Close-Range Photogrammetry, pp. 1-18, 1971.

[Ahmed1999] - M. T. Ahmed, E. E. Hemayed, A. A. Farag, “Neurocalibration: A Neural Network That Can Tell Camera Calibration Parameters”, International Conference on Computer Vision, pp. 463-468, 1999.

[Alvarez2007] - J.M. Alvarez, A.M. López, R. Baldrich, “Shadow Resistant Road Segmentation from a Mobile Monocular System”, Pattern Recognition and Image Analysis, Vol. 4478, pp. 9-16, 2007.

[Badrinarayanan2017] - V. Badrinarayanan, A. Kendall, R. Cipolla, "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 12, pp. 2481-2495, 2017.

[Bazin2012] J. Bazin, M. Pollefeys, “3-line RANSAC for orthogonal vanishing point detection”, IEEE International Conference on Intelligent Robots and Systems, pp. 4282-4287, 2012.

[Bengio2009] - Y. Bengio, “Learning Deep Architectures for AI”, Foundations and Trends in Machine Learning, vol. 2, no. 1, pp. 1-127, 2009.

[Bertozzi1998] - M. Bertozzi, A. Broggi, “GOLD: a Parallel Real-Time Stereo Vision System for Generic Obstacle and Lane Detection”, IEEE Transactions on Image Processing, vol. 7, no. 1, pp. 62-81, 1998.

[Bileschi2009] - S. Bileschi, “Fully automatic calibration of LiDAR and video streams from a vehicle”, IEEE International Conference on Computer Vision Workshops, pp. 1457-1464, 2009.

[Bojarski2016] - M. Bojarski, et. al., “End to End Learning for Self-Driving Cars”, arXiv:1604.07316, 2016.

[Broggi1995] - A. Broggi, “Robust real time land and road detection in critical shadow conditions”, IEEE International Symposium on Computer Vision, pp. 19-21, 1995.

[Broggi1999] - Alberto Broggi, Massimo Bertozzi, Alessandra Fascioli, and Gianni Conte, “Automatic Vehicle Guidance: the Experience of the ARGO Vehicle”, World Scientific, Singapore, 1999.

[Bui2013a] - T.H. Bui, E. Nobuyama, T. Saitoh, “A texture-based local soft voting method for vanishing point detection from a single road image”, IEICE Transactions on Information and Systems, vol. E96-D, pp. 690-698, 2013.

[Bui2013b] - T.H. Bui, T. Saitoh, E. Nobuyama, “Road Area Detection Based on Texture Orientations Estimation and Vanishing Point Detection”, Proceedings of the 2013 Annual Conference Society of Instrument and Control Engineers, pp. 1138-1143, 2013.

[Canny1986] - J. F. Canny, “A Computational Approach to Edge Detection”, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 8, no. 6, pp. 679-698, 1986.

[Caprile1990] - B. Caprile, V. Torre, “Using vanishing points for camera calibration”, International Journal of Computer Vision, vol. 4, pp. 127-139, 1990.

[Casser2018] - V. Casser, S. Pirk, R. Mahjourian, A. Angelova, “Depth Prediction Without the Sensors: Leveraging Structure for Unsupervised Learning from Monocular Videos”, arXiv: 1811.06152, 2018.

[Chen2016a] - L.C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, A. L. Yuille, “Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs”, arXiv: 1412.7062, 2016.

[Chen2016b] - L.C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, A. L. Yuille, “DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs”, arXiv: 1606.00915, 2016.

[Chen2016c] - X. Chen, K. Kundu, Z. Zhang, H. Ma, S. Fidler, R. Urtasun, “Monocular 3D Object Detection for Autonomous Driving”, Computer Vision and Pattern Recognition, pp. 2147-2156, 2016.

[Chen2017] - L.C. Chen, G. Papandreou, F. Schroff, H. Adam, “Rethinking Atrous Convolution for Semantic Image Segmentation”, arXiv: 1706.05587, 2017.

