UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

Modelarea, Estimarea si Gestionarea Situatiilor Periculoase prin Analiza Continua a Sistemului Sofer-Vehicul-Mediu - MEDALS

Echipa de cercetare: Radu Dănescu, Răzvan Itu, Raluca Brehar, Mircea Paul Mureșan, Attila Fuzes

Etapa 2 – Implementare și analiză

Cuprins

Introducere – rezumatul etapei

În această etapă au fost continuate activitățile de colectare și adnotare a datelor, această activitate fiind orientată în patru direcții: adnotarea imaginilor pentru percepția elementelor de trafic, adnotarea automată folosind senzori activi, adnotarea stării șoferului observată de o cameră plasată în interiorul autovehiculului, și adnotarea generică a stării de periculozitate a scenariului de trafic. De asemenea, a fost dezvoltată o soluție bazată pe rețele neuronale pentru percepția de tip panoptic a mediului, din imagini monoculare, soluție care poate rula pe dispozitive mobile. Au fost, de asemenea, implementate metode pentru analiza în timp real a stării șoferului, și pentru estimarea gradului de periculozitate a scenei. A fost începută dezvoltarea aplicației demonstrator, bazată pe arhitectura cu GPU limitat nVidia Jetson. În această etapă au fost elaborate trei publicații, un articol de jurnal și două articole de conferință. În consecință, considerăm că obiectivele etapei au fost îndeplinite cu succes.

1. Achiziție si adnotare de date (partea 2)

În cadrul acestei activități s-a urmărit generarea de imagini de tip referință (Ground Truth) pentru antrenarea soluțiilor de percepție a diferitelor tipuri de obiecte relevante, pentru analiza stării șoferului, precum și a percepției situațiilor pentru analiza automată a scenariilor de pericol. De asemenea, a fost abordată o soluție pentru adnotarea automată prin utilizarea de senzori suplimentari, cu mare precizie și certitudine, împreună cu imaginile de flux video.

1.1. Adnotarea manuală pentru segmentarea semantică din imagini termale

Imaginile termale (IR – InfraRed) sunt imagini în care nivelul de intensitate este proporțional cu temperatura obiectului reprezentat. Deoarece pietonii au o temperatură a corpului caracteristică, ei sunt în mod special vizați de analiza în domeniul infraroșu. În urma adnotării din etapa trecută a 1000 de imagini, s-a constatat că acest set nu este suficient, astfel că a fost adnotat încă un set de 1000 de imagini, ce acoperă multiple scenarii dificile în condiții de luminozitate diferită, temperatură crescută, și diverse obiecte, pietoni sau animale prezente pe carosabil. De asemenea, setul nou de date conține imagini termale din perspective diferite, necesare sistemelor de percepție a mediului în momentul în care se dorește execuția unor manevre variate, precum întoarcerile. Spre exemplu, în Figura 1 este ilustrat un scenariu achiziționat vara pe timp de noapte. Din cauza faptului că pietonul este mult mai cald decât carosabilul, se pot observa chiar și urmele deplasării acestuia pe drum.

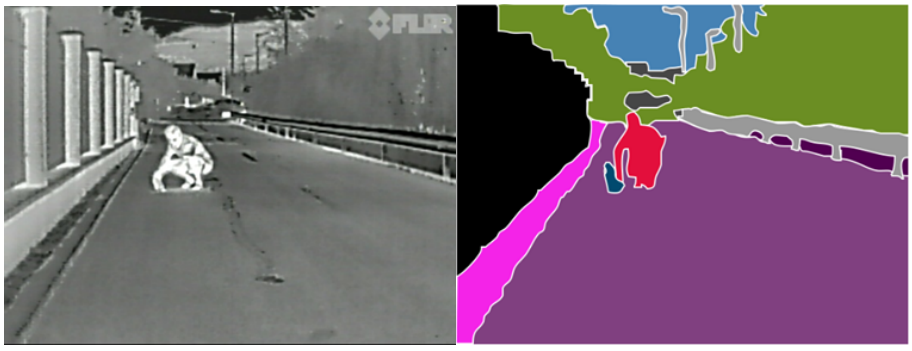

Figura 1. Persoană în mijlocul drumului, cu animalul de companie: imaginea termală și imaginea adnotată.

In Figura 2, este adnotata o imagine în care se poate observa fenomenul inversării polarității. În acest scenariu, pietonul este mult mai rece decât obiectele înconjurătoare, care sunt încălzite de, apărând

așadar mai închis la culoare. Imaginea a fost achiziționată în timpul amiezii unei zile călduroase de vara.

Figura 2. Imagine în care este ilustrat fenomenul de inversare a polarității.

1.2. Utilizarea de senzori activi pentru adnotarea automată

Adnotarea imaginilor în mod manual implică un efort considerabil, mai ales dacă această adnotare se realizează la nivel de pixel, așa cum este cazul adnotării pentru segmentarea semantică sau pentru segmentarea panoptică. O soluție alternativă este utilizarea de senzori suplimentari, calibrați împreună cu camerele video, dar care pot oferi în mod direct informație 3D, informație care poate fi procesată în mod direct pentru a determina dacă zona observată, și implicit un anumit grup de pixeli, reprezintă o zonă de drum, sau de obstacol.

Senzorii LiDAR sunt capabili să estimeze distanțele către obiectele din jur cu o precizie ridicată. De asemenea, acești senzori sunt capabili să funcționeze atât ziua cât și noaptea, fără a avea pierdere de performanță. Senzorii LiDAR oferă informația 3D sub formă de nor de puncte, care trebuie procesate pentru a identifica obiectele individuale, dar și suprafața drumului. Se propune o abordare în timp real, implementată folosind tehnologia CUDA, ce rulează pe un procesor grafic (GPU). Stadiile de procesare sunt ilustrate în Figura 3.

Figura 3. Stadiile de procesare principale ale algoritmului propus.

În faza de preprocesare se creează o hartă de tip caroiaj (grid), cu dimensiunea celulei de 10x10 cm, și în fiecare celulă se memorează punctul care are elevația (înălțimea) maximă. Se filtrează apoi punctele selectându-se acele puncte care au o elevație mai mică decât un prag predefinit. Filtrarea punctelor și crearea hărții de elevație se execută în paralel.

În pasul următor se detectează punctele de drum. Pentru a beneficia de arhitectura paralelă GPU, se implementează o versiune paralelă a algoritmului RANSAC. În loc să se facă R iterații, se generează în paralel R modele. Aceasta variantă de execuție este mai eficientă decât cea în care s-ar analiza toate punctele unui model și apoi s-ar trece la următorul model, iar fiecare nucleu ar fi responsabil să calculeze distanța de la un singur punct la un model, deoarece în implementarea curentă se folosește accesul “coalesced” la memoria globală. Atunci când un fir de execuție de pe GPU citește sau scrie în

memoria globală, el accesează o bucată mai mare din aceasta chiar daca are nevoie doar de un singur element. Daca alte fire de execuție accesează memoria într-un mod similar, atunci GPU-ul poate exploata acest lucru, reutilizând bucata de memorie deja adusa din memoria globală. GPU-ul este utilizat cu eficiență maximă atunci când firele de execuție accesează zone de memorie continue. Acest tipar de accesare a memoriei se numește coalesced.

După generarea modelelor, pentru fiecare punct se calculează distanța de la un punct la fiecare model în mod paralel. După selecția celui mai bun model, toate punctele care au fost inlier pentru model vor fi considerate puncte de drum. In Figura 4 se pot observa rezultatele separarii punctelor de drum in doua scenarii diferite.

