UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

Tehnici avansate de percepție a mediului folosind Deep Learning și estimatori probabilistici (DEEPSENSE)

Echipa de cercetare: Radu Dănescu, Răzvan Itu, Diana Borza, Andrei Vadan

Raport final

Cuprins

Introducere

Percepția mediului, în cazul sistemelor de asistență a șofatului sau a autovehiculelor autonome, este una dintre cele mai dificile sarcini, și de îndeplinirea cu succes a acesteia depind deciziile care pot face diferența dintre siguranță și dezastru. Deși șoferul uman se descurcă destul de bine doar cu doi ochi, sistemele computerizate se bazează, în mod tradițional, pe o mare gamă de senzori activi și pasivi, pentru a acoperi cât mai multe situații. Recent, apariția metodelor de învățare profundă (Deep Learning) a început să permită și sistemelor computerizate să pună tot mai mult accent pe informația vizuală, capturată prin intermediul camerelor video, nemaifiind nevoie de algoritmi specifici pentru extragerea fiecărei trăsături sau tip de obiect, ci doar de date multe și de putere de calcul. În același timp, realitatea se supune unor reguli clare, ce pot fi modelate matematic. Aceste modele, ce includ aspectul static al scenei, dinamica părților scenei, precum și incertitudinile asupra a ceea ce cunoaștem, sunt folosite în procesul de estimare continuă, numit și urmărire (tracking). Găsirea celor mai bune modele, și a algoritmilor pentru estimarea parametrilor lor, este un domeniu de cercetare important în robotică și în domeniul sistemelor de asistență a șofatului.

Proiectul DeepSense s-a aliniat eforturilor comunității științifice internaționale de a găsi noi soluții pentru percepția mediului de trafic auto, și a explorat cele două direcții principale: utilizarea de rețele convoluționale pentru detecție/segmentare, și modelarea și urmărirea

probabilistică pentru estimare continuă. Ca parte a acestui efort, s-au proiectat, antrenat și testat diferite soluții de rețele neuronale pentru extragerea de informații utile din imagini, și au fost proiectate noi modele pentru reprezentarea lumii, și algoritmi pentru estimarea parametrilor lor, folosind ca date de intrare rezultatele date de rețelele neuronale. Efortul a fost concentrat pe utilizarea secvențelor de imagini dintr-o singură sursă (sistem monocular de viziune), deoarece această configurație este ușor de folosit și beneficiază de disponibilitatea unei mari cantități de date în bazele de date internaționale.

Proiectul s-a desfășurat pe durata a 24 de luni, în 3 etape. În etapa 1 au fost analizate soluțiile de detecție existente, și au fost colectate și pregătite date pentru antrenarea soluțiilor proprii pentru segmentare bazată pe CNN. O atenție particulară a fost dedicată calibrării camerei, pentru că s-a dorit ca în final să se ajungă la un sistem autocalibrat, care să poată fi utilizat pe orice secvență de imagini.

În etapa 2 au fost elaborate majoritatea soluțiilor tehnice ale proiectului. Au fost proiectate și implementate mai multe rețele neuronale, pentru extragerea de multiple tipuri de informații din imagine. A fost proiectat un model probabilistic pentru integrarea măsurătorilor CNN în sistemul de urmărire a obiectelor bazat pe particule mobile în harta de ocupare, o soluție proiectată anterior tot de cercetătorii din echipa proiectului, și a fost dezvoltată o soluție funcțională pentru calibrarea automată a parametrilor camerei. A fost de asemenea începută dezvoltarea unei aplicații demonstrator, pentru achiziție de date proprii și pentru testarea algoritmilor.

În etapa finală au fost aduse îmbunătățiri rețelelor neuronale folosite, a fost experimentat un model cu multiple ieșiri și trunchi comun, și a fost extins modelul probabilistic pentru urmărirea spațiului liber și a identității obiectelor. De asemenea, a fost definitivată aplicația demonstrator, și au fost efectuate multiple teste.

În toate etapele proiectului au fost publicate lucrări în vederea diseminării rezultatelor, astfel că în final au fost publicate două articole de jurnal ISI, două articole de conferință indexate ISI, și două articole de conferințe BDI (volume Springer). Pe baza rezultatelor obținute a fost finalizată și o teză de doctorat a unui membru din echipa proiectului.

Etapa 1 - Analiza cerințelor, pregătirea datelor și a rețelelor neuronale

1. Colectarea imaginilor de trafic, ingineria și selecția datelor, adnotare

Pentru realizarea obiectivelor acestui proiect, au fost folosite 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 conțin imagini de dimensiuni diferite, și au datele de etichetă stocate în formate

diferite. În tabelul 1.1. se prezintă o comparație din mai multe puncte de vedere (dimensiune, scenarii) a bazelor de date existente pentru analiza automată a imaginilor de trafic.

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
BDD100K	100000	120000000	Da	Da	Da	Da

Tabelul 1.1. Prezentare comparativă a bazelor de date cu imagini de trafic

1.1. Baze de date pentru detecția obstacolelor

Baza de date KITTI [1] a fost creată de Karlsruhe Institute of Technology din Germania și oferă diferite seturi de date cu aplicații multiple: baza de date pentru obstacole (vehicule, autobuze, pietoni, etc.), segmentare, urmărire obstacole. Baza de date cu obstacole conține un total de 7481 imagini de antrenare cu vehicule și 7518 imagini de test, conținând un total de peste 80.000 de obiecte etichetate.

Baza de date oferită de Udacity [2] pentru detecția de obstacole conține imagini din traficul rutier împreună cu fișiere de tip “csv” asociate fiecărei imagini. În fișierele text sunt stocate informațiile despre locația obstacolelor în spațiul imaginii și sunt definite coordonatele dreptunghiului care le încadrează: coordonatele punctului din colțul stânga sus: (xmin, ymin) și colțul din dreapta jos: (xmax, ymax) al dreptunghiului, numele imaginii, tipul clasei etichetate. Opțional în fișiere text mai este un câmp numit “occluded” care va avea valoarea 0 dacă obstacolul nu este obstrucționat și valoarea 1 în caz contrar.

1.2. Baze de date pentru segmentare semantică

Baza de date CityScapes [3] cuprinde secvențe din diferite orașe din Germania și Elveția (ex: Aachen, Koln, Dusseldorf, Zurich). Datele sunt din traficul rutier urban în 5000 de imagini și cu 30 de clase de obiecte diferite. Imaginile sunt stocate pe 3 canale, de dimensiune 2048 x 1024 pixeli (un raport de 2:1). În imaginile adnotate culoarea clasei drumului este: (128, 64, 128). Baza de date CityScapes conține 2975 imagini adnotate cu carosabil.

Baza de date KITTI [1] adnotată cu zone de drum conține doar 289 imagini din scene din trafic în orașul german Karlsruhe. Drumul este marcat folosind culoarea: (255, 0 255) în imaginile adnotate.

Baza de date Berkeley Deep Drive [4] pentru segmentare conține peste 100.000 imagini etichetate. Dimensiunea imaginilor este de 1280 x 720 pixeli pe 3 canale color (RGB). Imaginile de etichetă sunt de aceeași dimensiune, unde fiecare clasă are asignata câte o culoare diferită. Pentru drum, culoarea este: (128, 64, 128). Datele sunt stocate pe 8 biți, ceea

ce înseamnă că intensitatea pe fiecare canal poate avea valori cuprinse între 0 și 255. Baza de date conține un total de 6575 imagini adnotate.

1.3. Definirea unui format standard pentru bazele de date

Format standard pentru segmentare semantica:

Prin folosirea și definirea unui format standard pentru toate cele 3 baze de date pentru segmentare, am ajuns la un total de 9839 imagini, din care 7871 de antrenare și 1968 imagini de test (validare). 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.

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.

Format standard pentru detecția obstacolelor:

Pentru a putea procesa toate bazele de date pentru obstacole, am ales definirea unei perechi imagine - fișier text. Toate bazele de date existente pentru detecție de obstacole au fost convertite în formatul comun definit în această secțiune. În cazul în care bazele de date nu dispun de toate informațiile senzoriale, au fost scrise valori de 0. De asemenea, în această etapă am implementat și un sistem propriu de achiziție, cu același format de date, folosind dispozitive mobile rulând sistemul de operare Android. Achiziția imaginilor este realizată prin camera foto a dispozitivului, iar restul de date senzoriale sunt preluate din senzorii disponibili pe dispozitivul mobil. Imaginile sunt salvate în format RGB pe 8 biți.

Fișierele text au următoarea structură:

- timestamp: UNIX timestamp - timpul achiziționării imaginii
- accX, accY, accZ: accelerația gravitațională în jurul axelor x, y și z
- gyroX, gyroY, gyroZ: rata de rotație a dispozitivului în jurul axelor x, y și z (radiani/sec)
- magX, magY, magZ: rotația din senzorul geo-magnetic în jurul axelor x, y și z
- yaw: unghiul de rotație în jurul axei x (rotație laterală a vehiculului de-a lungul axei drumului)
- pitch: unghiul de înclinație a dispozitivului mobil
- roll: unghiul de rotație în jurul axei y (rotația dispozitivului mobil pe axa verticală a drumului)
- yawRate: rata de rotație în jurul axei x (aceeași cu gyroX)
- speedMs: viteza de deplasare (metri / secundă)
- lat, long: coordonata de latitudine și longitudine obținută din senzorul GPS

Cele 3 unghiuri de rotație (yaw, pitch și roll) au fost obținute din valorile senzorului geo-magnetic și cel de accelerație gravitațională. Acest format de date poate fi accesat și procesat “offline” pe sisteme desktop.

2. Proiectarea, antrenarea și evaluarea rețelelor neuronale, și reglarea fină a hiperparametrilor

Inteligența artificială reprezintă un domeniu activ de interes și cercetare, mai ales în ultimii ani datorită popularității rețelelor neuronale convolutionale și apariției unor metode de a le antrena folosind hardware accesibil (plăci video grafice). În cadrul acestei etape am proiectat și antrenat o rețea neuronală convoluțională pentru segmentarea semantică a imaginilor din trafic. În această fază am ales segmentarea zonei de drum ("driveable road area") din imagini. Deși în această etapă am procesat și adus la un format comun și bazele de date pentru detecție de obstacole, în prima fază am implementat doar partea de detecție de carosabil, urmând ca în etapele următoare să implementăm o altă rețea neuronală pentru a detecta obstacolele.

Pentru segmentare am ales o arhitectură de rețea de tipul "encoder-decoder". Am ales folosirea unei rețele U-NET [5] modificată. Prima parte, cea de codificare, are rolul de a extrage informații relevante despre trăsături și caracteristici ale imaginii de intrare, în timp ce reduce dimensiunea spațială (datorită aplicării operațiilor de convoluție cu filtre având dimensiune de nucleu variabilă). Informațiile despre localizare sunt astfel pierdute, dar prin introducerea unor legături directe între nivelurile de codificare și cele de decodificare se va rezolva problema localizării trăsăturilor. Partea de decodificare, prin folosirea operațiilor de deconvoluție, are rolul de a reconstrui harta de segmentare. În rețelele de tip U-NET, partea de decodificare are același număr de niveluri ca și cea de codificare (figura 1.1).

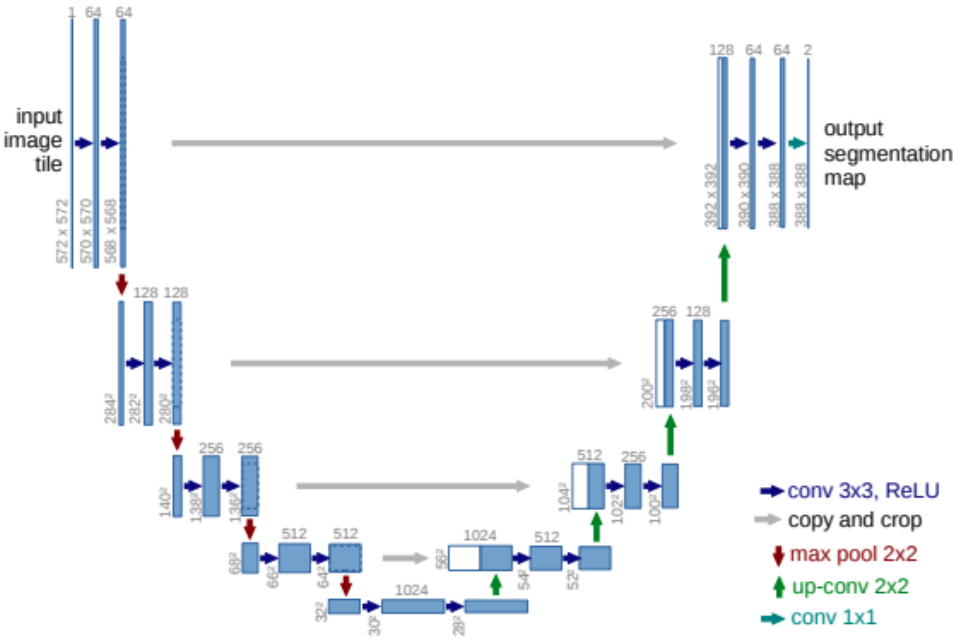


Figura 1.1. Arhitectura clasica a retelei U-NET. Sursa: https://arxiv.org/pdf/1505.04597.pdf

Rețeaua de tip U-NET folosită în această etapă are ca intrare perechi de imagini de dimensiune 256 x 256 pixeli. Predicția (ieșirea rețelei), adică imaginea segmentată, va fi de aceeași dimensiune. Perechile de imagini de intrare sunt stocate pe disc astfel: imaginea originală (cu scena de trafic) este de tip întreg pe 8 biți pe 3 canale de culoare, în timp ce

imaginea de etichetă (mască) este pe 1 canal pe 8 biți și au valoarea intensității 255 pentru carosabil, iar 0 pentru restul. Intensitățile din imaginile de etichetă sunt normalizate în intervalul [0, 1] pentru o antrenare mai eficientă. O pereche de imagini de intrare pentru rețea este ilustrată în figura 1.2.

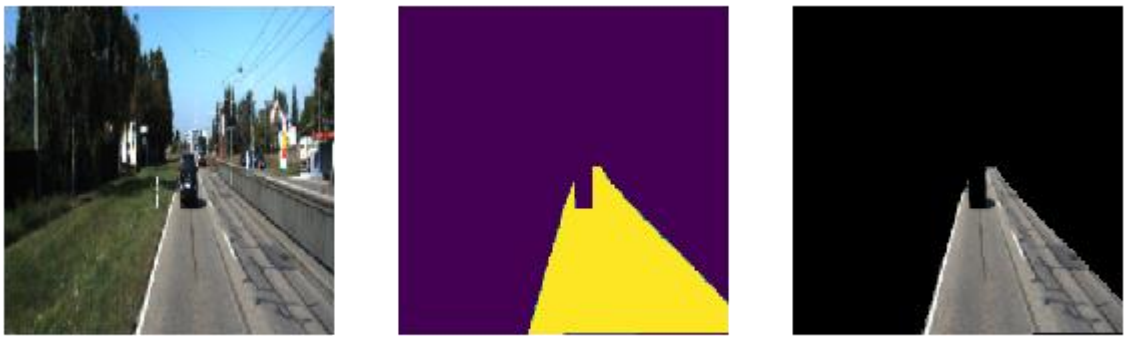


Figura 1.2. Date de intrare pentru rețea: stânga imaginea originală, centru: imaginea de etichetă, dreapta: rezultatul aplicării etichetei ca mască pe imaginea originală

În procesul de antrenare am ales minimizarea funcției de cost “intersect over union”. Implementarea funcției de cost a fost făcută prin utilizarea coeficientul Sorensen-Dice (“dice loss”).

Antrenarea unei rețele convoluționale este facilitată prin utilizarea mai multor date de antrenare. Pentru extinderea bazelor de date existente am ales să aplicăm procesul de augmentare a datelor. Am folosit următoarele tehnici de augmentare: variații aleatorii ale intensității imaginilor de intrare (în spațiul de culoare HSV), translatări și rotații aleatorii ale imaginilor, scalarea lor și oglindirea pe orizontală (“horizontal flipping”). Pentru antrenarea și predicția folosind rețele neuronale am folosit limbajul de programare Python și software-ul TensorFlow [6] și Keras [7]. Pre-procesarea imaginilor și generarea bazelor de date modificate, iar apoi generarea imaginilor augmentare am realizat-o folosind software-ul OpenCV [8] pentru Python.

Rețeaua aleasă are următoarea structură: 5 straturi de codificare, 1 strat central și 5 pentru decodificare. Un strat de codificare are următoarele operații:

- convoluție
- “batch normalization”
- funcție de activare ReLU
- convoluție
- “batch normalization”
- activare ReLU
- “max pooling”

Stratul central al rețelei are aceleași operații ca și cel de codificare, cu excepția operației de “max pooling” de la final.

Stratul de decodificare are urmatoarele operații:

- deconvoluție (“up sampling”)
- concatenare (legatura directă cu stratul echivalent din codificare)
- deconvoluție

- “batch normalization”
- activare ReLU
- deconvoluție
- “batch normalization”
- activare ReLU
- deconvoluție
- “batch normalization”
- activare ReLU

Predicția finală va fi rezultatul aplicării a unei convoluții cu nucleu 1 x 1 și a unei activări sigmoide.