[Chen2018] - L.C. Chen Chen, Y. Zhu, G. Papandreou, F. Schroff, H. Adam, “Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation”, arXiv: 1802.02611, 2018.

[Chira2010] - I. M. Chira, A. Chibulcutean, R. Danescu, “Real-Time Detection of Road Markings for Driving Assistance Applications”, International Conference on Computer Engineering and Systems, pp. 158-163, 2010.

[Chollet2017] - F. Chollet, “Xception: Deep Learning with Depthwise Separable Convolutions”, Computer Vision and Pattern Recognition, pp. 1800-1807, 2017.

[Ciresan2012] - D. C. Ciresan, A. Giusti, L. M. Gambardella, J. Schmidhuber, “Deep neural networks segment neuronal membranes in electron microscopy images”, Advances in Neural Information Processing Systems, pp. 2852-2860, 2012.

[Clarke1998] - T. A. Clarke, J. G. Fryer, “The development of camera calibration methods and models”, Photogrammetric Record, vol. 16, no. 9, pp. 51-66, 1998.

[Cordts2016] - M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, B. Schiele, “The Cityscapes Dataset for Semantic Urban Scene Understanding”, Computer Vision and Pattern Recognition, pp. 3213-3223, 2016.

[Cybenko1989] - G. Cybenko, "Approximations by superpositions of sigmoidal functions", Mathematics of Control, Signals, and Systems, vol. 2 no. 4, pp. 303-314, 1989.

[Daimler2016] - Daimler, “The PROMETHEUS project launched in 1986: Pioneering autonomous driving”, Daimler, 2016, online la adresa: https://media.daimler.com/marsMediaSite/en/instance/ko/The-PROMETHEUS-project-launched-in-1986-Pioneering-autonomous-driving.xhtml?oid=13744534

[Dalal2005] - N. Dalal, B. Triggs, “Histograms of oriented gradients for human detection”, Computer Vision and Pattern Recognition, vol. 1, pp 886-893, 2005.

[Danescu2011] - R. Danescu, F. Oniga, S. Nedevschi, “Modeling and Tracking the Driving Environment with a Particle-Based Occupancy Grid”, IEEE Transactions on Intelligent Transportation Systems, volume 12, no. 4, pp. 1331-1342, 2011.

[Danescu2016a] - R. Danescu, R. Itu, A. Petrovai, "Generic Dynamic Environment Perception Using Smart Mobile Devices", Sensors, vol. 16, no. 10, art. no. 1721, 2016.

[Danescu2016b] - R. Danescu, A. Petrovai, R. Itu, S. Nedevschi, "Generic Obstacle Detection for Mobile Devices Using a Dynamic Intermediate Representation", Advances in Intelligent Systems and Computing, vol. 427, pp. 629-639, 2016.

[Danescu2019] - R. Danescu, “Perceptia prin particule”, UT Press, Cluj-Napoca, 2018.

[Debattisti2013] - S. Debattisti, L. Mazzei, M. Panciroli, “Automated Extrinsic Laser and Camera Inter-Calibration Using Triangular Targets”, IEEE Intelligent Vehicles, pp. 696-701, 2013.

[Delcker2018] - J. Delcker, “The man who invented the self driving car (in 1986)”, Politico, 2018, online la adresa: https://www.politico.eu/article/delf-driving-car-born-1986-ernst-dickmanns-mercedes/

[Dhanachandra2015] - N. Dhanachandra, K. Manglem, Y.J. Chanu, “Image Segmentation Using K-means Clustering Algorithm and Subtractive Clustering Algorithm”, Procedia Computer Science, vol. 54, pp. 764-771, 2015.

[Dice1945] - Lee R. Dice, "Measures of the Amount of Ecologic Association Between Species", Ecology, vol. 26, no. 3, pp. 297-302, 1945.

[Dickmanns1998] - E. D. Dickmanns, "Vehicles capable of dynamic vision: a new breed of technical beings", Artificial Intelligence, vol. 103, no. 1-2, pp. 49-76, 1998.