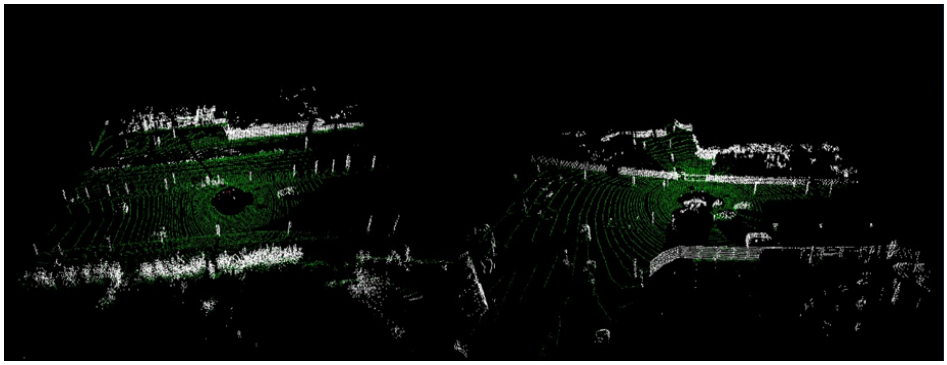

Figura 4. Rezultatele separării punctelor de drum de punctele obstacol. Punctele de drum sunt marcate cu verde.

După detecția punctelor de drum, se va continua cu identificarea obstacolelor, prin gruparea punctelor 3D care nu aparțin drumului. Pentru gruparea punctelor 3D, folosim o abordare care se bazează pe densitatea norului de puncte și joacă un rol important în identificarea structurilor neliniare. Această abordare este cunoscută în mod obișnuit sub numele grupare spațială bazată pe densitate a aplicațiilor cu zgomot (density-based spatial clustering of applications with noise) sau DBSCAN [Ref]. În DBSCAN sunt necesari doi parametri, o distanță de prag care este utilizată pentru cele două concepte de accesibilitate și conectivitate și un număr minim de puncte necesare pentru a crea un cluster. Metoda de clustering DBSCAN poate fi paralelizată deoarece nu există condiții de cursă în ceea ce privește ordinea în care luăm în considerare punctele care urmează să fie atribuite unui cluster. În abordarea noastră, pe lângă implementarea unei versiuni paralele a algoritmului DBSCAN, aplicăm și o codificare folosind piloni. Stâlpii (pilonii) sunt un tip de voxel care au o înălțime egală cu înălțimea scenei. Aceștia au dimensiuni egale, iar lățimea și lungimea lor sunt egale cu o dimensiune predefinită. Dimensiunea stâlpilor este importantă deoarece poate afecta performanța și precizia procesului de detectare a obiectelor.

În abordarea noastră, fiecare punct este alocat unui pilon pe baza coordonatelor lui x și y. Punctele sunt analizate în paralel și sunt atribuite stâlpului căruia îi aparțin. Primul pas în procesul de grupare este căutarea unui pilon de pornire de la care să poată începe generarea unui cluster. Pentru ca un pilon să fie valid și să fie luat în considerare, acesta trebuie să aibă un număr minim de puncte (valoare care este setată într-un fișier de configurare înainte de rularea programului) și să nu fie atribuit în prealabil niciunui alt cluster. După selectarea unui pilon de început, stâlpii învecinați sunt analizați în paralel și sunt alocați clusterului curent dacă pilonul este valid și accesibil. Procesul se termină atunci când este găsit un pilon invalid și gruparea nu continuă de la acel pilon. Algoritmul menționat mai sus îmbunătățește foarte mult timpul de rulare al soluției în comparație cu o abordare standard care se aplică doar punctelor.

Generarea clusterelor și a casetei de delimitare este urmată de o etapă de rafinare în care clusterele care sunt apropiate unele de altele sunt îmbinate într-unul singur. Analizăm fiecare pereche de casete de delimitare calculând cea mai scurtă distanță dintre colțurile lor. Două casete sunt îmbinate dacă cea mai scurtă distanță este mai mică decât un prag predefinit, care în cazul nostru are valoarea de 20 cm. După ce toate perechile de cutii de delimitare sunt comparate și casetele de delimitare care aparțin aceluiași obiect sunt îmbinate, procesul de rafinare este încheiat. În Figura 5 sunt ilustrate rezultatele detectorului de obiecte pe doi nori de puncte diferiți.

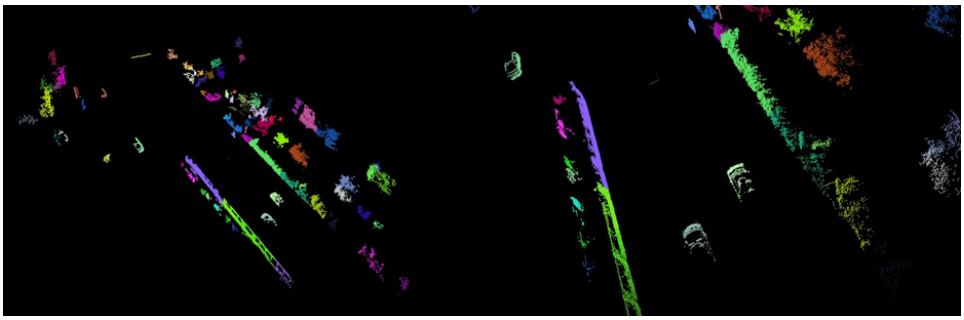

Figura 5. Ilustrarea obiectelor identificate din doua secvențe. Fiecare obiect individual are o culoare proprie.

În figura 6 putem observa rezultatele grupării obiectelor împreună cu punctele de drum.

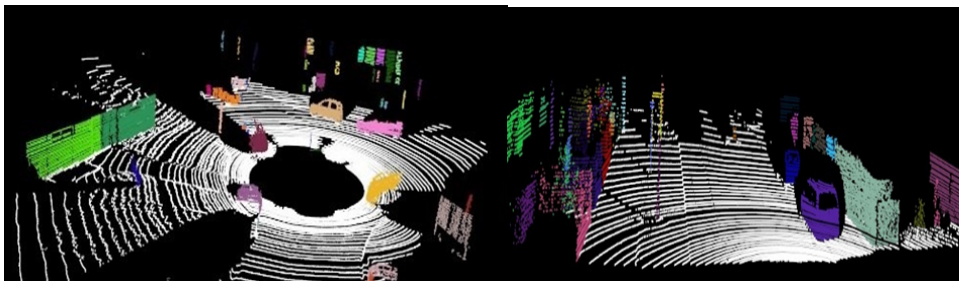

Figura 6. Obiectele identificate, și punctele de drum.

Timpul mediu de execuție al operației de extragere a drumului este de 0.34ms, iar timpul mediul pentru detectorul de obiecte este de 11.2 ms, evaluarea fiind făcută pe nori de puncte din setul de date KITTI. Acești timpi nu sunt adecvați pentru rularea în timp real, dar sunt suficienți pentru a putea utiliza aceste rezultate ca surse de date pentru adnotarea automată a imaginilor preluate în același timp.

1.3. Achiziție și adnotare de date pentru percepția stării șoferului

Deoarece cea mai sigură modalitate a determina starea de atenție a șoferului este monitorizarea directă a acestuia (conform studiului Asociației Automobilistice Americane AAA [1]), pentru monitorizarea acțiunilor șoferului a fost instalată o cameră web cu rezoluție Full HD, conectată prin USB 2.0 la sistemul mobil de procesare automată din autovehicul, sistem bazat pe placa nVidia Jetson. Camera a fost plasată în spatele oglinzii retrovizoare, într-un fel, în care nu ar împiedica șoferul. Camera are un câmp vizual de 135° și poate funcționa oarecum în condiții de lumină scăzută. Accesul la sursa de imagini se face prin metode standard ale bibliotecii OpenCV, sub sistemul de operare Linux. În figura 7 este ilustrată poziționarea camerei în autovehicul.