Pentru antrenare am folosit o dimensiune a lotului de 16 și un număr maxim de 50 de epoci. Timpul de antrenare pentru o epocă este de 172 secunde (în medie) pe un sistem desktop echipat cu 2 plăci video Nvidia 1080 Ti. O analiză a antrenării este ilustrată în figura 1.3, unde se observă minimizarea funcției de cost (“loss”) și convergența coeficientului Dice spre valoarea 1 (în metrică IoU utilizată, valoarea 1 reprezintă o suprapunere perfectă a datelor).

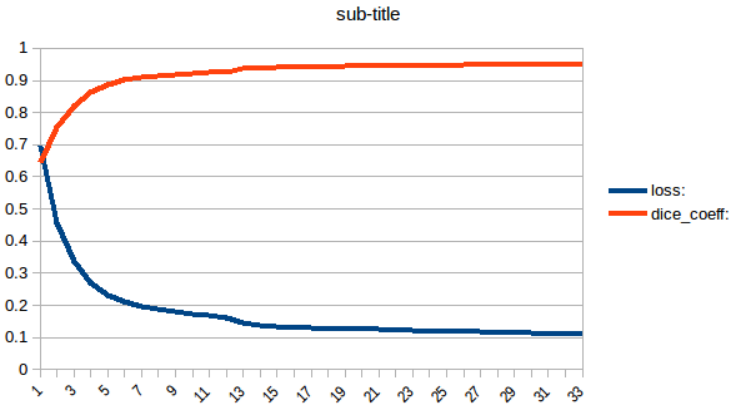


Figura 1.3. Funcția de cost (“loss”) scade, iar coeficientul Sorensen-Dice (“dice coeff”) converge spre valoarea 1 după antrenarea timp de 33 epoci.

Un exemplu de rezultat al segmentării folosind rețeaua neuronală este ilustrat în figura 1.4:

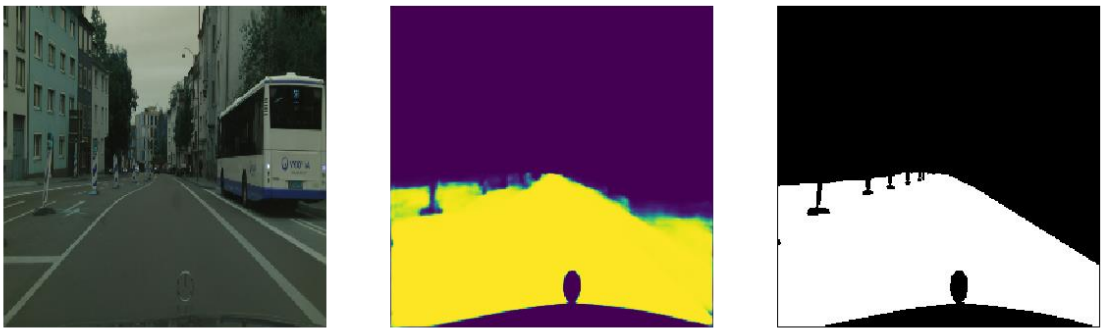


Figura 1.4. Exemple de predicție a rețelei. Stânga: imaginea de intrare color, centru: rezultatul segmentării (predicția), dreapta: imaginea etichetată (“ground truth”).

3. Detecția și măsurarea mișcării folosind rețele neuronale convoluționale

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 vehicule sau pietoni. Pentru implementarea acestei funcționalități am ales folosirea unei hărți de ocupare bazată pe un filtru de particule. Aceste abordări pot fi folosite pentru estimarea sistemelor neliniare, multi-modale.

Harta de ocupare a scenei bazată pe filtre de particule este o reprezentare dinamică a scenei unde efectul de perspectivă este eliminat. În implementarea din cadrul acestui proiect, obstacolele sunt compuse din particule, fiecare având o poziție și un vector de viteză. Harta de ocupare este împărțită în celule de 20 x 20 cm, iar probabilitatea unei celule de a fi ocupată este dată de numărul de particule din acea celulă. Totodată, particulele pot avea un rol dublu: de a forma ipoteze pentru posibile obiecte, 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). Partea de predicție a stării este realizată prin deplasarea particulelor.

Obstacolele din scenă sunt reprezentate de un set de particule, unde fiecare particulă are o poziție în harta de ocupare, o componentă de viteză dar și o variabilă pentru “vârsta” particulei din momentul creării ei. Acest parametru de “vârstă” este folosit în procesul de validare și estimare a vitezei pentru particulă, deoarece sunt luate în considerare numai particulele care “supraviețuiesc” pentru mai multe cadre. Probabilitatea unei celule de a fi ocupată este estimată ca fiind raportul dintre numărul de particule din celula (poziția lor coincide cu poziția celulei) și numărul total de particule permise într-o singură celulă *“Nc”*. Acest parametru este constant și a fost setat la 200 de particule totale într-o celulă. Alegerea acestei valori reprezintă un compromis între acuratețe și performanța sistemului: un număr mare de particule într-o celulă înseamnă că se mențin mai multe ipoteze de viteză simultan, iar sistemul de urmărire va gestiona mai bine obstacolele care deplasează rapid în scenă, dar viteza de execuție a algoritmului va scădea. Viteza unei celule din harta de ocupare este calculată ca fiind viteza medie a particulelor din acea celulă.

Primul pas al algoritmului de urmărire este cel de predicție și este aplicat pe fiecare particulă. Poziția unei particule este modificată în funcție de viteza ei proprie și în funcție de parametri de mișcare ai ego-vehiculului (viteza și rata de rotație “yaw rate”). Al doilea pas al algoritmului este procesul de actualizare, bazat pe harta binară de ocupare a celulelor creată prin procesarea imaginii segmentate.

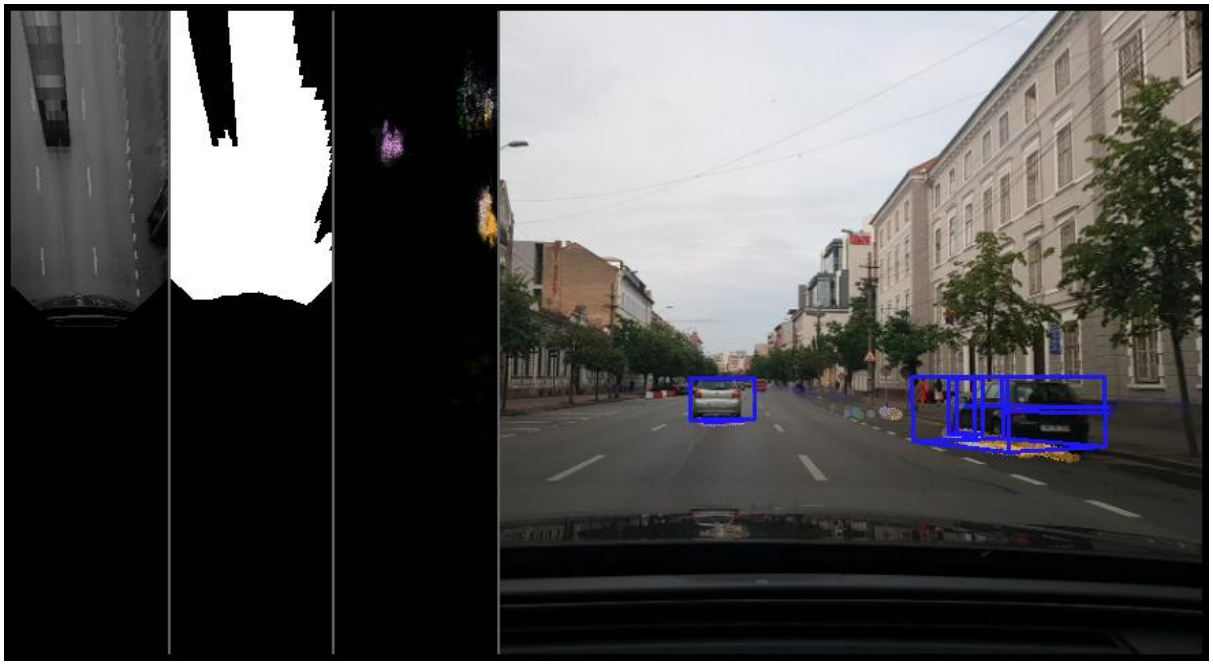


Figura 1.5. Framework-ul probabilistic de percepție a scenei.

Figura 1.5 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 viteza lor codificată în funcție de vectorii de viteză (albastru - obiect în mișcare, galben - obiect static). În imaginea din dreapta este prezentat rezultatul procesării și grupării particulelor și a celulelor în obstacole.

4. Rezultate suplimentare: Calibrarea automată a parametrilor extrinseci ai camerei

Unul dintre deziderateleacestui proiect este de a oferi un sistem de percepție a mediului care să poată funcționa cu resurse senzoriale limitate, inteligența artificială suplinind absența unor senzori costisitori dar foarte preciși, precum laserscannerele sau radarele. Șoferul va putea monta o cameră sau un dispozitiv mobil în spatele parbrizului, camera fiind orientată spre traficul din exterior, dar pentru a putea utiliza informațiile de la această cameră trebuie ca trăsăturile din imaginile achiziționate să poată fi puse în legătură cu trăsăturile 3D ale lumii reale. Acest lucru implică procesul de calibrare, ceea ce un utilizator obișnuit ar prefera să nu fie obligat să facă.

În mod tradițional, calibrarea se face în medii controlate, în condiții de laborator, de obicei prin măsurarea manuală a unor obiecte de referință. Dacă ne referim la scenariul în care utilizatorul pur și simplu montează camera în mașină, dorind apoi să pornească la drum, condițiile de laborator nu pot fi îndeplinite, și astfel avem nevoie de calibrare automată.

Pentru calibrarea automată, avem nevoie de repere cu dimensiuni 3D cunoscute, precum lățimea unei benzi de circulație sau dimensiunea unor obstacole cunoscute. Noi am

ales varianta a doua, și ne vom folosi de avantajele rețelelor neuronale pentru a selecta doar obstacolele de tip autovehicul din imagini necalibrate. Soluția pentru detecția obstacolelor este bazată pe o CNN pre-antrenată cu exemple formate din perechi imagini-obstacole reprezentate ca un dreptunghi în spațiul imagine. Am folosit API-ul TensorFlow pentru simplificarea procesului de antrenare, și am re-antrenat o soluție existentă de detecție a obstacolelor din imagini singulare numită MobileNet [10], care este o rețea proiectată în mod explicit pentru a avea un număr redus de parametri de antrenare, și care necesită putere de calcul redusă pentru predicție, putând astfel fi utilizată în timp real pe dispozitive mobile. Un exemplu de detecție automată a acestor obiecte este dat în figura 1.6.

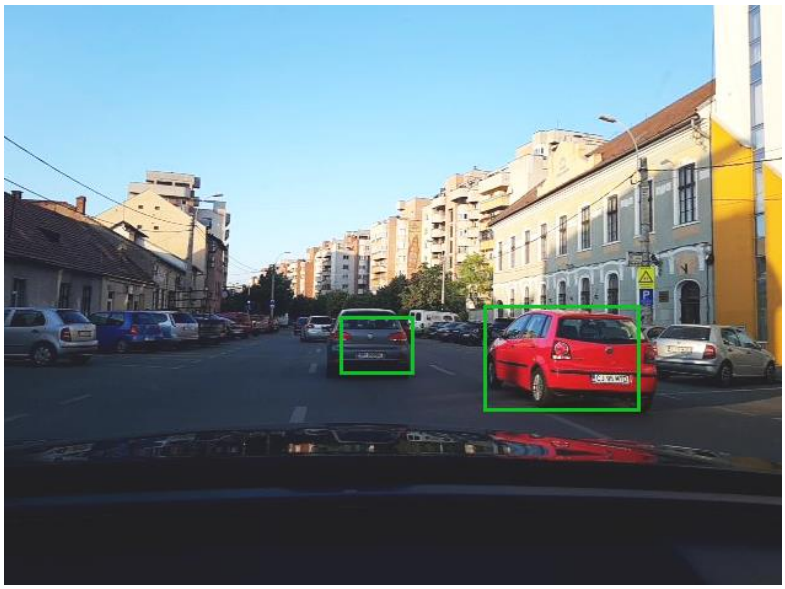


Figura 1.6. Exemplu de autovehicule detectate automat de MobileNet.

În funcție de distanța unui autovehicul față de noi, dimensiunea lui aparentă în imagine va fi variabilă. Distanța în lumea reală va avea o legătură directă cu coordonata rândului din imagine, iar obiectul va avea dimensiunea zero (va dispărea) când coordonata limitei lui inferioare va fi egală cu linia orizontului (linia de fugă, “vanishing line”). De fapt, între coordonata rândului limitei inferioare a obiectului și dimensiunea lui în imagine va exista o relație liniară, parametrii dreptei fiind direct influențați de parametrii extrinseci ai camerei video, unghiul de aplecare (pitch) și înălțimea camerei față de sol.

Datorită faptului că suprafața drumului nu e întotdeauna plană, dimensiunea vehiculelor nu este cunoscută cu exactitate, și detecția nu este perfectă (vehiculul nu este încadrat perfect), nu vom obține o dreaptă exactă pentru toate vehiculele din scenă. Colectând toate detecțiile dintr-o secvență de 5 minute, se va obține un grafic de genul celui prezentat în figura 1.7. Pentru identificarea dreptei care se potrivește cel mai bine acestor date, se folosește metoda RANSAC.

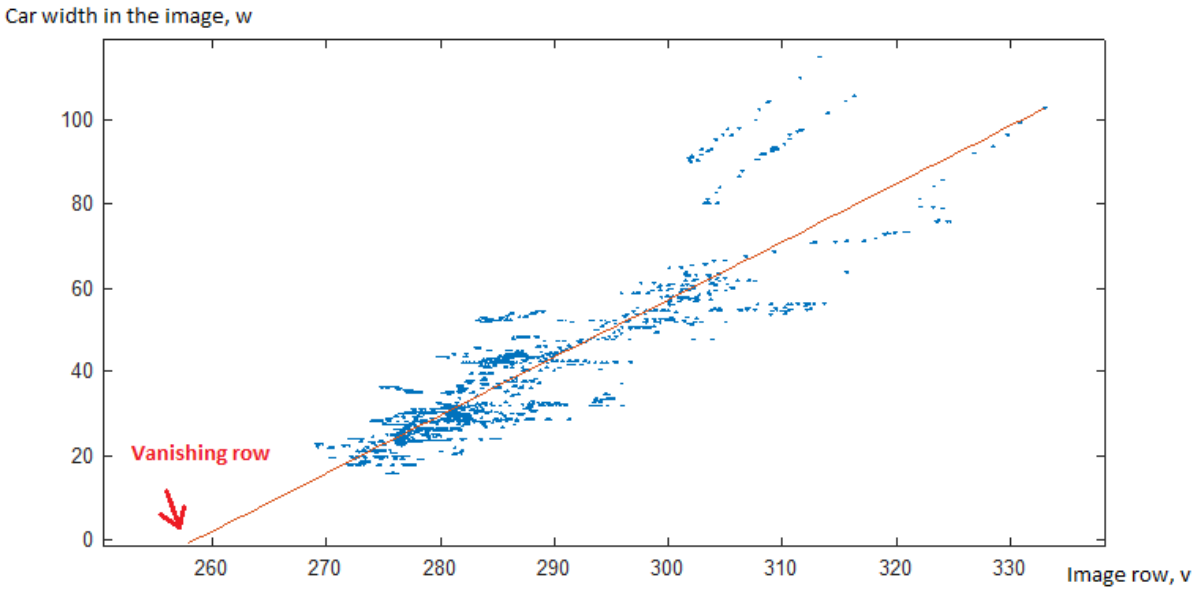


Figura 1.7. Exemplu de autovehicule detectate automat de MobileNet.

Din parametrii dreptei obținute vom estima înălțimea și unghiul de aplecare al camerei folosind o metodă bazată pe filtrul Kalman extins, unde starea estimată este un vector format din înălțimea camerei și unghiul de pitch, iar măsurătoarea este dată de două puncte de tip (coordonată rând, lățime) de pe dreapta estimată. Filtrul converge cam dupa 4-5 iterații.

Pentru măsurarea unghiului de orientare laterală a camerei față de axul longitudinal al mașinii, se folosesc perechi de detecții obținute în cadre succesive, presupunând că ele aparțin aceluiași obiect. Traiectoria obiectului trebuie să conveargă spre punctul de fugă, care este poziționat pe linia orizontului, pe care am estimat-o deja. În realitate, fără un algoritm de urmărire complex, nu putem garanta că detecțiile aparțin aceluiași obiect, dar vom utiliza din nou o abordare statistică, și vom selecta mediana ipotezelor acestor puncte de fugă, precum în figura 1.8.