[Drivea] - Drivea, “Driving Assistant App”, accesat 2019, online la adresa: http://www.appszoom.com/android_applications/transportation/drivea-driving-assistant-app_bdwmk.html

[Drozdzal2015] - M. Drozdzal, E. Vorontsov, G. Chartrand, S. Kadoury, C. Pal, “The Importance of Skip Connections in Biomedical Image Segmentation”, Workshop on Deep Learning in Medical Image Analysis, pp. 234-241, 2015.

[Duda1972] - R.O. Duda, P. E. Hart, "Use of the Hough Transformation to Detect Lines and Curves in Pictures", Communications of the ACM, vol. 15, no. 1, pp. 11-15, 1972.

[Enciso1997] - S. Enciso, T. Vieville, “Self-calibration from four views with possibly varying intrinsic parameters”, Image and Vision Computing, vol. 15, pp. 293-305, 1997.

[Everingham2010] - M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, A. Zisserman, “The PASCAL Visual Object Classes (VOC) Challenge”, International Journal of Computer Vision, vol. 88 no. 2, pp. 303-338, 2010.

[Faig1975] - W. Faig, “Calibration of close-range photogrammetry systems: Mathematical formulation”, Photogrammetric Engineering and Remote Sensing, vol. 41, no. 12, pp. 1479-1486, 1975.

[Fehlhaber2018] - N. Fehlhaber, “The Ghost Car”, Continental, 2018, online la adresa: https://www.continental-corporation.com/en/company/history/the-ghost-car-145316

[Felzenszwalb2004] - P. F. Felzenszwalb, D. P. Huttenlocher, ”Efficient Graph-Based Image Segmentation”, International Journal of Computer Vision, vol. 59, no. 2, pp. 167-181, 2004.

[Fischler1981] - M. A. Fischler, R. C. Bolles, “Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography”, Communications of the ACM, vol. 24, no. 6, pp. 381-395, 1981.

[Fod2002] - A. Fod, A. Howard, M. J. Mataric, “Laser-based people traking”, Proc. IEEE Int. Conf. Robotics and Automation (ICRA), pp. 3024-3029, 2002.

[Foot1967] - P. Foot, “The problem of abortion and the doctrine of double effect”, Oxford Review, vol. 5, pp. 5-15, 1967.

[Forsyth2002] - D. A. Forsyth, J. Ponce, “Computer Vision: A Modern Approach”, Prentice Hall Professional Technical Reference, 2002.

[Ganapaphy1984] - S. Ganapaphy, “Decomposition of transformation matrices for robot vision”, International Conference on Robotics and Automation, pp. 130-139, 1984.

[Garg2016] - R. Garg, V. Kumar BG, I. Reid, “Unsupervised CNN for single view depth estimation: Geometry to the rescue”, European Conference on Computer Vision, pp. 740-756, 2016.

[Garmin] - Garmin, “Garmin Dashcam 65W”, accesat 2019, online la adresa: https://buy.garmin.com/ro-RO/RO/p/587334

[Geiger2012] - A. Geiger, P. Lenz, R. Urtasun, “Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite”, Computer Vision and Pattern Recognition, pp. 3354-3361, 2012.

[Gennery1979] - D. B. Gennery, “Stereo-camera calibration”, Proceedings Image Understanding Workshop, pp. 101-108, 1979.

[Girshick2014] - R. B. Girshick, J. Donahue, T. Darrell, J. Malik, “Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation”, Computer Vision and Pattern Recognition, pp. 580-587, 2014.

[Girshick2015] - R. B. Girshick, “Fast R-CNN”, International Conference on Computer Vision, pp. 1440-1448, 2015.

[Godard2017] - C. Godard, O. M. Aodha, G. J. Brostow, “Unsupervised Monocular Depth Estimation with Left-Right Consistency”, Computer Vision and Pattern Recognition, pp. 6602-6611, 2017.