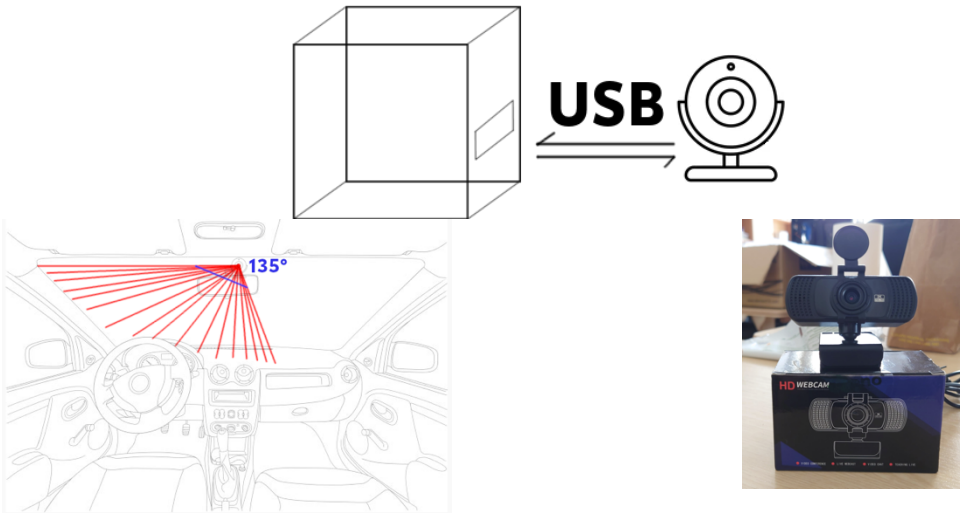

Figura 7. Schema de achiziție a imaginilor din interiorul mașinii

Pentru imagini s-a folosit o rezoluție de 640 x 480, care, împreună cu modelul bazat pe rețele neuronale artificiale FaceDetect, rulează la o viteză de zece imagini pe secundă pe dispozitivul de procesare mobil, viteză la care se face și salvarea imaginilor pe dispozitivul de stocare.

S-au captat mai multe secvențe atât pe timp de zi cât și pe timp de noapte, fiind analizați doi conducători auto, în două autovehicule diferite.

1.4. Adnotare de date pentru analiza pericolului

În afară de percepția scenei la nivel de elemente componente precum suprafața drumului, autovehicule, pietoni, dorim să avem și o percepție la un nivel global a unui anumit grad de periculozitate, care să nu derive din analiza individuală a obiectelor din scenă, ci direct din imaginea furnizată unui clasificator neuronal. O astfel de clasificare nu ar fi de ajutor unui sistem de conducere automată, dar este de mare ajutor în asistarea unui șofer uman, pentru a preveni accidentele de circulație.

Folosind un set de date publice (setul JAAD), dar și imagini achiziționate de noi, s-a realizat adnotarea pericolului pe grade: scăzut, mediu și înalt. Categoria de pericol scăzut, surprinsă în figura 8 de mai jos, cuprinde situații în care pietonii sunt localizați în afara drumului, sunt departe de mașini, merg sau stau pe trotuar. Această categorie de imagini reprezintă situații de relativă siguranță în trafic.

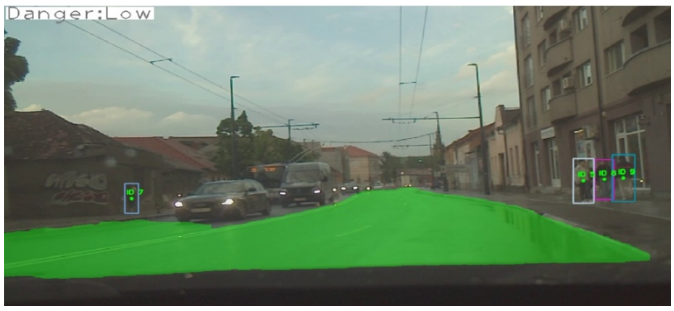

Figura 8. Exemplu de situație etichetată cu nivel de pericol scăzut.

Categoria de pericol sau risc mediu cuprinde situații în care pietonii sunt pe stradă și autovehiculul se îndreaptă spre ei (dar este încă la o distanta de siguranță fata de ei), sau situații în care pietonii urmează să treacă sau trec strada iar restul autovehiculelor angajate în trafic își încetinesc deplasarea sau chiar se opresc. O astfel de situație este surprinsa in figura 9.

Figura 9. Exemplu de situație etichetată cu nivel de pericol mediu.

Categoria de pericol ridicat conține situații în care autovehiculul propriu accelerează spre un pieton (care este pe stradă sau foarte aproape), sau situații în care pietonii sau bicicliști sunt pe stradă și sunt aproape de alte vehicule participante la trafic. Figura 10 arată o situație de acest fel (pietonul este pe stradă în timp ce noi accelerăm).

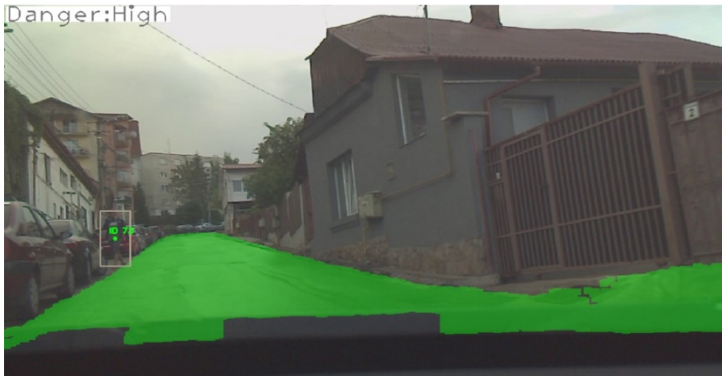

Figura 10. Exemplu de situație etichetată cu nivel de pericol ridicat.

Setul de date conține adnotări pe imagini de referință din JAAD [2], [3], pentru care s-au adnotat 3850 de imagini din care 1545 cu pericol ridicat, 1218 cu pericol mediu si 1087 cu pericol mic. Totodată, s-au făcut adnotări și pe secvențe achiziționate cu sistemul dezvoltat in cadrul acestui proiect. S-au adnotat 140 de imagini din care 8 cu risc ridicat, 22 cu risc mediu si 110 cu risc mic. Pe aceste imagini s-a adnotat atât gradul de pericol, cât și poziția pietonilor și a autovehiculelor în imagini.

2. Proiectare si implementare a unui sistem pentru percepție limitata a mediului pe dispozitive mobile (partea 2)

În cadrul acestei activități s-a realizat un sistem capabil să realizeze cel mai complex tip de segmentare, segmentarea panoptică, care combină segmentarea semantică cu identificarea instanțelor (deci cu identificarea fiecărui obiect în parte). Sistemul apoi aplică un algoritm de urmărire (tracking) atât pe cuboidele care conțin obiectele, cât și pe măștile (mulțimile de pixeli) ce aparțin obiectelor.

Sistemul utilizează trei modele de rețele neuronale pentru realizarea segmentării panoptice și pentru validarea acesteia, și un model bazat pe ingineria trăsăturilor pentru realizarea urmăririi obiectelor.