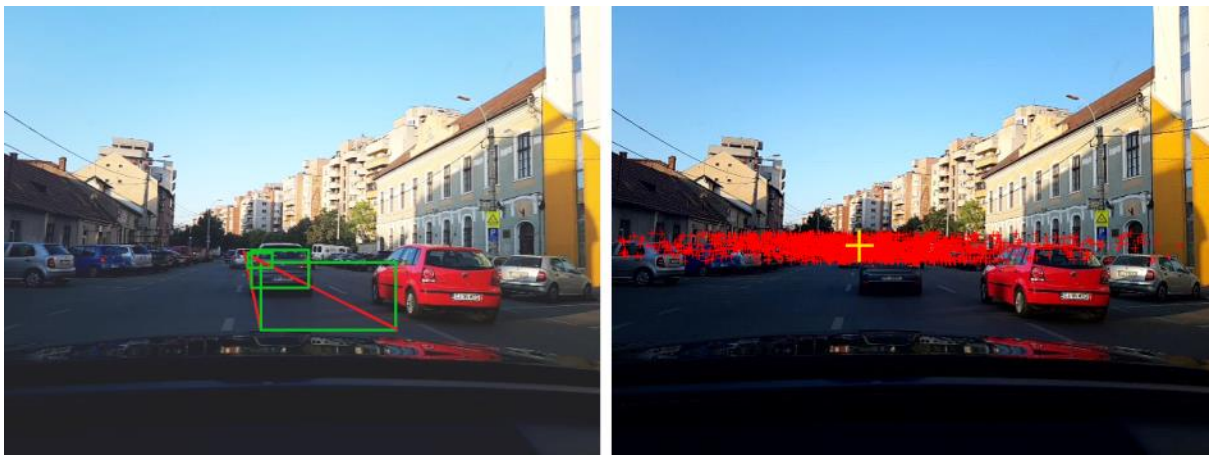


Figura 1.8. Principiul deteției punctului de fugă (stânga), și selectarea punctului de fugă din multiple ipoteze (dreapta).

Coordonata coloanei punctului de fugă este direct legată de unghiul de orientare laterală a camerei. O pozitie perfect centrală a acestui punct înseamnă o aliniere perfectă a

camerei cu axul autovehiculului propriu, iar orice deviație de la acest punct este dată de unghiul de orientare ψ, astfel:

$$tan\,\psi = (x0 - w/2)/f,$$

unde x0 este coordonata coloană a punctului de fugă, w este lățimea imaginii, și f este distanța focală a camerei în pixeli.

Etapa 2 - Proiectarea și implementarea algoritmilor

1. Proiectarea, antrenarea și evaluarea rețelelor neuronale, și reglarea fină a hiper- parametrilor

Rețeaua neuronală convoluțională folosită pentru extragerea zonei de drum este bazată pe U-Net [5] și a fost descrisă în raportul primei etape. Structura rețelei a rămas la fel, având 5 niveluri de codificare, un nivel central și tot 5 niveluri de decodificare. Am testat rețeaua și cu dimensiunea imaginii de intrare de 512 x 512 pixeli, având rezultate mai bune pe setul de validare, dar timpul de procesare a crescut de cel puțin 2 ori. În varianta de 256 x 256 pixeli predicția se face în 8-10ms, în timp ce imaginea de 512 x 512 pixeli necesită ~20ms. Structura rețelei convoluționale neuronale este ilustrată în figura 2.1.

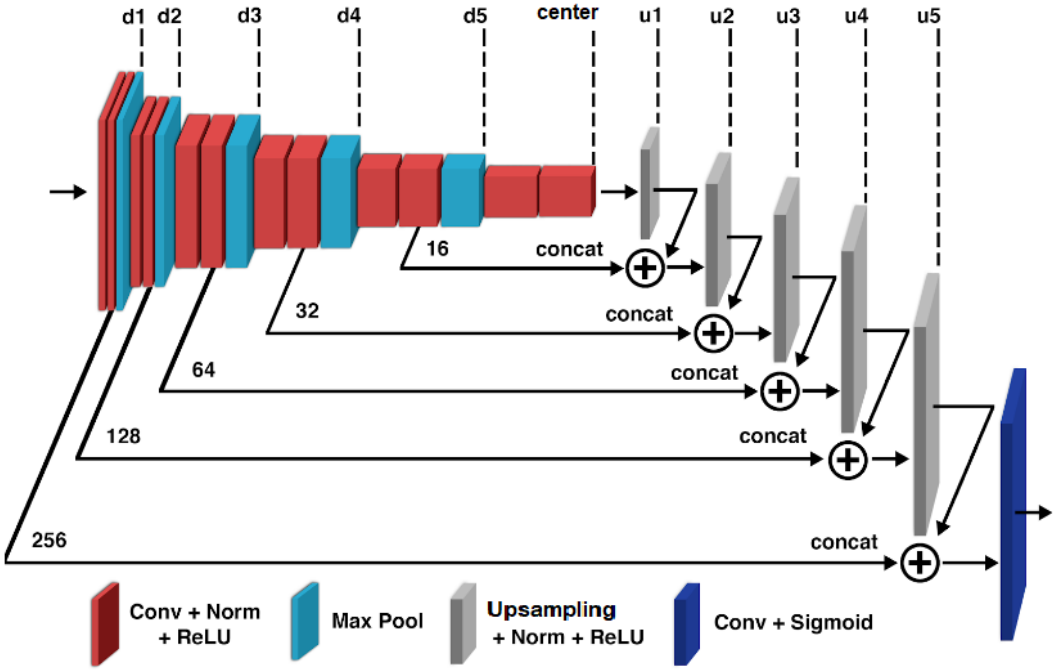


Figura 2.1. Rețeaua neuronală convoluțională pentru segmentare.

Am testat de asemenea și arhitectura existentă ERFNet [11], folosind software-ul PyTorch. Aceast model de rețea artificială folosește niveluri care au mai puțini parametri, de aceea necesită resurse hardware mai puține. Obiectivul acestui proiect este de a implementa un sistem de percepție direct pe o platformă mobilă, cu resurse hardware limitate, de aceea o abordare folosind ERFNet ar fi justificată. Intrarea și ieșirea rețelei a fost modificată pentru a folosi aceleași baze de date de antrenare (descrise pe larg în raportul etapei 1). Rețeaua necesită un timp de calcul al predicției mai redus decât prima soluție bazată pe U-Net, în detrimentul acurateții. Am antrenat rețeaua folosind aceleași funcții de cost (“binary cross entropy”), dar am monitorizat și coeficientul Dice, iar rezultatul este ilustrat în figura 2.2.

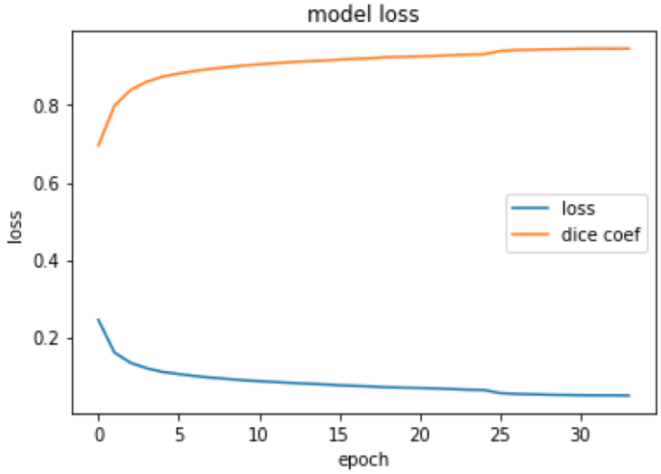


Figura 2.2. Antrenarea rețelei pentru segmentare folosind funcția de cost “binary cross entropy”. Este ilustrat și coeficientului Dice.

În figura 2.2 se poate observa că antrenarea este oprită automat dacă funcția de cost nu este îmbunătățită după cel puțin 5 epoci consecutive (în acesta caz antrenarea este oprită după 34 epoci). Antrenarea a folosit următoarele baze de date cu imagini din trafic și adnotări: CityScapes [3], Berkeley DeepDrive [4], Mapillary Vistas [12] și KITTI [1].

Segmentarea bazată pe rețele neuronale convoluționale a fost evaluata folosind un set de validare din baza de date CityScapes. Scorul obținut de noi folosind metrica “Intersect over Union” (IoU) este 0.91, iar metoda cea mai performantă la momentul actual [13], de la Google obține un scor de 0.98. Rezultatele comparative sunt ilustrate în tabelul 2.1.

Model rețea artificială	**Scor IoU (doar clasa de drum)**
DeepLab [13]	0.986
E-Net [14]	0.974
CNN propriu antrenat pe mai multe clase	0.922
CNN propriu antrenat pe o singură clasă (drum)	0.911

Tabelul 2.1. Evaluarea segmentării prin comparație cu alte abordări.

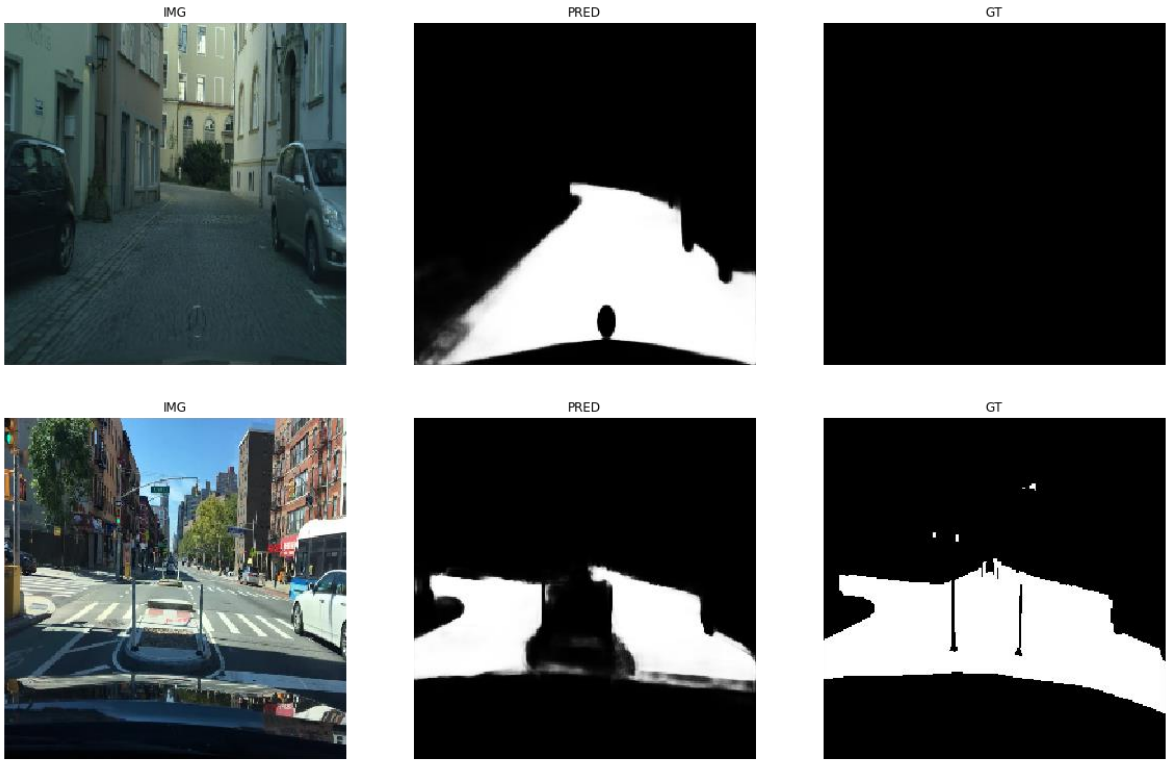


Figura 2.3. Exemple de adnotări eronate în baza de date CityScapes [3] și Berkeley Deep Drive [4]. Coloana din stânga reprezintă imaginea scenei de drum, în centru sunt predicțiile rețelei, iar în dreapta sunt adnotările greșite.

Evaluarea este realizată folosind o rețea antrenată pentru a oferi predicții asupra unei singure clase (cea de drum). Am observat faptul că la o antrenare pe mai multe clase, scorul obținut crește (0.92 vs. 0.91). Un alt aspect este faptul că metodele de top publicate deja sunt antrenate folosind imagini de dimensiuni mari, iar partea de predicție este post-procesată și rafinată. În cazul nostru, am ales o varianta care să favorizeze o viteză de predicție mare, în timp ce nu efectuam nicio post-procesare a hărții de predicție (am aplicat doar o binarizare cu prag fix).

Totodată, în timp ce am investigat rezultatele slabe ale evaluării, am observat prezența unor imagini adnotate greșit în setul de validare din bazele de date folosite.

Rețeaua antrenată de noi a folosit mai multe baze de date, iar numărul de adnotări greșite este redus proporțional, iar din figura 2.3 se poate observa faptul că predicția este corectă, deși adnotarea este eronată. Aceste imagini s-ar putea să afecteze unele metrici de evaluare și putem trage concluzia că sistemul propus de noi este suficient de robust pentru a putea fi folosit pentru a extrage zonele de drum într-un sistem de percepție bazat pe o singură cameră.

2. Detectarea si masurarea mișcării folosind rețele neuronale convolutionale

Viteza în spațiul 3D poate fi codificată în imagine sub forma componentelor ei pe axa X și axa Y, iar combinate cu componenta de ocupație (prezența sau absența obstacolelor) se poate genera o imagine pe trei canale (care pot fi scalate în intervalul 0...255, sau 0...1), care poate fi văzută ca o imagine pseudo-color (figura 2.4). În concluzie, se poate utiliza o rețea capabilă de generare de imagini color pentru a genera imaginea de viteze în mod direct, pe baza unui flux de imagini de intrare. Soluția care ar funcționa cel mai bine este cea a utilizării unor rețele neuronale convoluționale recurente, care mențin o stare ascunsă ce va fi folosită în procesul de clasificare, dar din păcate aceste rețele necesită o cantitate foarte mare de date de antrenare, și putere enormă de calcul, astfel că s-a renunțat la această variantă. O altă abordare a fost să luăm perechi de imagini la timpi consecutivi, care să compună o imagine pe mai multe canale (două imagini color produc o imagine pe șase canale, două imagini grayscale o imagine pe două canale), și să folosim o arhitectură de tip encoder-decoder, similară cu cea folosită pentru segmentare, pentru a genera imaginea de viteze. Experimentele pentru această abordare nu au produs încă rezultatele dorite de noi, astfel că s-a decis, deocamdată, folosirea mecanismului probabilistic bazat pe particule pentru extragerea vitezelor, iar utilizarea rețelelor a fost limitată, deocamdată, doar la producerea de informații statice care vor sta la baza modelului de măsură.

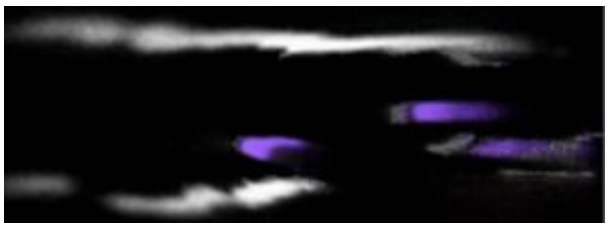


Figura 2.4. O imagine pseudo-color care codifică viteza sub forma canalelor de culoare.

3. Integrarea modulului de învățare profundă cu modulul de urmarire probabilistica

Rezultatele date de rețelele de tip CNN folosite sunt folosite ca date primare în procesul de calibrare automată a camerei, și în algoritmul de detecție și urmărire a obstacolelor.

3.1. Calibrarea automată a camerei

Calibrarea camerei este necesară pentru a stabili relația dintre spațiul imaginii, unde ne putem baza pe rezultatele date de rețelele neuronale convoluționale, și spațiul lumii 3D, unde vom realiza urmărirea probabilistică a obiectelor reale din traficul auto. Calibrarea poate fi realizată în mod manual sau supervizat, utilizând unelte de calibrare cunoscute, sau poate fi realizată automat, pentru a permite o punere în funcțiune a sistemului cât mai rapidă și care să solicite cât mai puțin utilizatorul.

Pentru calibrare este necesar să punem în corespondență dimensiuni cunoscute din lumea reală cu proiecția lor în spațiul imagine, detectată în mod independent de parametrii de calibrare.

O primă abordare pentru calibrare este de a folosi lățimea medie a unui vehicul, iar modul de calcul a fost prezentat în cadrul etapei 1. Principiul folosit este următorul: vehiculele apropiate vor avea o lățime mai mare (în pixeli) în spațiul imaginii decât cele situate la o distanță mai mare. Vehiculele apropiate de linia de orizont vor avea lățimea minimă (ele devin puncte). Totodată aici este și punctul de fugă din imaginile de trafic. Astfel, am creat o hartă de voturi din lățimea unui vehicul și coordonată sa din imagine pe axa rândului. Harta de voturi este folosită pentru a determina relația liniară care există între lățimea unui vehicul și coordonata sa de-a lungul axei orizontale. Am folosit metoda RANSAC [15] pentru a extrage această dependență. Linia orizontului din imagine (coordonată pe axa orizontală a punctului de fugă) va fi la coordonată în care lățimea unei mașini este 0 pe linia extrasă din RANSAC.

Extragerea dependenței liniare se poate face și prin modificarea transformatei Hough. Acumulatorul Hough va fi de fapt o hartă de voturi similară cu prima abordare: se incrementează valoarea în fiecare punct de coordonate (“lățime vehicul”, “index rând”). Astfel, prin folosirea transformatei Hough este necesară apoi determinarea parametrilor rho și theta ale liniei cu cele mai multe voturi din acumulator. Rezultatul acestei abordări este ilustrat în figura 2.5.