[Gordon1993] - N. J. Gordon, D. J. Salmond, A. F. M. Smith, “A novel approach to nonlinear/non-gaussian state estimation”, IEEE Procedings F, vol. 2, pp. 107-113, 1993.

[Grompone2010] - R. Grompone von Gioi, J. Jakubowicz, J.-M. Morel, G. Randall, “LSD: A Fast Line Segment Detector with a False Detection Control”, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 32, no. 4, pp. 722-732, 2010.

[Harris1988] - C. Harris, M. Stephens, “A combined corner and edge detector”, Alvey Vision Conference, vol. 15, pp. 147-151, 1988.

[Hartley1997] - R. I. Hartley, “Self-calibration of stationary cameras”, International Journal of Computer Vision, vol. 22, no. 1, pp. 5-23, 1997.

[He2016] - K. He, X. Zhang, S. Ren, J. Sun, “Deep Residual Learning for Image Recognition”, Computer Vision and Pattern Recognition, pp. 770-778, 2016.

[He2017] - K. He, G. Gkioxari, P. Dollár, R. B. Girshick, “Mask R-CNN”, International Conference on Computer Vision, pp. 2980-2988, 2017.

[Hochreiter1997] - S. Hochreiter, J. Schmidhuber, “Long short-term memory”, Neural computation, vol. 9, no. 8, pp. 1735-1780, 1997.

[Hol2006] - J. D. Hol, T. B. Schon, F. Gustafsson, “On resampling algorithms for particle filters”, Proc. IEEE Nonlinear Stat. Signal Process, Workshop, pp. 79-82, 2006.

[Hornik1991] - K. Hornik, "Approximation Capabilities of Multilayer Feedforward Networks", Neural Networks, vol. 4, no. 2, pp. 251-257, 1991.

[Hough1962] - P.V.C. Hough, “Method and means for recognizing complex patterns”, U.S. Patent 3,069,654, Dec. 18, 1962.

[Howard2017] - A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, H. Adam, “Mobilenets: Efficient convolutional neural networks for mobile vision applications”, arXiv:1704.04861, 2017.

[Huang2017] - G. Huang, Z. Liu, L. van der Maaten, K. Q. Weinberger, “Densely Connected Convolutional Networks”, Computer Vision and Pattern Recognition, pp. 2261-2269, 2017.

[Huber1964] - P. J. Huber, "Robust Estimation of a Location Parameter", Annals of Statistics, vol. 53, no. 1, pp. 73-101, 1964.

[iOnRoad] - iOnRoad, “iOnRoad Augmented Driving Pro”, accesat 2019, online la adresa: http://www.ionroad.com/

[Isard1998] - M. Isard, A. Blake, “CONDENSATION - Conditional density propagation for visual tracking”, International Journal of Computer Vision, vol. 29, pp. 5-28, 1998.

[Itu2014] - R. Itu, R. Danescu, "An Efficient Obstacle Awareness Application for Android Mobile Devices", International Conference on Intelligent Computer Communication and Processing, pp. 157-163, 2014.

[Itu2017] - R. Itu, D. Borza, R. Danescu, "Automatic extrinsic camera parameters calibration using Convolutional Neural Networks", International Conference on Intelligent Computer Communication and Processing, pp. 273-278, 2017.

[Itu2018] - R. Itu, R. Danescu, "Machine Learning Based Automatic Extrinsic Calibration of an Onboard Monocular Camera for Driving Assistance Applications on Smart Mobile Devices", International Forum on Advanced Microsystems for Automotive Applications, pp. 16-28, 2018.

[Jegou2016] - S. Jegou, M. Drozdzal, D. Vazquez, A. Romero, Y. Bengio, “The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation”, arXiv:1611.09326, 2016.

[Kalman1960] - R. E. Kalman, “A New Approach to Linear Filtering and Prediction Problems”, Journal of Basic Engineering, Transactions of the ASME, ser. D, vol. 82, no. 1, pp. 35-45, 1960.