Pentru segmentarea semantică s-a folosit un model de rețea neuronală bazat pe ERFNet, care a fost antrenat pe imaginile adnotate prezentate în capitolul 1. Acest model rulează în 8 ms pe GPU, pe imagini având rezoluție VGA, și este capabil să identifice clasele semantice relevante precum pieton,

autovehicul, drum. Pentru detecția instanțelor s-a folosit o combinație între modelele de rețele neuronale Yolact si YOLOv5, motivul fiind că în cazul în care modelul Yolact folosește un prag de detecție mai mare de 50% numărul detecțiilor este foarte mic, iar daca pragul este mai mic de 50% se afișează prea multe detecții false. Pentru a păstra detecțiile și măștile oferite de YOLACT, dar și pentru a valida dacă detecțiile sunt corecte, s-a folosit o rețea YOLOv5 antrenata pe setul de data FLIR ADAS. Se setează scorul de confidență a detecțiilor algoritmului YOLACT la o valoare foarte mică, pentru a se genera multe detecții, apoi se calculează IoU (raportul dintre intersecție și reuniune) între obiectele YOLOv5 și cele generate de YOLACT, și pentru fiecare detecție YOLOv5 se păstrează doua dintre obiectele YOLACT care au valoarea IoU calculată cea mai mare, cu precizarea că valoarea cea mai mică nu trebuie să fie sub un prag setat la valoarea 0.8.

În pasul de validare a detecțiilor se verifică apoi dacă clasele semantice ale obiectelor YOLOv5 corespund cu clasa semantică a cel pun unuia dintre cele două obiecte YOLACT. Dacă clasele corespund, obiectele sunt considerate valide. În caz contrar, se verifică clasa dominantă din chenarul de încadrare generat de YOLOV5, folosind imaginea de segmentare semantică. Astfel, a fost implementat un algoritm care face o histogramă a claselor semantice din chenarul generat de YOLOV5. Din această histogramă se alege clasa dominantă pentru a verifica clasa cuboidului generat. Algoritmul de votare a fost implementat pe GPU folosind frameworkul CUDA. În cazul în care două din cele trei clase semantice obținute folosind YOLOv5, Yolact si segmentarea semantica sunt identice, atunci chenarul este validat ca având acea clasă semantică. În cazul în care cele trei rețelele neuronal oferă valori diferite, clasa obiectului este raportată ca necunoscută. În ultimul pas se verifică dacă mai există obiecte YOLACT care au un scor de confidență peste 50% și care nu au fost asociate unor obiecte YOLOv5 din cauza inexistentei detecțiilor YOLOv5 in apropierea acestora. În cazul în care astfel de obiecte există, se verifică clasa semantică în chenarul generat de YOLACT, și în cazul în care cele doua clase semantice sunt identice obiectul este validat. În caz contrar, obiectului i se atribuie clasa necunoscut. De asemenea, se mai verifică și dacă există obiecte YOLOv5 ce încă nu au fost asociate unor obiecte YOLACT, caz în care se verifică clasa obiectelor YOLOv5 cu datele din segmentarea semantică. Algoritmul YOLACT funcționează în 60 ms, YOLOv5 are un timp de rulare de 20 de ms, iar toată asocierea și raționamentele descrise funcționează în medie în 5ms. Menționăm că generarea obiectelor prin cei trei algoritmi, cât și generarea imaginii de segmentare semantică au fost paralelizate, astfel că timpul total este timpul celui mai mai costisitor algoritm.

După etapa de validare, toate obiectele identificate sunt urmărite pentru a menține identitatea lor în cadre succesive. Fiecare obiect identificat va avea ca atribute un cuboid (chenar), o mască de pixeli, și histograma claselor semantice în regiunea de interes definită de chenarul obiectului. Pe lângă aceste trăsături, se mai extrag alți descriptori, care vor fi utilizați în etapa de asociere de date. Având în vedere că se dorește păstrarea unei performanțe de timp real, pentru scorul de aparență s-au folosit următoarele trăsături, care au fost combinate într-un mod ponderat: IoU (intersection over union), cost de clasificare, cost dimensiuni, cost de varianță și medie, cost LBP (local binary pattern), cost bazat pe histograma orientării gradienților. De asemenea, pentru a descrie scorul de mișcare s-a folosit distanța euclidiana între detecții și trackuri. Algoritmul Munkres a fost folosit pentru determinarea asocierilor optime între trackuri și detecții, iar filtrarea traiectoriilor a fost implementată folosind filtrul Kalman. În Figura 11 si Figura 12 sunt ilustrate rezultatele algoritmului pe doua secvențe diferite. Algoritmul de urmărire funcționează în medie în 5 ms pe CPU.

Figura 11. Rezultatele algoritmului folosit. In poza din stânga sunt evidențiate măștile suprapuse peste imaginea de segmentare semantica, iar in poza din dreapta sunt prezentate id-urile trackurilor si chenarele fiecărui obiect.

In Figura 12 sunt ilustrate rezultatele pe o scenă mai complexă, unde sunt multe autovehicule și pietoni.

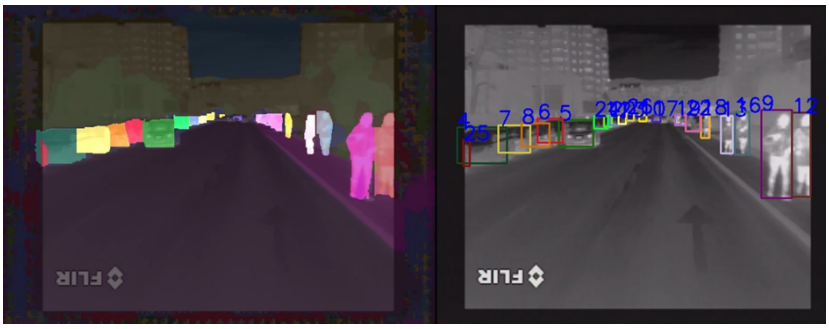

Figura 12. Rezultatele algoritmului pe o scenă mai complexă, cu multe obiecte.

3. Proiectare si implementare a algoritmilor de percepție a stării autovehiculului si a șoferului (partea 2)

3.1. Proiectarea si implementarea algoritmului de percepție a stării autovehiculului

Pentru perceperea stării autovehiculului avem două abordări independente. O abordare implică utilizarea de senzori externi, precum senzorul inerțial, care va determina accelerația sau rata de rotație (yaw rate), sau receptorul GPS care va determina viteza. Dezavantajul acestor senzori este că ei vor da aceste informații cu precizie redusă, sau, în cazul GPS, viteza va fi o viteză medie, cu o inerție a măsurătorii semnificativă.

O altă abordare este să utilizăm portul OBD, portul de diagnostic care se găsește pe aproape orice vehicul modern. Acest port poate furniza, în timp real, date despre viteză, despre turația motorului, sau despre starea unor indicatori de avarie ai autovehiculului. Pentru citirea acestor informații vom utiliza un adaptor de la OBD la interfața serială UART, apoi datele preluate din interfața serială vor fi procesate de un microcontroller Arduino, care apoi le va transmite către dispozitivele noastre mobile folosind o interfață Bluetooth de tip HC-05. Arhitectura acestui sistem este prezentată în figura 13.