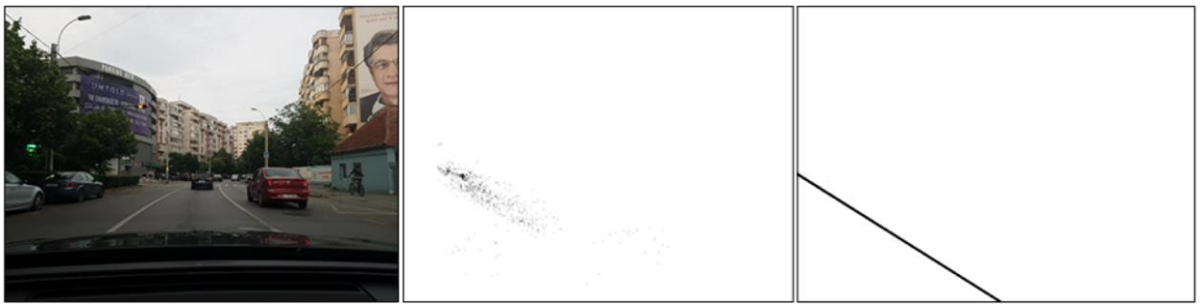


Figura 2.5. Stânga: imaginea de intrare, centru: acumulatorul Hough, dreapta: linia extrasă din acumulator.

Un alt reper de dimensiune aproximativ cunoscută este lățimea benzilor de circulație, care de asemenea devine mai mică în spațiul imagine pe măsură ce se apropie de punctul de fugă.

Calibrarea înălțimii și a unghiului de înclinare se poate face folosind relația dintre lățimea structurii 3D din lume (lățimea vehiculului sau a benzii) și lățimea proiectată în planul imaginii. În figura 2.6, relația dintre lățimea benzii de circulație în spațiul 3D al lumii și în spațiul 2D al imaginii poate fi definită folosind ecuația 1.

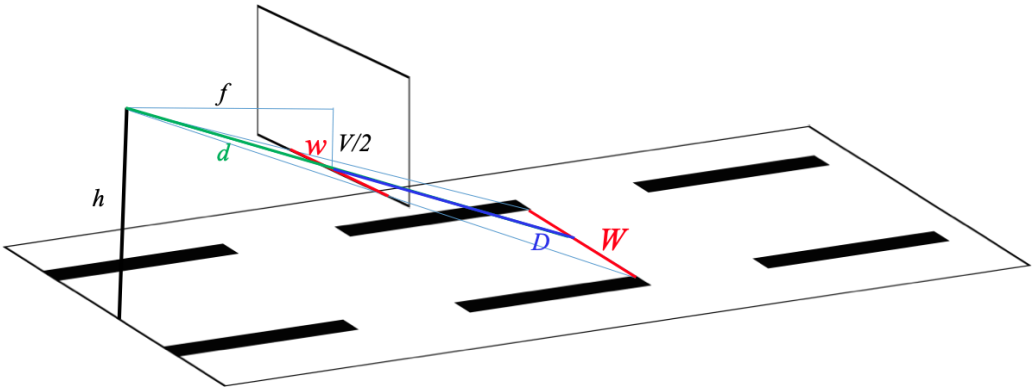


Figura 2.6. Proiecția în spațiul imaginii a lățimii unei structuri 3D cunoscute, situată în partea de jos a imaginii.

w/W = d/D (1)

Distanța “d” din ecuația 1 poate fi calculată folosind distanța focală (exprimată în pixeli) și din înălțimea imaginii, presupunând că poziția este în partea de jos a imaginii.

d=sqrt(f^2 + (img height/2 ^2) (2)

Putem folosi aceste ecuații pentru a determina distanța D dintre structura 3D (lățimea benzii sau a unui vehicul) și centrul optic al camerei. Dacă structura este o bandă de circulație situată pe planul drumului, atunci putem determina și înălțimea camerei față de sol folosind ecuația 2, și după cum este ilustrat în figura 2.7.

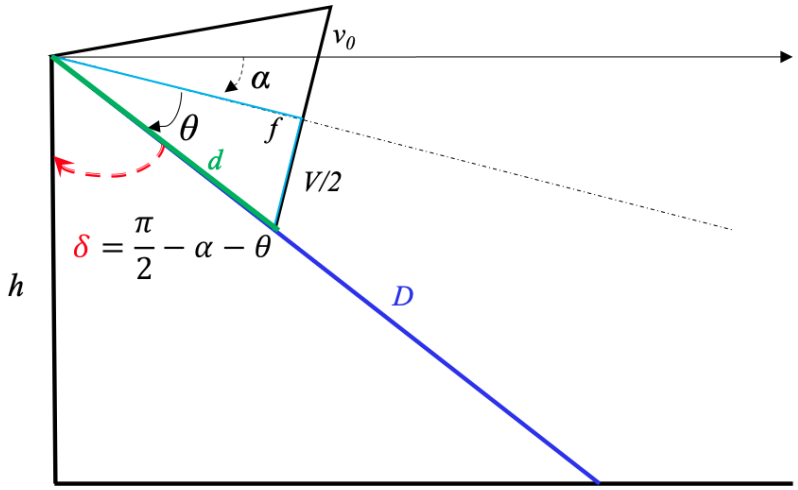


Figura 2.7. Relația dintre parametri camerei și distanța D.

h= D * cos(delta) (3)

Folosind dependența liniară calculată anterior, avem deja coordonata rând a punctului de fugă “v0” și putem determina unghiul de înclinare al camerei folosind ecuația 4.

tan (pitch) = (img height/2 - v0) / focal (4)
delta = pi/2 - pitch - theta (5)

În ecuațiile 4 și 5, θ reprezintă jumătate din câmpul vizual vertical al camerei. Am publicat rezultatele calibrării folosind lățimea benzilor de circulație în [16], iar prin folosirea lățimii unui vehicul în [17].

Punctul de fugă este punctul din imagine unde se intersectează elementele care în spațiul 3D sunt paralele, precum marcajele rutiere. Acest punct se poate folosi pentru a calcula unghiul de înclinare (“pitch”) și de girație (“yaw”).

Linia pe care se găsește punctul de fugă poate fi determinată prin statistica descrisă anterior, acumulând lățimi de bandă sau de obstacol și poziția lor în imagine. Punctul de fugă poate fi, de asemenea, determinat în mod direct din fiecare imagine în parte, iar combinația celor două metode ne va ajuta să avem o calibrare continuă, care să compenseze mișcările de oscilație ale vehiculului propriu. În lucrarea [16] am propus o metodă pentru a calcula punctul de fugă folosind gradientul și magnitudinea punctelor din imagine. Metodele existente sunt bazate pe extragerea punctelor de trăsături relevante (de exemplu muchii), în timp ce abordarea propusă de noi folosește orice punct din imagine. Totuși, pentru a reduce complexitatea computațională, am definit o zonă de interes de unde punctele din imagine pot vota în funcție de orientarea gradientului și magnitudine. Harta de voturi rezultată este filtrată și maximul local va fi ales ca fiind punctul de fugă. Figura 2.8 reprezintă harta de voturi și punctul de fugă calculat.

Figura 2.8. Harta de voturi (stânga) și punctul de fugă rezultat (dreapta).

O altă abordare de a detecta punctul de fugă în imagini este de a folosi o rețea neuronală convoluțională care oferă predicții asupra coordonatelor din imagine ale acestui punct. Am publicat această abordare în [18], dar o îmbunătățire obținută recent este de a folosi două rețele antrenate separat: una pentru predicția asupra coordonatei orizontale (“X”) și alta pentru predicția coordonatei verticale (“Y”). Cele două rețele au aceeași structură cu cea publicată deja: 5 niveluri convoluționale cu număr variabil de filtre, urmate de 3 niveluri complet conectate, ultimul fiind compus din 1 neuron care va reprezenta coordonata punctului de fugă din imagine. Intrarea rețelei este o imagine de dimensiune 200 x 70 pixeli și rețeaua a fost antrenată pentru un total de 200 epoci folosind dimensiunea lotului de 256 și funcția de cost RMSE. Soluția a fost implementată și pe dispozitive mobile Android, iar un exemplu de predicție este ilustrat în figura 2.9, unde cercul alb reprezintă predicția.

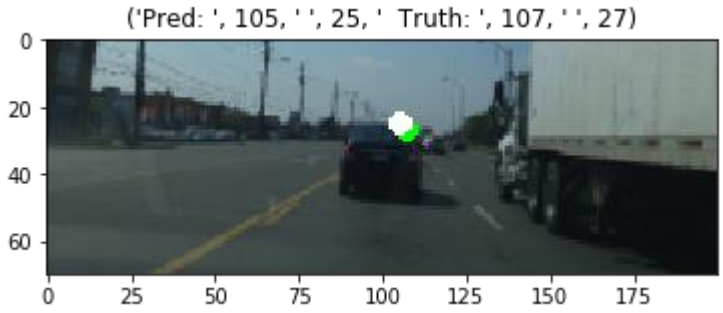


Figura 2.9. Exemplu de predicție (punctul alb) și valoarea reală (punctul verde).

În sistemul propriu de percepție am folosit și testat ambele implementări, dar am ales pentru integrare varianta algoritmică, bazată pe orientarea și magnitudinea punctelor.

Calibrarea înălțimii camerei față de sol și a unghiului de înclinare a obținut rezultate bune. Am efectuat 3 teste folosind vehicule diferite și camere diferite, dar am comparat și pe baza de date KITTI [1], în care avem informații despre înălțimea camerei față de sol. Rezultatele sunt prezentate în tabelul 2.2.

Scenariu	**Înălțimea camerei față de sol (calibrată)**	**Înălțimea camerei față de sol (valoarea reala)**	**Unghiul de înclinare (calibrat)**
Test 1	1268	1250	-1.68°
Test 2	1234	1200	-3.8°
Test 3	1468	1480	2.97°
KITTI	1649	1650	-0.85°

Tabelul 2.2. Rezultatele auto-calibrării înălțimii camerei față de sol și calculul unghiului de înclinare (“pitch”).

În testele efectuate de noi, dar și în baza de date KITTI, nu avem informații despre unghiul de înclinare. Astfel, singurul mod de a valida corectitudinea acestui unghi este de a construi matricea de proiecție și de a crea o imagine cu efectul de perspectivă eliminat. Analiza acestei imagini este folosită pentru validare. O imagine în care liniile benzilor de circulație sunt dispuse paralel va însemna că avem un unghi de înclinare corect (figura 2.10).

Figura 2.10. Generarea imaginii “bird’s eye view” folosind parametrii calculați din calibrare.

Punctul de fugă calculat cu 2 rețele artificiale este îmbunătățit față de metoda deja publicată (care folosește o singură rețea neuronală). Rezultatele comparative sunt ilustrate în tabelul 2.3, unde am testat folosind baza de date proprie care este împărțită în două: imagini din scenarii de oraș și de pe autostradă.

Metrică eroare	**Oraș (V1)**	**Oraș (V2)**	**Autostradă (V1)**	**Autostradă (V2)**
NormDist	0.05072	0.29725	0.02612	0.012905

Tabelul 2.3. Comparație între rețeaua artificială inițială (V1) și metoda îmbunătățită folosind 2 rețele artificiale (V2).

În tabelul 2.3, metrica de eroare pentru comparație este “NormDist” care este cea folosită și în lucrările deja publicate. Reprezintă de fapt eroarea RMSE dintre punctul de fugă calculat și cel real, iar rezultatul este împărțit la diagonala imaginii de intrare (astfel fiind o metrică invariantă la dimensiunea imaginii de intrare).

3.2. Modelul probabilistic al lumii

Scena de drum este modelată ca o hartă de ocupare, având 120 x 500 celule, în care fiecare celulă va reprezenta o zonă de 20 x 20 cm, ceea ce înseamnă că spațiul complet de procesare este o zonă de 24 x 100 m. Camera este orientată înainte și este poziționată în centrul hărții de ocupare, la coordonată de rând 250 și coordonată de coloană 60. Harta de ocupare bidimensională reprezintă reproiectarea spațiului 3D al lumii (al scenei observate) într-o vedere de sus a scenei (numită de obicei “bird’s eye view”). Spațiul vizibil este doar jumătate din harta de ocupare, iar cealaltă jumătate va fi actualizată de pasul de predicție folosind celulele vizibile din scenă. Prin folosirea acestei abordări, sistemul poate fi ușor extins prin adăugarea mai multor camere și senzori, de exemplu se poate monta o cameră orientată în spate care va actualiza măsurătorile pentru jumătatea inferioară a hărții de ocupare a scenei.

Harta de ocupare va reprezenta obiecte individuale, fiecare compus din particule dinamice având o viteză și o poziție. Migrarea particulelor dintr-o celulă în alta are la bază viteza proprie a particulei, dar și viteza de deplasare a vehiculului și rata de rotație. Algoritmul pentru predicția și actualizarea particulelor este descris pe larg în lucrarea [19] și [20]. Fiecare celulă din hartă va avea o probabilitate de a fi ocupată, determinată din numărul de particule din celulă raportat la numărul maxim de particule dintr-o celulă (Nc = 100, parametru configurabil în funcție de hardware-ul folosit).

3.3. Utilizarea rezultatelor CNN în urmărirea probabilistică

Urmărirea scenei de obiecte 3D se realizează prin estimarea continuă a probabilității de ocupare a celulelor din harta de ocupare, precum și a distribuției de probabilitate a vitezelor celulelor. Pentru realizarea acestui deziderat se va utiliza rezultatul segmentării semantice pe baza CNN pentru a genera modelul probabilistic de măsură. Rezultatul segmentării se proiectează folosind parametrii calibrați ai camerei în vederea de tip “bird eye”, corespunzătoare scenei observate de sus, fiecare pixel corespunzând unei celule din harta de ocupare. Deoarece proiecția este validă doar pentru suprafața drumului, punctele de interes sunt de fapt doar punctele de contact ale obstacolelor cu suprafața drumului, sau mai bine spus punctele de graniță dintre zonele segmentate de CNN ca obstacole și zonele segmentate ca drum.

În lucrările publicate deja am prezentat diferite metode de extragere a punctelor de contact ale obiectelor cu drumul. O abordare este de a proiecta raze cu originea în punctul focal al camerei spre obiecte, de-a lungul cărora se vor căuta tranziții de intensitate. Această abordare a fost publicată în [21] și are la bază idei și principii din lucrarea [22]. În [20] am prezentat o extensie a algoritmului de scanare folosind raze care va calculat 3 distanțe de-a lungul fiecărei raze pentru a îmbunătăți robustețea și precizia. Totuși, majoritatea obstacolelor și în special vehiculele nu vor atinge suprafața de drum complet. Mașinile ating drumul doar cu rotile, iar acest aspect este amplificat în imaginile cu efectul de perspectivă eliminat. Astfel, este nevoie să umplem zonele goale dintre roți pentru a genera o detecție robustă a distanțelor până la obstacole. Pentru aceasta, am implementat un algoritm de umplere a spațiilor convexe, iar întreg procesul este descris pe larg în lucrarea [23].

Principiul din toate abordările prezentate este de a folosi camera similar cu un scanner bazat pe laser, prin analiza razelor proiectate în imaginile de tip “bird’s eye view”. Aceste imagini procesate sunt folosite pentru a crea o hartă de măsurare pentru modelarea probabilității unei celule de a fi ocupată sau nu. Acest model trebuie să ia în considerare erorile care apar în timpul procesului de măsurare, de aceea am identificat următoarele limitări: erorile de-a lungul razelor longitudinale și limitările determinate de procesul de observație. Pe baza celor două tipuri de erori se va genera, pentru fiecare rază de observație, o probabilitate ideală pentru obstacol de a exista (până la punctul de contact probabilitatea de a exista este aproape de 0%, la punctul de contact aproape 100%, iar după punctul de contact avem zona neobservabilă, unde probabilitatea este de 50%). Probabilitatea ideală va fi supusă unui proces de convoluție cu un nucleu Gaussian unidimensional, cu deviația standard dependentă de modelul de eroare la determinarea distanței din maparea imaginii perspectivă în planul suprafeței drumului.

Aceste erori sunt modelate de ecuația 1, unde h reprezintă înălțimea camerei față de sol, z este distanța până la obiect și σ□ modelează erorile mici (deviațiile) în unghiul de

înclinare al camerei, cauzat de suprafețe de drum accidentate sau inegale. Această valoare am setat-o la 0.1 grade, în timp ce valoarea pentru alte surse de erori care nu se pot modela (σ0) este setată la 0.1 metri.

$$\sigma z = h\,(1 + (z/h)^2)\,\sigma\square + \sigma 0 \qquad (6)$$

Procesul de generare a modelului de măsură este redat în figura 2.11. Mai multe detalii pot fi găsite în [17].

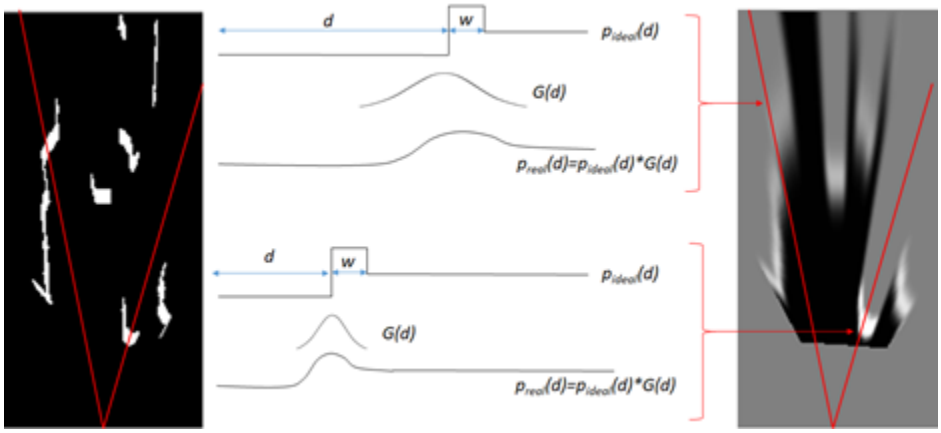


Figura 2.11. Generarea modelului de măsură pe baza rezultatelor CNN.

