

Metode de detecție a trecerilor de pietoni

LUCRARE DE DISERTAȚIE

Absolvent masterand:

Andreea-Simona GRIGOROVICI

Coordonator științific:

Conf. dr. ing. Florin ONIGA

Iulie 2019

Cuprins

Capitolul 1. Introducere – Contextul proiectului.....	1
1.1. Motivație.....	1
1.2. Scopul și obiectivele cercetării.....	1
1.3. Conținutul lucrării.....	2
Capitolul 2. Studiu bibliografic.....	4
2.1. Metode existente în literatură.....	4
Capitolul 3. Analiză, proiectare, implementare.....	9
3.1. Fundamentare teoretică.....	9
3.1.1. Metoda geometrică de detecție.....	9
3.1.1.1. Canny.....	9
3.1.1.2. Transformata Hough.....	10
3.1.1.3. Vanishing Point.....	13
3.1.1.4. Preprocesarea prin filtrare folosind operații morfologice.....	16
3.1.1.5. Filtrarea liniilor după proprietăți.....	18
3.1.2. Metoda ce folosește rețele neuronale.....	19
3.1.2.1. Mask R-CNN.....	21
3.2. Proiectarea și implementarea soluției.....	24
3.2.1. Arhitectura generală a sistemului de detecție.....	24
3.2.2. Metoda geometrică.....	25
3.2.2.1. Diagrama de structură a metodei propuse.....	25
3.2.2.2. Detalii de implementare.....	26
3.2.2.3. Canny.....	26
3.2.2.4. Transformata Hough.....	27
3.2.2.5. Vanishing point.....	28
3.2.2.6. Operații morfologice.....	29
3.2.2.7. Filtrare linii după proprietăți.....	31
3.2.3. Metoda ce folosește rețele neuronale.....	32
3.2.3.1. Diagrama de structură a metodei propuse.....	32
3.2.3.2. Detalii de implementare.....	33
3.2.3.3. Mask R-CNN.....	34
Capitolul 4. Rezultate experimentale.....	40
4.1. Descrierea setului de date de test.....	40
4.2. Timpi de execuție.....	41
4.3. Acuratețea detecției.....	42
Capitolul 5. Concluzii.....	47
5.1. Direcții de dezvoltare.....	48
Bibliografie.....	49

Capitolul 1. Introducere – Contextul proiectului

În zilele noastre, procesarea imaginilor joacă un rol esențial în numeroase domenii cum ar fi: fizica, optica, matematica și informatica. Vizualizarea imaginilor în domeniul astronomic, simularea chirurgicală simplă sau robotică, monitorizarea performanțelor sportive și detecția obiectelor sunt doar câteva aplicații ale tehnologiei de viziune artificială. Algoritmii de procesare a imaginilor stau la baza proceselor din care este constituit domeniul viziunii artificiale. Studiul proprietăților imaginilor și transformarea acestora constituie conceptul de bază utilizat în procesarea imaginilor. Prelucrarea informațiilor conținute într-o imagine este strâns legată de analiza, interpretarea și manipularea semnalelor. Imaginile se găsesc cel mai adesea sub forma unor fișiere în format binar. În cel mai simplu caz, când nu se aplică algoritmi de compresie, imaginile conțin câte o valoare sau un set de valori numerice pentru fiecare pixel. Aceste valori sunt utilizate de programele de prelucrare a imaginilor pentru a identifica și compune culorile pixelilor corespunzători. Astfel, folosindu-se valorile obținute se pot identifica diferite informații din imaginile procesate. Un bun exemplu în care sunt utilizate aceste informații este reprezentat prin sistemele de conducere autonomă. Navigarea vehiculelor autonome implică cunoașterea și detecția unui set de entități particulare dintr-un flux de informații. Printre acestea se numără și localizarea marcajelor rutiere, a semnelor de circulație și a trecerilor de pietoni.

Trecerea de pietoni reprezintă un set uniform de linii albe, paralele pe o suprafață de culoare închisă (negru, nuanțe de gri sau uneori roșu). Deși există treceri de pietoni de diferite forme și structuri, lucrarea de față se axează pe cele ce au fiecare componentă în forma unui patrulater, sau specific pentru cazul trecerii în diagonală, în forma unui paralelogram [1].

1.1. Motivatie

Având în vedere evoluția tehnologică constantă, procesarea de imagini a devenit un domeniu din ce în ce mai aplicabil în diferite situații. Detecția obiectelor se poziționează în centrul de interes al dezvoltării tehnologice, datorită impactului pe care îl are în îmbunătățirea calității unui set foarte mare de servicii: atât în cadrul proiectelor educaționale, cât și într-un cadru tehnologic. Lucrarea de față se ocupă de detecția trecerilor de pietoni din perspectiva coducerii autovehiculelor. Astfel, apare ca domeniu principal de aplicabilitate: navigarea vehiculelor autonome, urmat de alte domenii precum: crearea unor aplicații de asistență pentru nevăzători sau pentru persoane cu handicap locomotor. Ne propunem să analizăm comparativ metode de detecția a trecerilor de pietoni, punând accentul în principal pe timpul de execuție, eficiența detecției și acuratețea rezultatelor. Timpul de detecție a obiectelor din imagine trebuie să fie cât mai mic pentru a se apropia de performanța de detecție în timp real.

Lucrarea analizează utilitatea modalităților de analiză a imaginilor în funcție de context, dar se prezintă doar tehnici ce folosesc fie metode clasice, geometrice, fie metode bazate pe construirea unor rețele neuronale în prezentarea posibilităților de detecție a trecerii de pietoni. Tehnicile analizate pot fi ulterior utilizate pentru a optimiza algoritmi și metode deja existente în literatură sau pentru a dezvolta noi sisteme utilizabile în domeniul navigării vehiculelor autonome.

1.2. Scopul și obiectivele cercetării

Detectia trecerilor de pietoni este un subiect important atât din perspectiva pietonilor, cât și a coducerii autovehiculelor. Din acest punct de vedere utilitatea îmbunătățirii metodelor existente sau creării unor metode noi se justifică prin importanța siguranței participanților la trafic. Având în vedere importanța acestei teme, dar și interesul crescut în ceea ce privește

domeniul viziunii artificiale, s-a decis abordarea unui subiect de analiză a două metode de detectie a trecerilor de pietoni.

Scopul principal al lucrării este dezvoltarea și compararea a două soluții de detectare a prezenței trecerilor de pietoni în imaginea urmărită și de a decide care este metoda cea mai potrivită din perspectiva unui studiu de caz aplicativ **în domeniul navigării vehiculelor autonome**. Deși metodele dezvoltate nu sunt originale, acestea sunt implementate urmărind pași algoritmici și aspecte de proiectare adaptate pentru cazul aplicativ menționat. Astfel, această lucrare vine în sprijinul celor ce doresc să propună o nouă modalitate de detectare a trecerilor de pietoni, dar poate fi folosită și ca suport pentru îmbunătățirea metodelor existente din perspectiva acurateței și a eficienței.

Obiectivele principale ale lucrării de față sunt următoarele:

- implementarea a două metode de detectie a trecerilor de pietoni, diferite din punct de vedere conceptual, care pot servi drept soluții pentru cazul aplicativ propus;
- analiza comparativă a două metode principale de detectie a trecerilor de pietoni: metoda clasică, geometrică și metoda ce utilizează rețele neuronale (Mask R-CNN);
- evaluarea metodelor de detectie a trecerilor de pietoni în contextul aplicațiilor de navigare a vehiculelor autonome.

Pe lângă aceste obiective principale, ne propunem și să folosim diferite concepte din literatura de specialitate (e.g. Transformata Hough, Vanishing Point, Canny, Mask R-CNN, operații morfologice, adnotarea imaginilor etc.) pentru a implementa cele două metode de detectie a trecerilor de pietoni. Pentru achiziționarea secvențelor de imagini color utilizate în aplicație se folosește o cameră simplă, de la telefon.

1.3. Conținutul lucrării

Această lucrare poate fi utilizată ca suport în găsirea de soluții fezabile pentru dezvoltarea unor aplicații ce detectează treceri de pietoni corect și eficient, dar și pentru îmbunătățirea unor soluții deja existente.

Lucrarea este împărțită în patru părți, fiecare detaliind informații specifice domeniului abordat.

Primul capitol conține o introducere în domeniu motivând în acest mod alegerea și studiul temei. Tot aici sunt prezentate în linii generale contextul problemei abordate în cadrul unui cuprins mai extins. De asemenea se prezintă scopul lucrării, obiectivele și particularitățile abordării acestei teme prezentate sumar.