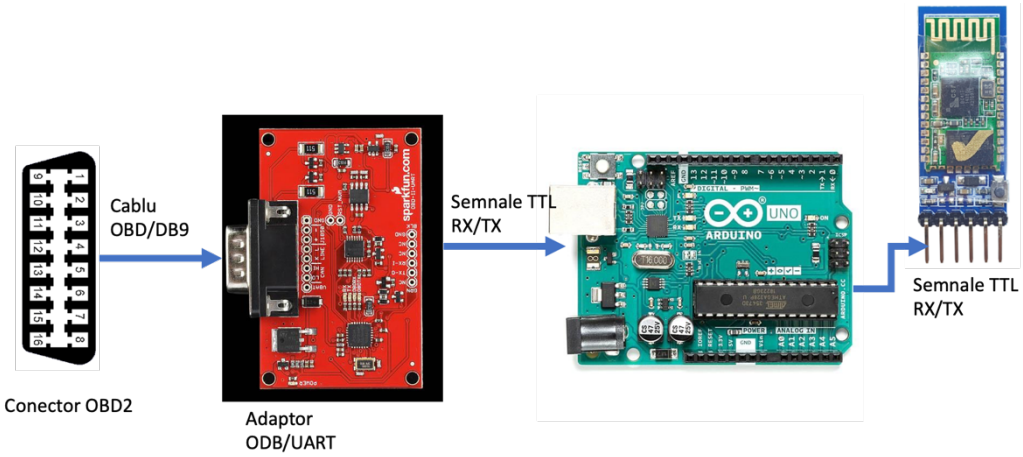

Figura 13. Citirea datelor de la autovehicul în mod wireless, prin Bluetooth.

3.2. Proiectarea si implementarea algoritmului de percepție a stării șoferului

Starea șoferului în timp ce acesta conduce autovehiculul poate varia, acesta putând trece prin diferite stări de spirit (obosit, trist, furios, etc), putând fi influențat de accesoriile pe care le are la îndemână (cel mai des acest lucru înseamnă utilizarea unui telefon mobil), sau de existența sau nu a altor pasageri în mașină. Menținerea atenției în timpul condusului a devenit o problemă destul de mare din cauza telefoanelor și a tehnologiei în general. Apelurile telefonice, scrierea de mesaje în timpul condusului, citirea notificărilor sau privitul în altă direcție prezintă un impact negativ asupra atenției șoferului. Alți factori care influențează mult starea șoferului și astfel cresc riscul de accidente sunt lipsa de somn sau conducerea sub influența alcoolului [2].

În această etapă a proiectului s-au studiat și implementat metode de recunoaștere a orientării feței șoferului (după cum se vede în Figura 14), metode de recunoașterea stării de somnolență a șoferului, cât și modele de recunoaștere a stării de ebrietate. Toate modele utilizează doar imagini captate cu o cameră video color montată în interiorul autovehiculului, orientată spre fața șoferului.

Acțiunile șoferului au fost clasificate în patru categorii, în funcție de orientarea feței, și anume: privind la drum, privind în dreapta sau în stânga, și privind înapoi, după cum se poate observa în figura 13.

Figura 14. Poziția capului șoferul - patru clase de orientare.

Pentru a determina orientarea actuală a feței, a fost utilizată o implementare BiSeNet [4] care a fost antrenată cu setul de date CelebAMask-HQ [5]. Analizând patru trăsături faciale principale, inclusiv

ochii, nasul, gura, urechile, este posibil să se estimeze orientarea feței șoferului, după cum se poate vedea în figura 15.

Figura 15. Rezultate analizei componentelor feței.

Starea de somnolenta este analizata prin prisma acțiunii ochilor șoferului și a căscatului. Primul pas constă în detecția fețelor persoanelor care sunt în cadru folosind un detector al bibliotecii dlib. În regiunea de interes în care este detectată fața se extrage un șablon de repere pentru punctele cheie ale feței. O parte din punctele cheie sunt utilizate pentru extragerea conturului ochilor. Șablonul este definit de 68 de puncte cheie pe conturul feței, conturul ochilor, al gurii și al nasului, după cum se observă în figura 16, unde persoana este cu ochii deschiși și apoi cu ochii închiși.

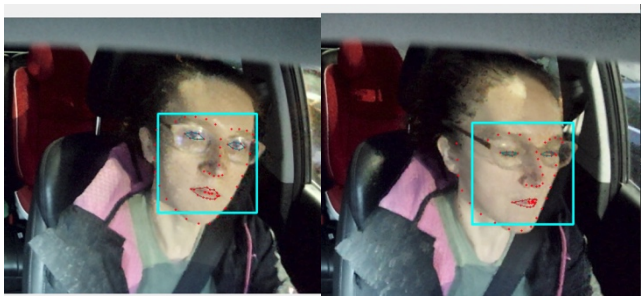

Figura 16: Șoferul clipind. În imaginea din stânga, ochii sunt deschiși, iar în dreapta ochii sunt închiși.

O parte din punctele din șablon sunt utilizate pentru a determina starea ochilor deschiși, respectiv închiși. Pentru aceasta se calculează factorul de aspect al fiecărui ochi. În urma stabilirii stării ochilor individului, în caz de ochi închiși se va porni un cronometru ce are menirea de a monitoriza durata acestei stări. Cronometrul se resetează la detectarea stării de ochi deschiși.

Starea șoferului este monitorizata continuu si dacă se colectează un minim de cinci stări cu ochii închiși se rulează un algoritm de predicție de tipul Long Short Term Memory, care are rolul de a prezice cat timp va sta persoana cu ochii închiși la următoarea acțiune de clipire. Întreg procesul este sumarizat în figura 17.

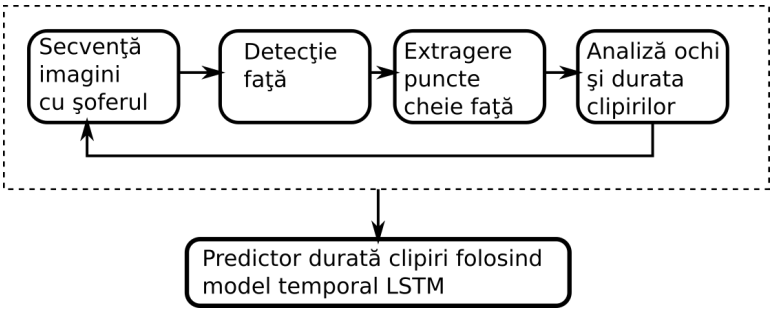

Figura 17. Procesul de identificare a somnolenței.

Pentru antrenarea modelului LSTM de predicție a duratei clipirilor s-a utilizat setul de date [6]. Modelul LSTM a fost antrenat pe mai multe secvențe care cuprind 15 participanți, dintre care patru femei și 11 bărbați, iar șapte persoane au experimentat stări de somnolență, pe când opt persoane nu au experimentat această stare. La antrenare s-a variat dimensiunea setului de date (numărul de secvențe video), numărul de epoci, cât și dimensiunea ferestrei (numărul de imagini consecutive din secvență) care se transmite ca intrare modelului. Pentru evaluarea modelului s-au măsurat eroarea pătratică medie, eroarea absolută medie, eroarea procentuală absolută medie și eroarea maximă. Aceste valori sunt prezentate în tabelul de următor.

Parametru	MSE	MAE	MAPE	ME
Rezultate arhitectura optimă	0.93	0.38	1.78	9.92

În figura 18 sunt prezentate o parte din predicțiile modelului pentru durata clipirilor atunci când persoana este în stare de somnolență. Se poate vedea astfel comparația pe coloane dintre valorile prezise și valorile adevărate pentru durata clipirii.

	FINAL predictions	FINAL sleep
0	0.150853	0.157817
1	0.150853	0.172782
2	0.155865	0.188156
3	0.149821	0.107174
4	0.151086	0.062500
..	...	...
173	0.276955	0.298476
174	0.291341	0.376604
175	0.297158	0.282720
176	0.306462	0.046842
177	0.309177	0.314164

Figura 18. Rezultatele predicției duratei clipirilor.