După actualizarea modelului lumii pe baza modelului de mișcare a particulelor și a modelului de măsură obținut pe baza segmentării CNN, celulele ocupate din hartă se vor grupa pe baza similarității, extrăgându-se obiectele individuale din scenă.Gruparea celulelor este realizată în funcție de proximitatea celulor, dacă vectorii lor de viteză sunt asemănători (diferența dintre orientarea lor este mai mică de 30°) și dacă celulele au o probabilitate de a fi ocupate de minim 0.5. Presupunând că o celulă conține un singur obstacol, viteza unei celule este calculată ca media vitezei particulelor din celulă. Din celulele grupate am creat cuboide, iar în final avem o listă de obiecte, fiecare având o poziție, orientare, viteză, lungime, lățime și o înălțime standard de 1.5 metri.

Întreg procesul este descris în figura 2.12.

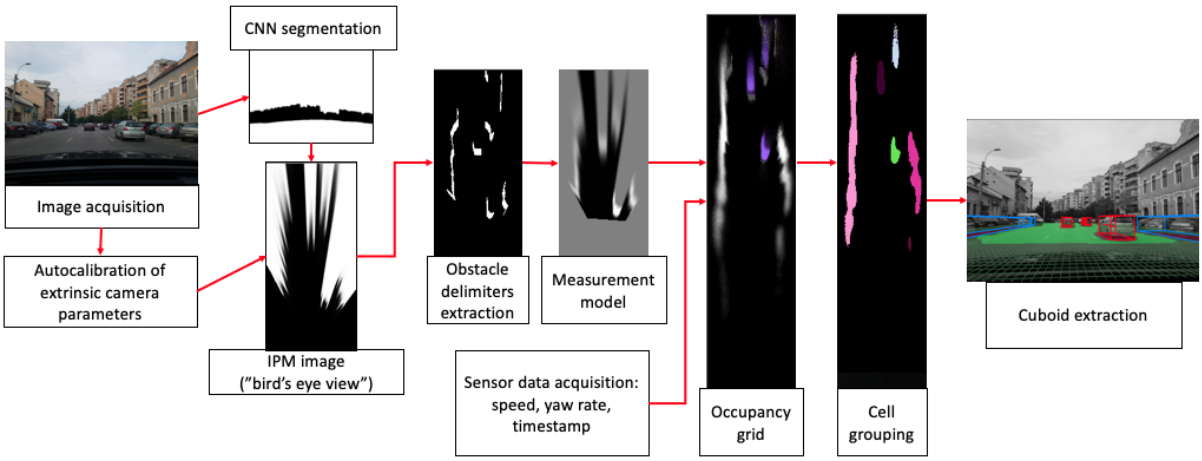


Figura 2.12. Fluxul de procesare al sistemului monocular de percepție a traficului rutier.

3.4. Evaluarea sistemului de detecție

Pentru evaluare am folosit și baza de date pentru evaluarea algoritmilor de urmărire KITTI [1]. Rezultatele pe secvențele numerotate cu 0005, 0010, 0011 și 0018 sunt ilustrate în tabelul 2.4. Am calculat eroarea Mean Absolute Error (MSE) dintre locația 3D a obiectului detectat de noi și locația reală din baza de date KITTI. Evaluarea este efectuată doar pentru poziția de pe axa Z, care reprezintă distanța până la obiectul detectat și este cea mai relevantă pentru estimarea adâncimii, în special în sisteme bazate pe o singură cameră. Potrivirea obiectelor este realizată prin calculul "Intersect over Union" aplicat la chenarele de încadrare. Pentru evaluare am filtrat unele obiecte și nu am folosit cele care sunt alungite (sistemul nostru va detecta orice tip de obstacol, uneori și bordurile sau separatoarele dintre benzi, ceea ce va afecta procesul de evaluare). Evaluarea se face pe intervale de distanță diferite, iar distanța maximă de procesare a sistemului nostru este limitată la 50 de metri.

Interval distanță (m)	**KITTI 0005 Mean Absolute Error (m)**	**KITTI 0010 Mean Absolute Error (m)**	**KITTI 0011 Mean Absolute Error (m)**	**KITTI 0018 Mean Absolute Error (m)**
0-10	-	-	0.8	-
10-20	1.30	2.6	3.29	3.91
20-30	3.65	2.74	5.59	5.59
30-40	2.07	6.03	11.72	6.2
40-50	2.08	4.35	19.06	9.32

Tabelul 2.4. Analiza erorii MAE pe secvențe din baza de date KITTI.

Interval distanță (m)	**KITTI 0005 Rata de detectie (%)**	**KITTI 0010 Rata de detectie (%)**	**KITTI 0011 Rata de detectie (%)**	**KITTI 0018 Rata de detectie (%)**
0-10	-	-	97.56	-
10-20	100	94.73	97.25	79.2
20-30	89.62	62.78	82.95	74.4
30-40	76.95	51.85	40.73	74.71
40-50	60	5.35	31.46	47.05

Tabelul 2.5. Rata de detecție pe diferite intervale pe secvențe din baza de date KITTI.

De asemenea, am folosit pentru evaluare și imagini cu informație 3D obținută prin stereoviziune, obținute din proiecte anterioare ale colectivului. Secvențele constau în imagini

achiziționate cu un sistem de stereoviziune precis calibrat. Valorile de “ground truth” la care ne raportăm vor fi considerate cele urmărite folosind un algoritm bazat pe stereoviziune. În tabelul 2.6 am prezentat analiza acestei evaluări.

Interval distanță (m)	Mean Absolute Error (m)	Rata de detectie (%)
0-10	0.78	88.66
10-20	1.37	96.03
20-30	2.62	92.32
30-40	7.21	80.61
40-50	17.44	57.43

Tabelul 2.6. MAE și rata de detecție pe secvența proprie unde valorile de referință sunt cele dintr-un algoritm de stereoviziune.

Harta de ocupare și de măsurare acoperă o zonă de drum cu o lățime de 24 metri, ceea ce înseamnă că uneori rata de detecție este mai mică de 100%. Algoritmul de scanare folosit va introduce limitări mai ales în cazul vehiculelor care sunt parțial acoperite. Totodată, rata de detecție în intervalul de măsurare minim (0-10 metri) este afectată de punctele de contact ale obstacolelor care sunt acoperite de capota vehiculul propriu. La distanțe mai mari, eroarea unui sistem monocular este mai mare, ceea ce înseamnă că probabilitatea de ocupare a particulelor dintr-o celulă este mică și de aceea obiectele nu sunt întotdeauna detectate.

4. Îmbunătățirea performanțelor rețelei neuronale folosind informația de la estimatorul probabilistic

În afară de puterea evidentă de clasificare și de segmentare a rețelelor neuronale convoluționale, tendința recentă este de a folosi aceste rețele pentru a rezolva mai multe etape din problematica percepției mediului și chiar a conducerii automate a autovehiculelor. Dacă rețelele sunt suficient de mari, datele de antrenare în cantitate enormă, și puterea de calcul nu este un impediment, Deep Learning poate rezolva orice problemă.

Deoarece nu dispunem de aceste resurse nelimitate, am încercat totuși să experimentăm utilizarea rețelelor neuronale pentru a suplini diferite etape ale algoritmului de detecție a obstacolelor. Au fost abordate două sub-probleme: generarea automată a imaginii de tip “bird-eye” din imaginea perspectivă, și generarea automată a modelului de măsură din imaginea reproiectată. Pentru ambele abordări au fost utilizate mai multe arhitecturi de tip enconder-decoder, iar ca date de antrenare au fost utilizate datele generate de algoritmii analitici proiectați. Rezultatele preliminare sunt foarte promițătoare, după cum se poate observa în figura 2.13, unde rețeaua este capabilă să estimeze în mod corect zonele cu mare probabilitate de a fi obstacol, zonele libere, și zonele incerte.

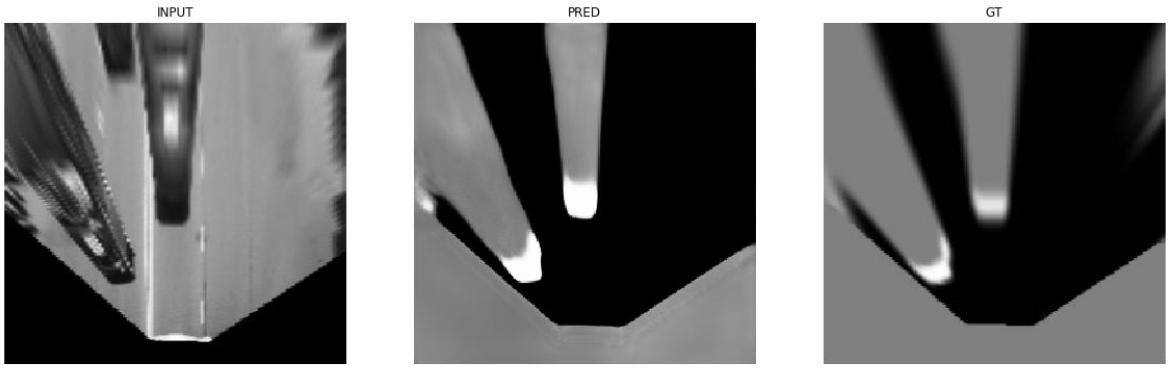


Figura 2.13. Generarea hărții de probabilitate (măsurare) direct dintr-o rețea artificială. Stânga: imaginea de intrare “bird’s eye view”, centru: predicția rețelei, dreapta: imaginea de adnotare (“ground truth”).

5. Realizarea aplicației demonstrator

În această fază a proiectului au fost dezvoltate două componente independente ale aplicației finale. O componentă este dedicată achiziției datelor, iar a doua componentă este dedicată procesării datelor achiziționate. În faza următoare se va realiza unificarea celor două componente pentru realizarea unui sistem care va percepe obstacolele din trafic în timp real, la bordul unui vehicul de test.

5.1. Aplicația de achiziție a datelor

Sistemul propriu pentru achiziția datelor este scris în limbajul de programare Java și utilizat pe aplicații mobile Android. Achiziția datelor pe dispozitive mobile folosește următoarele componente: camera, accelerometru, giroscop și senzorul de orientare și cel de poziționare prin satelit.

Camera principală este cea folosită pentru achiziția imaginilor, iar restul senzorilor sunt folosiți pentru a stoca sub formă de text informațiile despre poziție, orientare și viteză. Din procesul de achiziție am reușit să obținem un total de 27 de secvențe de trafic, în condiții de iluminare diferită și în momente diferite ale zilei. În figura 2.14 sunt ilustrate toate traseele parcurse în împrejurul orașului Cluj-Napoca.

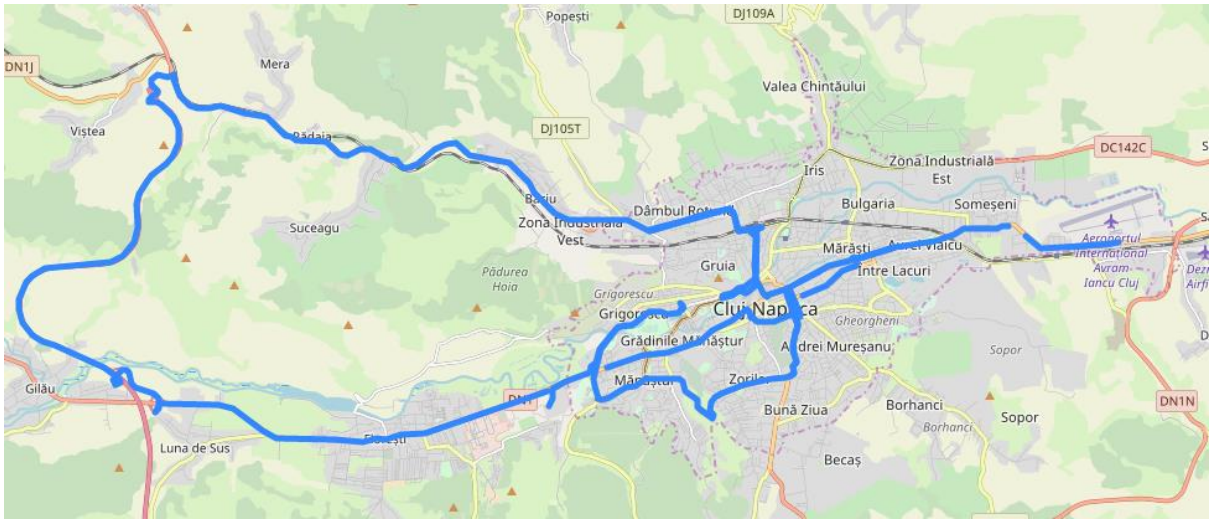


Figura 2.14. Traseele din GPS pentru călătoriile înregistrate în Cluj-Napoca și împrejurimi.

5.2. Aplicația de procesare a secvențelor achiziționate

Aplicația de procesare este compusă din modulul de clasificare și predicție pe bază de CNN, și implementarea algoritmilor de urmărire probabilistică. Rețelele neuronale sunt dezvoltate în Python, iar algoritmii sunt dezvoltați în C++. O comunicare între cele două procese a fost stabilită prin mecanismul de tip socket, dar această soluție este doar temporară. O altă abordare este exportarea rețelei neuronale din Python în C++, iar această soluție este mult mai eficientă din punct de vedere al vitezei de calcul și al memoriei folosite, dar datorită particularităților bibliotecilor folosite nu s-a reușit exportarea tuturor tipurilor de rețele utilizate.

Cu exportarea rețelelor în C++, întreg fluxul de procesare, incluzând segmentarea semantică, crearea de particule, actualizare și măsurare pentru particule este realizată într-un timp mediu cuprins între 70-80 ms, în funcție de numărul de obstacole din scenă. Cu optimizare și prin reducerea informațiilor de depanare care sunt afișate, acest timp ar putea fi redus suplimentar la sub 40-50 ms. Partea de segmentare semantică este realizată într-un timp mediu de 6 ms în C++ cu modelul exportat din Python.

O captură de ecran cu aplicația de procesare realizată este ilustrată în figura 2.15.

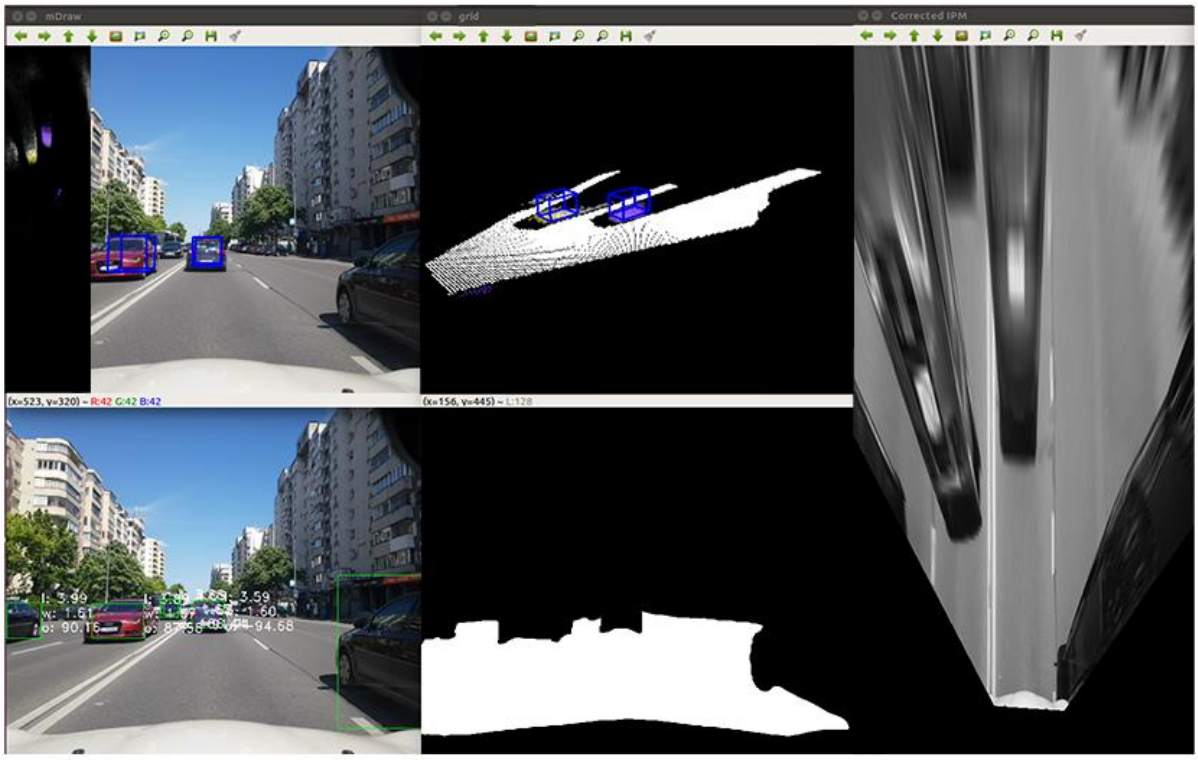


Figura 2.15. Sistemul de procesare în care sunt afișate cuboidele extrase (colțul din stânga sus și centru sus), chenarele de încadrare a vehiculelor (stânga jos), rezultatul segmentării (centru jos) și imaginea cu efectul de perspectivă eliminat (dreapta).