Capitolul 2, denumit studiu bibliografic are ca scop fixarea referențialului în care se situează tema, prezentarea cercetărilor similare și raportarea lucrării date în contextul viziunii artificiale. Astfel, se prezintă principale metode existente în literatură și care au fost utilizate ca punct de start pentru realizarea lucrării.

Capitolul 3 cuprinde prezentarea principalelor concepte teoretice, detalii la nivel de proiectare, structura generică a aplicației implementate și detalii de implementare ale conceptelor prezentate.

Ultimile două capitole reprezintă un rezumat a contribuțiilor lucrării de față în domeniul abordat. Capitolul 4 conține exemplificarea rezultatelor obținute din perspectiva timpului și al acurateței detectiei. Capitolul 5 conține o analiză critică a rezultatelor obținute, incluzând avantajele și dezavantajele rezultatelor și metodelor folosite. De asemenea, sunt descrise posibilele dezvoltări și îmbunătățiri ce pot fi implementate ulterior. Rezultatele obținute prin metodele implementate sunt evaluate atât din punct de vedere al acurateței, cât și din punct de

vedere al timpului de execuție, **din perspectiva unui studiu de caz aplicativ în domeniul navigării vehiculelor autonome.**

În cadrul acestei lucrări sunt exemplificate etapele parcurse și metodele implementate, atât din punct de vedere teoretic, al proiectării și al implementării, cât și din punct de vedere al rezultatelor obținute.

Capitolul 2. Studiu bibliografic

În ultimii ani au fost realizate numeroase studii în ceea ce privește semnele de circulație de pe stradă, marcajele de atenționare pentru șoferi, mașinile, intersecțiile, comportamentul pietonilor etc. Câteva exemple de astfel de proiecte sunt:

- recunoașterea automată a plăcuței de înmatriculare a unui autovehicul,
- aplicații de recunoaștere a semnelor de circulație,
- aplicații ce ajută la verificarea stării traficului la un moment dat,
- detecția prezenței pietonilor pe carosabil.

În ceea ce privește detectarea trecerilor de pietoni studiile făcute pe acest subiect se împart în mai multe categorii în funcție de tipul imaginii și domeniul de utilizare al algoritmului folosit. Imaginile utilizate pentru procesare pot fi imagini color, imagini cu nivele de gri, imagini ce conțin informații de adâncime, imagini ce conțin informații tri-dimensionale etc. În cadrul acestui capitol sunt prezentate o serie de metode propuse în literatura de specialitate pentru detectarea trecerilor de pietoni.

2.1. Metode existente în literatură

Detectia trecerilor de pietoni într-o secvență de imagini apare deseori ca subiect principal în literatura de specialitate. Numeroase studii se îndreaptă spre acest domeniu fie pentru a îmbunătăți metode deja existente, fie pentru a dezvolta metode noi, mai eficiente și cu rezultate mai bune.

Stephen Se [2] a propus o metodă de detecție a trecerii de pietoni bazându-se pe proprietatea paralelismului în spațiul 3D al dreptelor din care aceasta se compune. Figura 2.1 prezintă patru exemple de rezultate obținute de autor prin aplicarea acestei metode.



(a)



(b)



(c)



(d)

Figura 2.1: Rezultate obținute bazat pe proprietăți de paralelism [2].

Această metodă are avantajul de a putea fi extinsă și pentru detecția scărilor din imagini. Mai întâi se detectează muchiile și liniile concurente din imagini ce ulterior sunt grupate pe criterii geometrice (i.e., paralelism). Se elaborează trei metode pentru estimarea poziției: o metodă de căutare homografică utilizând un model a priori, o altă metodă pentru găsirea normalei folosind linia de dispariție calculată de la linii egal distanțate și o treia metodă ce utilizează două puncte de dispariție. Pentru a calcula linia de dispariție se utilizează algoritmul RANSAC (Random Sample Consensus), adaptat pentru problema definită. Acest algoritm este folosit pe scară largă pentru a estima un model matematic pornind de la un set de date predefinite ce conține atât date potrivite, cât și date considerate a fi aberante. Ulterior, dreptele finale ce fac parte din trecerile de pietoni sunt extrase utilizând diferențele de variații ale intensității pixelilor din imagine.

Mergând în aceeași direcție, Wang et. al., [3] au propus o metodă de detectare a trecerilor de pietoni din imagini folosind forma, culoarea și adâncimea informațiilor recepționate. Imaginile utilizate sunt preluate folosind o cameră RGB-D (Red Green Blue-Depth)¹. Această metodă folosește de asemenea transformata Hough, dar și informații de culoare pentru a extrage dreptele paralele. Ulterior, se folosește de informația de adâncime preluată de la camera utilizată pentru a recunoaște trecerile de pietoni finale și scările din imagini. Pentru a realiza acest pas se extrage caracteristica unidimensională ale informațiilor de adâncime în funcție de direcția celei mai lungi linii detectate din imaginea de adâncime. Informațiile cu privire la orientare și poziție sunt calculate pentru ca apoi, această caracteristică să fie utilizată ca intrare pentru un clasificator bazat pe SVM (Support Vector Machine, algoritm de învățare supervizată) pentru a realiza o diferențiere între trecerile de pietoni și scările detectate. Framework-ul propus de acești autori are o rată de acuratețe ridicată (peste 90%), având totuși dezavantajul de a nu putea fi folosit fără posibilitatea de a obține o imagine de adâncime ca element de intrare. Figura 2.2 conține un exemplu de linii detectate pentru o imagine oferită ca parametru de intrare.

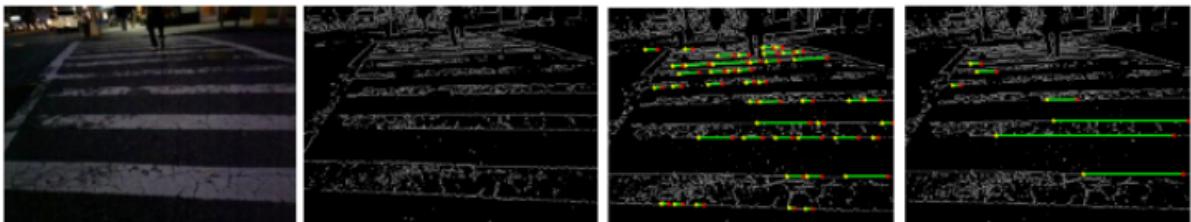


Figura 2.2: Liniile detectate prin metoda propusă de Wang [3].

Aceste metode funcționează foarte bine atunci când imaginile sunt clare, lipsite de zgomot, componentele sunt lizibile și nu conțin un număr mare de elemente asemănătoare. Pe de altă parte, transformata Hough este instabilă ca etapă de preprocesare în condițiile întâlnite în lumea reală (e.g., scene aglomerate cu umbre, linii curbate, efecte de saturatie și lizibilitate redusă datorită condițiilor de mediu).

Coughlan et. al., [4] au dezvoltat o metodă pentru a detecta trecerile de pietoni bazându-se pe segmentarea imaginilor. Aceasta se bazează pe faptul că trecerile de pietoni pot fi privite ca o textură: un aranjament de elemente similare, cvasi-periodic. Elementele din imagine sunt extrase și marcate ca obiecte sau ca parte din calea de rulare. Ulterior, se construiește într-un model grafic pentru a grupa caracteristicile geometrice într-o structură coerentă, astfel încât caracteristicile pozitive tind spre a se organiza în structuri coerente, iar cele nedeterminate sunt distribuite aleatoriu. Pentru evaluarea structurilor deja create, din punct de vedere probabilistic, modelul construit este de tip MRF (Markov Random Field). În domeniul probabilităților modelul este reprezentat ca un model grafic ce deține proprietăți Markov fiind descris ca un graf

¹ RGB-D sunt cele 3 canale principale din imagini color (i.e., roșu, verde, albastru) și adâncimea.

unidirecțional. Acest model grafic este similar unei rețele Bayesiene în reprezentarea dependențelor. Diferențele sunt în faptul că rețelele Bayesiene pot fi direcționate și aciclice, în timp ce rețelele Markov sunt nedirecționate și aciclice. Astfel, o rețea Markov poate reprezenta dependențe ciclice, însă nu poate reprezenta dependențe induse. Spre deosebire de alte metode asemănătoare, această metodă maximizează utilizarea informațiilor geometrice, scăzând aportul adus de informațiile de intensitate. Această abordare aduce un plus în momentele în care condițiile de iluminare sunt slabe. Figura 2.3 conține o serie de exemple de rezultate obținute aplicând metoda descrisă.