Recunoașterea stării de ebrietate

Persoanele aflate sub influenta alcoolului prezintă un factor principal de risc de accident rutier. În cadrul proiectului s-au studiat, implementat și comparat mai multe metode de recunoaștere a stării de ebrietate din imagini monoculare color. Fluxul de procesare este prezentat în figura 19.

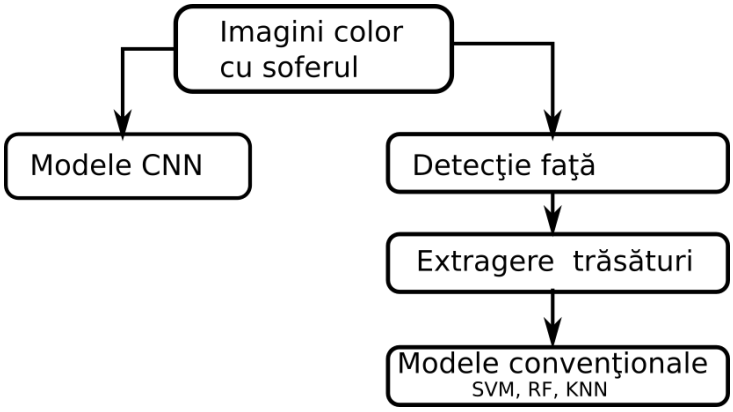

Figura 19. Abordarea pentru recunoașterea stării de ebrietate.

S-au studiat si implementat mai mulți algoritmi de recunoaștere a stării de ebrietate din imagini monoculare color. Datorită lipsei unui set de date de dimensiune mare pentru astfel de imagini, dezvoltarea s-a axat pe două direcții: modele de tip CNN (Convolutional Neural Networks) și modele convenționale de învățare bazate pe trăsături.

Pentru realizarea setului de date de antrenare și testare s-a pornit de la setul [7] care conține un număr de doar 212 imagini, dintre care 75% dintre acestea aveau persoane în stare de ebrietate. Deoarece acest set de date a fost insuficient pentru antrenarea algoritmilor de învățare profundă, s-au colectat mai multe imaginii digitale din mediul online în care au fost surprinși indivizi care se încadrează în cele două clase de interes. Pentru fiecare clasă s-au colectat imagini în care erau prezenți unul sau mai mulți indivizi, fie cu simptome de ebrietate, fie fără. S-a obținut în final un set de imagini cu 1300 de persoane în stare de ebrietate și 1010 cu persoane care nu sunt sub influența alcoolului. Acest set de date a fost augmentat pentru antrenarea modelelor de învățare profundă. Setul a fost împărțit în date de antrenare, testare și validare, în proporție de 70%-15%-15%. Modelele de tip CNN considerate au fost ResNet50, InceptionNetV3 și EfficientNetB0. Structura lor a fost adaptată pentru a clasifica imagini de cele două tipuri (ebrietate și non-ebrietate). Modelele au fost antrenate pentru cel puțin 100 de epoci. S-au realizat augmentări ale datelor (de tipul rotații, translații, oglindire orizontală). Rezultatele cele mai bune din punctul de vedere al metricii de acuratețe sunt prezentate în tabelul de mai jos:

Model	Acuratețe
ResNet50	74 %
InceptionNetV3	75 %
EfficientNetB0	72 %

Pentru metodele de învățare convenționale s-au extras trăsături din imagini. S-a utilizat modulul DLib pentru detecția feței și extragerea punctelor cheie pe față, iar din aceste puncte cheie s-au selectat zonele obrajilor, frunții, gurii și ale ochilor.

Având în vedere faptul că persoanele în stare de ebrietate prezintă roșeață în jurul obrajilor, nasului și frunții din cauza circulației sângelui afectată de consumul de alcool, din aceste zone a fost extrasă media de culoare a canalului care indică culoarea roșie. În zonele obrajilor și a frunții se calculează

media acestui canal, iar în zona nasului este extrasa valoare canalului întâi din imaginea digitală RGB din cadrul punctelor identificate. Astfel din aceste zone sunt extrase 5 caracteristici. Celelalte trăsături sunt obținute din zona ochilor și a cavității bucale. Pentru a nu extrage o mulțime de trăsături care ar putea afecta procesul de antrenare a modelului de învățare, în zona ochilor s-au prelucrat distanțe în funcție de coordonatele primite, pentru a observa starea ochilor dacă aceștia sunt închiși sau deschiși, astfel că pentru fiecare ochi s-au extras 4 puncte la extremitățile ochiului și s-a calculat distanța normalizată dintre ele, rezultând două trăsături pentru fiecare ochi. Pentru zona cavității bucale s-au considerat punctele de interes și distanța dintre ele. Toate trăsăturile descrise mai sus au fost folosite ca date de intrare pentru algoritmi de clasificare clasici bazați pe trăsături: RandomForest, SupportVector Machine, Decision Tree. Acuratețea obținută este prezentată în tabelul de mai jos.

Model	Acurateţe
Random Forest	80 %
Support Vector Machine	85 %
Decision tree	78 %

Performanța cea mai bună este obținută cu un clasificator de tip SVM. Modelul antrenat poate fi integrat în faza următoare a proiectului în modulul de răspuns la pericol.

4. Proiectarea si implementarea sistemului multidimensional de analiza a pericolului (partea 2)

Fluxul de procesare pentru identificarea gradului de pericol este arătat in Figura 20. Modulele principale sunt reprezentate de detecția si urmărirea pietonilor si a autovehiculelor, segmentarea semantică a drumului, achiziția de informații de mișcare despre vehiculul propriu, modulul de extragere de trăsături și antrenarea unor modele de clasificare bazate pe trăsături pentru recunoașterea celor trei categorii discrete de pericol.

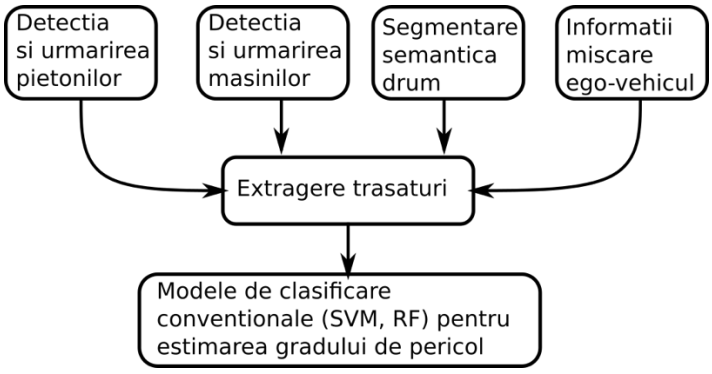

Figura 20. Abordarea pentru detecția nivelului de pericol.

Modulele de detecție si urmărire a pietonilor si a mașinilor cuprind un detector de obiecte (pietoni si mașini) bazat pe Yolo [8] care a fost antrenat si validat pe adnotările din setul de date JAAD, și o componentă de urmărire a obiectelor care îmbunătățește detecția si totodată oferă informații despre modul în care evoluează poziția obiectelor în timp în scena de trafic captată.

Pentru modulul de segmentare a drumului s-au experimentat doua abordări si anume: ERFNet [9] și Yolop [10]. Ambele sunt arhitecturi de tip encoder-decoder, cu rezultate bune pentru segmentarea semantica a drumului.