Etapa 3 - Integrare, testare și validare

1. Introducere - rezumatul etapei

În cadrul etapei finale a proiectului DEEPSENSE, având ca titlu “Integrare, testare și validare”, a fost pus accentul pe rafinarea soluțiilor tehnice, pe testarea acestora, dar și pe publicarea unui articol de jurnal ce descrie în detaliu rezultatele principale ale proiectului.

Procesul de rafinare a tehnologiilor a fost orientat pe două direcții: rafinarea interpretării imaginilor prin rețele neuronale convoluționale, un proces ghidat de date și bazat pe enormele colecții de imagini disponibile la nivel internațional, și pe tendințele recente de utilizare a rețelelor cu ieșiri multiple și trunchi comun (“backbone”), iar procesul de rafinare a algoritmilor probabilistici a fost orientat pe extinderea modelului bazat pe particule pentru a include informații suplimentare precum spațiul liber și identitatea obiectelor individuale.

Majoritatea algoritmilor de calibrare, segmentare și urmărire au fost integrați într-o aplicație demonstrator capabilă de procesare în timp real a imaginilor, care a fost testată pe imagini proprii și comparată cu alte aplicații existente.

Rezultatele au fost publicate într-un articol de jurnal ISI, din categoria Q1 (zona roșie).

2. Îmbunătățirea performanțelor rețelei neuronale folosind informația de la estimatorul probabilistic

2.1. Îmbunătățirea rețelei neuronale

În cadrul acestei etape am experimentat soluții pentru obținerea a cât mai multe informații, de nivele diferite de prelucrare și complexitate, folosind o singură rețea neuronală convoluțională. Aceste eforturi se aliniază cu tendințele generale din comunitatea științifică și din industrie, care urmăresc înlocuirea soluțiilor bazate pe algoritmi (“algorithm-driven”) cu cele bazate pe date (“data-driven”), soluții care pot fi în permanență rafinate prin date suplimentare.

Rețeaua folosită de noi are ca intrare o imagine, iar ca ieșire are o parte de segmentare semantică ceea ce presupune o imagine pe mai multe canale și o altă ieșire partea de detecție de obiecte care constă într-un șir de chenare de încadrare și probabilități de a aparține unei clase pentru fiecare chenar (“bounding box”).

Arhitectura rețelei propuse este ilustrată în figura 3.1.

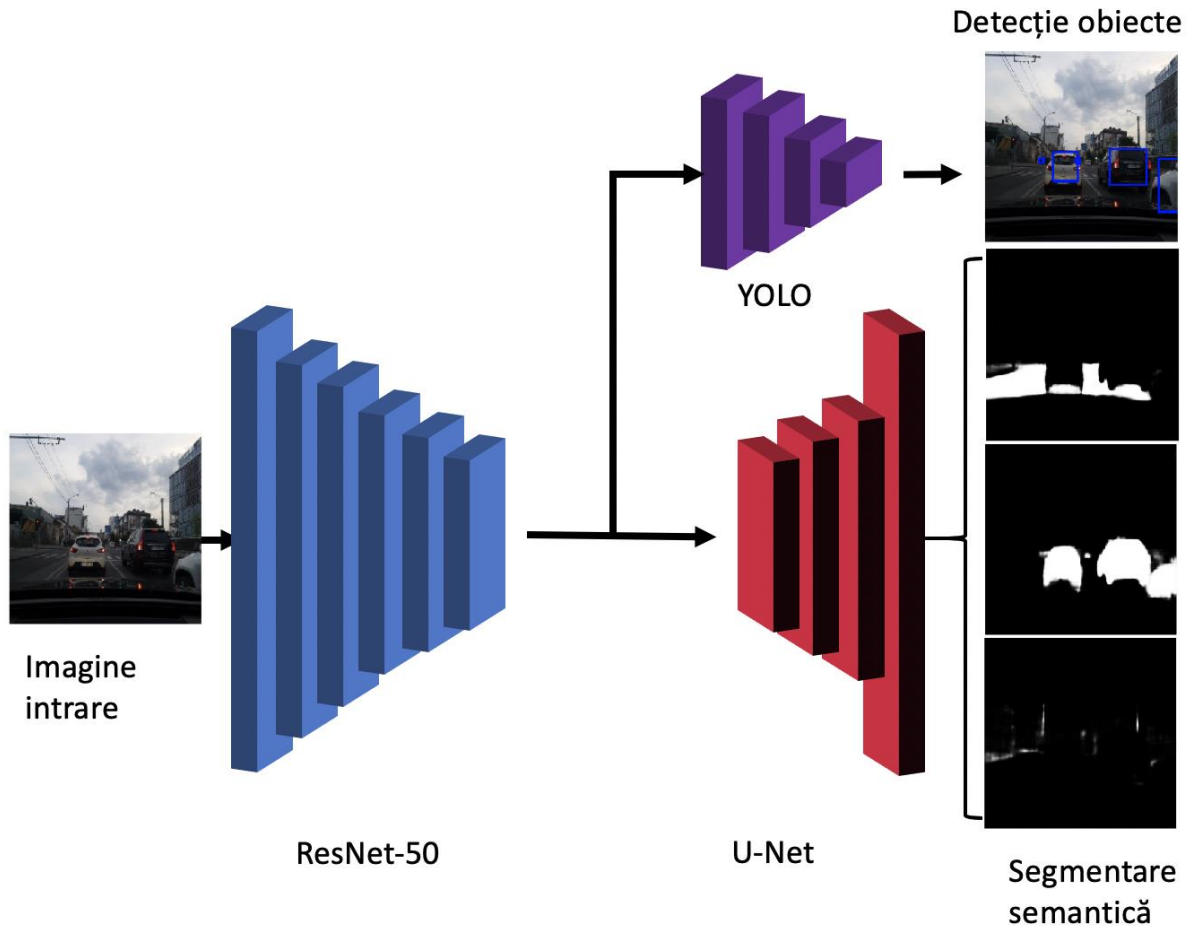


Figura 3.1. Structura rețelei de detecție obiecte și segmentare semantică într-un singur pas.

Prima parte a rețelei, cea care are rolul de a extrage informația relevantă (trăsăturile relevante) din imagine folosește structura arhitecturii ResNet [24], mai exact o abordare modificata numită ResNet-50. Rețeaua ResNet este una foarte populară și a câștigat concursul ImageNet în 2015. Această abordare a introdus conceptul de conexiuni “skip” între nivelurile rețelei, ceea ce a dus la îmbunătățirea perfomanțelor în special pentru rețele complexe având număr mare de niveluri (“layers”).

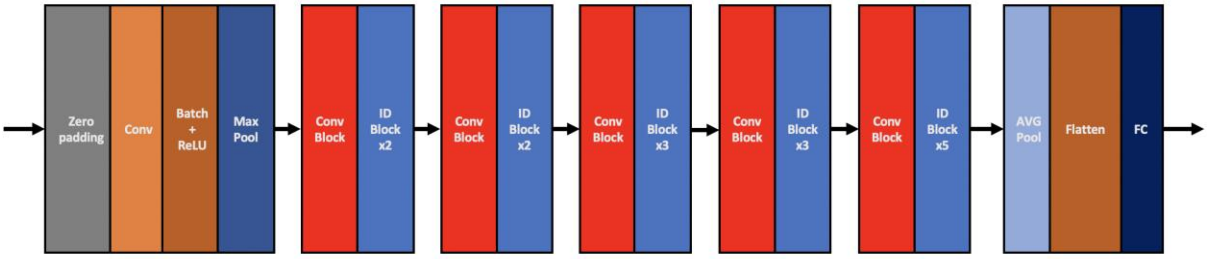


Figura 3.2. Structura rețelei ResNet-50.

Partea de construcție a informației semantice folosește structura rețelei U-Net [5]. Având trăsăturile extrase de ResNet, nivelurile de reconstrucție ale U-Net vor determina ieșirea rețelei: o imagine pe 3 canale de aceeași dimensiune ca și cea de intrare, unde fiecare canal va reprezenta o clasă de segmentare diferită. Cele 3 clase alese sunt: drumul, obiecte dinamice și obiecte statice. Un canal va reprezenta zona de drum pe care se poate conduce (“driveable road area”), obiectele dinamice sunt compuse din vehicule, autobuze, camioane, motociclete, pietoni, în timp ce obiectele statice le-am ales ca fiind: trotuarele și delimitatoarele de benzi (bariere sau garduri).

Pentru clasificarea obiectelor prin generarea chenarelor de încadrare am folosit structura rețelei Yolo [25] cu unele modificări [26]. Abordarea prezentată în lucrarea Yolo presupune împărțirea imaginii de intrare a rețelei în celule, iar pentru această implementare am folosit 11x11 celule. Am utilizat un număr maxim de 2 chenare de încadrare pentru fiecare celulă, iar pentru fiecare chenar de încadrare se va face o predicție asupra probabilității de a aparține unei clase (am ales un total de 3 clase: pieton, vehicul sau camion/autocar), iar totodată se va face predicția asupra dimensiunii și poziției chenarelor (poziția de început pe axa x, pe axa y, lățime, înălțime). Pentru fiecare chenar dintr-o celulă se va face și o predicție a încrederii în respectivul chenar de încadrare (“confidence”). Pentru fiecare celulă se vor face următoarele predicții: 3 valori pentru probabilitățile pentru clasificare, 4 valori pentru dimensiuni și poziție, 1 valoare pentru încrederea predicției unui obiect (chenar), iar ținând cont că sunt maxim 2 chenare într-o celulă, rezultă un șir de dimensiune: (11 x 11) x (3 + 2 x (1 + 4)) = 1573.

Pentru antrenare am folosit trei baze de date cunoscute: CityScapes [3], BDD [4] și Mapillary [12] deoarece conțin informații atât despre segmentare, cât și despre obiecte și poziția lor în imagine. Astfel, în urma procesării și filtrării am folosit un total de 2975 imagini din CityScapes, 2759 imagini din BDD și 17109 imagini din Mapillary. Imaginile au fost filtrate pentru a alege doar acelea care conțin un număr mare de pixeli ai drumului. În procesul de antrenare am folosit augmentare prin translatare și scalarea imaginilor, dar și prin ajustarea intensității și saturației în spațiul de culoare HSV.

Pentru segmentare am folosit funcția de cost “binary cross entropy” și Sorensen-Dice (“Dice loss” - o versiune modificată a funcției “Intersect over Union”). Funcția de cost folosită pentru clasificare este aceeași cu cea din lucrarea Yolo și consta în insumarea functiilor de cost pentru coordonatele chenarului, dimensiunea chenarului, increderea în detectie (“confidence”) și probabilitatea de a apartine unei clase de obiect (probabilitatea clasificarii).

Cele două funcții de cost sunt folosite simultan, fiecare având ponderi care pot fi configurate. Am testat și cu ponderea pentru partea de segmentare (U-Net) de 10 ori mai mare decât partea de detecție (Yolo), iar rezultatele sunt mai bune decât în cazul antrenării cu ponderi egale între cele două funcții de cost.

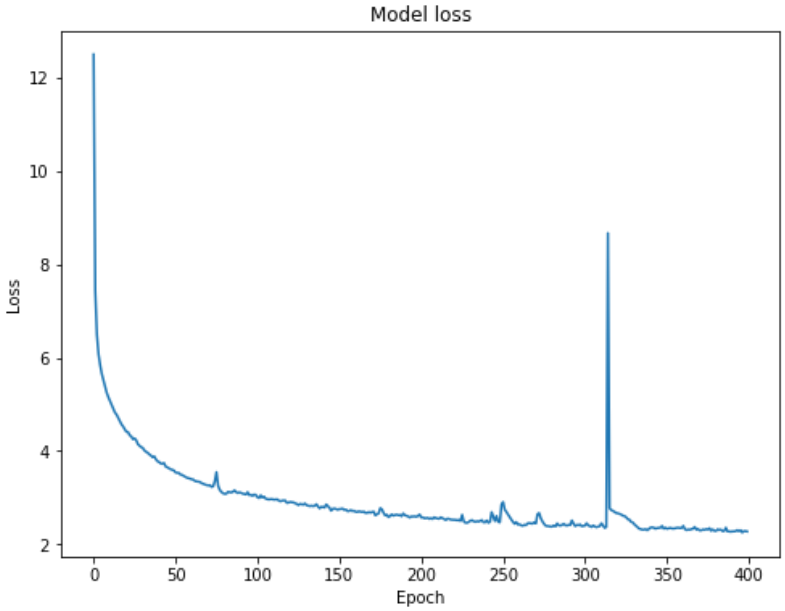


Figura 3.3. Antrenare rețea, evoluția funcției de cost.

Rețeaua neuronală a fost evaluată pe setul de validare al CityScapes, compus din 500 de imagini care nu au mai fost văzute de rețea în timpul antrenării. Rezultatul este similar cu cel obținut în abordările noastre precedente: 0.90 IoU. Scorul este mai mic față de abordările de top, fiind limitat de dimensiunea redusă a imaginii de intrare și de numărul redus de clase folosit pentru antrenare. Am folosit imagini de dimensiune 256 x 256 pixeli pentru a favoriza un timp de procesare cât mai redus, iar analiza timpilor de predicție este prezentată în figura 3.4.

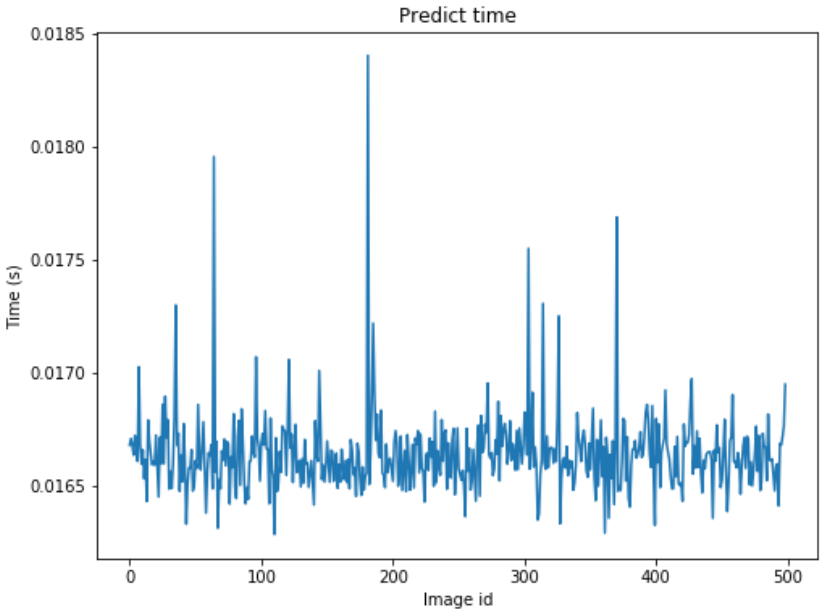
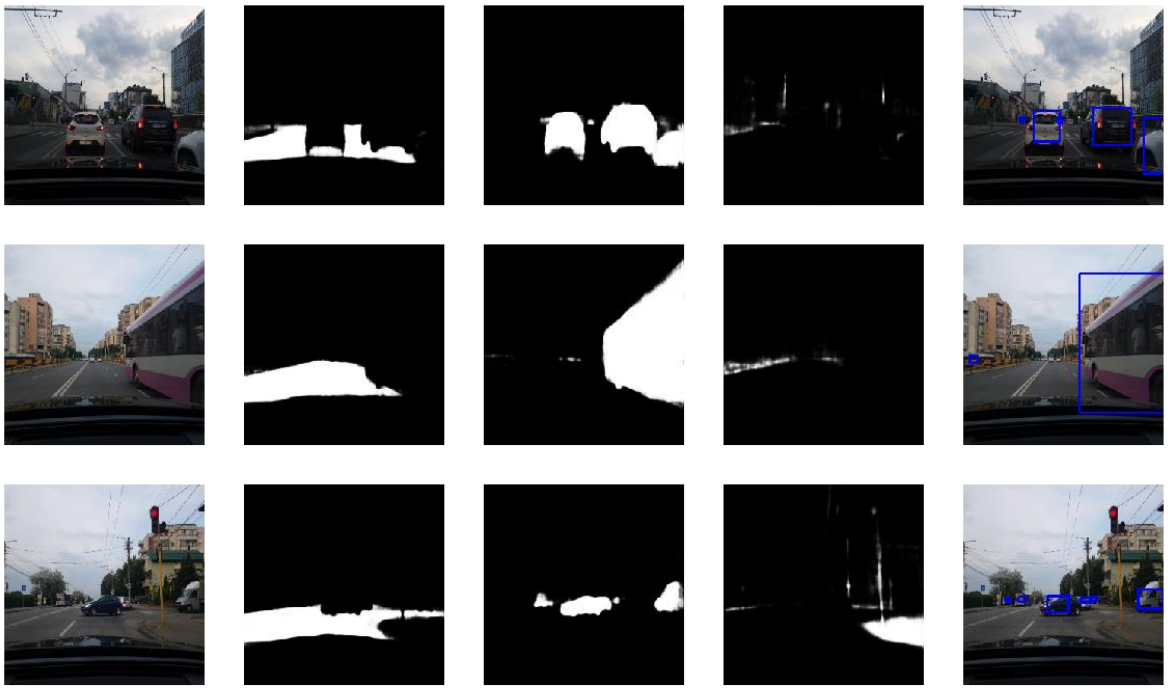


Figura 3.4. Analiză timpi de predicție (exprimat în secunde).

