UNIVERSITATEA BABEŞ-BOLYAI CLUJ-NAPOCA

FACULTATEA DE MATEMATICĂ ŞI INFORMATICĂ

SPECIALIZAREA INFROMATICĂ

LUCRARE DE LICENŢĂ
Aplicații ale inteligenței artificiale și computer vision în domeniul asistenței auto

Conducător ştiinţific
Bogdan MURSA, Doctorand
Dr. Anca ANDREICA, Profesor Universitar

Absolvent
Marius Vasile CHIRILĂ

2019

Abstract

We live in the speed century where the majority of road accidents happen because of the lack of attention generated by the rush to arrive on time at destination. Fortunately, the progress of technology helps us out solving this kind of errors and making driving safer. One of the branches of this technological industry is based on image recognition, and because of it we can achieve extraordinary things. In this thesis I will present an application which uses image processing and image recognition concepts to detect and fit the vehicle between the lines and it achieves the detection of nearby vehicles by using only the camera located in from of the car. In this thesis we want to present the possibilities offered by image recognition in driver assistance systems, that in chapter two theoretical aspects in this direction will be presented, related to image recognition and neural networks, after that in chapter three some practical methods of usage will be presented, and at the end, in chapter four which is meant for conclusions, some improvements for the model provided in the previous chapter will be presented. This work is the result of my own activity. I have neither given nor received unauthorized assistance on this work.

Cuprins

1 Introducere 4

2 Aspecte teoretice 5

2.1 Descrierea generală a problemei 5

2.2 Prelucrarea imaginilor 5

2.2.1 Calibrarea imaginii 5

2.2.2 Proiecția în perspectivă a unei imagini 10

2.2.3 Spațiul de culoare HSL 11

2.2.4 Operatorul Sobel 13

2.2.5 Gradientul imaginii 14

2.3 Rețele neuronale artificiale 16

2.3.1 Noțiuni introductive 16

2.3.2 Detectarea de obiecte 19

2.3.3 Rețele neuronale convoluționale 21

3 Parte experimentlă 24

3.1 Detectarea și încadrarea în bandă 24

3.1.1 Calibrarea imaginii 25

3.1.2 Schimbarea perspectivei 26

3.1.3 Filtrarea culorilor în imagine 27

3.1.4 Măsurarea curburii și calcularea poziției mașinii 29

3.1.5 Proiectarea benzii în perspectiva normală 30

3.2 Detectarea mașinilor 31

3.2.1 Crearea modelului 32

3.2.2 Obținerea zonei de interes și calcularea hărții de predicții 33

3.2.3 Extragerea chenarelor și crearea hărții de „temperatură” 34

3.2.4 Gruparea chenarelor 35

3.3 Arhitectura și utilizarea aplicației 36

4 Concluzie 37

5 Referințe 38

1 Introducere

Trăim în secolul vitezei, unde necesitatea deplasării a crescut în mod ridicat. Cu toții ne deplasăm, fie la muncă, fie la scoală, fie la un prieten drag care are nevoie de noi, însă această deplasare trebuie să fie rapidă și eficientă. Majoritatea persoanelor circulă cu vehicule, fie ele proprii sau mijloace de transport în comun. Datorită creșterii ridicate al populației, numărul de vehicule a crescut în mod masiv, mai ales în orașe, unde se crează fluxuri continue de mașini.

Din cazua dorinței de ajunge într-un timp cât mai scurt la destinația propusă, uneori, apar erori în judecata noastră, care pot produce accidente, din păcate uneori chiar fatale. Din fericire, tehnologia vine în ajutorul nostru, oferindu-ne un suport în această direcție. Acest suport este reprezentat de către anumite sisteme care au scopul de a reduce, uneori chiar de a rezolva, erorile produse de către om în timpul condusului, ele find așa-numite sisteme avansate de asistență auto.

Sistemele avansate de asistență auto (SAAA) sunt sisteme care, dupa denumire, ajută conducătorul auto in timpul șofatului. Astfel, ele sunt create pentru a reduce riscul accidentelor cauzate de neatenție.

Cele mai multe accidente rutiere apar din cauza erorilor umane. Sistemele avansate de asistență a conducătorilor auto sunt sisteme dezvoltate pentru a automatiza, adapta și îmbunătăți sistemele autovehiculelor pentru a oferi o siguranță și conducere mai bună. Sistemul automatizat furnizat de SAAA vehiculului se dovedește a reduce mortalitatea rutieră prin minimizarea erorii umane. Elementele de siguranță sunt concepute pentru a evita coliziunile și accidentele oferind tehnologii care avertizează șoferul cu privire la posibile probleme sau pentru a evita coliziunile prin punerea în aplicare a măsurilor de siguranță și preluarea controlului asupra vehiculului.

Unele dintre aceste sisteme se bazează pe recunoașterea în imagini, o direcție înspre care această lucrare va fi orientată, iar cu ajutorul ei putem realiza lucruri remarcabile, după cum afirmă Elon Musk (deținătorul companiei Tesla): „Putem fi superoameni utilizând pur și simplu camere”.

Astfel, în această lucrare se dorește prezentarea posibilităților oferite de către recunoașterea în imagini în domeniul sistemelor de asistență auto, încât în capitolul doi vor fi prezentate aspecte teoretice în această direcție, legate de procesarea de imagini și chiar despre rețele neuronale, după care în capitolul trei vor fi prezentate cateva metode practice de utilizare al acestora, iar la final în capitolul patru destinat concluzilor, se vor prezenta îmbunătățiri ulterior posibile modelului prezentat anterior în capitolul trei.

2 Aspecte teoretice

2.1 Descrierea generală a problemei

În esență, se dorește prezentarea unui sistem bazat pe recunoașterea în imagini, care să reducă din greșelile produse în timpul șofatului. Privind în perspectivă asupra problemelor care pot apărea din cauza neatenției în timpul condusului, putem spune că printre ele se numără neîncadrarea corespunzătoare în banda de circulație și coliziunea cu alte vehicule datorată faptului că acesta nu a fost observat. Deci, se dorește prezentarea unui sistem care să ofere suport în aceste direcții.

2.2 Prelucrarea imaginilor

Prelucrarea imaginilor este o metodă de conversie a unei imagini în formă digitală și efectuarea unor operații pe aceasta, pentru a obține o imagine îmbunătățită sau pentru a extrage câteva informații utile din aceasta. Este un tip de dispensare a semnalului în care datele de intrare sunt reprezentate de către imagine, de exemplu cadre ale unei filmări sau fotografii, iar datele de ieșire pot fi o nouă imagine sau caracteristica acelei imagini. De obicei, sistemul de procesare a imaginilor include tratarea imaginilor ca semnale bidimensionale în timp ce aplică metode de procesare a semnalelor deja setate.

Prelucrarea imaginiilor se numără printre tehnologiile în plină expansiune de astăzi, cu aplicațiile sale în diverse aspecte, ea formează aria centrală de cercetare în domeniile ingineriei și informaticii.

2.2.1 Calibrarea imaginii

Calibrarea camerei în contextul vizualizării tridimensionale este procesul de determinare a caracteristicilor geometrice și optice ale camerei interne (parametrii intrincesi), poziția în planul 3D și orientarea cadrului camerei față de un anumit sistem de coordinate (parametrii extrinseci) [8]. În multe cazuri, performanța generală a sistemului de vizionare a mașinii depinde foarte mult de precizia calibrării aparatului foto. Mai multe metode de calibrare a camerei geometrice sunt prezentate în literatură. Abordarea clasică [7] care provine din domeniul fotogrametriei rezolvă problema minimizând o funcție de eroare neliniară. Datorită încetinirii și a ponderii computaționale a acestei tehnici, au fost de asemenea sugerate soluții de formă închisă (de exemplu, [8], [1], [5]).

Există, de asemenea, proceduri de calibrare în care se utilizează atât minimizarea neliniară, cât și o soluție de formă închisă (de exemplu, [5], [10]). În aceste metode cu două etape, valorile inițiale ale parametrilor sunt calculate liniar, iar valorile finale sunt obținute prin minimizarea neliniară. Metodele în care modelul camerei se bazează pe parametri fizici, cum ar fi lungimea focală și punctul principal, se numesc metode explicite. În majoritatea cazurilor, valorile acestor

parametri sunt în sine inutili, deoarece este necesară numai relația dintre coordonatele de referință 3D și coordonatele imaginii 2D. În calibrarea implicită a camerei, parametrii fizici sunt înlocuiți de un set de parametri impliciți non-fizici care sunt utilizați pentru a realiza interpolarea între unele puncte de legătură cunoscute (de exemplu [9]).

Lucrarea lui Janne Heikkila and Olli Silven [11] prezintă o procedură de calibrare în patru etape, care este o extensie a procedurii în două etape. În aceste etape se vor lua în considerare numai caracteristicile circulare, însă se pot face analize similar pentru forme ale caracteristicilor arbitrare. Există și alte surse de eroare în extragerea caracteristicilor, cum ar fi modificările iluminării, dar acestea sunt discutate în [4]. Această lucrare oferă o rezolvare pentru problema corecției imaginii. Corecția imaginii se realizează utilizând un nou model implicit care interpolează punctele de imagine corecte pe baza parametrilor fizici ai camerei. În continuare se va face referire la două dintre aceste etape.

2.2.1.1 Calibrarea explicită a camerei

Parametrii camerei fizice sunt împărțiți în parametri extrinseci și intrinseci. Parametri extrinseci sunt necesari pentru a transforma coordonatele obiectului intr-un set de coordinate centrate pe cadru. Modelul cu camera cu miniatură se bazează pe principiul colinearității, în care fiecare punct din spațiul obiectului este proiectat de o linie dreaptă prin centrul de proiecție în planul imaginii.

Originea sistemului de coordonate al camerei este în centrul de proiecție din locația (X_0, Y_0, Z_0) în raport cu sistemul de coordonate al obiectului, iar axa Z al cadrului camerei este perpendiculară pe planul imaginii. Rotația este reprezentată folosind unghiurile Euler ω, φ și κ care definesc o secvență de trei rotații elementare în jurul axelor x, y și z. Rotițiile sunt realizate în sensul acelor de ceasornic, mai întâi în jurul axei x, apoi axa y care este deja rotită și în final în jurul axei z care este rotită de două ori în etapele anterioare. Pentru a exprima un punct obiect arbitrar P la locația (X_i, Y_i, Z_i) în coordonatele imaginii, trebuie mai întâi să îl transformăm în coordonatele camerei (x_i, y_i, z_i). Această transform are constă dintr-o traducere și o rotație și se poate realiza folosind următoarea ecuație de matrice:

$$\begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \\ Z_i \end{bmatrix} + \begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix}$$

unde

$m_{11} = \cos\varphi \cos\kappa$ $m_{21} = \cos\varphi \sin\kappa$ $m_{31} = -\sin\varphi$

$m_{12} = \sin\omega \sin\varphi \cos\kappa - \cos\omega \sin\kappa$ $m_{22} = \sin\omega \sin\varphi \sin\kappa - \cos\omega \cos\kappa$ $m_{32} = \sin\omega \cos\varphi$

$m_{13} = \cos\omega \sin\varphi \cos\kappa + \sin\omega \sin\kappa$ $m_{23} = \cos\omega \sin\varphi \sin\omega - \sin\omega \cos\kappa$ $m_{33} = \cos\omega \cos\varphi$

Parametrii intrinseci ai camerei includ, de obicei, distanța focală efectivă *f*, factorul de scalare s_u și centrul de imagine (u_0, v_0) numit și punctul principal. Aici, originea sistemului de coordonate al imaginii se află în colțul din stânga sus al matricii. Unitatea de măsură a coordonatelor imaginilor este pixelul și, prin urmare, coeficienții D_u și D_v sunt necesari pentru a schimba unitățile metrice în pixeli. Acești coeficienți pot fi obținuți în mod obișnuit din datele

aparatului foto și ale cadrelor grafice. De fapt, valorile lor precise nu sunt necesare, pentru că ele sunt dependente liniar de lungimea focală *f* și factorul de scalare s_x. Prin folosirea modelului „pinhole", proiecția punctului (x_i, y_i, z_i) la planul imaginii este:

$$\begin{bmatrix}\tilde{u}_i\\ \tilde{v}_i\end{bmatrix} = \frac{f}{z_i}\begin{bmatrix}x_i\\ y_i\end{bmatrix}$$

Coordonatele corespondente în pixeli ale imaginii sunt obținute din proiecția ($\tilde{u}_i$, $\tilde{v}_i$) aplicând următoarea transformare:

$$\begin{bmatrix}u_i'\\ v_i'\end{bmatrix} = \begin{bmatrix}D_u s_u \tilde{u}_i\\ D_v \tilde{v}_i\end{bmatrix} + \begin{bmatrix}u_0\\ v_0\end{bmatrix}$$