Figura 2.3: Rezultate obținute prin metoda propusă de Coughlan [4].

Ivanchenko et. al., [5] au îmbunătățit algoritmii existenți pentru a detecta localizarea și orientarea trecerii de pietoni pentru persoanele nevăzătoare utilizând o cameră a unui telefon mobil. Acest algoritm poate fi folosit de un public mult mai larg datorită posibilității utilizării telefonului mobil. Prototipul sistemului propus poate lucra în timp real pe Nokia N95, dar poate fi extins spre a putea fi folosit și cu alte dispozitive. Telefonul preia în mod automat câteva imagini pe secundă, analizând fiecare imagine într-o fracțiune de secundă și răspunde cu un semnal sonor dacă detectează trecerea de pietoni. Performanțele aplicației și posibilitatea de a lucra în timp real pe telefonul mobil, a cărui resurse de calcul sunt limitate în comparație cu platformele de tip desktop utilizate în mod obișnuit în domeniul viziunii artificiale, este

realizabilă datorită folosirii Symbian C++² ca limbaj de programare. Algoritmul implementat constă în două etape: inițial se realizează extragerea caracteristicilor din imagini (i.e., se determină segmente de dreaptă) și ulterior se folosește segmentarea imaginilor pentru a eticheta fiecare caracteristică. De asemenea, folosind metode de învățare supervizată, sistemul dezvoltat este demonstrat experimental de către autori ca fiind fezabil și performant. Figura 2.4 conține două exemple de rezultate obținute prin aplicarea metodei propuse.

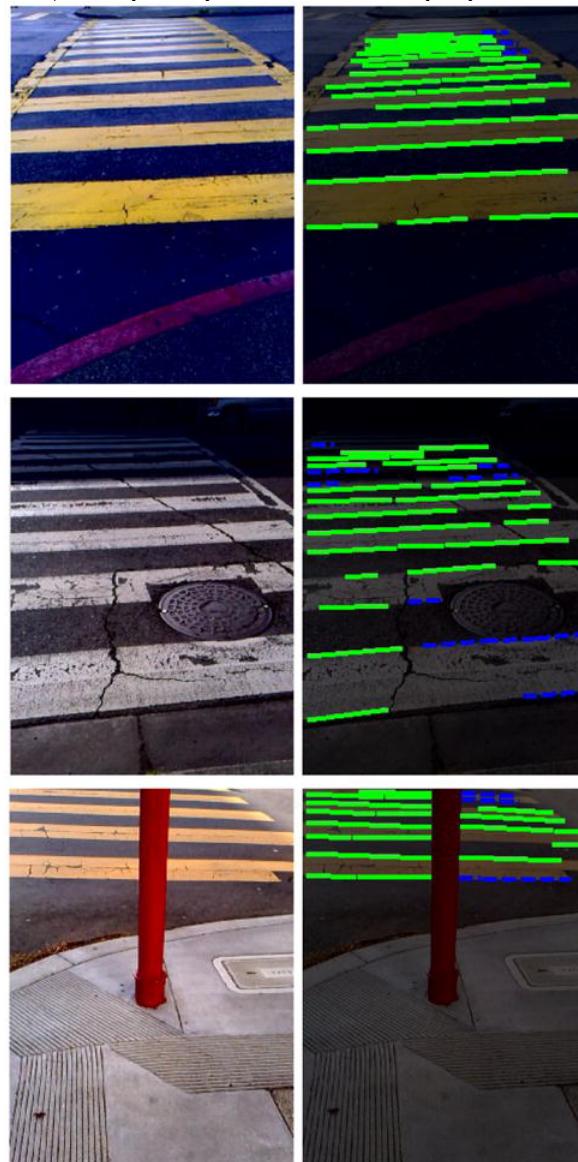


Figura 2.4: Rezultate obținute prin aplicarea metodei propuse de către Ivanchenko [5].

Radvanyi et. al., [6] merge mai departe, propunând utilizarea ochelariilor bionici pentru a furniza nevăzătorilor sau indivizilor cu deficiențe de vedere, orientarea bazându-se pe procesarea cularilor și îmbunătățirea lor. Ochelarii bionici, creați de oamenii de știință de la Oxford, folosesc niște camere minusculă și un calculator de buzunar prin care purtătorii lor sunt alertați de eventualele obiecte și obstacole din fața lor. De asemenea, detecția trecerilor de pietoni a fost realizată cu succes folosind algoritmul CNN (Convolutional Neural Networks) adaptiv, ce conține o rețea neuronală conlovuțională. Rețelele neuronale conlovuționale, datorită

² Symbian C++ limbaj de programare utilizat pe scară largă pentru dezvoltarea programelor pe platformele și sistemele de operare de pe telefoanele mobile.

arhitecturii ce conține perceptroni³ multistrat, aduc îmbunătățiri din punct de vedere al rezultatelor obținute pentru recunoaștere și detecție. În acest mod, utilizându-se o metodă nouă de segmentare ce oferă estimări mult mai bune pentru suprafața de transport, se obțin rate de recunoaștere ale trecerilor de pietoni mult mai mari, precum se poate observa în Figura 2.5.



Figura 2.5: Rezultate obținute prin metoda propuse de Radványi [6].

3 Perceptronul este un algoritm pentru învățarea supravegheată a clasificatorilor binari.

Capitolul 3. Analiză, proiectare, implementare

Acest capitol cuprinde structura generală a aplicației implementate, astfel pe parcursul acestui capitol sunt descrise toate modulele existente. De asemenea sunt prezentate principalele concepții teoretice, detalii la nivel de proiectare, tehnologiile utilizate și detalii de implementare ale conceptelor prezentate.

3.1. Fundamentare teoretică

În această lucrare ne vom axa pe detectarea trecerilor de pietoni nesemaforizate. Prin urmare, ne propunem pe lângă identificarea prezenței trecerilor de pietoni în imagini, și determinarea cât mai precisă a pixelilor din care se compun acestea. Pentru îndeplinirea acestui scop se va utiliza segmentarea semantică care reprezintă descompunerea imaginilor în părți componente, identificând astfel obiecte la nivelul pixelilor. Astfel, ne propunem realizarea unei comparații între o metodă clasică, geometrică, de detecție a liniilor candidate pentru a face parte din trecerea de pietoni și o metodă ce se bazează pe o ramură principală a inteligenței artificiale: rețelele neuronale.

3.1.1. Metoda geometrică de detecție

În acest subcapitol ne propunem detalierea metodei geometrice utilizate pentru a detecta trecerile de pietoni în imagini color.

Primul pas spre a detecta treceri de pietoni folosindu-se metoda geometrică este de a detecta dreptele din imagini. Imaginile ce conțin mai multe puncte de interes (i.e., muchii) pot fi prelucrate astfel încât din acestea să se poată identifica toate dreptele importante. Realizarea acestui lucru este posibil prin efectuarea a doi pași: detecția frontierelor folosindu-se Canny, și ulterior utilizarea transformatei Hough pentru detecția liniilor propriu zise. Următorul pas este de a filtra setul de linii detectat anterior. Acest pas se realizează folosindu-se vanishing point-ul⁴ și o serie de proprietăți geometrice ale liniilor detectate. Ulterior, pentru a reconstrui liniile care nu au fost detectate complet se utilizează operații morfologice precum dilatarea și eroziunea.

3.1.1.1. Canny

Orice imagine este caracterizată de prezența frontierelor. Detecția frontierelor este o problemă fundamentală în prelucrarea imaginilor, deoarece permite extragerea de informație utilă. Frontierele sunt zone de imagini unde există varianții bruse ale intensității. Acestea sunt obiecte subțiri sau contururi ale unor zone din imagine cu caracteristici de intensitate similară. Majoritatea detectoarelor de frontiere utilizează derivata I-a și derivata a-II-a a funcției imagine.

Detectorul de frontiere Canny este considerat a fi printre principalele detectoare de frontiere. De asemenea, dintre detectoarele cunoscute, acesta este cel mai optim, având multe avantaje față de detectoarele tradiționale. Canny [7] a propus definirea unui detector de frontiere care să aducă îmbunătățiri detectoarelor existente, din diferite puncte de vedere, cum ar fi:

- pentru scăderea ratei de eroare se dorește ca detectorul să nu piardă puncte de frontieră și să nu ia în considerare puncte care nu sunt puncte de frontieră (maximizarea raportului semnal zgomot);
- punctele de frontieră să fie bine localizate;
- distanța dintre punctele de frontieră detectate și cele reale să fie minimă;

⁴ Vanishing point din engl., poate fi tradus ca punct de dispariție.