Este important de menționat faptul că graficul reprezintă numărul de secunde necesar pentru a face o predicție completă (segmentare dar și detecție de obiecte) pe o singură imagine. Timpul mediu pe setul de validare al CityScapes a fost de 16ms pentru detecție și segmentare ceea ce reprezintă un avantaj major față de abordările clasice bazate pe algoritmi sau chiar și față de alte rețele neuronale convoluționale care implementează în două rețele diferite partea de segmentare și cea de detecție, ceea ce necesită două predicții separate.

Rezultatele obținute folosind această rețea artificială sunt ilustrate în figura de mai jos. Imaginile de intrare sunt complet noi, nefolosite în procesul de antrenare.



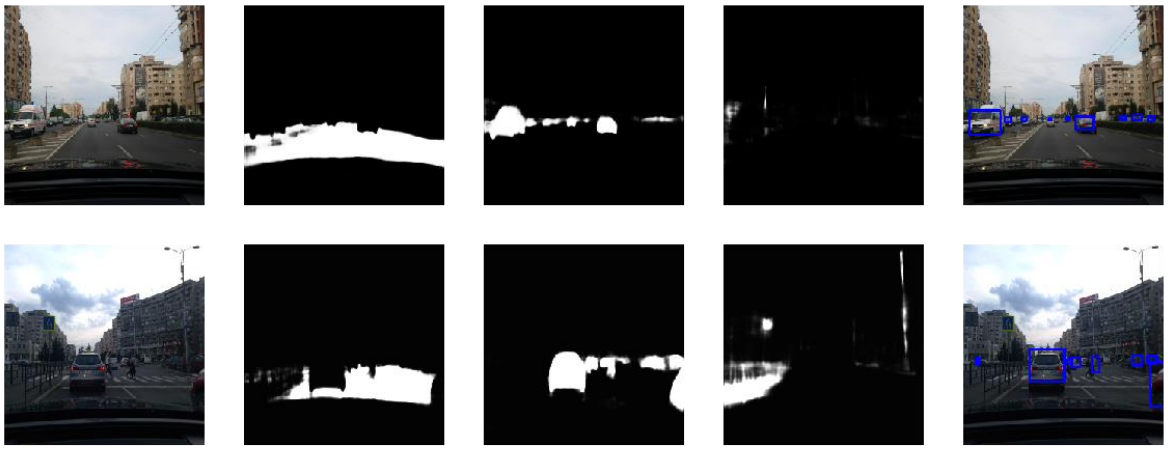


Figura 3.5. Exemple de predicție ale rețelei artificiale: prima coloană reprezintă imaginile de intrare, a doua coloană reprezintă segmentarea zonei de drum, urmată de segmentarea obstacolelor și apoi a trotuarelor (zone statice), iar ultima coloană reprezintă obstacolele detectate.

2.2. Îmbunătățirea estimatorului probabilistic

Estimarea scenei de trafic folosind particulele mobile grupate în harta de ocupare are avantaje datorită flexibilității modelului, care nu este legat de o anumită formă a obiectelor, și a ușurinței modelării dinamicii scenei. Totuși, acest model are două dezavantaje majore: nu permite urmărirea corectă a spațiului liber (nu există diferență clară între spațiu liber cu siguranță și spațiul liber ca absență a obstacolelor cunoscute, care poate fi datorat lipsei de percepție), și nu permite identificarea unică a obiectelor în cadre succesive (identitatea unică a obiectelor). În această etapă a fost extins modelul probabilistic, și procesul de inferență, pentru a remedia aceste două probleme.

2.2.1. Urmărirea spațiului liber

A fost proiectat un sistem pentru urmărirea în mod explicit a spațiului liber, folosind un alt set de particule, statice, numite particule de spațiu liber. Modelul de măsură se bazează pe segmentarea semantică, și zonele observate ca drum sunt folosite pentru a accentua prezența particulelor de tip spațiu liber. De asemenea, modelul acesta interacționează cu particulele de tip obstacol: atunci când particulele de tip obstacol migrează într-o celulă liberă, numărul particulelor libere scade.

Procesul este ilustrat în figura de mai jos. Modelul de măsură (primul panou) este folosit pentru a actualiza mulțimea de particule obstacol (al doilea panou), dar și mulțimea particulelor libere (al treilea panou). Estimarea finală este obținută prin fuzionarea celor două mulțimi de particule (panoul al patrulea).

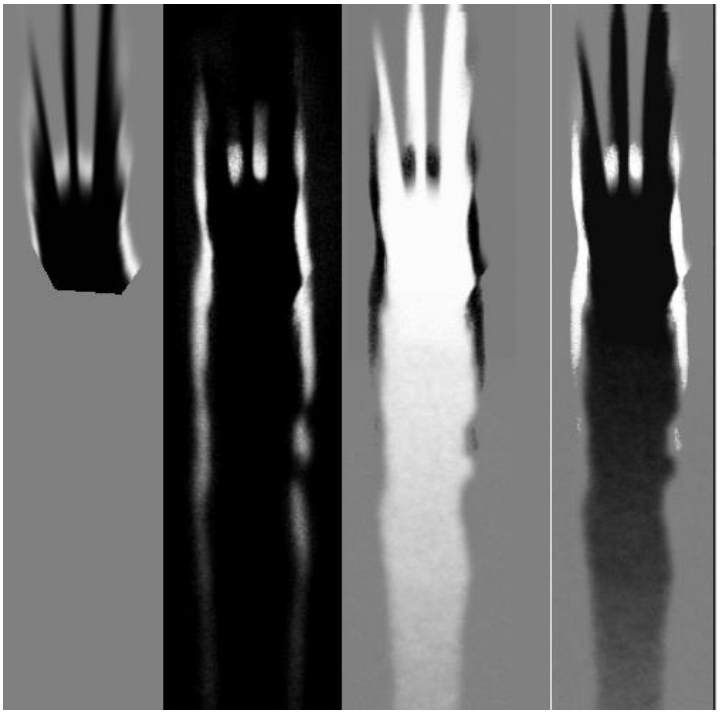


Figura 3.6. Modelarea și urmărirea mediului folosind particule obstacol și particule de tip spațiu liber. De la stânga la dreapta: modelul de măsură, modelul obstacolelor, modelul spațiului liber, modelul fuzionat. Intensitatea reprezintă probabilitatea de ocupare.

2.2.2. Urmărirea obiectelor individuale cu identitate proprie

Pentru realizarea unui sistem capabil de a urmări obiecte individuale, cu identitate proprie, au fost analizate soluțiile tipice pentru urmărirea obiectelor, dar acestea implică un model al geometriei obiectelor și un model global de mișcare, ceea ce înlătură flexibilitatea dată de sistemul de urmărire bazat pe particule. Modelul propus pentru urmărirea obiectelor este astfel următorul:

- Identitatea obiectelor este alocată particulei individuale, iar obiectul în sine este identificat prin gruparea particulelor cu identitate egală.
- Un nou obiect este creat din procesul de grupare a particulelor folosind criteriile de adiacență, dacă particulele din acel grup nu au identitate proprie.
- Identitatea particulei se păstrează prin replicarea între cadre.
- Un obiect unic este eliminat ca identitate dacă nu mai există particule cu identitatea respectivă.
- Identitatea obiectului nu depinde de o formă anume, starea lui fiind dată de starea particulelor componente. O formă cuboidală se poate genera ulterior, dar urmărirea nu depinde de ea.
- Nu este nevoie de utilizarea de filtre Kalman sau de alte filtre estimatoare suplimentare, pentru că procesul de estimare a poziției și a vitezei este implementat la nivel de particulă.

Procesul este ilustrat în figura 3.7. Identitatea obiectelor este prezentată în panoul din mijloc, care, spre deosebire de panoul etichetelor de grupare, păstrează culoarea între cadre.

Modelul cuboidal generat pe baza grupurilor identitare este prezentat în imaginile perspectivă din dreapta.

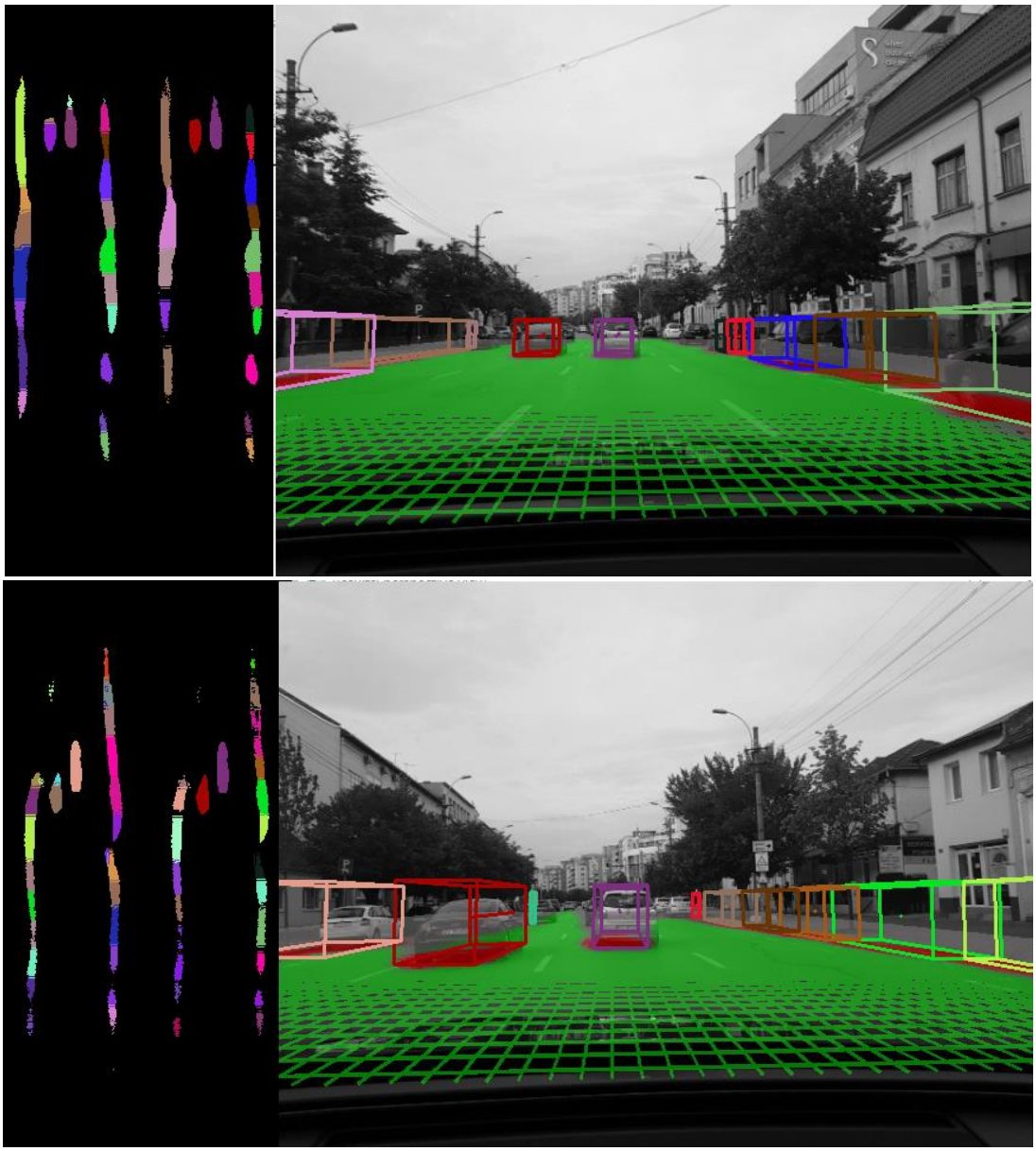


Figura 3.7. Urmărirea scenei, în două situații la 20 de cadre diferență: panoul din stânga reprezintă etichetarea grupurilor de celule ocupate, fără păstrarea identității, panoul din mijloc reprezintă celule etichetate cu identitatea particulelor componente, iar imaginea perspectivă prezintă obiectele cu culoarea identității lor, păstrată în timp.

3. Realizarea aplicației demonstrator

3.1. Arhitectura aplicației demonstrator

Aplicația demonstrator a fost implementată în limbajul C++, utilizând biblioteca TensorFlow pentru încărcarea rețelelor antrenate în mediul Python și folosirea lor pentru

segmentarea semantică, și facilitățile bibliotecii OpenCV pentru manipularea fișierelor imagine și pentru afișarea rezultatelor. Majoritatea modulelor aplicației, conținând algoritmii pentru procesarea fluxului de imagini, sunt cod sursă propriu scris de membrii echipei de proiect. Pentru integrarea în aplicația demonstrator nu au fost utilizate toate rezultatele obținute individual cu diferite configurații de rețele neuronale, ci doar partea de segmentare semantică, ce permite detecția generică a obstacolelor și a spațiului liber.

Structura logică a sistemului este prezentată în figura 8. Datele de intrare sunt secvențe de imagini preluate de la o cameră video color, iar datele odometrice (viteza și rata de rotație, sau viteza angulară) privind deplasarea vehiculului în care este montată camera pot fi preluate de la senzorii cu care este echipat vehiculul, sau de la un telefon mobil echipat cu GPS și giroscop. Deoarece majoritatea telefoanele mobile sunt echipate cu cameră, giroscop, accelerometru si GPS, toate datele de intrare au fost preluate de către un telefon.

Pentru ca imaginile să poată fi utilizate, sistemul trebuie calibrat, determinând astfel parametrii intrinseci și cei extrinseci. Distanța focală a camerei este calibrată o singură dată, analizând deplasarea laterală dintre cadre consecutive pe măsură ce vehiculul se rotește, cunoscând rata de rotație din datele de la senzorul giroscopic. Deoarece și determinarea deplasamentului în imagini, și datele de la senzorul giroscopic conțin erori/zgomote, se colectează mai multe eșantioane și se calculează distanța focală prin mediana distanțelor focale candidate.

Independent de calibrare, imaginile achiziționate sunt procesate folosind o rețea neuronală convoluțională de tip U-Net, pentru segmentarea semantică ce va separa zonele de tip drum, traversabile, de zonele de tip obstacol. Zonele de tip drum sunt apoi folosite pentru a extrage marcajele delimitatoare de bandă, ce vor fi apoi utilizate pentru calibrarea extrinsecă (determinarea înălțimii camerei față de sol, și a unghiurilor de aplecare / tangaj, și de orientare în plan orizontal / azimut). Unghiul de aplecare este rafinat apoi în fiecare cadru prin calcularea punctului de fugă / orizont, algoritm care compensează variația acestui unghi dată de mișcarea corpului autovehiculului datorită imperfecțiunilor suprafeței drumului.

Folosind parametrii camerei obținuți prin calibrare, imaginea segmentată prin CNN este reproiectată sub forma unei scene văzută de sus (birds eye view), și se vor identifica zonele de delimitare dintre drum și obstacole, sub forma unor linii poligonale. Pe baza acestor linii poligonale, se va genera modelul probabilistic de măsură, ce va ține cont de particularitatea procesului de observație și reproiectare (ne putem baza doar pe punctul de contact dintre obstacol și drum), și pe incertitudinile care variază cu distanța.

Modelul probabilistic de măsură este utilizat pentru a actualiza modelul probabilistic al lumii, bazat pe particule care formează o hartă de ocupare. Datele privind mișcarea autovehiculului sunt integrate cu modelul de mișcare al particulelor pentru a genera o predicție, care apoi este corectată de măsurători prin eliminarea sau multiplicarea particulelor.

Zonele din harta de particule cu probabilitate mare de a fi ocupate sunt grupate în obiecte individuale, folosind criterii de adiacență în harta de ocupare, dar și de compatibilitate a vitezelor celulelor. Obiectele generate sunt clasificate în două tipuri, pe baza vitezei lor: statice și dinamice.