Modelul „pinhole" este doar o aproximare a unei proiecții reale a camerei. El fiind un model simplu care exprimă matematic relația dintre coordonatele obiectului si imaginii. Din păcate, nu oferă o precizie mare și trebuie folosit un model mai amplu. De obicei, acest model este extins pentru realizarea corecțiilor. Cea mai frecventă corecție utilizată distorsiunea radială a lentilei care determină deplasarea radială a punctului real al imaginii înplajjnul imaginii [7]. Ea poate fi aproximată utilizând urmatoarea expresie:

$$\begin{bmatrix}\delta u_i^{(r)}\\ \delta v_i^{(r)}\end{bmatrix} = \begin{bmatrix}\tilde{u}_i(k_1 r_i^2 + k_2 r_i^4 + \cdots)\\ \tilde{v}_i(k_1 r_i^2 + k_2 r_i^4 + \cdots)\end{bmatrix}$$

unde k_1, k_2, … sunt coeficienții de distorsiune radială și $r_i = \sqrt{\tilde{u}_i^2 + \tilde{v}_i^2}$. În multe cazuri, doi coeficienți sunt suficienți pentru compensa distorsiunea. Este important de observant faptul că centrele curburii lentilelor nu sunt neapărat coliniare. Acest lucru anticipează o altă distorsiune, „decentering distorsion" care conține atât o componentă radial, cât și una tangențială. Expresia pentru distorsiunea tangențială este adesea scrisă în următoarea formă:

$$\begin{bmatrix}\delta u_i^{(t)}\\ \delta v_i^{(t)}\end{bmatrix} = \begin{bmatrix}2p_1\tilde{u}_i\tilde{v}_i + p_2(r_i^2 + 2\tilde{u}_i^2)\\ p_1(r_i^2 + 2\tilde{v}_i^2 + 2p_2\tilde{u}_i\tilde{v}_i)\end{bmatrix}$$

unde *p1* și *p2* sunt coeficienții distorsiunii tangențiale. De asemenea, alte tipuri de distorsiuni au fost propuse in literatură, de exemplu Melen [5] aduce în discuție termenul de corecție pentru distorsiuni liniare. El este relevant atunci când axele imaginii nu sunt ortogonale. În cele mai multe situații, eroarea este atât de mică, încât componenta de denaturare este nesemnificativă. O altă distorsiune este distorsiunea prismei subțire. Aceasta apare atunci când lentilele sunt imperfecte din fabricație sau există erori de măsură la asamblarea camerei. Acest tip de distorsiune poate fi simulată prin adăugarea unei prisme subțire la sistemul optic. Adăugarea provoacă cantități suplimentare de distorsiuni radiale și tangențiale [2], [10].

Poate fi obținut un model care oferă o calibrare mai precisă prin combinarea modelului „pinhole" cu corecția pentru distorsiunile radiale si tangențiale:

$$\begin{bmatrix}u_i\\ v_i\end{bmatrix} = \begin{bmatrix}D_u s_u(\tilde{u}_u + \delta u_i^{(r)} + \delta u_i^{(t)})\\ D_v(\tilde{v}_i + \delta v_i^{(r)} + \delta v_i^{(t)})\end{bmatrix} + \begin{bmatrix}u_0\\ v_0\end{bmatrix} \qquad (1)$$

În acest model, parametrii intrinseci (f, s_u, u_0, v_0) este înmulțit cu coeficienții de distorsiune k_1, .. , k_n, p_1 și p_2. Acești parametrii sunt cunoscuți ca parametrii fizici ai camerei, deoarece au un anumit sens fizic. În general, scopul procedurii de calibrare a camerei este de a determina valorile optime pentru acești parametri pe baza observațiilor din planul real.

2.2.1.2 Corecția proiecției asimetrice

Proiecția în perspectivă nu este o transformare care păstrează forma. Numai liniile sunt mapate ca linii în planul imaginii. Obiectele bidimensionale și tridimensionale cu o suprafață de proiecție diferită de zero sunt distorsionate dacă nu sunt coplanare cu planul imaginii. Acest lucru este valabil pentru forme arbitrare, în continuare se va discuta doar despre cerc.

Punctele centrale ale cercurilor sunt localizate în imagini cu o foarte bună precizie (până la subpixel), însă distorsiunea cauzată de proiecția in perspectivă nu este de obicei luată in considerare. Proiecția în perspectivă distorsionează forma trăsăturilor circulare în planul imaginii în funcție de unghiul și deplasarea între suprafața obiectului și planul imaginii. Proiecțiile rămân circulare atunci când suprafața de proiectare si planul imaginii sunt paralele.

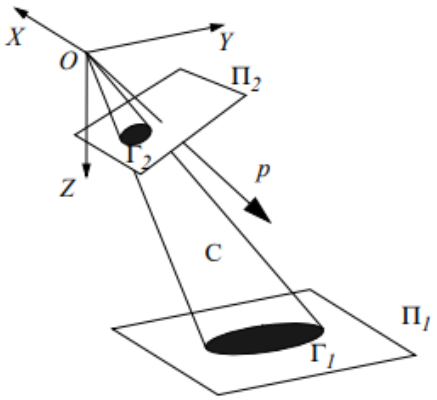

Figura 1. Proiectarea în perspectivă a unui cerc

2.2.1.3 Corecția imaginii

Modelul camerei oferit în ecuația (1) exprimă proiecția punctelor 3D pe planul imaginii. Cu toate acestea, nu oferă o soluție direct la problema „back-projection”, din care dorim să recuperăm viziunea reală din coordonatele imaginii. Dacă sunt luate în considerare atât componentele de distorsiune radială, cât și tangențială se observă că nu există o soluție analitică pentru o asociere inversă. De exemplu, doi coeficienți pentru distorsiunea radial determină modelul camerei in ecuația (1) pentru a devein un polinom de ordinul cinci:

$$u_i = D_u s_u\left(k_2\tilde{u}_i^5 + 2k_2\tilde{u}_i^3\tilde{v}_i^2 + k_2\tilde{u}_i\tilde{v}_i^4 + k_1\tilde{u}_i^3 + k_1\tilde{u}_i\tilde{v}_i^2 + 3p_2\tilde{u}_i^2 + 2p_1\tilde{u}_i\tilde{v}_i + p_2\tilde{v}_i^2 + \tilde{u}_i\right) + u_0 \quad (2)$$

$$v_i = D_v\left(k_2\tilde{u}_i^4\tilde{v}_i + 2k_2\tilde{u}_i^2\tilde{v}_i^3 + k_2\tilde{v}_i^5 + k_1\tilde{u}_i^2\tilde{v}_i + k_1\tilde{v}_i^3 + p_1\tilde{u}_i^2 + 2p_2\tilde{u}_i\tilde{v}_i + 3p_1\tilde{v}_i^2 + \tilde{v}_i\right) + v_0$$

Putem deduce din ecuația (2) că este necesară o cautare neliniară pentru a se recupera $(\tilde{u}_i\ ,\ \tilde{v}_i)$ din $(u_i\ ,\ v_i)$. O alternativă este aproximarea asocierii inverse. Doar câteva soluții pot fi găsite în literatură legate de „back-projection", desi problema este evidentă în multe aplicații. Melen [5] a folosit o abordare iterativă pentru a estima coordonatele imaginii nedistorsionate. El a propus următorul proces de două iterații:

$$q_i' = q_i'' - \delta(q_i'' - \delta(q_i''))$$

unde vectorii q_i'' și q_i' conțin coordonatele imaginii distorsionate și corectate.

O metodă propusă de Wei și Ma [9], în care se utilizează două planuri, rezolvă problema „back-projection" determinând un set de parametrii impliciți pentru a compensa distorsiunea. Datorită numărului mare de parametrii necunoscuți, această tehnică necesită o atenție sporită pentru a oferi o acuratețe ridicată. Cu toate acestea, dacă știm parametrii fizici ai camerei pe baza calibrării explicite, este posibilă rezolvarea parametrilor necunoscuți prin generarea unei rețele dense de puncte $(\tilde{u}_i\ ,\ \tilde{v}_i)$ și calculând coordonatele corespondente imaginii distorsionate $(u_i\ ,\ v_i)$ prin utilizarea modelului din ecuația (1). Pe baza modelului implicit al camerei propuse de Wei și Ma [9] putem exprima asocierea de la $(u_i\ ,\ v_i)$ la $(\tilde{u}_i\ ,\ \tilde{v}_i)$ după cum urmează:

$$\tilde{u}_i = \frac{\sum_{0\le j+k\le N} a_{jk}^{(1)} u_i^j v_i^k}{\sum_{0\le j+k\le N} a_{jk}^{(3)} u_i^j v_i^k} \qquad \tilde{v}_i = \frac{\sum_{0\le j+k\le N} a_{jk}^{(2)} u_i^j v_i^k}{\sum_{0\le j+k\le N} a_{jk}^{(3)} u_i^j v_i^k} \quad (3)$$

Este de asteptat faptul că există parametri redundanți care pot fi eliminați. După mai multe teste, s-a constatat că urmatoarea expresie a compensate distorsiunile astfel încât eroarea reziduală maximă ajunge mai mica de 0,01 unități pixel, chiar și cu o cantitate de distorsiune mare:

$$\begin{bmatrix}\tilde{u}_i \\ \tilde{v}_i\end{bmatrix} = \frac{1}{G}\begin{bmatrix}\tilde{u}_i' + \tilde{u}_i'(a_1 r_i^2 + a_2 r_i^4) + 2a_3\tilde{u}_i'\tilde{v}_i' + a_4(r_i^2 + 2\tilde{u}_i^2) \\ \tilde{v}_i' + \tilde{v}_i'(a_1 r_i^2 + a_2 r_i^2) + a_3(r_i^2 + 2\tilde{v}_i^2) + 2a_4\tilde{u}_i'\tilde{v}_i')\end{bmatrix}$$

și

$$G = (a_5 r_i^2 + a_6\tilde{u}_i' + a_7\tilde{v}_i' + a_8)r_i^2 + 1$$

unde $\tilde{u}_i' = (u_i - u_0)/(D_u s_u) \qquad \tilde{v}_i' = (v_i - v_0)D_v \qquad r_i = \sqrt{\tilde{u}_i'^{\,2} + \tilde{v}_i'^{\,2}}$.

Dacă comparăm acest model invers implicit cu modelul cu modelul camerei din ecuația (6), observăm că și modelul invers are componente care seamănă cu distorsiuni radiale și tangențiale. Parametrii similari de distorsiune k_1, k_2, p_1 și p_2 sunt coeficienții a_1, .,. , a_4.

Modelul de mai sus conține doar opt parametri necunoscuți, în loc de 63 de parametri care au fost în modelul original de ordinal cinci din ecuația (3). Dacă se folosește acest model pentru rezolvarea „back-projection", va necesita mai putin calcul decât abordarea iterative sugerată de Melen, oferind, de asemenea, rezultate mai precise [11].

2.2.2 Proiecția în perspectivă a unei imagini

Perspectiva în arta grafică este o reprezentare aproximativă a ceea ce observăm cu ochiul liber pe o suprafață,în general plană. Două caracteristici importante ale perspectivei sunt următoarele: obiectele apar mai mici pe măsură ce distanța față de observator crește și acestea sunt supuse unei reduceri de precizie, ceea ce înseamnă că dimensiunile obiectului de-a lungul liniei vizuale apar mai aproape decât ar trebui.

Proiecția obiectelor tridimensionale reprezintă orice metodă prin care un obiect tridimensional este transpus pe un plan bidimensional. Dat fiind faptul că cele mai multe metode actuale de afișare a datelor grafice se bazează pe medii plane, utilizarea acesui tip de proiecție este larg răspândită.

Proiecția în perspectivă este o proiecție liniară în care obiectele tridimensionale sunt proiectate pe un plan de imagine. Acest lucru are ca efect faptul că obiectele îndepărtate apar mai mici decât obiectele mai apropiate. De asemenea, înseamnă că liniile care sunt paralele în realitate, par a se intersecta în imaginea proiectata, de exemplu linile unei căi ferate într-o imagine proiectată în perspectivă, par că se vor intersecta într-un punct, numit punct de disparitie. Lentilele fotografice și ochiul uman lucrează în același mod, prin urmare proiecția de perspectivă pare a fi cea mai realistă [29].

Proiecția în perspectivă este de obicei clasificată în perpectiva dintr-un punct, din două puncte și din trei puncte, în funcție de orientarea planului de proiecție spre axele obiectului descris. Metodele de proiectare grafică se bazează pe dualitatea dintre linii și puncte, prin care două linii drepte determină un punct în timp ce două puncte determină o linie dreaptă. Proiecția ortogonală a punctului ocular pe planul imaginii este numită punctul principal de dispariție.

Două relevante ale unei linii sunt:

- Punctul de intesecție cu planul imaginii
- Punctul de dispariție

Punctul principal de dispariție este punctul de dispariție al tuturor liniilor orizontale perpendiculare pe planul imaginii. Punctele de dispariție ale tuturor liniilor orizontale se află pe linia orizontului. Dacă, adesea, planul imaginii este vertical, toate liniile verticale sunt desenate vertical și nu au un punct de dispariție finit pe planul imaginii. Diferite metode grafice pot fi utile pentru proiectarea formelor geometrice.

De exemplu, liniile trase de ochi la 45 ° pe planul imaginii intersectează aceasta din urmă de-a lungul unui cerc a cărui rază este distanța de la punct la ochi, urmărind astfel cercul care ajută la construirea tuturor punctelor de dispariție al linilor cu unghiul de 45 ° (particular, intersecția acelui cerc cu linia orizontului este formată din două puncte de distanță). Acestea sunt utile pentru desenarea tablelor de șah, care, la rândul lor, servesc la obținerea formei obiectului.