- detectorul trebuie să furnizeze doar un singur răspuns pentru un singur punct de frontieră (eliminarea non-muchiilor).

Având în vedere aceste considerente, detectorul Canny realizează inițial o netezire a imaginii, pentru a elmina zgomotul, apoi calculează gradientul imaginii pentru a evidenția zonele cu variație mare a intensității. Calculul gradientului reprezintă măsura variațiilor din imagini. Următorul pas este de a parcurge aceste zone și de a elmina orice pixel care nu este detectat ca fiind maxim local pe direcția gradientului. Matricea gradientului este mai departe redusă prin metoda histerezis⁵. Ulterior sunt utilizate două praguri pentru a decide care dintre pixeli sunt detectați ca făcând parte din frontiere.

Algoritmul Canny este un algoritm foarte folosit pentru a detecta frontiere în imagini, acesta poate fi împărțit în 5 pași descriși în cele ce urmează [8].

Primul pas este realizat pentru a atenua zgomotul din imagini. Zgomotul în imagini este un semnal aleator care afectează informația utilă din acestea. Astfel, se aplică un filtru Gausian pentru a netezi imaginea. Un filtru poate fi specificat de o matrice ce poartă numele de mască de filtrare, caracterizată de formă, valorile coeficienților și de origine. Operația de filtrare este realizată suprapunând masca de filtrare peste fiecare pixel al imaginii originale, astfel încât originea măștii să coincidă cu pixelul considerat. Ulterior, se calculează toate produsele dintre coeficienții măștii și valorile pixelilor suprapuși, iar suma acestor produse reprezintă noua valoare a pixelului considerat. Această operație poartă numele de convoluție bidimensională. Cu cât lățimea măștii utilizată este mai mare, cu atât este mai mică sensibilitatea detectorului la zgomot. Totodată, eroarea de localizare a fronturilor crește odată cu lățimea măștii Gaussiene.

Al doilea pas realizează calculul amplitudinii gradientului. Astfel, pentru a se obține matricea amplitudinilor gradientului se aplică inițial operatorul Sobel imaginii rezultate din primul pas.

Următorul pas este utilizat pentru a obține matricea direcțiilor θ , calculând direcția gradientului în fiecare punct:

$$\theta(x, y) = \begin{cases} \arctan(Dy/Dx), Dx \neq 0 \\ 0, Dy = 0 \\ \pm 90, Dx = 0, Dy \neq 0 \end{cases} . \quad (1)$$

Cel de-al patrulea pas se folosește pentru a ajusta θ la direcția cea mai apropiată de valoarea sa din spațiul discret al imaginii.

În pasul 5 sunt eliminate pixelii care nu au amplitudinea maximă locală. Matricea amplitudinilor poate conține zone late în jurului frontierei deoarece operatorul Sobel produce mai multe puncte de front pentru același punct de frontieră. Astfel, sunt declarate puncte de frontieră acele puncte a căror amplitudine este maxima locală pe direcția gradientului, iar restul punctelor sunt eliminate. Efectul principal al acestui pas este de a subția frontieră, dar fără a o întrerupe.

Ultimul pas este utilizat pentru a elimina zgomotele fără a întrerupe contururile detectate. Cu toate că în primul pas s-a efectuat o filtrare a imaginii, matricea rezultată după pasul 5 încă mai conține zgomote care se manifestă ca pixeli cu contrast mai mic. Pentru a elmina zgomotul se utilizează operația histerezis care folosește două praguri pentru filtrare.

3.1.1.2. Transformata Hough

Atunci când se utilizează imagini în scopul recunoșterii de obiecte, este importantă reducerea cantității de informație, păstrând în același timp datele importante și caracteristice. Prin

⁵ Metoda histerezis este utilizată și pentru binarizarea imaginilor, folosindu-se două praguri, față de metoda clasică ce folosește un singur prag.

detectia dreptelor din imagini se reduce considerabil cantitatea de informatie. Transformata Hough a fost dezvoltata initial pentru recunoasterea dreptelor din imagini, iar ulterior a fost generalizata si pentru a recunoaste forme arbitrate. Desi transformata Hough se foloseste cel mai des pentru detectia dreptelor, ea poate fi folosita si pentru detectia curbelor mai complexe (i.e., cercuri) atat timp cat se foloseste o parametrizare adevarata [9].

In aceasta lucrare transformata Hough se foloseste cu scopul gasirii dreptelor intr-o imagine ce contine o multime de puncte de interes. Aceasta metoda a fost propusa si patentata de Peter Hough [10], cu scopul direct de a imbunatatiti metoda directa de calcul a dreptelor din fiecare pereche de puncte.

Metoda directa are o complexitate computațională mare, $O(n^2)$, ceea ce o face ineficientă de aplicat pentru un număr mare de puncte. În varianta sa inițială, transformata Hough, a fost o metoda de timp real pentru a număra câte puncte sunt plasate pe fiecare dreaptă posibilă dintr-o imagine. Dreptele pot fi reprezentate în formă pantă-termen liber:

$$y = a * x + b \quad . \quad (2)$$

Pentru fiecare punct de interes din imagine, se calculează toate dreptele posibile ce trec prin acest punct și se incrementează elementele din spațiul parametric, numit acumulator Hough. Dreptele căutate sunt astfel localizate în maximele locale ale spațiului Hough [11].

Aceasta reprezentare este sub-optimală deoarece nu este marginita, iar pentru a reprezenta toate posibilele drepte din imagine, panta și termenul liber trebuie să varieze în domeniul $-\infty$ și $+\infty$. De asemenea, prin aceasta formă, nu pot fi reprezentate dreptele verticale. Aceste probleme sunt rezolvate prin parametrizarea normală. Aceasta parametrizare consistă din reprezentarea dreptei prin vectorul ce trece prin origine și este perpendicular pe dreaptă, Figura 3.1. Astfel, Ecuatia (1) poate fi rescrisă, în formă polară, folosindu-se unghiul liniei, θ și distanța de la linie la origine, ρ :

$$\rho = x * \cos(\theta) + y * \sin(\theta) \quad . \quad (3)$$

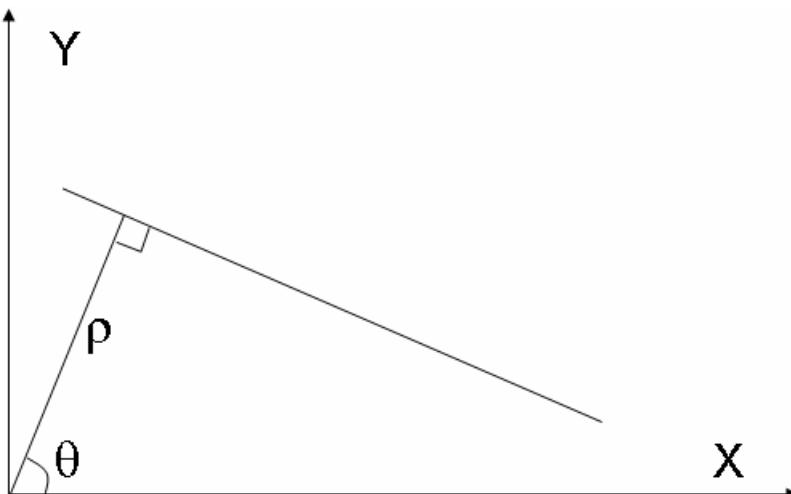


Figura 3.1: Vectorul perpendicular pe dreapta ce trece prin origine [8].

Orice dreaptă detectată în imagine poate fi reprezentată sub această formă polară atunci când $\theta \in [0^\circ, 180^\circ]$, iar $\rho \in \mathbf{R}$.

Pe de altă parte, imaginile au o dimensiune finită, astfel încât intervalul se restrâne. Astfel, prin introducerea cuantizării parametrilor se realizează descreșterea complexității computaționale. Valoarea maximă pentru ρ este diagonala imaginii ρ_{max} . În funcție de intervalul ales pentru θ , există mai multe configurații echivalente pentru domeniul parametrilor, însă cea

mai folosită este:

$$\theta \in [0^\circ, 180^\circ], \rho \in [-\rho_{max}, \rho_{max}] . \quad (4)$$

Unei drepte din planul imaginii îi corespunde un punct în spațiul Hough, aşa cum este reprezentat în Figura 3.2. De asemenea, reciproc, unui punct din planul imaginii îi corespunde o linie în spațiul Hough.