[Kalman1961] - R. E. Kalman, R. S. Bucy, “New Results in Linear Filtering and Prediction Theory”, Journal of Basic Engineering, Transactions of the ASME, vol. 83, no. 3, pp. 95-107, 1961.

[Keng2017] - H. C. Keng, “Carvana Image Masking Challenge”, 2017, online la adresa: https://www.kaggle.com/c/carvana-image-masking-challenge/discussion/37208

[Keras] - Keras, accesat 2019, online la adresa: https://keras.io/

[Kingma2015] - D. Kingma, J. Ba, “Adam: A method for stochastic optimization”, International Conference for Learning Representations, 2015.

[Kluger2018] - F. Kluger, H. Ackermann, M. Y. Yang, B. Rosenhahn, “Deep learning for vanishing point detection using an inverse gnomonic projection”, German Conference on Pattern Recognition, pp. 17-28, 2017.

[Kong2009] - H. Kong, J.-Y. Audibert, J. Ponce, “Vanishing point detection for road detection”, Computer Vision and Pattern Recognition, pp. 96-103, 2009.

[Korosec2018] - K. Korosec, “Waymo’s autonomous vehicles are driving 25,000 miles every day”, TechCrunch, online la adresa: https://techcrunch.com/2018/07/20/waymos-autonomous-vehicles-are-driving-25000-miles-every-day/

[Krizhevsky2012] - A. Krizhevsky, I. Sutskever, G. E. Hinton, “Imagenet classification with deep convolutional neural networks”, Advances in Neural Information Processing Systems, pp. 1097-1105, 2012.

[Krähenbühl2011] - P. Krähenbühl, V. Koltun, “Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials”, Advances in Neural Information Processing Systems, pp. 109-117, 2011.

[Lambert2018] - F. Lambert, “Tesla’s fleet has accumulated over 1.2 billion miles on Autopilot and even more in shadow mode”, Electrek, 2018, online la adresa: https://electrek.co/2018/07/17/tesla-autopilot-miles-shadow-mode-report/

[LeCun1995] - Y. Le Cun, Y. Bengio, “Convolutional networks for images, speech, and time series”, The Handbook of Brain Theory and Neural Networks, Cambridge, MIT Press, pp. 255-258, 1995.

[Levinson2013] - J. Levinson, S. Thrun, “Automatic online calibration of cameras and lasers”, Robotics Science Systems Conference, pp. 1-8, 2013.

[Lin2014] - T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollar, C. L. Zitnick, "Microsoft COCO: Common objects in context", European Conference on Computer Vision, pp. 740-755, 2014.

[Long2015] - J. Long, E. Shelhamer, T. Darrell, “Fully convolutional networks for semantic segmentation”, Computer Vision and Pattern Recognition, pp. 3431-3440, 2015.

[Lowe2004] - D.G. Lowe, “Distinctive image features from scale-invariant keypoints”, International Journal of Computer Vision, vol. 60, no. 2, pp. 91-110, 2004.

[Lucas1981] - B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision", Proceedings of Imaging Understanding Workshop, pp. 121-130, 1981.

[Lynch1991] - M. Lynch, C. Dagli, "Backpropagation neural net- work for stereoscopic vision calibration", International Society for Optics and Photonics, vol. 1615, pp. 289-298, 1991.

[Magee1984] - M. Magee, J. Aggarwal, “Determining vanishing points from perspective images”, Computer Vision, Graphics, and Image Processing, vol. 26, pp. 256-267, 1984.

[Mallot1991] - H. A. Mallot, H. H. Bulthoff, J. J. Little, S. Bohrer, “Inverse perspective mapping simplifies optical flow computation and obstacle detection”, Biological Cybernetics, vol. 64, pp. 177-185, 1991.

[Martins1981] - H. A. Martins, J. R. Birk, R. B. Kelly, “Camera models based on data from two calibration planes”, Computer Graphics and Image Processing, vol. 17, pp. 173-180, 1981.

[Masoud2004] - O. Masoud, N. Papanikolopoulos, “Using geometric primitives to calibrate traffic scenes”, IEEE/RSJ International Conference on Intelligent Robots and Systems, vol. 2, pp. 1878-1883, 2004.