Deși proiecția ortografică ignoră perspectiva pentru a permite măsurători exacte, proiecția perspectivă arată obiecte îndepărtate ca fiind mai mici pentru a oferi un realism suplimentar.

2.2.3 Spațiul de culoare HSL

Numele spațiului de culoare HSL este prescurtarea de la „Hue, Saturation and Lightness” (nuanță, saturație și luminozitate). Acest concept a fost introdus la începutul aniilor 70’ din necesitatea de selectare a culorilor, iar in prezent este utilizat in industrie la nivel global. Sindicatul WWW (World Wide Web) a incorporat paletele de culori HSL și HSLA în „W3C Recommendation CSS Color Module Level 3” în anul 2011, ei afirmând faptul că alegerea unei culori este mai intuitivă în spațiul de culori HLS decât în spațiul RGB. "De asemenea, este mai ușor să creați seturi de culori potrivite, menținând aceași nuanță și variind luminozitatea / întunericul și saturația"[12].

Este utilizat masiv în industrie, de exemplu în cazul aplicațiilor web și al designului web, HSL este principalul spațiu de culoare utilizat pentru alegerea culorilor. Toate sistemele majore de management al conținutului și scheletele pentru aplicații front-end, cum ar fi Drupal sau Wordpress, utilizează selectorul de culoare HSL în ultima lui versiune [14][15].

HSL este un spațiu de culoare cilindric, cu două dimensiuni liniare (saturația și luminozitatea) și o dimensiune unghiulară (nuanța). Nuanța este un unghi pe cercul de culori care variază de la 0° la 360°. Saturația poate fi considerată intensitatea percepută sau „vivacitatea” culorii [13]. O saturație de 100% are ca rezultat o culoare puternică, în timp ce o saturație de 0% are ca rezultat o nuanță de gri. Luminozitatea, după nume, este caracteristica care determină cât de deschisă este o imagine. Orice imagine cu 100% luminozitate este albă și orice imagine cu 0% luminozitate este neagră.

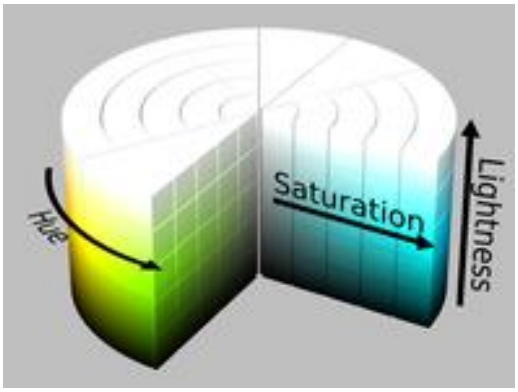

Figura 2. Reprezentarea cilindrică a spațiului de culori HSL

De asemenea, HSL este un spațiu de culoare utilizat adesea în analiza imaginilor pentru detectarea caracteristicilor sau segmentări. Aplicațiile acestor tip de instrumente includ detectarea și / sau obiectelor (de exemplu: detectarea pietoniilor pe stradă, recunoașterea gesturilor feței, recunoașterea textului, etc.).

În cea mai mare parte, algoritmii utilizați în „computer vision” pentru imaginile color sunt doar extensii directe al algoritmilor concepuți pentru nuanțe de gri (de exemplu: algoritmul de detectare al marginilor). Altfel spus, fiecare componentă de culoare este trecută separat prin același algoritm. Prin urmare, este important ca particularitățiile de interes ale imaginii să poată fi distinse. Cele trei componente (roșu, verde, albastru) care la un loc formează culoarea

obiectului, în combinație cu cantitatea de lumină expusă asupra lui nu oferă caracteristici clare asupra obiectului. Astfel, caracteristicile oferite de nuanță, saturație și lumină sunt mult mai relevante.

Chiar dacă acest spațiu de culoare servește suficient de bine pentru a oferi o singură culoare, ignoră o mare parte din complexitatea aspectului culorii. În esență, ele compensează relevanța percepută pentru viteza de calcul încă din vremea stațiilor grafice alea anilor 70', când modelele mai sophisticate ar fi fost scumpe din punct de vedere computational. Din fericire, putem realiza conversia de la spațiul de culori HSL la spațiul de culori RGB. Având o culoare cu nuanță $H \in [0°, 360°]$, saturație $S_{HSL} \in [0, 1]$ și luminozitate $L \in [0, 1]$, prima dată căutăm cromatica:

$$C = (1 - |2L - 1|) \times S_{HSL}$$

Apoi putem găsi un punct (R_1, G_1, B_1) de-a lungul celor trei fețe de jos ale cubului RGB, cu aceeași nuanță și cromatică identică cu culoarea noastră (folosind valoarea intermediară X pentru a doua componentă cea mai mare a acestei culori).

$$H' = \frac{H}{60°}$$

$$X = C \times (1 - |H' \bmod 2 - 1|)$$

$$(R_1, G_1, B_1) = \begin{cases} (0,0,0), dacă\ H\ este\ nedefinit \\ (C, X, 0), \quad dacă\ 0 \le H' \le 1 \\ (X, C, 0), \quad dacă\ 1 \le H' \le 2 \\ (0, C, X), \quad dacă\ 2 \le H' \le 3 \\ (0, X, C), \quad dacă\ 3 \le H' \le 4 \\ (X, 0, C), \quad dacă\ 4 \le H' \le 5 \\ (C, 0, X), \quad dacă\ 5 \le H' \le 6 \end{cases}$$

Se observă că atunci când H' este număr întreg avem fie $X = C$, fie $X = 0$, fie $C = 0$, etc.

În cele din urmă, putem extrage culorile R, G și B adăugând aceași sumă fiecărei componente pentru a se potrivi cu luminozitatea $m = L - \frac{C}{2}$.

$$(R, G, B) = (R_1 + m_1 G_1 + m_1 B_1 + m)$$

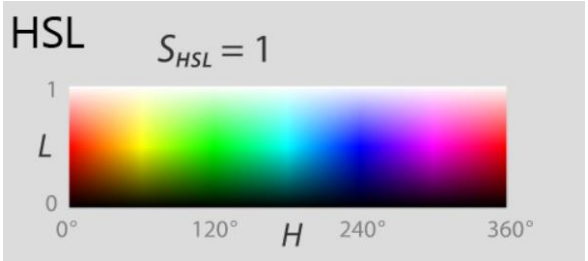

Figura 3. Reprezentarea plană a spectrului HSL

2.2.4 Operatorul Sobel

Operatorul Sobel, uneori numit operatorul Sobel-Feldman sau filtrul Sobel, este utilizat în procesarea imaginilor și în „computer vision”, în special în cadrul algoritmilor de detectare a marginilor, unde creează o imagine care accentuează marginile. Este numit după Irwin Sobel și Gary Feldman, colegi de la Laboratorul de Inteligență Artificială Stanford (SAIL). Sobel și Feldman au prezentat ideea unui „Isotropic 3x3 Image Gradient Operator”, la o discuție la SAIL în 1968 [16]. Din punct de vedere tehnic, este un operator de diferențiere discret, calculând o aproximare a gradientului funcției de intensitate a imaginii. În fiecare punct al imaginii, rezultatul operatorului Sobel-Feldman este fie vectorul de gradient corespunzător, fie norma acestui vector. Operatorul Sobel-Feldman se bazează pe conveierea imaginii cu un filtru mic, separabil și cu valoare integrată în direcțiile orizontale și verticale și, prin urmare, este relativ ieftin în ceea ce privește calculele. Pe de altă parte, aproximarea gradientului pe care o produce este relativ brută, în special pentru variațiile de înaltă frecvență ale imaginii.

Operatorul utilizează două kerneluri de dimensiune 3 × 3 care se imbină cu imaginea originală pentru a calcula aproximările derivatelor, o derivată pentru modificările orizontale și alta pentru cele verticale. Fie A imaginea și G_x și G_y sunt două imagini care la fiecare punct conțin aproximări ale derivatei orizontale și ale derivatei verticale, astfel:

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} * A \qquad G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} * A$$

unde * este operația de convoluție bidimensională

Deoarece kernelurile lui Sobel pot fi descompuse ca produse ale unui kernel de medie și de diferențiere, ele calculează gradientul cu precizie. De exemplu:

$$G_x = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} * ([-1 \quad 0 \quad 1] * A \qquad G_y = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} * ([1 \quad 2 \quad 1] * A$$

Coordonata x este își menține direcția „către dreapta”, iar coordonata *y* „în jos”. În fiecare punct al imaginii, aproximările gradientului rezultat pot fi combinate pentru a obține mărimea lui, folosind:

$$G = \sqrt{G_x^2 + G_y^2}$$

de asemenea, folosind această informație, putem calcula direcția gradientului:

$$\theta = \tan^{-1} \frac{G_y}{G_x}$$

Deoarece funcția de intensitate a unei imagini este cunoscută doar in punctele discrete, derivatele acestei funcții nu pot fi definite decât dacă se presupune că există o funcție de intensitate diferențială care să eșantoneze punctele din imagine. Cu unele presupuneri suplimentare, derivata funcției continue de intensitate poate fi calculată ca o funcție asupra funcției de intensitate a eșantionului. S-a descoperit că derivatele în oricare punct particular sunt

funcții ale valorilor intensității la aproape toate punctele din imagine. Cu toate acestea, aproximările acestor funcții derivate pot fi definite cu o precizie variată.

Rezultatul operatorului Sobel-Feldman este o hartă bidimensională a gradientului (despre care voi menționa in subcapitolul următor) în fiecare punct. Poate fi procesată și văzută ca și cum ar fi ea însăși o imagine, zonele cu un gradient înalt (marginile posibile) vizibile ca linii albe.

Figura 4. Exemplu de imagine alb-negru

Figura 5. Valoarea gradientului normalizat

2.2.5 Gradientul imaginii

Gradientul unei imagini este reprezentat de o schimbare direcțională a intensității sau culorii dintr-o imagine. Gradientul imaginii este unul dintre componentele fundamentale ale procesării de imagini. De exemplu, detectorul de margini Canny utilizează gradientul imaginii pentru detectarea de margini. În software-ul grafic pentru editare digitală a imaginilor, gradientul de culoare este, de asemenea, utilizat pentru combinarea treptată a culorilor, care poate fi considerat ca o gradare uniformă de la valori joase la valori mari, așa cum se utilizează de la alb la negru în imaginile din dreapta.

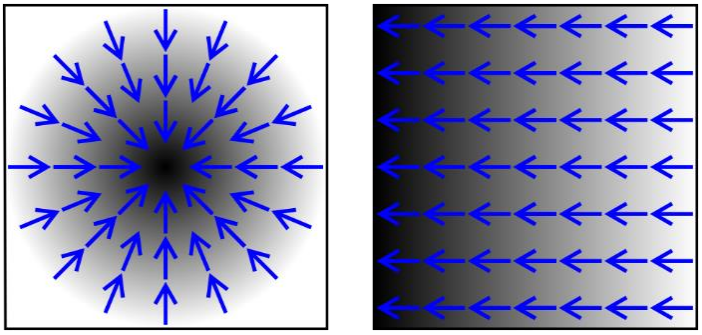

Figura 6. Două exemple are gradientului în imagini alb-negru

Din punct de vedere matematic, gradientul unei funcții cu două variabile (aici funcția de intensitate a imaginii) în fiecare punct din imagine este un vector bidimensional cu componentele date de derivatele de pe axa orizontală și axa verticală. La fiecare punct din imagine, vectorul de gradient indică direcția celei mai mari creșteri de intensitate posibile, iar lungimea vectorului de gradient corespunde ratei de schimbare în acea direcție [17].

Deoarece funcția de intensitate a unei imagini digitale este cunoscută numai în punctele discrete, derivatele acestei funcții nu pot fi definite decât dacă presupunem că există o funcție de intensitate continuă care a fost prelevată în punctele din imagine.

Gradienții imaginilor sunt adesea utilizați în hărți și în alte reprezentări vizuale ale datelor pentru a transmite informații suplimentare. Instrumentele GIS utilizează progresele culorilor pentru a indica altitudinea și densitatea populației.

De asemenea, gradientul unei imagini este folosit în domeniul „computer vision”. El poate fi folosit pentru a extrage informații. Imaginile gradient sunt create din imaginea originală, în general prin aplicarea unui filtru (de exemplu filtrul Sobel, vezi subcapitolul 2.2.4). Fiecare pixel al unei imagini gradient măsoară modificarea intensității aceluiași punct din imaginea originală într-o anumită direcție. Pentru a obține întreaga gamă de direcții, se calculează atât imaginea gradient pe verticală, cât și pe orizontală.

Cel mai frecvent este utilizat pentru detectarea marginilor. După ce a fost calculată imaginea gradient, pixelii cu valori de gradient mare devin posibili pixeli de margine. Pixelii cu cele mai mari valori de gradient în direcția gradientului devin pixeli margini, iar marginile având direcția perpendiculară pe direcția gradientului. Gradientul unei imagini mai poate fi folosit pentru potrivirea texturii. Diferența de luminozitate sau proprietățiile aparatului pot cauza diferențe majore ale valorilor pixelilor pentru două imagini ale aceluiași fundal.