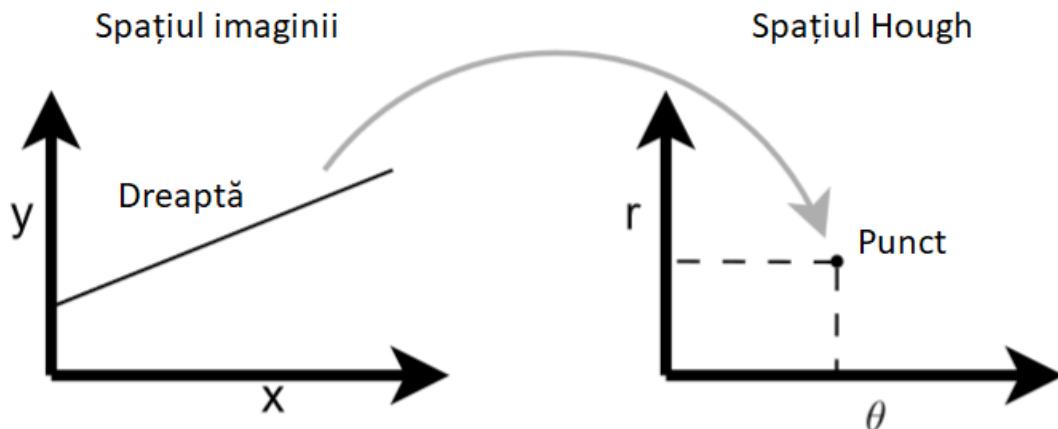


Figura 3.2: Maparea unei drepte din planul imaginii în spațiul Hough [8].

Plecând de la această explicație și ținând cont că dreptele pot fi reprezentate în spațiul polar, se poate generaliza astfel încât unor puncte coliniare din planul (x,y) le corespunde un set de sinusoide ce se intersectează într-un singur punct, Figura 3.3.

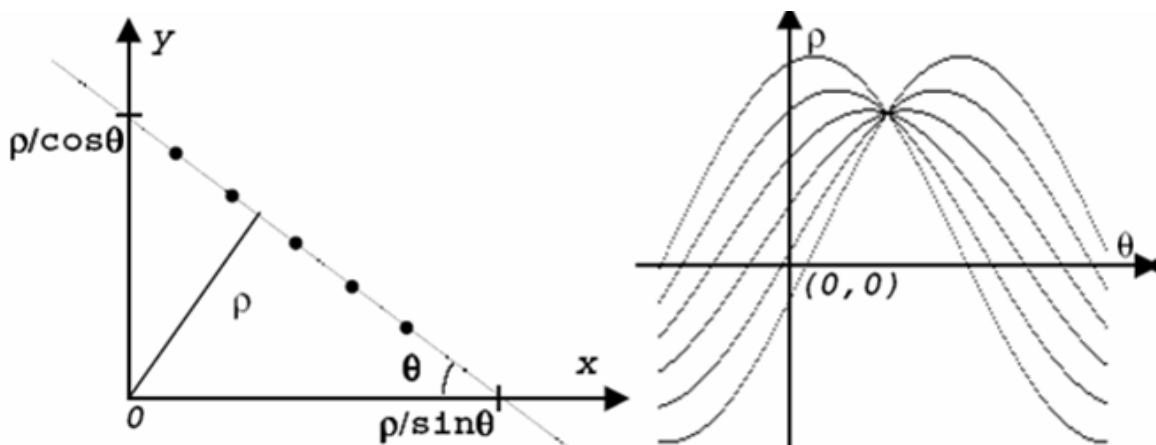


Figura 3.3: Maparea unui set de puncte coliniare din planul imaginii în spațiul Hough [11].

Având în vedere că acumulatorul Hough reprezintă spațiul parametrilor cuantizați ai dreptei, pașii de cuantizare pentru ρ și θ sunt $\Delta\rho$ și respectiv $\Delta\theta$, iar valorile lor maxime sunt ρ_{max} și θ_{max} . Atunci, dimensiunea acumulatorului este:

$$\begin{aligned} lățime &= \rho_{max}/\Delta\rho \\ înălțime &= \theta_{max}/\Delta\theta \end{aligned} . \quad (5)$$

Acumulatorul Hough, notat în continuare cu H , se construiește pe baza următorilor pași:

1. Fiecare celulă din H se initializează cu 0,
2. Pentru fiecare punct de muchie $P(x,y)$ se determină ecuațiile dreptelor aferente și se incrementează locațiile asociate din H astfel:

pentru fiecare θ pornind de la 0 la θ_{\max} , folosind pasul $\Delta\theta$:
 se calculează ρ folosind Ecuăția (2),
 și dacă ρ aparține intervalului $[0, \rho_{\max}]$ se incrementează celula aferentă din H .

După ce am construit acumulatorul, dreptele relevante se extrag ca fiind maxime locale ale acestuia. Un maxim local este un punct unde valoarea din acumulator este mai mare decât toate valorile dintr-o vecinătate.

Foarte importantă este alegerea unui nivel de cuantizare potrivit, deoarece dacă se face o cuantizare prea fină, rezoluția crește în același timp cu timpul de procesare și cresc şansele ca puncte aparent coliniare să incrementeze celule diferite din acumulator. Acest lucru va cauza detecții multiple ale aceleiași drepte sau fragmentarea unor drepte.

Transformata Hough clasica detectează drepte și oferă ca rezultat doar parametrii ρ și θ . Astfel, nefiind furnizată nicio informație privind lungimea, toate dreptele detectate sunt infinite. Dacă se doresc linii finite, atunci o modalitate de a le obține este folosind Transformata Hough Progresivă Probabilistică. Ideea acestei metode este de a transforma pixelii, aleși la întâmplare din imaginea ce conține muchii, în acumulatori. Această metodă se bazează pe voturile obținute de pixeli ca făcând parte din liniile căutate [12].

3.1.1.3. Vanishing Point

Vanishing point este un punct din planul de proiecție a unei imagini construite în perspectivă. Proiecțiile în perspectivă bidimensională a liniilor paralele din spațiul tridimensional par să converge în acest punct. Atunci când setul de liniile paralele este perpendicular pe un plan din imagine, construcția este cunoscută ca perspectivă dintr-un punct (*one-point perspective*), iar vanishing point-ul corespunde unui *eye-point*⁶ din care imaginea trebuie privită pentru ca geometria să să fie corectă, Figura 3.4. Desenele liniare tradiționale folosesc obiecte cu unul sau până la trei seturi de liniile paralele, astfel definind unul sau chiar până la trei vanishing point-uri [13].

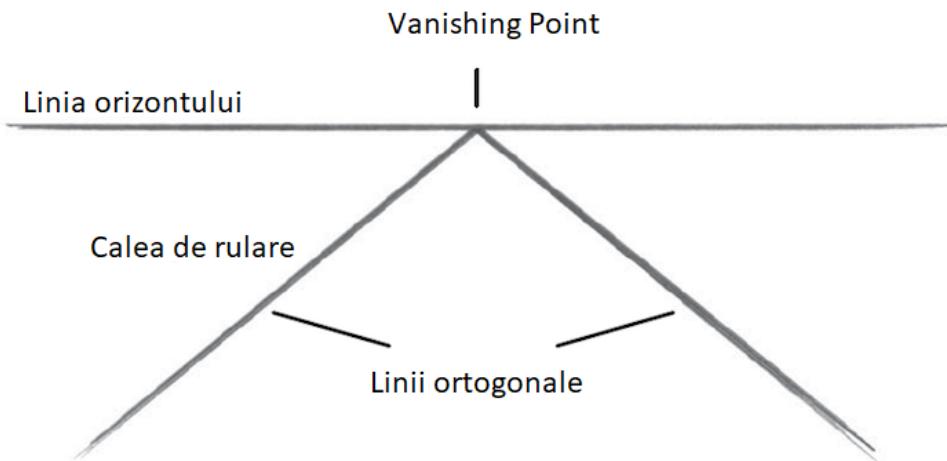


Figura 3.4: Vanishing point în 2D.

Vanishing point-ul este uneori denumit și „punctul de direcție” sau „punct de dispariție”, deoarece liniile care au același vector de direcție, vor avea același vanishing point.

Din punct de vedere matematic se poate defini un punct q , situat în planul imaginii:

$$q \equiv (x, y, f) , \quad (6)$$

unde f este focal lenght-ul camerei.