Modulul de extragere a trăsăturilor este responsabil de construirea vectorului de trăsături care cuprinde:
a) Trăsături pentru pietoni:

- Poziția pietonilor în scena care este furnizata de detectorul de obiecte sub formă de coordonate x, y, lățime, înălțime a pietonului în pixeli.
- Distanța pietonului față de mașina proprie, distanța pietonului față de drum și distanța pietonului față de celelalte mașini din trafic.
- Acțiuni ale pietonului: trece sau nu trece strada, privește spre ego-vehicul sau nu.

b) Trăsături pentru vehicul:

- Tipul de mișcare: mașina oprita, mașina se mișcă încet, accelerează, frânează, se mișca repede. Pentru setul de date JAAD aceste trăsături sunt inferate din adnotări, iar pentru setul de date construit cu modulul de achiziție al acestui proiect, aceste informații sunt date de senzori.

În cadrul componentei de clasificare s-au experimentat mai multe metode de învățare convenționale, bazate pe trăsături, cum ar fi AdaBoost, Random Forest, Arbori de Decizie, Support Vector Machines (SVM), dar și modele bazate pe rețele neuronale, precum Long Short Term Memory (LSTM). Deoarece setul de date este de dimensiune medie, modelele convenționale, care utilizează trăsături, sunt potrivite pentru a obține o acuratețe satisfăcătoare, mai bună decât a modelului bazat pe rețele neuronale.

Rezultate experimentale
S-au implementat, testat si comparat mai multe modele de clasificare a gradului de pericol. Evaluarea modelelor s-a realizat pe setul standard JAAD, dar și pe setul de imagini construit în cadrul acestui proiect. Rezultatele obținute pe setul JAAD sunt prezentate în tabelul de mai jos.

Clasificator	**Acuratete**	**Senzitivitate**	**Precizie**	**Scor F1**
Arbore de decizie	94 %	93%	95%	94%
Random Forest	97%	97%	97%	97%
AdaBoost	82%	81%	80%	81%
SVM	81%	78%	81%	80%
LSTM	77%	73%	75%	74%

Rezultatele obținute pe setul de imagini adnotat în cadrul acestui proiect sunt prezentate in tabelul de mai jos:

Clasificator	**Acuratete**	**Senzitivitate**	**Precizie**	**Scor F1**
Arbore de decizie	62%	52%	57%	54%
Random Forest	80%	89%	86%	87%
AdaBoost	78%	74%	80%	77%
SVM	67%	63%	68%	65%
LSTM	72%	69%	71%	70%

După cum se poate observa din tabelul care arata rezultatele experimentale, clasificatorii de tip Random Forest furnizează cele mai bune rezultate. Acești clasificatori agregă mai mulți clasificatori de tip arbori de decizie. Totodată, Random Forest sunt potriviți și pentru clasificarea multi-clasă, cum este

cazul de față, deoarece avem de recunoscut trei tipuri de pericol: mic, mediu, ridicat. Clasificatorii de tip LSTM au o performanta mai redusa deoarece, prin structura lor, au nevoie de mult mai multe date de antrenare decât clasificatorii convenționali, pentru a obține rezultate bune. Modelul de analiza si identificare a gradului de pericol a fost descris si in articolul [11].

5. Proiectarea si implementarea modelului de răspuns la pericol (partea 1)

Răspunsul la o situație periculoasă poate fi de mai multe tipuri, incluzând diferite moduri de avertizare, recomandarea frânării sau chiar recomandarea opririi autovehiculului dacă șoferul nu este considerat capabil să conducă în continuare. Abordarea noastră este de a avea un sistem inteligent, bazat pe rețele neuronale recurente, care să recomande cea mai bună soluție. Totuși, aceste sisteme au nevoie de date luate în situații cu diferite grade de periculozitate, iar colectarea acestor date în condiții de trafic real nu este recomandată. Pentru a trece peste această dificultate, a fost studiat simulatorul de trafic CARLA [12].

Acest simulator poate fi folosit pentru a genera date de antrenare pentru rețele artificiale cu scopul de a estima starea vehiculului propriu, pentru a determina starea viitoare a vehiculului și a altor participanți la trafic, dar și pentru a învăța manevrele necesare pentru evitarea accidentelor, el incluzând și datele generate de interacțiunea șoferului cu autovehiculul. Acest simulator este “open source” si a fost dezvoltat cu sprijinul comunității științifice, dar și cu sprijinul companiilor Intel, Nvidia, Toyota Research Institute, etc.

Simulatorul CARLA suportă diferite scenarii și condiții meteo variate: se pot face simulări în condiții de ceață, soare, etc., sau în momente diferite ale zilei. Totodată, numărul de participanți din scena de trafic poate fi configurat ușor: se pot adăuga sau exclude anumite categorii de vehicule sau chiar pietoni. CARLA oferă informații din diferite orientări sau puncte ale vehiculului propriu, dar și date senzoriale despre poziție, viteză, accelerație, oferind și acces la informații despre situațiile periculoase precum încălcări ale legilor de circulație (ex. trecerea pe culoarea roșie a semaforului, coliziuni, încălcarea marcajelor de linie continuă), după cum se poate vedea în figurile 21 și 22.

Figura 21. Captura de ecran din simulatorul Carla, într-un scenariu în care marcajul continuu este încălcat neregulamentar.

Figura 22. Captura de ecran din simulatorul Carla: coliziune cu un alt autovehicul.

Vom utiliza simulatorul Carla pentru a genera situații periculoase, generând set de date ce include fluxul de imagini video, care apoi îl putem achiziționa și noi cu camerele proprii, dar și informațiile privitoare la gradul de periculozitate identificat de simulator, și manevrele șoferului, mai ales cea de frână pentru evitarea unui pericol iminent. Aceste date vor fi interpretate prin rețele neuronale recurente, mai exact prin folosirea unei rețele de tip "LSTM" (Long Short Term Memory), cu ajutorul căreia se va genera automat recomandarea de acțiune către șofer. Structura unei astfel de abordări, în care sunt folosite imagini și date senzoriale, este ilustrată în figura 23.

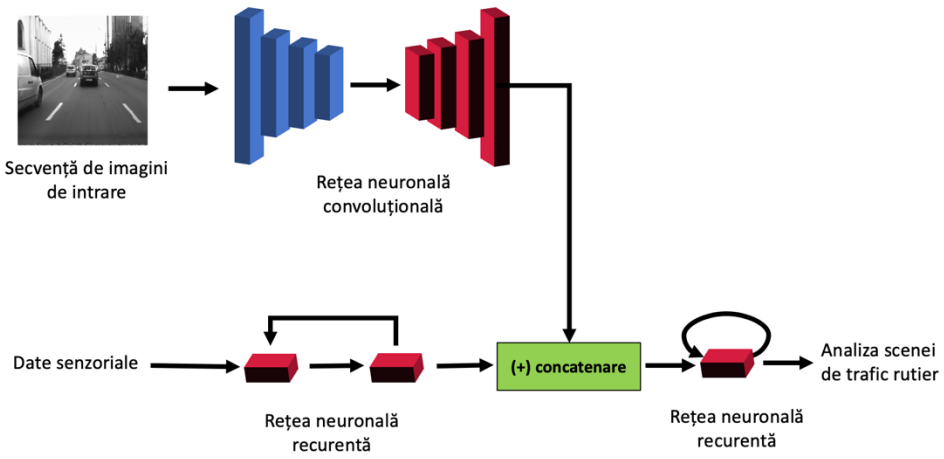

Figura 23. Structura rețelei artificiale recurente pentru analiza situațiilor și determinarea răspunsului la pericol.

6. Dezvoltarea unei aplicații demonstrator (partea 1)

Pentru realizarea aplicației demonstrator ne-am concentrat pe placa de dezvoltare nVidia Jetson Nano, care a fost echipată cu două camere video, una pentru monitorizarea traficului din fața autovehiculului, pentru detecția participanților la trafic și a pericolului posibil creat de aceștia, a doua cameră fiind pentru monitorizarea șoferului și identificarea posibilelor stări periculoase ale acestuia.