Matematic, gradientul este un vector:

$$\nabla f = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\delta f}{\delta x} \\ \frac{\delta f}{\delta y} \end{bmatrix}$$

unde:

$\frac{\delta f}{\delta x}$ este derivata în raport cu x (gradientul pe direcția x)

$\frac{\delta f}{\delta y}$ este derivate în raport cu y (gradientul pe direcția y)

Derivata unei imagini poate fi aproximată prin diferențe finite. Dacă se folosește diferența centrală, pentru a calcula $\frac{\delta f}{\delta y}$ putem aplica un filtru unidimensional pe imaginea A prin convoluție:

$$\frac{\delta f}{\delta y} = \begin{bmatrix} -1 \\ 1 \end{bmatrix} * A$$

unde * indică operația de convoluție unidimensională. Acest filtru 2 × 1 va modifica imaginea cu jumătate de pixel. Pentru a evita acest lucru, se poate utiliza un filtru 3 × 1:

$$\begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$$

2.3 Rețele neuronale artificiale

2.3.1 Noțiuni introductive

Studiul creierului uman este o preocupare de mii de ani a umanității. Traversăm o perioadă a dezvoltării tehnologice care face posibilă valorificarea proceselor de gândire mai mult ca niciodată. Simularea artificală a gândirii se poate realiza acum prin aplicarea unor metode noi pe care știința le pune la dispoziție, ele find așa-numitele rețele neuronale artificiale. Primul pas spre dezvoltarea rețelelor neuronale artificiale a fost făcut în 1943 de către Warren McCulloch, un neurofiziolog și Walter Pitts, un tânăr matematician, care au scris o lucrare despre modul în care ar putea funcționa neuronii. Ei au modelat o rețea simplă neurală cu circuite electrice. Rețelele neuronale au abilitatea remarcabilă de a obține informații relevante din date complicate sau imprecise și pot fi folosite pentru a extrage modele și a detecta tendințe care sunt prea complexe pentru a fi observate de către oameni sau chiar de către tehnici avansate computaționale. O rețea neuronală are următoarele avantaje:

- Învățare adaptivă: este abilitatea de a învăța cum să indeplinească anumite sarcini pe baza experienței dobândite din antrenarea pe un set de date inițial
- Auto-organizare: este abilitatea unei rețele neuronale artificiale de a-și crea propria viziune asupra datelor în timpul învățării
- Operații efectuate în timp real: o rețea neuronală artificială poate efectua calcule în paralel
- Toleranța erorilor prin codificarea informațiilor redundante: au capacitatea de a rămâne funcționale, chiar dacă apar erori in rețea

Rețelele neuronale utilizează utilizează o abordare diferită față de cea a calculatoarelor convenționale. Calculatoarele utilizează o abordare algoritmică, adică urmează un set de instrucțiuni pentru a rezolva o problemă. Dacă nu se cunosc pașii specifici pe care trebuie să-i urmeze, nu va putea rezolva problema. Acest fapt limitează capacitatea de rezolvare a problemelor la situații pe care deja le înțelegem și știm cum să le rezolvăm. Calculatoarele ar fi mult mai utile dacă ar putea deduce lucruri pe care nu le întelegem exact. Rețelele neuronale procesează informații într-un mod similar creierului uman, deoarece ele învață prin exemple. În informatică, ori de câte ori vorbim despre o rețea neuronală, de fapt ne referim la termenul oficial de rețea neuronală artificială.

Rețelele neuronale artificiale sunt modele electronice relativ brute, bazate pe structura neronală a creierului uman. Creierul învață practic din experiență. Această modelare a creierului promite o abordare mai puțin tehnică pentru rezolvarea problemelor computaționale. De asemenea, această nouă abordare oferă o degradare mai grațioasă în timpul supraîncărcării sistemelor decât echivalenții săi tradiționali. Aceste modele inspirate din biologie sunt considerate următorul progres major în industria de calcul. Inclusiv creierul animal simplu este capabil de săvârșirea anumitor sarcini pe care calculator nu le poate executa. Chiar dacă computerele reușesc să efectueze calcule matematice complexe, au dificultăți în recunoașterea modelelor, chiar dacă acestea sunt simple.

Chiar dacă ele au fost inspirate din biologie, ar fi greșit să considerăm că neuronii artificiali sunt identici cu cei biologici. Creierul uman conține aproximativ 100 de miliarde de neuroni care operează in paralel [19]. Neuronii artificiali au funcții implementate pe calculatoare seriale (mai mult sau mai puțin). Cercetările privind rețelele neuronale sunt în mare parte ghidate de evoluțiile din inginerie și matematică, nu din biologie [9].

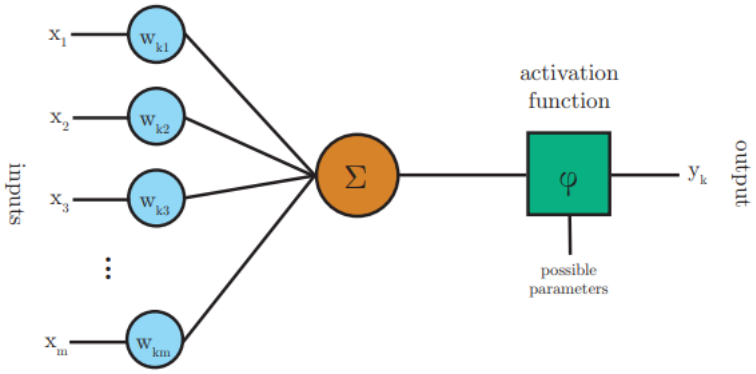

Figura 7. Reprezentarea unui neuron artificial

Un neuron artificial bazat pe modelul McCulloch-Pitts este prezentat în Figura 7 [6]. Neuronul k primește m parametrii de intrare cu valorile x_j. Neuronul are, de asemenea, m parametrii de pondere („weights parameters”) w_{kj}. Parametrii de pondere includ adesea o valoare de influență („bias”) care potrivesc datele de intrare necorespunzătoare cu valoarea fixă 1. Datele de intrare și parametrii de pondere sunt combinate liniar și sunt însumate. Suma este apoi alimentată la o funcție de activare φ care produce ieșirea y_k a neuronului:

$$y_k = \varphi(s_k) = \varphi\left(\sum_{j=0}^{m} w_{kj} x_j\right)$$

Neuronul este antrenat prin alegerea cu grijă a parametriilor de pondere pentru a produce un rezultat optim pentru fiecare dată de intrare.

O rețea neurală este o combinație de neuroni artificiali. Neuronii sunt de obicei grupați în straturi. Într-o rețea cu mai multe straturi complet conectate, fiecare rezultat a unui strat de neuroni este dată de intrare pentru fiecare neuron al următorului strat. Astfel, unele straturi procesează datele de intrare inițiale, în timp ce alte straturi procesează datele rezultate din straturile anterioare. Fiecare neuron are un număr de parametrii de pondere egal cu numărul de neuroni din stratul anterior [6].

O rețea cu mai multe straturi include de obieci trei tipuri de straturi: un strat de intrare, unul sau mai multe straturi ascunse și un strat de ieșire [6]. Stratul de intrare, de obicei, doar transmite datele fără a le modifica. Cea mai mare parte din calcule au loc în straturile ascunse. Stratul de ieșire convertește rezultatele stratului ascuns la o ieșire, asemănător unei clasificări. O rețea cu mai multe straturi care are cel puțin un strat ascuns poate funcționa ca un aproximator universal, adică poate fi construită pentru a calcula aproape orice funcție [18].

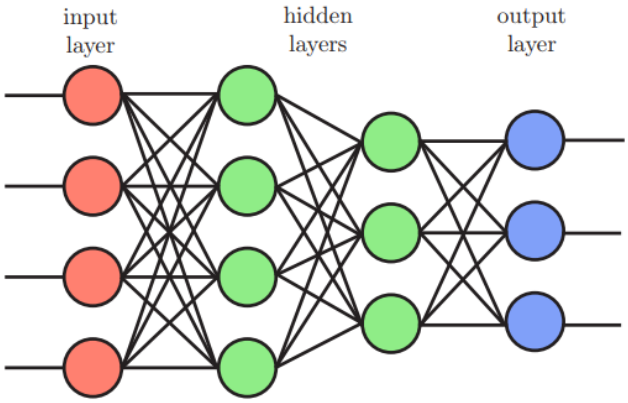

Figura 8. Reprezentarea unei rețele neuronale cu mai multe straturi

Antrenarea unei rețele neuronale se realizează prin selectarea parametrilor de pondere a tuturor neuronilor astfel încât rețeaua să învețe să aproximeze rezultatele din datele de intrare cunoscute. Deoarece este dificil să rezolvăm analitic valorile parametrilor de pondere, a fost dezvoltat conceptul de propagare înapoi („back-propagation”). Algoritmul de propagare înapoi [6] oferă o soluție simplă și eficientă pentru rezolvarea valorilor parametrilor de pondere într-un mod iterativ. Versiunea clasică a acestui algoritm folosește metoda gradientului descendent pentru optimizare. Metoda gradientului descendent poate fi costisitoare din punct de vedere al timpului și nu e garantat că va extrage eroarea minimă globală, dar dacă utilizează o configurație adecvată (cunoscută în „machine learning” drept „hyperparameters”) funcționează destul de bine în practică.

În prima fază a algoritmului, un vector de intrare este propagat înainte prin rețeaua neruonală. Înainte de aceasta, valorile parametrilor de pondere au fost inițializați cu valori aleatorii între 0 și 1. Rezultatul obținut în urma parcurgerii rețelei neuronale este comparat cu rezultatul așteptat (rezultat care ar trebui să fie cunoscut pentru datele de test) utilizând o funcție numită „loss function”. Apoi, se calculează gradientul funcției de pierdere. Acest gradient este așa-numit valoarea erorii. Când se utilizează media pătrată a erorii ca funcție de pierdere, valoarea erorii stratului de ieșire este diferența dintre rezultatul curent și cel dorit.

Aceste valori ale erorii sunt apoi propagate înapoi prin rețea pentru a calcula valorile de eroare ale neuronilor din stratul ascuns. Valoarea funcției de pierdere a neuronilor ascunși poate fi calculată utilizând regula lanțului de derivate. În cele din urmă, valoarile parametrilor de pondere ale neuronilor sunt actualizate prin calcularea gradientului valorilor parametrilor de pondere pe care îl scădem din parametrii de pondere actuali. Acest raport este numit rata de învățare. Rata de învățare poate fi fixă sau dinamică. După ce parametrii de pondere au fost actualizați, algoritmul continuă executând din nou aceași pași cu aceleași date de intrare până când rezultatele converg.

Funcția de activare φ determină ieșirea finală a fiecărui neuron. Este important să fie selectată o funcție adecvată pentru ca rețeaua să funcționeze bine.

În urma cercetărilor s-a descoperit că perceptronii și alte sisteme liniare au avut dezavantaje majore, fiind în imposibilitatea de a rezolva probleme care nu au fost separabile liniar, cum ar fi problema XOR. Uneori, sistemele liniare pot rezolva astfel de probleme folosind detectoare, dar nu este cea mai avantajoasă soluție în „machine learning”. Adăugarea straturilor ascunse nu ajută deoarece o dacă o rețea compusă din neurori este liniară atunci ea va rămâne liniară, indiferent câte straturi sunt adăugate [20].

O metodă ușoară și eficientă de creare a unei rețele neliniare este adăugarea de funcții liniare rectificate (ReLu [20]). O funcție liniară rectificată calculează rezultatul folosind o funcție de „rampă”, de exmplu:

$\varphi(s) = \max(0, s)$

Acest tip de funcție este ușor de calculat și de diferențiat pentru propagarea înapoi. Chiar dacă funcția nu poate fi diferențiată în 0, dar acest lucru nu a împiedicat utilizarea sa în practică. Funcția ReLu a devenit destul de populară, adesea înlocuiind funcția de activare sigmoidală, care are derivatele continue, însă gradientul are probleme de saturație și calculul este mai lent. Pentru problemele de clasificare cu mai multe clase, funcția de activare softmax [6] este utilizată în stratul de ieșire al rețelei:

$$\varphi(s) = \frac{\exp s_k}{\sum_{k=1}^{K} \exp s_k}$$

Softmax folosește un vector de K valori de dimensiune mare arbitrare și returnează un vector cu K valori cuprinse între 0 și 1și suma lor este egală cu 1. Valorile extrase de această funcție pot fi folosite ca probabilități de clase.

Rețelele neuronale moderne sunt numite adesea rețele neuronale profunde. Deși rețelele neuronale cu mai multe straturi au existat încă din anii 1980, mai multe motive au împiedicat formarea eficientă a rețelelor cu mai multe straturi ascunse [20].

2.3.2 Detectarea de obiecte

Detectarea de obiecte este o tehnologie care înrudită cu „computer vision” și procesarea de imagini, care se ocupă cu detectarea anumitor clase (cum ar fi oameni, mașini, etc.) în clipuri video sau în imagini.