[Matlab2019] - Matlab Calibration Toolbox, accesat 2019, online la adresa: http://www.vision.caltech.edu/bouguetj/calib_doc/

[Minsky1969] - M. L. Minsky, S. A. Papert, “Perceptrons”, Cambridge, MIT Press, 1969.

[Mocan2018] - I. Mocan, R. Itu, A. Ciurte, R. Danescu, R. Buiga, "Automatic Detection of Tumor Cells in Microscopic Images of Unstained Blood using Convolutional Neural Networks", 2018 IEEE 14th International Conference on Intelligent Computer Communication and Processing (ICCP 2018), pp. 319-324.

[Moghadam2012] - P. Moghadam, J.A. Starzyk, W.S. Wijesoma, “Fast vanishing-point detection in unstructured environments”, IEEE Transactions on Image Processing, vol. 21, pp. 425-430, 2012.

[More1977] - J. More, “The Levenberg-Marquardt algorithm, implementation and theory”, Numerical Analysis, pp. 105-116, 1977.

[Moreno2012] - D. Moreno, G. Taubin, “Simple, Accurate, and Robust Projector-Camera Calibration”, International Conference on 3D Imaging, Modeling, Processing, Visualization & Transmission, pp. 464-471, 2012.

[Mousavian2017] - A. Mousavian, D. Anguelov, J. Flynn, J. Kosecka, “3D Bounding Box Estimation Using Deep Learning and Geometry”, Computer Vision and Pattern Recognition, pp. 5632-5640, 2017.

[Movon] - Movon Corporation, “Movon FCW”, accesat 2019, online la adresa: https://play.google.com/store/apps/details?id=com.movon.fcw

[NCSA2018] - National Center for Statistics and Analysis, “2017 fatal motor vehicle crashes: Overview”, Traffic Safety Facts Research Note, Report No. DOT HS 812 603, 2018, online la adresa: https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/812603

[Nair2010] - V. Nair, G. E. Hinton, “Rectified linear units improve restricted boltzmann machines”, International Conference on Machine Learning, pp. 807-814, 2010.

[Neuhold2017] - G. Neuhold, T. Ollmann, S. R. Bulò, P. Kontschieder, "The Mapillary Vistas Dataset for Semantic Understanding of Street Scenes", IEEE International Conference on Computer Vision, pp. 5000-5009, 2017.

[Nieto2007] - M. Nieto, L. Salgado, F. Jaureguizar, J. Cabrera, “Stabilization of Inverse Perspective Mapping Images Based on Robust Vanishing Point Estimation”, Proceedings of IEEE Conference on Intelligent Vehicles Symposium, pp. 315-320, 2007.

[Nock2004] - R. Nock, F. Nielsen, “Statistical region merging”, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 26, no. 11, pp. 1452-1458, 2004.

[OpenCV] - OpenCV, accesat 2019, online la adresa: https://opencv.org/

[OpenStreetCam] - Open Street Cam, accesat 2019, online la adresa: https://www.openstreetcam.org/

[Paszke2016] - A. Paszke, A. Chaurasia, S. Kim, E. Culurciello, "ENet: A Deep Neural Network Architecture for Real-Time Semantic Segmentation", arXiv: 1606.02147, 2016.

[Pereira2016] - M. Pereira, D. Silva, V.M.F. Santos, P. Dias, “Self calibration of multiple LIDARs and cameras on autonomous vehicles”, Robotics and Autonomous Systems, vol. 83, pp. 326-337, 2016.

[Pomerleau1989] - D. Pomerleau, “ALVINN: an autonomous land vehicle in a neural network”, Advances in Neural Information Processing Systems, 1989.

[Pomerleau1995] - D. Pomerleau, “RALPH: Rapidly adapting lateral position handler”, IEEE Intelligent Vehicles, pp. 506-511, 1995.