⁶ Eye-Point (din engl) reprezintă punctul de vedere din care se dorește ca imaginea să fie observată.

Pe lângă acesta se poate asocia v_q , vectorul unității pentru q :

$$v_q \equiv \left(\frac{x}{h}, \frac{y}{h}, \frac{f}{h} \right) , \quad (7)$$

unde h este definit precum:

$$h = \sqrt{x^2 + y^2 + f^2} . \quad (8)$$

Dacă luăm în considerare o linie dreaptă în spațiul S cu vectorul n_s asociat și vanishing point-ul v_s , atunci vectorul unitar asociat cu v_s este egal cu n_s , presupunând că ambele sunt considerate a se îndrepta spre planul imaginii.

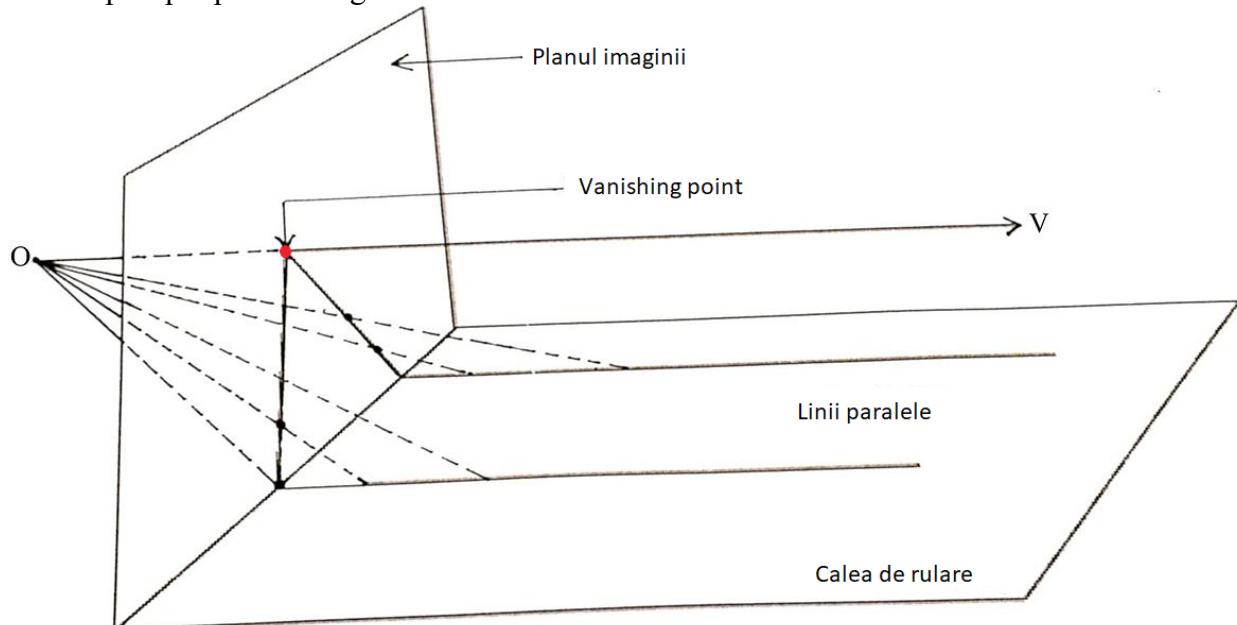


Figura 3.5: Construcție în perspectivă ce arată modalitatea de formare a vanishing point-ului.

Când planul imaginii este paralel cu două axe de coordonate al lumii, liniile paralele cu axa tăiată de acest plan de imagine vor avea imagini care se întâlnesc într-un singur vanishing point. Liniile paralele cu celelalte două axe nu vor forma puncte de dispariție deoarece sunt paralele cu planul imaginii. Aceasta este perspectiva dintr-un punct, cum este și ilustrat în Figura 3.5⁷. În mod similar, atunci când planul imaginii intersectează două axe de coordonate mondiale, liniile paralele cu aceste planuri se vor întâlni din două puncte de dispariție în planul imaginii. Aceasta se numește perspectivă în două puncte. În perspectivă în trei puncte, planul imaginii intersectează toate axele și, prin urmare, se intersectează liniile paralele cu aceste axe, rezultând trei vanishing point-uri diferite [14].

Există diferite metode utile pentru a detecta vanishing point-ul. În lucrarea de față se vor exemplifica două metode, cea de a doua fiind cea utilizată practic.

RANSAC este o paradigmă de potrivire a unui model la date experimentale, robustă la erori mari, ce a fost introdusă de Martin A. Fischler și Robert C. Bolles [15]. Aceștia au declarat că „Procedura RANSAC este opusul tehniciilor convenționale de filtrare: în loc să folosim cât mai multe date posibile pentru a obține o soluție inițială, și apoi să eliminăm punctele invalide, RANSAC folosește un set inițial de date cât mai mic posibil și apoi mărește acest set cu date valide atunci când este posibil”. Încă de atunci RANSAC a devenit o procedură esențială în domeniul viziunii artificiale și al procesării imaginilor. Cu toate acestea, metoda are dezavantajele sale (i.e., sensibilitate la alegerea pragului pentru zgomot sau la alegerea setului de parametri definiitorii). Pentru a compensa efectele nedorite, Torr et. al., [16] au propus două

⁷ Imagine preluată din [14] și modificată pentru a prezenta ideile de interes.

metode ce aduc modificări parțiale: MSAC (M-estimator Sample and Consensus) și MLESAC (Maximum Likelihood Estimation Sample and Consensus).

În mod similar, Chum et. al., [17] au propus să se ghideze procedura de eșantionare, dacă sunt cunoscute informații despre datele de intrare a priori (i.e., dacă acestea sunt mai probabil date de tip inlier⁸ sau outlier⁹). Abordarea propusă a fost numită PROSAC (PROgressive Sample Consensus.). Pe lângă această idee, Chum et. al., [18] au propus o variantă ce folosește estimarea aleatorie pentru RANSAC, numită R-RANSAC (Random-RANSAC). Această strategie se numește schemă preventivă și are rolul de a aduce îmbunătățiri din punct de vedere al timpului de execuție atunci când numărul de date de tip inlier este mare.

Totuși RANSAC în forma lui inițială are multe avantaje precum abilitatea de a realiza estimări robuste (i.e., metoda poate estima parametri precis atunci când un număr mare dintre acești sunt de tip inlier). Această metodă interpretează și netezește datele care conțin un procent semnificativ din erorile brute și, prin urmare, este ideal pentru aplicații ce realizează analiza automată a imaginilor, în cazul în care interpretarea se bazează pe datele furnizate de detectoarele cu caracteristici predispuse la erori. Algoritmul RANSAC este o tehnică de învățare ce estimează parametrii unui model, folosind o eșantionare aleatorie a unui set de date observate. Seturile de date pot conține atât date de tip inlier, cât și de tip outlier, iar algoritmul folosește un sistem de votare pentru a obține rezultatul optim de potrivire. Implementarea acestei scheme de votare este bazată pe două ipoteze [15]:

- caracteristicile ce conțin mult zgomot nu vor vot în mod consecvent pentru model (există puține date aberante);
- există suficiente caracteristici care pot defini un model corect.

Algoritmul este compus din doi pași care sunt repetăți iterativ [15]:

- În prima etapă, un subset de eșantioane care conține date minime este selectat aleator din setul de date de intrare. Este calculat un model de potrivire și parametrii corespondenți modelului, folosind doar elementele din subsetul selectat. Cardinalitatea acestui subset este cea mai mică, suficientă pentru a determina parametrii caracteristici modelului;
- În cel de-al doilea pas, algoritmul verifică care elemente ale întregului set de date sunt în concordanță cu modelul instantiat de parametrii estimati din primul pas. Un element va fi considerat ca fiind o deviere dacă nu se potrivește cu modelul reprezentat de setul de parametri deja aflați, având în vedere un prag de eroare, ce reprezintă deviația maximă admisă pentru a nu introduce zgomot excesiv.

Setul de date de tip inlier obținut pentru modelul de potrivire poartă numele de set de consens („consensus set”). Algoritmul descris anterior va repeta cei doi pași până când consensul obținut va avea destule date de tip inlier.