Această tehnologie dorește să rezolve una dintre problemele clasice pe care „computer vision” dorește să le rezolve, detectarea de obiecte find considerată una dintre cele mai dificile sarcini. În multe privințe, această sarcină este similară cu majoritatea celor din „computer vision” deoarece implică o soluție invariabilă pentru deformarea provocată de schimbările de lumină și de perspectivă. Ceea ce face ca detectarea de obiecte să reprezinte o problemă deosebită este faptul că implică atât localizarea regiunilor de interes, cât și clasificarea lor. În anumite situații partea de localizare nu este utilizată (atunci când folosim intreaga imagine).

Pentru a detecta un obiect, trebuie să avem o idee despre locul unde ar putea fi obiectul în imagine și despre felul cum poate fi împărțită imaginea. Această situație creează urmatoarea

problemă: pentru a recunoaște forma și clasa unui obiect, trebuie să cunoaștem locația acestuia și pentru a cunoaște locația lui, trebuie să-i cunoaștem forma [21]. Unele trăsături sunt vizibile, cum ar fi hainele și chipul unei ființe umane, iar ele pot fi considerate părți ale aceluiași obiect, dar este dificil să știm acest lucru fără a recunoaște mai înâi obiectul. Alte obiecte se remarcă doar puțin în fundal, iar ele necesită o separare înainte de recunoaștere.

Locația și dimensiunea sunt de obieci definite printr-o cutie de încadrare („bounding box”), care este definită prin coordonatele colțurilor. Folosirea unui dreptunghi este mai simplă decât utilizarea unui poligon în formă arbitrară, iar multe operații, cum ar fi convoluția, sunt realizate în orice situație pe dreptunghiuri. Sub-imaginea obținută în caseta de încadrare este apoi clasificată printr-un algoritm care a fost instruit folosind „machine learning”. Marginile obiectului pot fi rafinate ulterior iterativ.

În anii 2000, soluțiile de detecție a obiectelor foloseau descriptorii de caracteristici, cum ar fi histograma orientărilor gradientului (HOG), populară in anul 2005 [22]. În 2010, s-a înregistrat o schimbare majoră, îndreptată spre utilizarea rețelelor neuronale convoluționale [23].

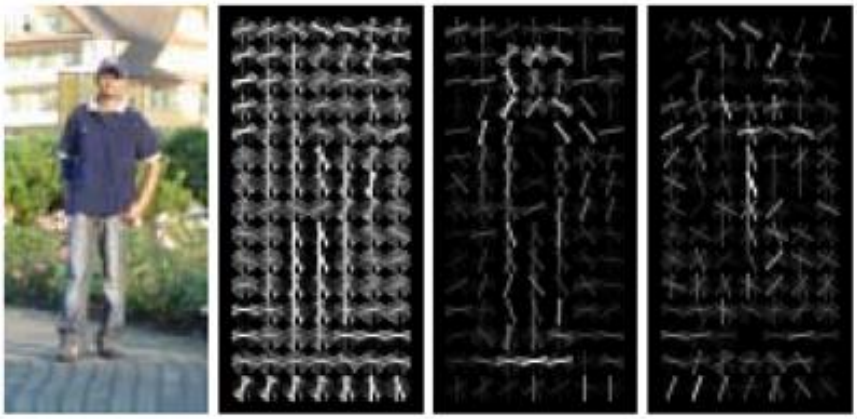

Figura 9. Exemplu de descriptor HOG pentru recunoașterea de persoane

Înainte de utilizarea pe o scară largă a rețelelor neuronale convoluționale, au existat două soluții pentru generarea cutiilor de încadrare. Prima soluție propune generarea unui set de propuneri de regiune și majoritatea acestora vor fi respinse [24]. În a doua soluție se propune generarea unui set restrâns de cutii de încadrare folosind o metodă de propunere a regiunii, cum ar fi „Selective Search” [25].

2.3.3 Rețele neuronale convoluționale

Rețelele neuronale convoluționale au fost o evoluție firească a rețelelor neuronale clasice și cu ajutorul lor se dorește rezolvarea problemelor întâlnite în computer vision (până și o imagine de dimensiuni modeste conține o cantitate enormă de informații).

O imagine monocromă de dimensiune 620x480 conține 297.600 de pixeli. Dacă fiecare intensitate a pixelilor din această imagine este introdusă separat într-o rețea complet conectată, fiecare neuron necesită 297.600 de parametrii de pondere. O imagine Full HD de dimensiune 1920x1080 ar necesita 2.073.600 de parametrii de pondere. Dacă imaginile sunt policrome, cantitatea de greutăți este înmulțită cu cantitatea de canale de culoare (de obicei trei). Astfel, putem observa că numărul total de parametri liberi din rețea devine rapid extrem de mare, pe măsură ce mărimea imaginii crește. Modelele prea mari provoacă „overfitting” și performanțe lente [6].

În plus, multe sarcini de detecție a modelelor necesită ca soluția să fie invariabilă în translație. Este ineficient să instruim neuronii să recunoască separat același model în colțul din stânga sus și în colțul din dreapta-jos al unei imagini. O rețea neuronală complet conectată nu ia în considerare acest tip de structură.

Ideea de bază a rețelelor neuronale convoluționale a fost inspirată dintr-un concept din biologie numit domeniul receptiv [3]. Suprafețele receptorilor sunt o caracteristică a cortexului vizual al animalului. Acestea acționează ca detectori care sunt sensibili la anumite tipuri de stimuli, de exemplu, marginile. Acestea se găsesc pe întreaga suprafață vizuală și se suprapun între ele.

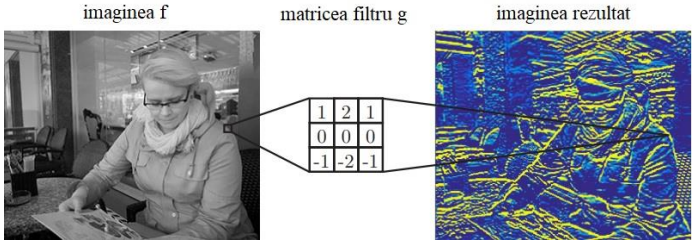

Figura 10. Detectarea marginilor orizontale dintr-o imagine utilizând filtrarea prin convoluție

În procesarea imaginilor, imaginile pot fi filtrate folosind convoluția pentru a produce diferite efecte vizibile. Figura 10 arată modul în care un filtru convoluțional selectat manual detectează marginile orizontale dintr-o imagine care funcționează similar cu un câmp receptiv.

Operația de convoluție discretă dintre o imagine f și o matrice filtru g este definită ca:

$$h[x,y] = f[x,y] * g[x,y] = \sum_n \sum_m f[n,m]g[x-n, y-m]$$

De fapt, produsul scalar al filtrului g și sub-imaginea f (cu aceleași dimensiuni ca g) centrate pe coordonatele x, y produce valoarea pixelului h la coordonatele x, y [20]. Dimensiunea câmpului receptiv este ajustată de dimensiunea matricei de filtrare. Alinierea succesiv a filtrului

cu fiecare sub-imagine a lui *f* produce matricea de pixeli de ieșire *h*. În cazul rețelelor neuronale, matricea de ieșire reporezintă o hartă a caracteristicilor [20] (sau o hartă de activare după calculul funcției de activare). Marginile trebuie tratate ca un caz special.

Un set de filtre convoluționale pot fi combinate pentru a forma un strat convoluțional dintr-o rețea neurală [19]. Valorile matricei filtrelor sunt tratate ca neuroni și instruiți prin „machine learning”. Operația de convoluție înlocuiește operația de multiplicare a unui strat ascuns dintr-o rețea neuronală. Ieșirea stratului este de obicei descrisă ca un volum. Înălțimea și lățimea volumului depind de dimensiunile hărții de activare. Adâncimea volumului depinde de numărul de filtre.

Deoarece aceleași filtre sunt utilizate pentru toate zonele imaginii, numărul de parametri liberi este redus drastic în comparație cu un strat conectat complet. Neuronii din stratul convoluțional au în cea mai mare parte aceiași parametri și sunt conectați doar la o regiune locală a datelor de intrare. Partajarea parametrilor care rezultă din convoluție asigură invarianța translației. O modalitate alternativă de a descrie stratul convoluțional este de a vă imagina un strat complet conectat, cu un prefix plasat pe parametri de pondere [20]. Acest lucru forțează neuronii să împartă parametri de pondere în diferite locații spațiale și să aibă greutate zero în afara câmpului receptiv.

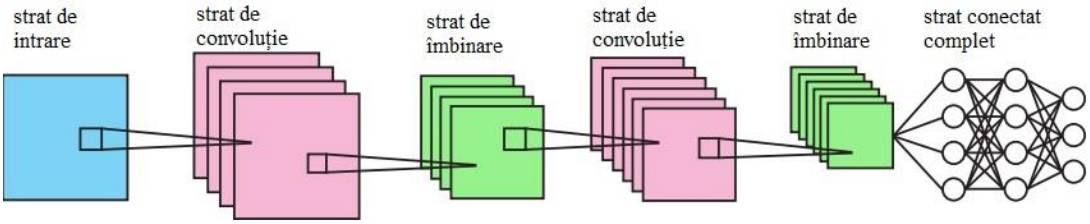

Figura 11. Exemplu de rețea neuronală convoluțională

Straturile convoluționale succesive (care sunt de obicei combinate cu alte tipuri de straturi, vezi Figura 11) formează o rețea neuronală convoluțională. Algoritmul de antrenare care utilizează propagarea înapoi, descris în secțiunea 2.3.1, este de asemenea aplicabil rețelelor convoluționale [20]. Teoretic, straturile mai aproape de intrare ar trebui să învețe să recunoască caracteristicile de nivel inferior ale imaginii, cum ar fi marginile și colțurile, iar straturile mai aproape de ieșire ar trebui să învețe să combine aceste caracteristici pentru a recunoaște forme mai semnificative [9].

Pentru ca rețeaua să fie mai ușor de gestionat pentru clasificare, este util să fie micșorată dimensiunea hărții de activare în adâncimea rețelei. În general, straturile profunde ale rețelei necesită mai puține informații despre locațiile spațiale exacte ale caracteristicilor, dar necesită mai multe matrici de filtrare pentru a recunoaște mai multe modele la nivel înalt [22]. Prin reducerea înălțimii și lățimii volumului de date, putem crește adâncimea volumului de date și putem menține timpul de calcul la un nivel rezonabil.

Există două modalități prin care se poate reduce dimensiunea volumului de date. O modalitate este de a include un strat de îmbinare („pooling layer”) după un strat convoluțional [20]. Stratul elimină eficient hărțile de activare. Îmbinarea are efectul de a transforma rețeaua rezultat într-o rețea mai invariantă din punct de vedere al translației, fortând detectorii să fie mai

imprecișii. Cu toate acestea, punerea în comun poate distruge informații despre relațiile spațiale dintre părtile formelor. Metoda specifică de combinare este maximizarea îmbinărilor („max-pooling”). Maximizarea îmbinărilor reprezintă alegerea valorii maxime într-o vecinătate dreptunghiulară a hărții de activare [20].

O altă modalitate de a reduce dimensiunea volumului de date este ajustarea parametrului de deplasare al operației de convoluție. Parametrul de deplasare controlează rezultatul convoluției prin modul în care este calculată maximizarea îmbinărilor cu ajutorul numărului de pixeli peste care se deplasează. Cercetările au arătat că straturile de îmbinare pot fi deseori eliminate, fără pierderi de precizie, prin utilizarea straturilor convoluționale cu o valoare mai mare a deplasării [26]. Operația de deplasare este echivalentă cu utilizarea unui chenar fix pentru punerea în comun.

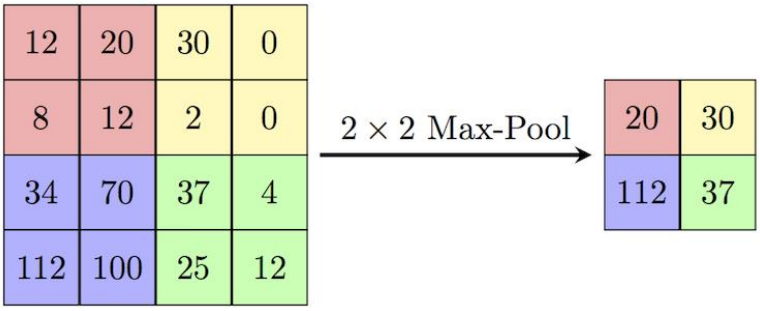

Figura 12. Exemplu de maximizare a îmbinărilor

Straturile convoluționale include de obicei o funcție de activare neliniară, cum ar fi o funcție de activare liniară rectificată (vezi secțiunea 2.3.1). Activitățile sunt uneori descrise ca un strat separat între stratul convoluțional și stratul de îmbinare.

Unele sisteme, implementează un strat numit stratul de normalizare a răspunsului local („local response normalization”) care este folosit ca tehnică de regularizare. Replicarea locală a răspunsului imită o funcție a neuronilor biologici denumită inhibiție laterală, care determină neuronii excitați să scadă activitatea neuronilor învecinați.