[Qt] - Qt, “Qt Creator”, accesat 2019, online la adresa: https://www.qt.io/

[Ren2015] - S. Ren, K. He, R. B. Girshick, J. Sun, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”, Neural Information Processing Systems, pp. 91-99, 2015.

[Ronneberger2015] - O. Ronneberger, P. Fischer, T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation", Medical Image Computing and Computer-Assisted Intervention (MICCAI), Springer, Vol. 9351, pp. 234-241, 2015.

[Rosenblatt1958] - F. Rosenblatt, “The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain”, Cornell Aeronautical Laboratory, Psychological Review, vol. 65, no. 6, pp. 386-408, 1958.

[Rosten2005] - E. Rosten, T. Drummond, “Fusing points and lines for high performance tracking”, International Conference on Computer Vision, vol. 2, pp. 1508-1515, 2005.

[Rosten2010] - E. Rosten, R. Porter, T. Drummond, "Faster and better: A machine learning approach to corner detection", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 32, no. 1, pp. 105-119, 2010.

[SAE2018] - Society of Automotive Engineers, “Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles”, 2018, online la adresa: https://www.sae.org/standards/content/j3016_201806/

[Santana2016] - E. Santana, G. Hotz, "Learning a driving simulator", arXiv:1608.01230, 2016.

[Shannon1949] - C.E. Shannon, W. Weaver, “The Mathematical Theory of Communication”, University of Illinois Press, 1949

[Shotton2008] - J. Shotton, M. Johnson, R. Cipolla, “Semantic texton forests for image categorization and segmentation”, Computer Vision and Pattern Recognition, pp. 1-8, 2008.

[Shotton2011] - J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, A. Blake, “Real-time human pose recognition in parts from a single depth image”, Computer Vision and Pattern Recognition, pp. 1297-1304, 2011.

[Simonyan2014] - K. Simonyan, A. Zisserman, “Very deep convolutional networks for large-scale image recognition”, arXiv:1409.1556, 2014.

[Sistu2019] - G. Sistu, I. Leang, S. Chennupati, S. Milz, S. Yogamani, S. Rawashdeh, “NeurAll: Towards a Unified Model for Visual Perception in Automated Driving”, arXiv:1902.03589, 2019.

[Strat1984] - T. M. Strat, “Recovering the camera parameters from a transformation matrix”, DARPA Image Understanding Workshop, pp. 264-271, 1984.

[Suttorp2006] - T. Suttorp, T. Bucher, “Robust vanishing point estimation for driver assistance”, in Proc. IEEE Intelligent Transportation Systems Conference, pp. 1550-1555, 2006.

[Swerling1959] - P. Swerling, “First-order error propagation in a stagewise smoothing procedure for satellite observations”, J. Astronaut. Sci., vol. 6, pp. 46-52, 1959.

[Sørensen1948] - T. Sørensen, "A method of establishing groups of equal amplitude in plant sociology based on similarity of species and its application to analyses of the vegetation on Danish commons", Kongelige Danske Videnskabernes Selskab, vol. 5, no. 4, pp. 1-34, 1948.

[Tan2006] - S. Tan, J.L. Dale, A. Anderson, A. Johnston, “Inverse perspective mapping and optic flow: A calibration method and a quantitative analysis”, Image and Vision Computing, vol. 24, no. 2, pp. 153-165, 2006.

[Tensor] - TensorFlow, accesat 2019, online la adresa: https://www.tensorflow.org/

[Tesla2019] - Tesla, “Tesla Autopilot”, accesat 2019, online la adresa: https://www.tesla.com/autopilot

[Thrun2005] - S. Thrun, W. Burgard, D. Fox, “Probabilistic Robotics”, MIT Press, Cambridge 2005.

[Thrun2006] - S. Thrun, et. al., “Stanley: The robot that won the DARPA Grand Challenge”, Journal of Field Robotics, vol. 23, no. 9, pp. 661-692, 2006.

[Tsai1986] - R. Y. Tsai, “An efficient and accurate camara calibration technique for 3Dmachine vision”, Computer Vision and Pattern Recognition, pp. 364-374, 1986.