Valoarea pragului utilizat pentru a determina când o dată se încadrează în modelul dorit și când numărul de date de tip inlier este minim pentru a putea declara un model ca fiind potrivit este aflat având în vedere cerințe specifice pentru aplicația dorită sau pe bază experimentală (i.e., empiric). Pe de altă parte, numărul de eșantioane N este ales pentru a asigura, cu o probabilitate p , că cel puțin unul din eșantioane nu conține zgomot. Fie q probabilitatea ca o dată selectată să fie validă. Atunci probabilitatea ca toate cele s date selectate să fie valide este q^s . Evenimentul complementar este ca cel puțin una dintre date să fie invalide și are probabilitatea $1-q^s$. Atunci, putem exprima valoarea numărului de eșantioane ca fiind [19]:

⁸ Inlier este un tip de valori potrivite pentru contextul cerut.

⁹ Outlier este un tip de valori aberante, care nu se potrivesc unui anume context.

$$N = \frac{\log(1-p)}{\log(1-q^s)} . \quad (9)$$

Pentru a detecta vanishing point-ul folosind metoda RANSAC, se poate seta ca obiectiv determinarea punctului care verifică cele mai multe ecuații ale dreptelor detectate în imagine. Alegând câte două drepte aleator, se calculează intersecția lor și se folosește un sistem de votare pentru a obține valoarea dorită, iar din restul setului de date sunt selectate dreptele care sunt considerate de tip inlier și cele de tip outlier. Se decide astfel dacă punctul ce reprezintă intersecția selectată poate fi considerat ca fiind potrivit pentru sistemul ales. După executarea unui număr suficient de mare de pași, calculat folosind Ecuația (9), se alege ca fiind vanishing point-ul potrivit, punctul care are cel mai mare număr de voturi.

O altă metodă de a calcula vanishing point-ului, este prin estimarea obiectivă a acestuia. Această metodă este mai rapidă din punct de vedere computațional pentru imagini de dimensiuni mici și medii. Imaginele procesate au dimensiunea 1024x876 și tocmai de aceea s-a luat decizia de a se utiliza această metodă în lucrarea de față.

Se utilizează o structură bidimensională în care se incrementează poziția tuturor punctelor ce fac parte din dreptele din imagine. Dreptele, detectate utilizând transformata Hough sunt caracterizate prin parametrii ρ și θ , aşa cum am prezentat în Subcapitolul 3.1.1.2.

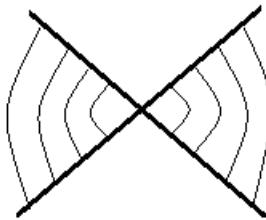


Figura 3.6: Parcugerea imaginii pe verticală.

Pentru a putea găsi toate punctele care fac parte dintr-o dreaptă se verifică dacă direcția dreptei este pozitivă sau negativă. Datorită semnificației parametrului θ (unghiul dreptei din origine, perpendiculară pe dreapta detectată), punctele dorite se obțin parcurgând imaginea pe verticală (Figura 3.6) atunci când este îndeplinită condiția prezentată în Ecuația (10), iar pe orizontală în rest.

$$|\tan(\theta)| > 1 \quad (10)$$

Ulterior se parcurge structura obținută prin pașii anteriori pentru a găsi poziția maximului local. Pentru a localiza vanishing point-ul final, se caută punctele apropriate de maxim determinat și se calculează centrul casetei determinate de toate aceste puncte.

3.1.1.4. Preprocesarea prin filtrare folosind operații morfologice

Pentru a modifica sau pentru a aduce îmbunătățiri asupra formei sau structurii unor elemente sunt necesare anumite operații de post-filtrare sau pre-filtrare, printre acestea numărându-se și operațiile morfologice. Steven W. Smith [20], prezintă o serie de tehnici de procesare a imaginilor folosind operațiile morfologice.

Operațiile morfologice se aplică în general asupra unor imagini binare¹⁰ pentru a se obține ca rezultat reprezentări ale formei obiectelor și regiunilor. De asemenea, prin filtrarea morfologică se pot elimina imperfecțiuni introduse prin binarizare.

Binarizarea imaginii este una dintre cele mai simple metode de segmentare. Separarea

¹⁰ O imagine digitală binară este o imagine alb negru, ce conține doar pixeli albi sau pixeli negri.

regiunilor din imagine poate fi realizată prin impunerea unui prag de binarizare (threshold) pe baza căruia fiecare pixel din imagine va fi clasificat ca fiind pixel de fond sau pixel element:

$$g(i,j) = \begin{cases} 1, & \text{dacă } (i,j) \geq T \\ 0, & \text{în rest} \end{cases}. \quad (11)$$

Pixelii de fond vor avea valoarea 0 (negru), iar pixelii obiect vor avea valoarea 1 (alb).

Printre cele mai utilizate operații morfologice se numără dilatarea și eroziunea. Dilatarea este utilizată pentru a mări obiecte, pentru a umple lipsuri sau pentru a conecta obiecte disjuncte. Eroziunea are ca efect micșorarea obiectelor prin erodarea marginilor acestora. Atât dilatarea, cât și eroziunea utilizează un element structural, o mască binară compusă din elemente de 0 și 1, ce poate avea diferite forme. Cele mai des întâlnite forme sunt: cruce, elipsă sau dreptunghi, exemplificate în Figura 3.7¹¹. Elementul structural este folosit pentru a determina care dintre pixelii vecini vor contribui la calcularea valorii noi a pixelului curent din imagine. Acest element structural este suprapus peste imaginea originală într-o manieră similară conveției¹².

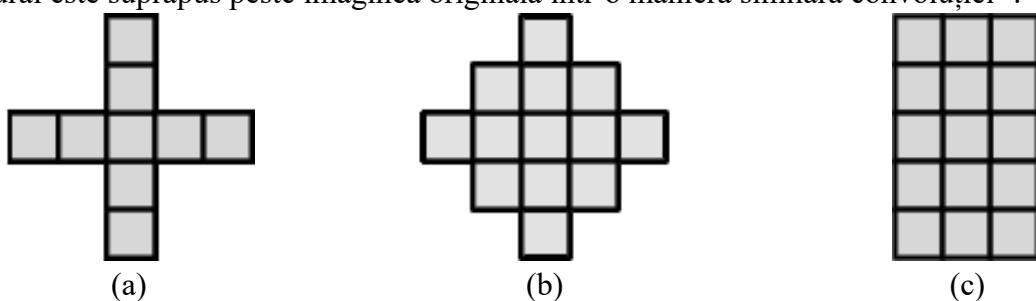


Figura 3.7: Exemplu de forme ale elementului structural: (a) – cruce; (b) – elipsă; (c) – dreptunghi.

Dilatarea se realizează prin suprapunerea și mutarea elementului structural peste imaginea de prelucrat prin următorii pași:

- Dacă originea elementului structural se suprapune cu un pixel de culoare albă, se trece la următorul pixel, neefectuându-se nicio modificare;
- Dacă originea elementului structural se suprapune cu un pixel de culoare neagră, atunci toți pixelii acoperiți de elementul structural devin negri.

Operația de eroziune este realizată invers față de cea de dilatare, efectuându-se următorii pași:

- Dacă originea elementului structural se suprapune cu un pixel de culoare albă, se trece la următorul pixel, neefectuându-se nicio modificare;
- Dacă originea elementului structural se suprapune cu un pixel de culoare neagră a imaginii și există cel puțin un pixel de culoare neagră al elementului structural ce se suprapune peste un pixel alb din imagine, atunci pixelul curent din imagine devine alb.

În Figura 3.8 se prezintă un exemplu clasic de aplicare a operațiilor de dilatare și eroziune, folosind un element structural de tip cruce, de dimensiune 3x3. Prin combinarea operațiilor de dilatare și eroziune se obțin diferite rezultate ce pot aduce îmbunătățiri imaginii originale.

Deschiderea constă într-o eroziune urmată de o dilatare și este folosită pentru eliminarea pixelilor din regiunile care sunt prea mici pentru a conține elementul structural. Astfel, elementul structural filtrează elementele prea mici din imagine. Închiderea este formată dintr-o dilatare urmată de o eroziune, și este folosită pentru a umple goluri și mici discontinuități.

11 Sursa: <https://ihrchive.files.wordpress.com/2010/06/1eeaa-strels.png>

12 Convoluția este o modalitate matematică de a realiza combinații între două imagini, intrări, pentru a forma o imagine ieșire. Pixelul rezultat la ieșire este o combinație liniară a setului de pixeli sursă.

În aplicația implementată se utilizează deschiderea și închiderea pentru reconstruirea liniilor detectate în imaginile primite ca parametri de intrare, dar care au părți componente lipsă. Aceste operații aduc îmbunătățiri rezultatelor obținute.