Straturile finale ascunse ale unei rețele neuronale convoluționale sunt de obicei straturi complet conectate [6]. Un strat complet conectat poate capta unele relații interesante de partajare a parametrilor pe care straturile convoluționale nu pot. Cu toate acestea, un strat complet legat necesită o dimensiune a volumului de date mică pentru a fi practic. Setările de îmbinare și de deplasare pot fi utilizate pentru a reduce dimensiunea volumului de date care ajunge la straturile conectate complet. O rețea convoluționară care nu include straturi conectate în întregime, se numește o rețea complet convoluțională (FCN) [27].

3 Parte experimentlă

Din cazua numărului mare de accidente petrecute în trafic și datorită evoluției tehnologiei, au apărut treptat diferite sisteme inteligente de asistență auto (vezi capitolul 1). Astfel, scopul acestei lucrări este de a prezenta un prototip al unui asemenea sistem. Acest sistem va conține două mari funcționalități:

- Detectarea benzii curente
- Detectarea mașinilor

Deci, sistemul va fi capabil să detecteze atât banda pe care se află, cât si mașinile din trafic. Pentru crearea și testarea sistemului, s-a folosit un videoclip dintr-o provocare oferită de către un curs de învățare aprofundată („Udacity Self-Driving Car Engineer Nanodegree Program”).

Figura 13. Rezultatul produs de către prototip

Sistemul este proiectat în așa manieră cât să funcționeze videoclipul oferit în provocare, însă poate fi îmbunătățit și extins. În următoarele secțiuni vor fi prezentate etapele necesare pentru a obține programul.

3.1 Detectarea și încadrarea în bandă

În timpul deplasării, autovehiculul trebuie să fie încadrat pe o bandă de circulație, iar pentru a evita accidentele, autovehiculul trebuie să fie încadrat corespunzător și să își mențină banda (cu excepția depășirilor și schimbărilor intenționate de bandă). Pentru o încadrare mai bună în banda curentă, propotitpul oferă o detecție a benzii curente (ea find marcată vizual în clip) și oferă informații asupra poziției în bandă, astfel șoferul se poate încadra mai ușor.

Figura 14. Exemplu de detecție a benzii

Acest sistem poate fi obținut urmând urmatorii pași:

1. Calibrarea imaginii – acest lucru este realizat prin calcularea matricii de calibrare și a coeficienților de distorsiune ai obiectivului camerei, utilizându-se un set de imagini (realizate cu același obiectiv) ale unei table de șah.
2. Schimbarea perspectivei – aplicarea schimbării de perspective se realizează pentru a obține o vedere de sus a benzilor
3. Filtrarea culorilor în imagine– acest lucru se realizează prin aplicarea operatorului Sobel și a gradientului imaginii pentru a obtine o imagine binară care să conțină doar infomațiile utile, acestea find chiar marcajul benzilor.
4. Măsurarea curburii și calcularea poziției mașinii – se realizează prin calcularea polinomului care trece prin cele mai multe puncte ale fiecărui marcaj.
5. Proiectarea benzii în perspectiva normală – după ce au fost obținute marcajele benzii, se proiectează banda, și se revine la perspectiva normală.

3.1.1 Calibrarea imaginii

Datorită proprietăților fizice ale obiectivului, imaginea bidimensională capturată nu este perfectă. Există distorsiuni ale imaginii care modifică dimensiunea și forma aparentă a unui obiect. Mai important, unele obiecte apar mai aproape sau mai departe decât sunt. Din fericire, putem măsura aceste distorsiuni și le putem corecta. Putem extrage toate informațiile de distorsiune de care avem nevoie, având imagini cu obiecte pe care le știm unde ar trebui să fie teoretic anumite puncte. În mod obișnuit, sunt folosite tablele de șah aflate pe o suprafață plană deoarece ele au modele regulate de contrast înalt și cunoaștem forma tablei nedistorsionată.

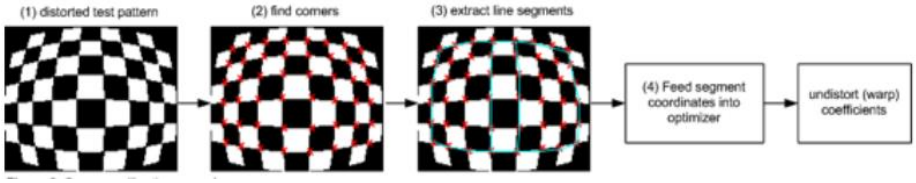

Figura 15. Procedura de calibrare a imaginii

Dacă avem mai multe imagini ale aceleiași table aflate pe o suprafață plană, putem obține diferența dintre dimensiunea si forma aparentă a imaginilor, comparativ cu ceea ce ar trebui să fie de fapt. Poate fi creată o transformare care să direcționeze punctele distorsionate către puncte nedistorsionate. Această transformare va putea fi folosită mai târziu pentru a calibra orice imagine realizată de acel obiectiv.

Pentru calibrarea imaginilor, am creat clasa CameraCalibration care calculează matricea de calibrare și coeficienții de distorsiune pe baza a douazeci de imagini din data/calibration/ utilizând funcții predefinite din OpenCV (findChessboardCorners() și calibrateCamera()). De asemenea, după ce matricea și coeficienții au fost calculați, va putea să elimine distorsiunea din

orice imagine realizată de acel obiectiv prin utilizarea funcției „undistort” regăsită în OpenCV. Rezultatul obținut în urma eliminării distorsiunilor arată astfel:

Figura 16. Rezultatul obținut în urma calibrării imaginii

Chiar dacă diferențele nu sunt de cele mai multe ori vizibile cu ochiul liber când vine vorba de distorsiuni cauzate de către obiectiv, ele joacă un rol important în ansamblul imaginii, iar dacă ar fi să privim imaginea din punct de vedere matematic (ca pe o matrice), diferențele sunt suficient de mari cât să influențeze operațiile următoare.

3.1.2 Schimbarea perspectivei

După ce imaginea a fost corectată, vom avea o imagine nedistorsionată a unui drum din perspectiva normală a unui vehicul. Pentru a extrage toate informațiile necesare pentru detectarea benzii curente, această imagine trebuie adusă într-o perspectivă din care să putem observa mai bine delimitările benzilor. Aceasta ar trebui să fie perpendiculară pe drum, astfel încât delimitările să se asemene cu liniile dintr-un plan bidimensional. Putem spune că această perspectivă „de deasupra” se aseamănă cu perspectiva unei păsări („bird perspective”) care zboară deasupra unui drum, iar de în continuare, de fiecare dată când vom face o referire la această perspectivă, ea va fi asociată cu perspectiva unei păsări.

Pentru a tranforma o imagine a drumului care oferă perspectiva unui vehicul într-o imagine cu perspectiva unei păsări, tot ce avem nevoie sunt coordonatele locației dorite. Mai exact, în imaginea de intrare trebuie localizate punctele care oferă perspectiva în care dorim să transformăm imaginea. Numesc aceste coordonate puncte sursă („source_points”) și puncte destinație („destination_points”). Punctele sursă conturează zona perspectivei în care ne aflăm, iar punctele destinație reprezintă zona pe care dorim să o vizualizăm asemenea celei oferite de punctele sursă.

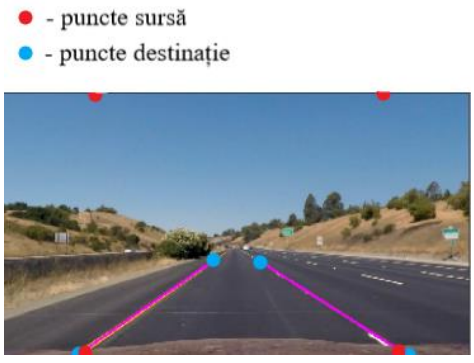

Figura 17. Reprezentarea coordonatelor sursă si destinație

Pentru schimbarea perspectivei, am creat clasa BirdsEye care transformă orice imagine din perspectiva normală în perspectiva unei păsări și invers. Clasa foloseste punctele sursă si punctele destinație specifice setului meu de imagini (imagini preluate din videoclip) pentru a crea aceste perspective utilizând funcția oferită de OpenCV getPerspectiveTransform(), iar pentru schimbarea efectivă a perspectivei utilizez funcția warpPerspective(). Astfel, în urma aplicării schimbării de perspective, rezultatul obținut este următorul:

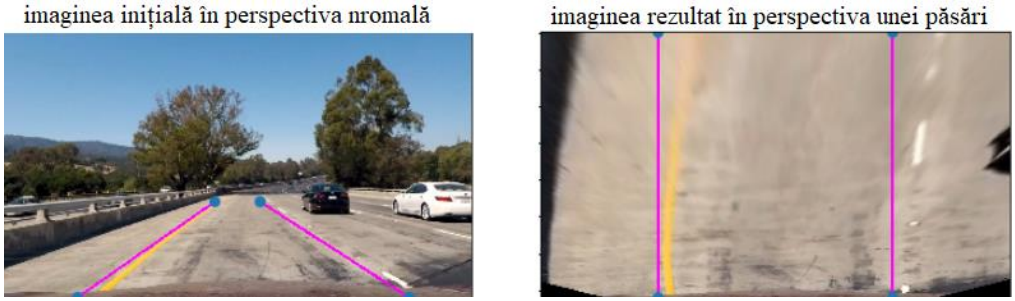

Figura 18. Rezultatul obținut în urma schimbării perspectivei

3.1.3 Filtrarea culorilor în imagine

Pentru a extrage din imagine banda împreuna cu curbura sa, avem nevoie de liniile care delimitează benzile, restul informațiilor din imagine find inutile. Deci, trebuie să extragem liniile care marchează banda, iar acest lucru poate fi realizat prin filtrarea culorilor. În videoclipul ales, drumul este marcat cu linii de culoare albă și linii de culoare galbenă, deci tot ce nu este de culoare albă sau galbenă în imagine poate să fie mascat. Ca o măsură de precauție suplimentară, pentru a evita extragerea anumitor pixeli care nu aparțin marcajului (de exemplu: poate exista culoarea galbenă în imagine, chiar dacă aceasta nu face parte din marcajul benzilor) va fi folosit așa-numitul operator Sobel (discutat in secțiunea 2.2.4). În principu, operatorul Sobel măsoară rata schimbării valorilor (cât și a direcției) dintre două locații diferite ale imaginii. Tehnic, operatorul Sobel reprezintă gradientul imaginii.

Prima dată, am reprezentat imaginea în format HSL (vezi secțiunea 2.2.3). Pe scurt, nuanța („hue”) reprezintă combinația culorilor primare, saturația („saturation”) măsoară intensitatea culorii, iar luminozitatea („lightness”) ne spune cât de deschisă este o culoare, deci cât de aproape de alb este. Liniile de culoare galbenă sunt evidențiate printr-o combinație de luminozitate și o saturație peste o anumită valoare de prag, iar cele albe sunt evidențiate printr-o valoare foarte mare a luminozității, care nu ține cont nici de nuanță, nici de saturație. Aceste valori de prag nu sunt predefinite și nu există valori fixe pentru ele. În cazul de față, valorile de prag au fost alese în urma experimentelor, iar rezultatul obținut în urma extragerii culorilor este de forma:

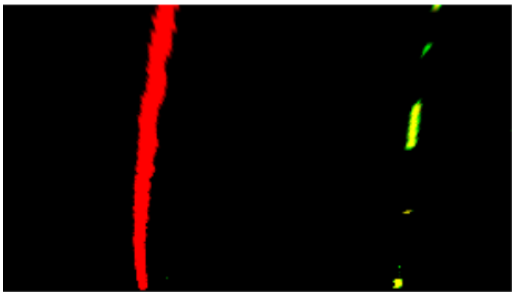

Figura 19. Rezultatul obținut în urma extragerii culorilor

După, am aplicat operatorul Sobel pe valoarea luminozității imaginii, asupra căruia am folosit o combinație de praguri dintre gradientul componentei orizontale, magnitudinea gradientului și direcției sa. Am făcut acest lucru pentru a elimina din imagine zonele care nu conțin o schimbare suficient de mare a luminozității. Utilizarea unui valori de prag pentru magnitudine și componenta x a gradientului ajută la scopul dorit. De asemenea, iau în considerare doar gradienții de o anumită orientare, care au între 0.7 (puțin peste 0°) și 1.4 radiani (sub 90°). Zero grade implică linii orizontale și nouăzeci de grade implică verticale, iar în perspectiva normală benzile se află între ele.

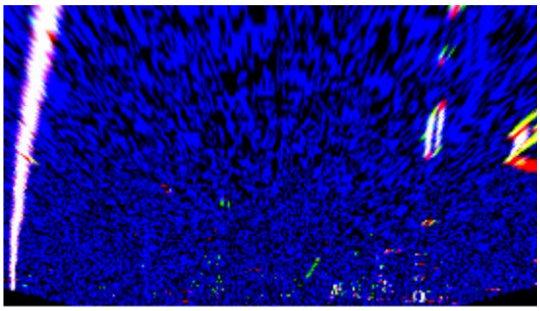

Figura 20. Rezultatul obținut în urma aplicării operatorului Sobel

Prin combinarea celor două rezultate obținute prin aplicarea filtrului de culoare si a operatorului Sobel obținem marcajul drumului sub o formă destul de promițătoare. Chiar dacă apar câteva valori străine din cauza umbrelor din imagine, ele pot fi extrase ulterior. Rezultatul final al combinării celor două filtre este următorul:

Figura 21. Rezultatul obținut în urma combinărilor filtrelor

Pentru realizarea acestor operații, am creat clasa LaneFilter care filtrează culorile dintr-o imagine și oferă ca rezultat o imagine binară de forma celei din Figura 20. Pentru realizarea filtrărilor au fost folosite operații matematice disponibile în NumPy și OpenCV.