[Udacity2019] - Udacity Vehicle dataset, accesat 2019, online la adresa: https://github.com/udacity/self-driving-car/tree/master/annotations

[Uijlings2013] - J. R. Uijlings, K. E. van de Sande, T. Gevers, A. W. Smeulders, ”Selective search for object recognition”, International Journal of Computer Vision, vol. 104, no. 2, pp. 154-171, 2013.

[Urmson2007] - Chris Urmson, et. al., “Tartan racing: A multi-modal approach to the DARPA Urban Challenge”, Technical report, Carnegie Mellon University, 2007.

[Wahab2011] - M. N. A. Wahab, N. Sivadev, and K. Sundaraj, "Development of monocular vision system for depth estimation in mobile robot - Robot soccer", Sustainable Utilization and Development in Engineering and Technology, pp. 36-41, 2011.

[Wang2004] - Y. Wang, E. K. Teoh, D. Shen, “Lane detection and tracking using b-snake", Image and Vision Computing, pp. 269–280, 2004.

[Wang2007] - K. Wang, H. Huang, Y. Li, F.Y. Wang, “Research on lane-marking line based camera calibration”, International Conference on Vehicular Electronics and Safety, pp. 1-6, 2007.

[Waugh2013] - R. Waugh, “How the first driverless car was invented in Britain in 1960”, Yahoo News, 2013, online la adresa: https://uk.news.yahoo.com/how-the-first--driverless-car--was-invented-in-britain-in-1960-093127757.html

[Wei2016] - L. Wei, et. al., “SSD: Single Shot MultiBox Detector”, European Conference on Computer Vision, pp. 21-37, 2016.

[Werbos1974] - P. Werbos, “Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences”, PhD thesis, Harvard University, 1974.

[Wildenauer2012] - H. Wildenauer, A. Hanbury, “Robust camera selfcalibration from monocular images of Manhattan worlds”, Computer Vision and Pattern Recognition, pp. 2831-2838, 2012.

[Wu2016] - Z. Wu, W. Fu, R. Xue, W. Wang, “A Novel Line Space Voting Method for Vanishing-Point Detection of General Road Images”, Sensors, vol. 16, no. 7, art. no. 948, 2016.

[Xie2016] - J. Xie, R. Girshick, A. Farhadi, “Deep3d: Fully automatic 2d-to-3d video conversion with deep convolutional neural networks”, European Conference on Computer Vision, pp. 842-857, 2016.

[Yakimovsky1978] - Y. Yakimovsky, R. Cunningham, “A system for extracting three dimensional measurements from a stereo pair of TV cameras”, Computer Graphics and Image Processing, vol. 7, pp. 195-210, 1978.

[Yoon2019] - D.E. Yoon, H.I. Choi, "Method for Road Vanishing Point Detection Using DNN and Hog Feature", Journal of the Korea Contents Association, vol. 19, no.1, pp. 125-131, 2019.

[Yu2015] - F. Yu, V. Koltun, “Multi-Scale Context Aggregation by Dilated Convolutions”, arXiv: 1511.07122, 2015.

[Yu2018] - F. Yu, W. Xian, Y. Chen, F. Liu, M. Liao, V. Madhavan, T. Darrell, “BDD100K: A Diverse Driving Video Database with Scalable Annotation Tooling”, arXiv: 1805.04687, 2018.

[Zhang2000] - Z. Zhang, “A flexible new technique for camera calibration”, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, no. 11, pp. 1330-1334, 2000.

[Zhang2006] - S. Zhang, P. S. Huang, “Novel method for structured light system calibration”, Optical Engineering, vol. 45, no. 8 (083601), 2006.

[Zhuang2003] - Y. Zhuang, X. Xu, X. Pan, W. Wang, "Mobile robot indoor navigation using laser range finder and monocular vision", Robotics, Intelligent Systems and Signal Processing, vol. 1, pp. 77-82, 2003.