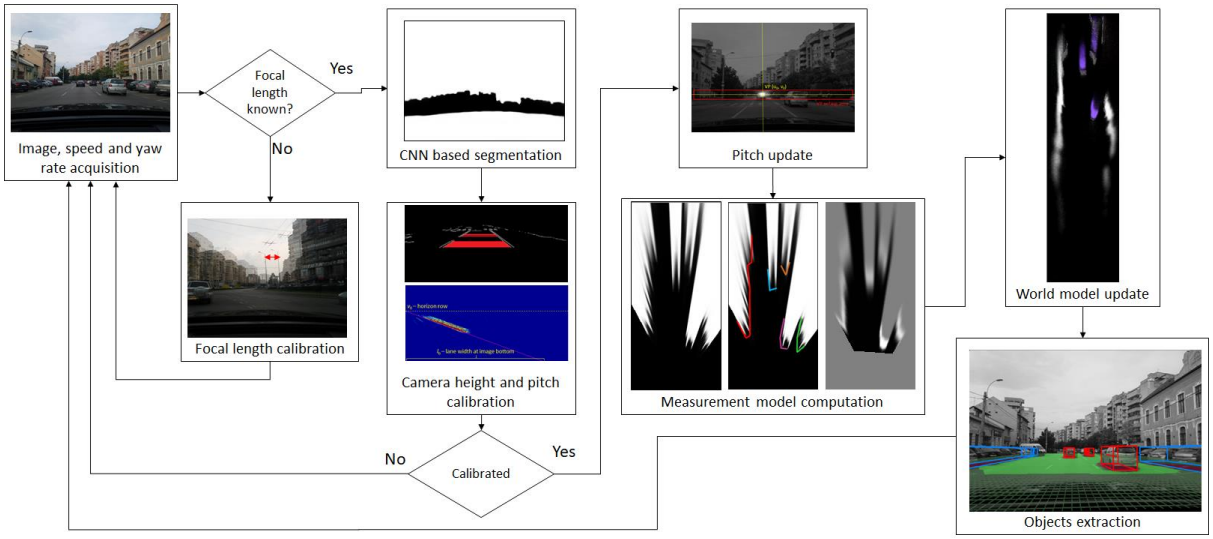


Figura 3.8. Vedere de ansamblu asupra sistemului integrat de procesare.

3.2. Evaluarea sistemului de detecție bazat pe urmărire

O evaluare a sistemului de detecție bazat pe urmărire folosind filtrul de particule a fost deja prezentată în raportul precedent și a fost efectuată pe secvențe din baza de date KITTI [1]. în acest raport prezentăm și o evaluare nouă detecției prin urmărire cu filtrul de particule, obținută prin compararea sistemului nostru cu alte sisteme bazate pe o singură cameră. Majoritatea soluțiilor existente oferă o performanță scăzută și nu sunt actualizate periodic. în timpul căutării sistemelor similare cu al nostru am observat că sistemul UGV Driver Assistance [27] este cel mai performant și are la bază o rețea neuronală convoluțională pentru vehiculele dintr-o scenă de trafic rutier. În tabelul 3.1 este prezentată o comparație cu sistemul UGV Driver Assistance. Rezultatele au fost obținute prin numărarea manuală a vehiculelor detectate la fiecare 10 cadre pe aceeași secvență de test pentru fiecare sistem în parte. Numărul total de vehicule din secvența de test este de 89.

	Numărul de vehicule detectate	**Rata de detecție (%)**	**Numărul de detecții fals-pozitive**	**False Discovery Rate (%)**
Sistemul nostru	88	92.6	5	0.05
UGV [9]	87	91.5	49	34.0

Tabelul 3.1. Analiza detecției în comparație cu un sistem comercial.

Din tabelul de mai sus putem observa faptul că sistemul UGV oferă o rată de detecție mult mai mare datorită eficienței rețelei artificiale care produce un număr mare de detecții (chenare de încadrare). Dezavantajul acestor abordări este că uneori rețeaua poate produce prea multe detecții și va oferi multe rezultate fals-pozitive. Figura următoare ilustrează aceste observații și prezintă și comparația cu sistemul nostru.

Figura 3.9. Comparație detecție vehicule cu o abordare bazată exclusiv pe rețele artificiale. în coloana din stânga este abordarea noastră, în coloana din dreapta este sistemul UGV Driver Assistant.

Din figura 3.9 putem observa că o rețea neuronală convoluțională poate oferi uneori predicții complet eronate (figura din dreapta din ultimul rând) sau poate să nu detecteze deloc din lipsă de trăsături relevante (figura din dreapta din al doilea rând). Soluția propusă de noi are o rază de detecție redusă, datorită modelului lumii folosit, dar detecțiile sunt robuste, în special datorită faptului că sunt și urmărite în timp.

În tabelul 3.2 este prezentată o comparație a sistemului propus de noi cu alte sisteme de detecție de obiecte. Comparația este efectuată în funcție de capabilități și caracteristici.

Metoda	Senzor principal	Calibrare automată	Detecție viteză obiecte	Detecție orientare obiecte	Suport pentru senzori multipli	Detecție obiecte generice
Mono 3D [28]	Sistem monocular	Nu	Nu	Da	Nu	Nu
OFT-Net [29]	Sistem monocular	Nu	Nu	Da	Nu	Nu
3DOP [30]	Sistem stereo-viziune	Nu	Nu	Da	Nu	Nu
Danescu [19]	Sistem stereo-viziune	Nu	Da	Da	Da	Da
Li [31]	LIDAR	Nu	Nu	Da	Da	Nu
Hu [32]	Sistem monocular	Nu	Nu	Da	Nu	Nu
Sistemul propriu	Sistem monocular	Da	Da	Da	Da	Da

Tabelul 3.2. Comparație caracteristici cu alte tehnici și abordări din literatura de specialitate.

Primele două metode prezentate în tabelul anterior și descrise în [28] și [29] sunt bazate pe rețele neuronale convoluționale. Limitarea lor principală este faptul că se bazează pe clase de obiect pentru extragerea informațiilor 3D și de aceea sunt limitate la detecția doar claselor de obiect folosite în procesul de antrenare al rețelei artificiale. Metoda descrisă în [30] folosește un sistem bazat pe stereoviziune și rețele neuronale pentru a genera propuneri de obiecte 3D sub forma unor cuboide orientate. Metoda prezentată în [19] are la bază un sistem de stereoviziune și o hartă de ocupare dinamică și reprezintă una dintre ideile principale din abordarea de față. Sistemul de stereoviziune oferă informații 3D utile, ceea ce oferă o rază de detecție mare și precizie mare, dar va necesita o calibrare precisă într-un mediu controlat. Totodată, acest sistem nu va putea integra și alte surse de măsurare a scenei (alți senzori).

Abordarea descrisă în [31] este bazată pe date obținute din senzori de tip LIDAR care măsoară distanțele folosind lumina (pulsații de laser). Astfel se obține o reprezentare 3D a scenei sub formă de puncte, iar această reprezentare este convertită într-una 2D, care apoi este analizată de o rețea artificială pentru a genera cuboide orientate pentru obiectele din scenă. Această metodă este capabilă de a detecta obiecte de orice tip (detecție generică) și oferă suport pentru achiziția de date din senzori multipli. Totuși, folosirea unui senzor de tip LIDAR necesită o calibrare specifică și necesită o montare precisă în exteriorul vehiculului.

Soluția din lucrarea [32] folosește o rețea neuronală convoluțională pentru extragerea zonelor de obiect candidat și apoi utilizează o altă rețea artificială pentru a estima orientarea și dimensiunea obiectelor. De asemenea este folosită și o rețea neuronală recurentă de tip LSTM (“long short-term memory”) pentru urmărirea obiectelor detectate. Principalul dezavantaj al acestei abordări este dat de necesitatea unor date de antrenare foarte bune,

autorii au menționat și faptul că au folosit imagini sintetice în procesul de antrenare și implementare al soluției. O altă limitare este dată de tipurile de obiecte folosite în timpul antrenării, de aceea această soluție nu va detecta obiecte generice într-o scenă de trafic rutier. Totuși, datorită detecție și apoi a urmăririi obiectelor, soluția tratează ocluzia obiectelor foarte bine.

Metoda propusă de noi este bazată pe o singură cameră, dar are avantajul că se poate calibra automat în timpul conducerii vehiculului (presupunând că există un număr suficient de cadre de trafic rutier procesate). Soluția noastră are capabilitatea de a detecta obstacole generice prezente pe suprafața șoselei și apoi de a determina viteza și orientarea lor. Scena de drum care este urmărită în timp este reprezentată de o hartă de ocupare care suportă diferite intrări senzoriale, ceea ce înseamnă că sistemul propus de noi poate fi extins prin adăugarea de mai multe camere.

Diseminarea rezultatelor

Rezultatele obţinute în prima etapă au fost publicate în următoarele jurnale/conferinţe internaționale:

- Diana Borza, Razvan Itu, Radu Danescu, “In the Eye of the Deceiver: Analyzing Eye Movements as a Cue to Deception“, Journal of Imaging, Vol. 4, No. 10, 2018, Art. No. 120. **[jurnal indexat ISI, fara factor de impact]**
- 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. **[Capitol carte Springer]**

Rezultatele obținute în cadrul etapei a doua au fost publicate în următoarele lucrări, în volume de conferințe internaționale:

- Radu Danescu, Razvan Itu, "Camera Calibration for CNN Based Generic Obstacle Detection", EPIA Conference on Artificial Intelligence, 2019, pp. 623-636. **[Capitol carte Springer]**
- Radu Danescu, Razvan Itu, Diana Borza, "Dynamic 3D Environment Perception Using Monocular Vision and Semantic Segmentation", 2019 IEEE 15th International Conference on Intelligent Computer Communication and Processing (ICCP 2019). **[conferință indexată ISI]**
- Sorana Capalnean, Florin Oniga, Radu Danescu, “Obstacle Detection Using a Voxel Octree Representation”, 2019 IEEE 15th International Conference on Intelligent Computer Communication and Processing (ICCP 2019). **[conferință indexată ISI]**

Rezultatele obținute în cadrul ultimei etape au fost publicate într-un jurnal cotat ISI, din categoria **Q1 (zona roșie), factor de impact 3.031**:

- Razvan Itu, Radu Danescu, "A Self-Calibrating Probabilistic Framework for 3D Environment Perception Using Monocular Vision", Sensors, Vol. 5, No. 20, 2020, Art. No. 1280.

De asemenea, rezultatele obținute în cadrul acestui proiect au stat la baza finalizării tezei de doctorat cu titlul: "**Sistem de percepție monoculară folosind viziune și inteligență artificială**", elaborată de As. Ing. Razvan Itu sub îndrumarea Prof. Dr. Ing. Radu Danescu.

Concluzii

În urma activităților din cadrul proiectului DeepSense, au fost aduse contribuții atât pe partea de interpretare a imaginilor folosind rețelele neuronale convoluționale (sau Deep Learning), cât și pe partea de modelare probabilistică a mediului din traficul auto, și estimarea stării acestui mediu folosind modelul probabilistic propus și datele de măsură extrase prin CNN. Soluția dezvoltată în cadrul acestui proiect folosește avantajele ambelor metode: extragerea informațiilor prin CNN nu necesită calibrare, și sistemul își găsește propriile reguli atâta timp cât există suficiente date de antrenare, iar abordarea probabilistică a urmăririi mediului folosește avantajul legilor de mișcare pentru prezicerea stărilor în cadre succesive, și poate integra diferite informații, chiar dacă acestea nu sunt lipsite de erori.

Un alt rezultat semnificativ este calibrarea automată a sistemului, atât din punct de vedere intrinsec cât și a parametrilor extrinseci.

Noutatea și relevanța rezultatelor au fost confirmate prin publicarea în șase lucrări internaționale, din care două în jurnale ISI. De asemenea, a fost finalizată o teză de doctorat.

Pe viitor ne propunem să integrăm complet rețeaua neuronală convoluțională cu multiple predicții (detaliată în acest raport) în aplicația demonstrator realizată în C++. Acest pas necesită o conversie a întregului cod într-un format comun care să poată fi citit și interpretat în C++ și va necesita mai mult timp. În momentul de față aplicația demonstrator în C++ folosește doar partea de segmentare semantică având predicții în timp real ale unei rețele convoluțioanle de tip U-Net realizată în software-ul PyTorch și exportată folosind software-ul TensorRT. De asemenea, ne propunem să folosim toate rezultatele segmentării pentru procesul de creare a hărții de ocupare dinamică a filtrului de particule și nu doar imaginea cu zona de drum segmentată, astfel să implementăm o fuziune mai completă și mai complexă a datelor despre scena de trafic rutier.

Considerăm că prin rezultatele obținute, și prin lucrările publicate, am îndeplinit cu succes obiectivele proiectului. Cu toate acestea, tehnologia în domeniu este în continuă evoluție, și interesul industriei în domeniul vehiculelor autonome este ridicat, și noi rezultate, mai bune decât cele existente, sunt oricând posibile.

Bibliografie

[1] - Geiger, A, Lenz, P, Urtasun, R, “Are we ready for autonomous driving?”, Computer Vision and Pattern Recognition Conference (CVPR), pp. 3354-3361, 2012.
[2] - Udacity Vehicle Dataset. Disponibil online: https://github.com/udacity/self-driving-car/tree/master/annotations
[3] - M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, “The Cityscapes Dataset for Semantic Urban Scene Understanding,” in Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.
[4] - Yu F., et. al, “BDD100K: A Diverse Driving Video Database with Scalable Annotation Tooling“, preprint arxiv:1805.04687, 2018, online: https://arxiv.org/abs/1805.04687.
[5] - O. Ronneberger, P. Fischer, T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation", Medical Image Computing and Computer-Assisted Intervention (MICCAI), Springer, LNCS, Vol. 9351: 234-241, 2015, available at arXiv:1505.04597.
[6] - TensorFlow. Disponibil online: https://www.tensorflow.org/
[7] - Keras. Disponibil online: https://keras.io/
[8] - OpenCV. Disponibil online: https://www.opencv.org/
[9] - King, D.E., “Dlib-ml: A machine learning toolkit”, J. Mach. Learn. Res. 2009, 10, 1755 - 1758.
[10] - Howard, A, et al, “Mobilenets: Efficient convolutional neural networks for mobile vision applications”, 2017, arXiv:1704.04861(preprint).
[11] - E. Romera, J. M. Alvarez, L. Miguel Bergasa, R. Arroyo, "ERFNet: Efficient Residual Factorized ConvNet for Real-Time Semantic Segmentation", IEEE Transactions on Intelligent Transportation Systems, vol. 19, no. 1, pp. 263-272, 2018.
[12] - 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.
[13] - L.C. Chen, Y. Zhu, G. Papandreou, F. Schroff, H. Adam, “Encoder- Decoder with Atrous Separable Convolution for Semantic Image Segmentation”, arXiv: 1802.02611, 2018.
[14] - A. Paszke, A. Chaurasia, S. Kim, E. Culurciello, "ENet: A Deep Neural Network Architecture for Real-Time Semantic Segmentation", arXiv: 1606.02147, 2016.
[15] - M. Fischler, R. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography", Communications of the ACM, Volume 24, pp. 381–395, 1981.
[16] - R. Danescu, R. Itu, "Camera Calibration for CNN Based Generic Obstacle Detection", EPIA Conference on Artificial Intelligence, 2019, pp. 623-636.
[17] - R. Itu, R. 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.
[18] - R. Itu, D. Borza, R. 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.
[19] - 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.

[20] - R. Danescu, R. Itu, A. Petrovai, "Generic Dynamic Environment Perception Using Smart Mobile Devices", Sensors, vol. 16, no. 10, art. no. 1721, 2016.
[21] - R. Itu, R. Danescu, "An Efficient Obstacle Awareness Application for Android Mobile Devices", International Conference on Intelligent Computer Communication and Processing (ICCP 2014), 2014, pp. 157-163.
[22] - M. Bertozzi and A. Broggi, “GOLD: a Parallel Real-Time Stereo Vision System for Generic Obstacle and Lane Detection”, IEEE Trans. on Image Processing, pp. 62-81, 1998.
[23] - R. Danescu, R. Itu, D. Borza, "Dynamic 3D Environment Perception Using Monocular Vision and Semantic Segmentation", 2019 IEEE 15th International Conference on Intelligent Computer Communication and Processing (ICCP 2019).
[24] - K. He, X. Zhang, S. Ren, J. Sun, “Deep Residual Learning for Image Recognition”, Computer Vision and Pattern Recognition, pp. 770-778, 2016.
[25] - Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi, “You Only Look Once: Unified, Real-Time Object Detection”, arXiv 1506.02640, 2016.
[26] - YOLO ResNet. Disponibil online: https://github.com/makatx/YOLO_ResNet (accesat aprilie 2020).
[27] - UGV Driver Assistant. Disponibil online: https://play.google.com/store/apps/details?id=com.infocomltd.ugvassistant&hl=en (accesat aprilie 2020).
[28] - X. Chen, et al., “Monocular 3D Object Detection for Autonomous Driving”. Disponibil online: https://www.cs.toronto.edu/~urtasun/publications/chen_etal_cvpr16.pdf (accesat aprilie 2020).
[29] - T. Roddick, A. Kendall, R. Cipolla, “Orthographic Feature Transform for Monocular 3D Object Detection”, arXiv:1811.08188, 2019.
[30] - X. Chen, et al., “3D Object Proposals for Accurate Object Class Detection”. Disponibil online: https://papers.nips.cc/paper/5644-3d-object-proposals-for-accurate-object-class-detection (accesat aprilie 2020).
[31] - B. Li, T. Zhang, T. Xia, “Vehicle detection from 3d lidar using fully convolutional network”, arXiv:1608.07916, 2016.
[32] - H.-N. Hu, et al., “Joint Monocular 3D Vehicle Detection and Tracking”. Disponibil online: http://openaccess.thecvf.com/content_ICCV_2019/papers/Hu_Joint_Monocular_3D_Vehicle_Detection_and_Tracking_ICCV_2019_paper.pdf (accesat aprilie 2020).

30.04.2020
Director proiect
Prof. Dr. Ing. Radu Danescu