3.1.4 Măsurarea curburii și calcularea poziției mașinii

Curbura marcajului benzii (care a fost obținut anterior) se poate calcula utilizând o funcție de gradul al doilea $x = Ay^2 + By + C$. Deci, problema se reduca la aflarea coeficienților [A, B, C], iar pentru aflarea acestora se poate utiliza funcția polyfit() inclusă în NumPy. Această funcție primeste mai multe puncte și pentru un anumit grad al funcției returnează coeficienții care oferă cea mai potrivită funcție de acel grad (funcție de care aparțin cele mai multe puncte dintre cele oferite în datele de intrare). Pentru a decide care dintre pixelii din imagine (puncte într-un plan bidimensional) fac parte din marcajul benzii, vom folosi un algoritm explicat în următoarea parte.

Prin utilizarea unei histograme a tuturor punctelor grupate pe coloane din jumătatea inferioară a imaginii obținem un grafic similar cu cel din Figura 22 care are două zone de vârf. Aceste zone de vârf sunt indicatori buni ai poziției orizontale (pe axa Ox) a marcajelor, așa că vor fi folosite ca poziții de plecare.

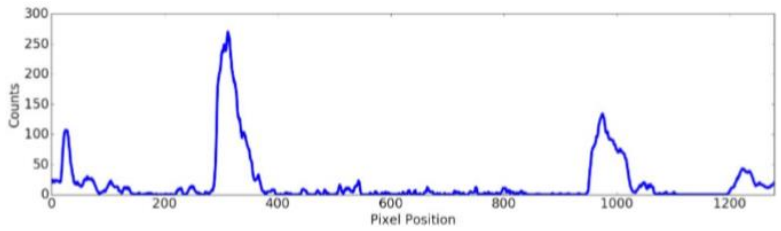

Figura 22. Histograma punctelor grupate pe coloane
din jumătatea inferioară a imaginii

Folosesc un „sliding window algorithm" pentru a indentifica zonele unde se află cele mai multe puncte de interes. Această tehnică folosește „ferestre" glisante pe oriozontală pentru încadra punctele de interes, aceste ferestre find generate una deasupra celelilalte de jos până sus. Pixelii aflați în interiorul unei ferestre sunt marcați ca pixeli de interes și sunt adăugați în lista de puncte pentru marcaj. Cu ajutorul mediei pixelilor din fereastra curentă putem oferi o indicație bună pentru punctul de bază a ferestrei de mai sus. Acest lucru se repetă până când ajungem la în capătul superior al imaginii. În acest fel, acumulăm toți pixelii de interes necesari pentru utilizarea funcției polyfit(). Putem calcula raza de curbură a fiecărui marcaj de banda după ce cunoaștem ecuațiile curbei cu ajutorul următoarei formule:

$$R_c = \frac{\left[1+\left(\frac{dx}{ddy}\right)^2\right]^{\frac{3}{2}}}{\left|\frac{d^2x}{dy^2}\right|}, \text{ unde } f(y) = Ay^2 + Bx + C$$

Pentru estimarea poziției vehiculului, se calculează lățimea benzii în pixeli în partea de jos a imaginii, care este cea mai apropiată de cameră. Sa presupunem că lățimea este de 1000 de

pixeli, atunci înseamnă că fiecare pixel corespunde la 3.7 metri pe scara reală. Apoi, se poate calcula decalajul dintre marcaje față de centrul imaginii.

Am creat clasa CurveCalculation care să ofere toate aceste informații, ea funcționând pe baza principilor explicate mai sus. La primirea unei imagini binare de forma celei din Figura 21, clasa va returna atât valorile razelor de curbură, cât și estimarea poziției vehiculului.

```
REAL WORLD: left fit curve parameters: [ 4.74348611e-04 -2.65924759e-02  2.20001476e+00]
REAL WORLD: right fit curve parameters: [ 2.81413819e-04 -2.71999855e-02  5.74662125e+00]
--------------------------------------------------------------------
PIXEL: left best-fit curve parameters: [ 1.55801439e-04 -2.09625373e-01  4.16219009e+02]
PIXEL: right best-fit curve parameters: [ 9.24313406e-05 -2.14414300e-01  1.08719861e+03]
--------------------------------------------------------------------
LEFT: current radius of curvature: 1054.082603437538 m
RIGHT: current radius of curvature: 1777.0264661826368 m
--------------------------------------------------------------------
VEHICLE POSITION: 0.12m right of center.
```

Figura 23. Rezultatele obținute în urma calculelor

Chiar mai mult, va returna o imagine în care este explicat vizual felul cum algoritmul de glisare a ferestrelor își alege poziția ferestrei în imagine.

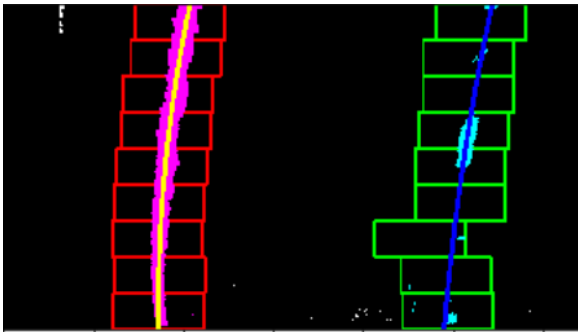

Figura 24. Reprezentarea ferestrelor glisante

3.1.5 Proiectarea benzii în perspectiva normală

Datorită faptului că anterior am obținut parametrii cuburiilor marcajelor, putem folosi funcția predefinită fillPoly() pentru a crea o imagine în care să evidențiem zona dintre marcaje, altfel spus, pentru a crea o imagine care să conțină banda însăși. După ce banda a fost scoasă în evidență, readucem imaginea în perspectiva normală (vezi secțiunea 3.1.2). Tot ce mai avem de făcut este să suprapunem imaginea originală nedistorsionată cu imaginea de evidențiere a benzii, iar rezultatul este următorul:

Figura 25. Rezultatul obținut în urma evidențierii benzii

3.2 Detectarea mașinilor

Detectarea mașinilor în timpul șofatului poate fi utilă pentru că reduce riscul accidentelor prin carambol și a contactelor produse prin neatenție. Acest prototip oferă o detecție bazată pe recunoașterea autovehiculelor prin utilizarea camerei frontale.

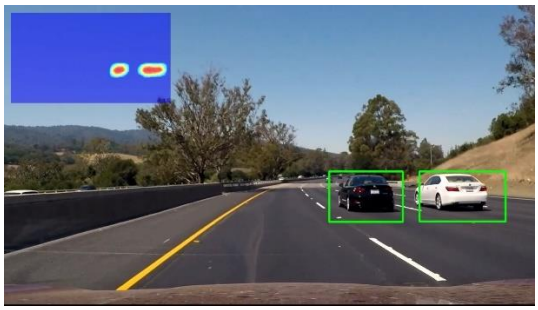

Figura 26. Exemplu de detecție a mașinilor

Prototipul folosește „deep learning" pentru recunoașterea autovehiculelor în imagine. Mai specific, utilizează o rețea neuronală convoluțională. Însă, scopul final nu este doar de a detecta prezența unui autovehicul în imagine, ci și poziția acestuia. Din fericire, rețele neuronale convoluționale sunt potrivite pentru realizarea acestui lucru.

În principiu, problema de clasficare este una binară (vehicul / non-vehicul), deci modelul ar putea fi construit astfel încat să fie antrenat pe imagini de dimensiuni relativ mici (de exemplu 64x64) și stratul rezultat să fie de forma „one_hot" cu dimensiune 1x1, care să reprezinte probabilitatea imaginii în clasificarea aleasă (cât de probabil este să fie vehicul și cât de probabil este sa nu fie). Dupa crearea modelului, diminesiunile imaginii de intrare pot fi extinse, transformând dimensiunile stratului rezultat dintr-un strat 1x1 într-o „hartă" de probabilități pentru zonele imaginii.

Această detecție poate fi realizată urmând următorii pași:

1. Crearea modelului – se creează o rețea neronală convoluțională antrenată pe imagini de dimensiuni mici, de exemplu 64x64
2. Împărțirea imaginii și detectarea în părți – imaginea originală este împărțită în mai multe fragmente mici, iar pentru fiecare dintre el se va verifica probabilitatea unui vehicul.
3. Crearea hărții de predicții – toate fragmentele obținute anterior, cu probabilitatea lor, sunt îmbinate sub aceași formă ca în imaginea inițială
4. Extragerea chenarelor și crearea hărții de „temperatură"– se realizează utilizând harta de predicții prin etichetarea zonelor unde s-a identificat un vehicul
5. Gruparea chenarelor – se grupează mai multe chenare păstrate într-un istoric al succesiunii imaginilor, este necesară pentru o acuratețe mai bună a detecției

3.2.1 Crearea modelului

Rețeaua neuronală convoluțională abordată pentru detecția de autovehicule este următoarea:

Strat	Dimensiunea de ieșire
Lambda	(None, 64, 64, 3)
Conv2D	(None, 64, 64, 128)
Dropout	(None, 64, 64, 128)
Conv2D	(None, 64, 64, 128)
Dropout	(None, 64, 64, 128)
Conv2D	(None, 64, 64, 128)
MaxPooling2D	(None, 8, 8, 128)
Dropout	(None, 8, 8, 128)
Conv2D	(None, 1, 1, 128)
Dropout	(None, 1, 1, 128)
Conv2D	(None, 1, 1, 1)

Setul de date utilizat pentru antrenarea acestei rețele neuronale a fost cel oferit de către „KITTI vision benchmark suite” [28], Acest set ne oferă 17760 de imagini, din care 8968 în care nu se găsesc mașini și 8792 în care se găsesc masini, toate find de dimensiunea 64x64 și color. Vom eticheta imaginile pentru a obține două mari clase: vom eticheta cu 1.0 toate imaginile în care sunt descoperite mașini și cu 0.0 toate imaginile care nu conțin mașini.

Figura 27. Etichetarea imaginilor

Acest total de 17760 se poate dubla, adăugând o rotire pe verticală tuturor imaginilor. Setul este împărțit astfel: 80% datele de antrenare, 4% datele de validare și 16% datele de test. Imaginile sunt alese aleator fiecarui set pentru preveni obținerea rezultatelor eronate.

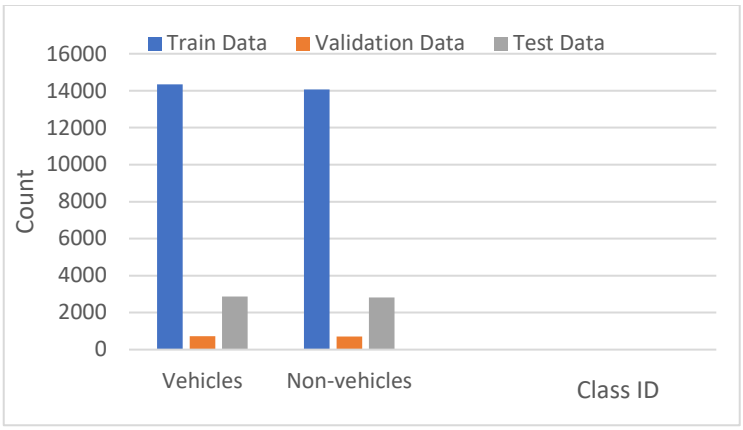

Figura 28. Graficul împărțirii setului de date

În urma antrenării rețelei, rezultatele obținute au fost satisfăcătoare, deoarece am obținut o valoare de 99.2% al acurateții și o valoare a funcții „loss function” de 0.0063.

```
Epoch 5/5
607/607 [==============================] - 48s - loss: 0.0063 - acc:
0.9923 - val_loss: 0.0073 - val_acc: 0.9926

Evaluating accuracy on test set.
test accuracy:  [0.0065823850340600764, 0.99373970345963758]
```

Figura 29. Rezultatul antrenării rețelei

3.2.2 Obținerea zonei de interes și calcularea hărții de predicții

Deoarece prototipul este realizat pentru un anumit videoclip, în continuare vom considera următoarea imagine pentru testare:

Figura 30. Exemplu de imagine fără distorsiuni

În Figura 30 se poate observa că doar într-o anumită zonă a imaginii se pot identifica vehicule, numim acea zonă ca find zona de interes, iar pentru acea zonă vom realiza segmentarea.

Figura 31. Zona de interes pentru detecție

În urma aplicării modelului asupra zonei de interes, stratul rezultat se va transforma din (?,1,1,1) în (?,25,153,1), unde 25 și 153 reprezintă înălțimea si lungimea hărții miniaturate de predicții, care vor fi proiectate în cele din urmă pe imaginea originală.