Sistemul este alimentat de la priza de brichetă a autovehiculului, și poate achiziționa și procesa date în timpul mersului.

Figura 24. Sistemul de achiziție și procesare instalat în interiorul autovehiculului.

La pornire, aplicația inițializează patru fire de execuție: un fir este responsabil pentru citirea accelerometrului, giroscopul și a direcției magnetice de la un senzor inerțial IMU, prin magistrala I2C, al doilea fir este responsabil pentru citirea vitezei autovehiculului de la un senzor GPS prin interfața serială UART, iar celelalte fire sunt dedicate achiziționării imaginilor de la cele două camere.

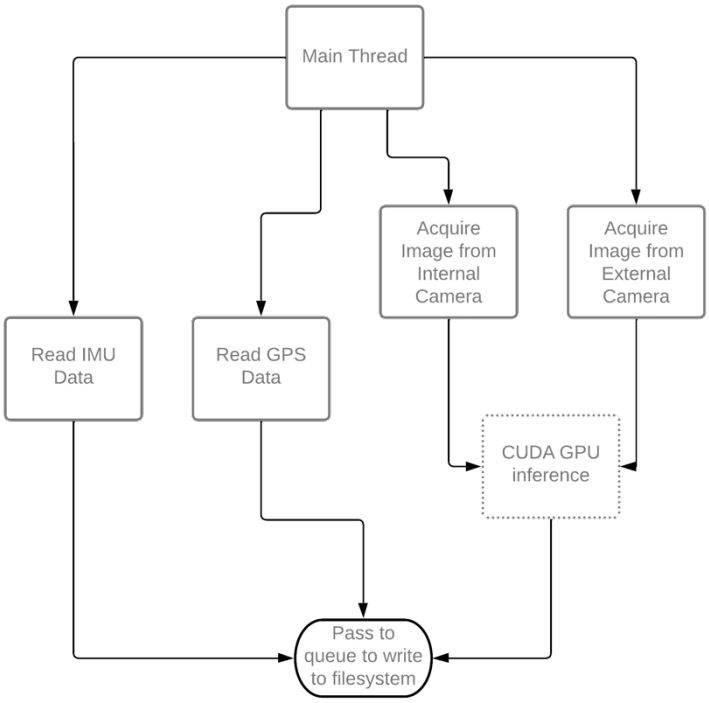

Figura 25. Aplicația demonstrator – firele de execuție.

Datele citite de la senzorul IMU ICM-20948 ajung la CPU prin două canale I2C diferite, un canal trimițând date legate de giroscop și accelerometru, iar celălalt date de la busola digitală (magnetometru). Analizând aceste valori putem identifica manevrele bruște, cum ar fi rotirea rapidă a volanului, și pot fi corelate cu imaginile interne/exterioare pentru determinarea unei surse de pericol.

Modulul GPS NEO-8M trimite mai multe mesaje prin UART, raportând totodată timpul precis sincronizat cu timpul global UTC. Diferitele propoziții NMEA sunt analizate, iar viteza curentă în km/h este citită din mesajul GPVTG.

Imaginile de la camera externă și cea internă sunt procesate în GPU pentru detectarea obiectelor. Imaginile de la camera cu fața spre exterior sunt procesate printr-o rețea SSD MobileNet V2 care s-a optimizat cu mediul de lucru TensorRT pentru a rula mai rapid pe sistemul Nano suportat de CUDA. Această rețea acceptă 91 de clase de obiecte COCO. Imaginile de la camera internă sunt deduse printr-o rețea FaceNet care a fost optimizată în același mod cu TensorRT ca și cea anterioară, responsabilă cu detectarea feței șoferului.

7. Diseminare rezultate (partea 2)

În cadrul acestei etape au fost realizate trei publicații, un articol de jurnal (ISI) și două articole de conferință.

Articol de jurnal

1. R. Itu and R. Danescu, "Part-Based Obstacle Detection Using a Multiple Output Neural Network," Sensors, vol. 22, no. 12, p. 4312, Jun. 2022, doi: 10.3390/s22124312 [ISI – Q1, zona roșie]

Articole de conferință

1. R. D. Brehar, R. O. Băbuţ, A. Füzes, R. Dănescu, "Outdoor Traffic Scene Risk Estimation in the Context of Autonomous Driving", IEEE Intelligent Computer Communication and Processing (ICCP 2022), Cluj-Napoca, Romania.

2. M. Mureșan, R. Schlanger, R. Danescu, S. Nedevschi, "Real-Time Obstacle Detection using a Pillar-based Representation and a Parallel Architecture on the GPU from LiDAR Measurements", 2023 International Conference on Computer Vision Theory and Applications (VISAPP), în review.

Bibliografie

[1] AAA newsroom, "Effectiveness of Driver Monitoring Systems", 2021, disponibil online https://newsroom.aaa.com/wp-content/uploads/2022/01/Driver-Monitoring-Full-Report-February-2022-Final.pdf

[2] Amir Rasouli, Iuliia Kotseruba, and John K Tsotsos. Are they going to cross? A benchmark dataset and baseline for pedestrian crosswalk behavior. In ICCVW, pages 206–213, 2017.

[3] Amir Rasouli, Iuliia Kotseruba, and John K Tsotsos. It's not all about size: On the role of data properties in pedestrian detection. In ECCVW, 2018

[4] Lee, Cheng-Han and Liu, Ziwei and Wu, Lingyun and Luo, Ping – MaskGAN: Towards Diverse and Interactive Facial Image Manipulation – https://arxiv.org/abs/1907.11922

[5] Yu, Changqian and Wang, Jingbo and Peng, Chao and Gao, Changxin and Yu, Gang and Sang, Nong – BiSeNet: Bilateral Segmentation Network for Real-time Semantic Segmentation – https://arxiv.org/abs/1808.00897

[6] Drowsiness dataset.[Online].Available:http://vlm1.uta.edu/~athitsos/projects/drowsiness/

[7] C. Willoughby, I. Banatoski, P. Roberts, and E. Agu, “Drunkselfie: Intoxication detection from smartphone facial images,” in 43rd IEEE Annual Computer Software and Applications Conference, COMPSAC 2019, Milwaukee, WI, USA, July 15-19, 2019, Volume 2, IEEE, 2019, pp. 496–501. [Online]. Available: https://doi.org/10.1109/COMPSAC.2019.10255

[8] Zhanchao Huang and Jianlin Wang. DC-SPP-YOLO: dense connection and spatial pyramid pooling based YOLO for object detection. CoRR, abs/1903.08589, 2019.

[9] Eduardo Romera, José M. Álvarez, Luis M. Bergasa, and Roberto Arroyo. Efficient convnet for real-time semantic segmentation. In 2017 IEEE Intelligent Vehicles Symposium (IV), pages 1789–1794, 2017.

[10] Dong Wu, Manwen Liao, Weitian Zhang, and Xinggang Wang. Yolop: You only look once for panoptic driving perception, 2021.

[11] R. D. Brehar, R. O. Băbuţ, A. Füzes, R. Dănescu, “Outdoor Traffic Scene Risk Estimation in the Context of Autonomous Driving”, IEEE Intelligent Computer Communication and Processing (ICCP 2022), Cluj-Napoca, Romania.

[12] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, Vladlen Koltun, CARLA: An Open Urban Driving Simulator, 2017, arXiv: 1711.03938

Director proiect,

Prof. Dr. Ing. Radu Dănescu