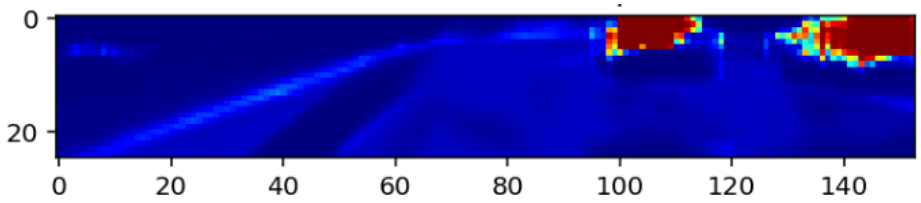

Figura 32. Harta de detecție
obținută prin aplicarea modelului

Pentru a obține o clasificare eficientă, trebuie să impunem un prag peste care să considerăm că predicția este corectă. În cazul de față, pragul ales este 0.7. Asta înseamnă că dacă predicția în imagine este mai mare ca 0.7 din 1, atunci considerăm că acolo regăsim o masină. Aplicăm acest prag hății obținute pentru a obține o imagine binară cu zonele unde se află mașini.

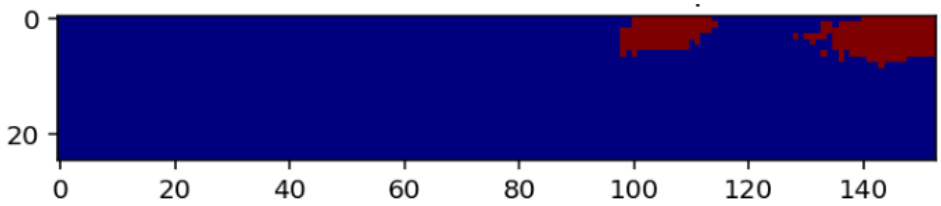

Figura 33. Harta de detectare
în urma aplicării pragului

3.2.3 Extragerea chenarelor și crearea hărții de „temperatură”

Pentru etichetarea zonelor unde s-a detectat un vehicul vom folosi metoda label() oferită de scipy.ndimage.measurements. Acest pas permite evidențierea marginilor etichetelor care, la rândul lor, vor ajuta la păstrarea fiecărui chenar în limitele dimensiunii lui în momentul construirii hărții de temperatură.

După extragerea etichetelor, ele vor fi proiectate în spațiul de coordonate al imaginii originale prin transformarea fiecărui punct intr-un chenar de dimensiune 64x64.

Figura 34. Proiecția chenarelor pe imaginea originală

Harta de temperatură se realizează prin supapunerea tuturor chenarelor extrase anterior. După ce obținem harta de temperatură, o etichetăm pentru a obține chenarele finale, care pentru exemplul dat, vor fi două ca număr.

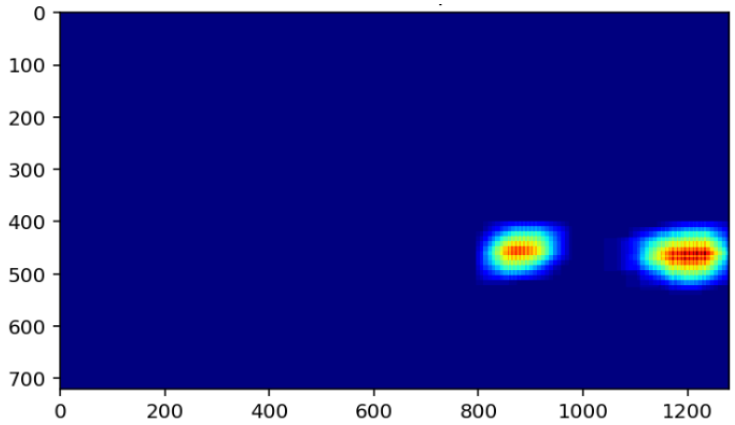

Figura 35. Harta de temperatură

3.2.4 Gruparea chenarelor

Acest lucru se realizează ca o asigurare în plus că în zona prezisă de model există un vehicul. Un videoclip presupune o serie continuă de imagini, astfel pentru fiecare imagine extragem un anumit număr de chenare (zero, unul sau două în videoclipul nostru). Ceea ce presupune de fapt gruparea chenarelor, este memorarea unui istoric de chenare pentru fiecare imagine, iar în momentul când atinge pragul chenarelor necesare pentru o incadrare (prag decis de către noi, în cazul de față am ales 10) atunci putem să le grupăm și astfel rezultatul grupării lor va fi chenarul unde se află vehiculul.

Pentru gruparea chenarelor folosim funcția oferită de către OpenCV numită groupRectangles(), care primește ca date de intrare o listă de chenare, o valoare minimă de chenare și o diferență maximă acceptată dintre locațiile lor. Dacă valoarea minimă de chenare este mai mare decât dimensiunea listei, atunci funcția nu va returna nimic.

Figura 36. Rezultatul în urma grupării chenarelor

3.3 Arhitectura și utilizarea aplicației

Aplicația este compusă din două părți independente (car_detection și lane_detection), care sunt utilizate conform umrătoarei diagrame:

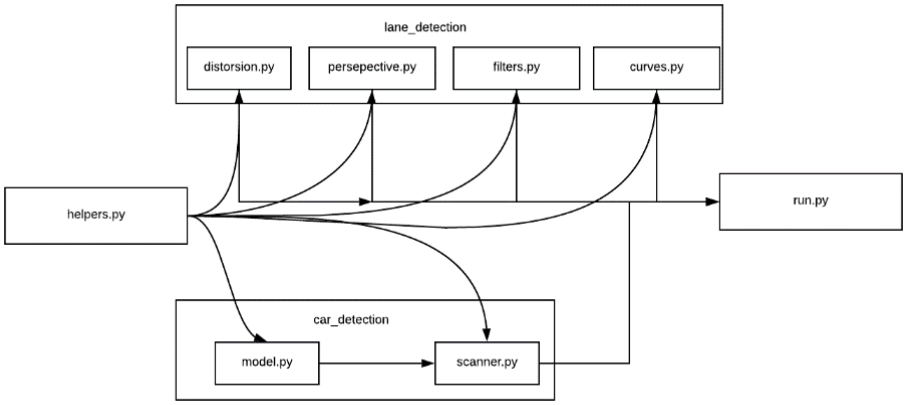

Figura 37. Diagrama de dependințe

Datorită faptului că ele coexistă independent, pot apărea îmbunătățiri la nivelul fiecărei părți, fără a o afecta pe cealaltă. De exemplu, dacă detecția de mașini prin rețele neuronale ar fi înlocuită cu alt tip de detecție (cum ar fi HOG + SVM), atunci detecția de linii nu ar fi afectată. De asemena, pot fi adăugate noi funcționalități, cum ar fi recunoașterea semnelor de circulație, fără a fi nevoie să modificăm funcționalitățile deja existente.

Programul principal al aplicației este dat de run.py, iar în momentul apelului este necesară oferirea unui video de intrare care mai apoi va fi procesat pentru a obține videoclipul rezultat.

4 Concluzie

Aplicația prezentată oferă soluții pentru diminuarea problememlor care pot apărea din cauza neatenției șoferilor. În timpul dezvoltării sale au apărut momente în care decizile privind implementarea sa au fost dificile. De exemplu, pentru detecția benzii de circulație există o vastă arie de metode, cum ar fi rețelele neuronale, însă metodele care implică detecția bazată pe învățare aprofundată necesită un set de date de antrenament de dimensiuni mari. Chiar dacă detecția bazată pe procesarea imaginii necesită convenții legate de tipul de videoclip procesat (de exemplu obiectivul camerei de filmat) și alte convenții legate de drumul pe care circulă mașina (culorile marcajelor), acest tip de detecție este eficient, însă nu e rapid computațional.

Un alt exemplu, pentru detecția de mașini există o mulțime de alte metode prin care se poate obține același rezultat, cum ar fi detecția bazată pe utilizarea clasificatorului SVM în combinație cu funcțiile HOG, însă sunt de părere că utilizarea rețelelor neuronale convoluționale este mult mai potrivită pentru această sarcină.

De asemena, aplicația poate fi îmbunătățită deoarece oferă această posibilitate prin construcția sa. O posibilă funcționalitate următoare ar fi calcularea distanței dintre mașini, care să ofere o avertizare în cazul în care mașina detectată se află prea aproape. O altă funcționalitate posibilă următoare poate fi integrarea detecției semnelor ce circulație, care să ofere informații în momentul în care un semn este detectat.

În final, doresc să închei cu un citat a lui Elon Musk, care spune „Aș putea ori să observ cum tehnologia evoluează, ori să fac parte din evoluția sa.”. Așadar, chiar dacă aplicația nu este din practică în acest moment, poate fi îmbunătățită până în acel punct și consider că este un start promițător pentru a obține rezultate utilizabile practic.

5 Referințe

1. Abdel-Aziz, Y. I. & Karara, H. M. (1971) Direct linear transformation into object space coordinates in close-range photogrammetry. Proc. Symposium on Close-Range Photogrammetry, Urbana, Illinois, p. 1 – 18
2. Faig, W. (1975) Calibration of close-range photogrammet- ric systems: Mathematical formulation. Photogrammetric Engineering and Remote Sensing 4l(12): 1479-1486.
3. Fukushima, K. Neocognitron: A hierarchical neural network capable of visual pattern recognition. Neural networks 1, 2 (1988), 119–130.
4. Heikkila, J. & SilvCn, 0. (1996) Calibration procedure for short focal length off-the-shelf CCD cameras. Proc. 13th Intemational Conference on Pattem Recognition. Vienna, Austria, p. 166- 170.
5. Melen, T. (1994) Geometrical modelling and calibration of video cameras for underwater navigation. Dr. ing thesis, Norges tekniske hflgskole, Institutt for teknisk kybemetikk
6. Bishop, C. M. Pattern Recognition and Machine Learning (Information Science and Statistics). Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.
7. Slama, C. C. (ed.) (1980) Manual of Photogrammetry, 4th ed., American Society of Photogrammetry, Falls Church, Virginia.
8. Tsai, R. Y. (1987) A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off- the-shelf TV cameras and lenses. IEEE Joumal of Robotics and Automation RA-3(4): 323-344.
9. Fukushima, K. Neocognitron: A hierarchical neural network capable of visual pattern recognition. Neural networks 1, 2 (1988), 119–130.Weng, J., Cohen, P. & Herniou, M. (1992) Camera calibration with distortion models and accuracy evaluation. IEEE Transactions on Pattem Analysis and Machine Intelligence PAMI-14(IO): 965-980.
10. Janne Heikkila and Olli Silven A Four-step Camera Calibration Procedure with Implicit Image Correction. FTN-90570 Oulu, Finland
11. Çelik, Tantek, Chris Lilley, and L. David Baron. "CSS Color Module Level 3." (2012).
12. . Schwarz, Michael W., William B. Cowan, and John C. Beatty. "An experimental comparison of RGB, YIQ, LAB, HSV, and opponent color models." ACM Transactions on Graphics (TOG) 6.2 (1987): 123-158
13. "Color: changing the color scheme of certain themes" (2014). Drupal Color module, included as core module in versions 6.x/7.x/8.x. Retrieved from https://www.drupal.org/documentation/modules/col
14. Matt Wiebe, "New color picker in WP 3.5" (2012). Retrieved from https://make.wordpress.org/core/2012/11/30/newcolor-picker-in-wp-3-5/
15. Irwin Sobel, 2014, History and Definition of the Sobel Operator
16. Jacobs, David. "Image gradients." Class Notes for CMSC 426 (2005)
17. Felzenszwalb, P. F., and Huttenlocher, D. P. Efficient graphbased image segmentation. International journal of computer vision 59, 2 (2004), 167–181.

18. Hoiem, D., Efros, A. A., and Hebert, M. Geometric context from a single image. In Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on (2005), vol. 1, IEEE, pp. 654–661.
19. Long, L. N., and Gupta, A. Scalable massively parallel artificial neural networks. Journal of Aerospace Computing, Information, and Communication 5, 1 (2008), 3–15.
20. Goodfellow, I., Bengio, Y., and Courville, A. Deep Learning. MIT Press, 2016. http://www.deeplearningbook.org.
21. Walther, D., Itti, L., Riesenhuber, M., Poggio, T., and Koch, C. Attentional selection for object recognition - a gentle way. In International Workshop on Biologically Motivated Computer Vision (2002), Springer, pp. 472–479.
22. Dalal, N., and Triggs, B. Histograms of oriented gradients for human detection. In Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on (2005), vol. 1, IEEE, pp. 886–893.
23. Girshick, R., Donahue, J., Darrell, T., and Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition (2014), pp. 580–587.
24. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. Ssd: Single shot multibox detector. In European Conference on Computer Vision (2016), Springer, pp. 21–37.
25. Uijlings, J. R. R., van de Sande, K. E. A., Gevers, T., and Smeulders, A. W. M. Selective search for object recognition. International Journal of Computer Vision 104, 2 (2013), 154–171.
26. Springenberg, J. T., Dosovitskiy, A., Brox, T., and Riedmiller, M. A. Striving for simplicity: The all convolutional net. CoRR abs/1412.6806 (2014).
27. Ren, S., He, K., Girshick, R., and Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. In Advances in neural information processing systems (2015), pp. 91–99.
28. http://www.cvlibs.net/datasets/kitti/
29. D. Hearn, & M. Baker (1997). Computer Graphics, C Version. Englewood Cliffs: Prentice Hall], chapter 9