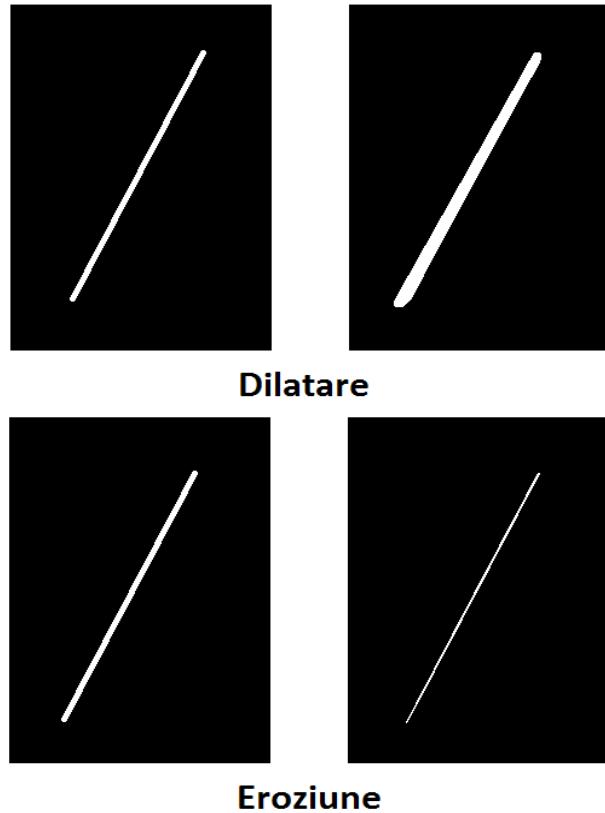


Figura 3.8: Exemplu de aplicare a operațiilor morfologice.

3.1.1.5. Filtrarea liniilor după proprietăți

Trecerile de pietoni se formează dintr-un set uniform de linii albe, paralele pe o suprafață de culoare închisă (negru, nuanțe de gri sau uneori roșu). Se dorește detectarea unui grup de linii paralele din imaginea prelucrată, care fac parte din trecerile de pietoni. Liniile paralele în imagini au o proprietate comună: trec prin același punct, numit vanishing point, deoarece construcția este realizată folosind perspectiva dintr-un punct. Planul imaginii este paralel cu cele două axe de coordonate ale lumii, liniile paralele cu axa tăiată de acest plan din imagine se întâlnesc într-un singur vanishing point.

Având la dispoziție atât dreptele împreună cu cele două puncte care le mărginesc, cât și poziția vanishing point-ului se poate detecta numărul dreptelor cărora le aparține acesta din urmă.

Dreptele sunt formate din puncte geometrice ce îndeplinesc ecuația dreptei. Cu alte cuvinte, ecuația unei drepte reprezintă o relație care este respectată de toate punctele aflate pe dreaptă. Forma generală a ecuației unei drepte în sistemul xOy este:

$$a*x + b*y + c = 0 \quad . \quad (12)$$

De asemenea, fiind date două puncte $A(x_1, y_1)$ și $B(x_2, y_2)$, ecuația dreptei determinată de ele se poate scrie [21]:

$$\frac{x - x_1}{x_2 - x_1} = \frac{y - y_1}{y_2 - y_1} . \quad (13)$$

Se cunoaște că orice dreaptă împarte planul în două semiplane: cel cu puncte pentru care, dacă aplicăm ecuația, vom obține o valoare strict pozitivă, și cel pentru care vom obține o valoare strict negativă. De aceea, dacă avem o dreaptă ce trece prin două puncte $A(x_1, y_1)$ și $B(x_2, y_2)$ de pe aceasta, atunci punctul $V(x_3, y_3)$ va apartine dreptei AB dacă și numai dacă:

$$\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix} = 0 . \quad (14)$$

Pe de altă parte pentru a calcula distanța de la un punct la o dreaptă se construiește o dreaptă $d2$ perpendiculară pe dreapta $d1$ care trece prin punctul V (Figura 3.9).

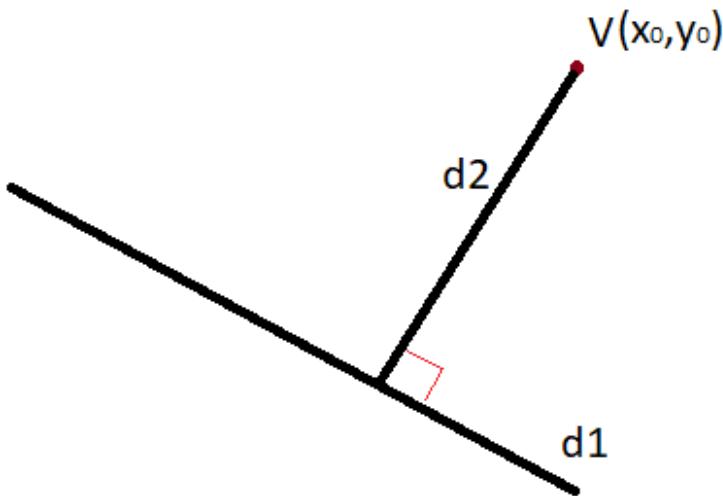


Figura 3.9: Distanța de la un punct V la o dreaptă $d1$

De asemenea, există și o formulă pentru a determina distanța de la un punct la o dreaptă:

$$d(V, d) = \frac{|ax_0 + by_0 + c|}{\sqrt{a^2 + b^2}} . \quad (15)$$

Folosindu-se metodele descrise mai sus, se pot afla dreptele care trec prin vanishing point-ul calculat anterior.

Dreptele detectate sunt apoi grupate pe baza unui scor bazat pe anumite constrângeri precum: lungime, număr total de linii detectate. Pentru ca liniile să fie semnalate ca facând parte din treceri de pietoni, acestea trebuie să fie într-un număr de cel puțin 5, iar capetele acestora trebuie să fie în proporție de 80% similară pe axa oY . În acest mod se detectează dreptele care fac parte din treceri de pietoni din setul de imagini de intrare.

3.1.2. Metoda ce folosește rețelele neuronale

O altă modalitate de a detecta trecerile de pietoni se bazează pe o ramură principală din știința inteligenței artificiale. Rețelele neurale artificiale caracterizează ansambluri de elemente de procesare de bază, interconectate și care operează în paralel. Acestea urmăresc să interacționeze cu mediul înconjurător într-un mod asemănător creierelor biologice și care prezintă capacitatea de a învăța. Rețelele neuronale se diferențiază față de alte sisteme de prelucrare a informației prin capacitatea acestora de a învăța în urma interacțiunii cu mediul

înconjurător și, ca urmare, de a-și îmbunătăți în timp performanțele după anumite criterii predefinite. Fundamental este ca modul de reprezentare internă a informațiilor să permită interpretarea, predicția și oferirea unui răspuns corect la un stimul provenit din mediul înconjurător. O reprezentare corectă și eficientă îi va permite rețelei neuronale să construiască un model al procesului analizat în stare să se comporte satisfăcător în condițiile în care la intrare i se va aplica stimuli care nu au fost utilizati în procesul prealabil de învățare. Există mai multe criterii în funcție de care se pot clasifica algoritmii de învățare printre care se numără și disponibilitatea răspunsului dorit la ieșirea rețelei neuronale. Astfel, învățarea poate fi clasificată ca: învățare supravegheată, învățare nesupravegheată și învățare folosind un „critic”. Dacă în cazul învățării supravegheate setul de antrenare conține perechi de tipul intrare-ieșire dorite, iar rețea neuronală trebuie să realizeze maparea intrărilor către ieșiri, la învățarea prin auto-organizare setul de antrenare conține numai mărimi de intrare, iar rețea neuronală construită încearcă să învețe structura datelor de intrare, chiar și în cazul inexistenței unei informații despre această structură. Rețea neuronală învăță singură, fără a-i se indica răspunsul corect pentru un model prezentat la intrare. Învățarea folosind un "critic" este denumită uneori și recompensă/pedeapsă. În această situație, rețea neuronală nu beneficiază de un semnal dorit, ca în cazul învățării supravegheate, ci de un semnal care oferă o informație calitativă ilustrând cât de corect funcționează sistemul.

Domeniul viziunii artificiale include diferite atribuții, dintre care vom enumera patru teme care se identifică ca fiind printre principalele metode utilizate pentru a identifica obiecte în imagini. O primă metodă este clasificarea care oferă ca răspuns doar confirmarea prezenței obiectului căutat în imagine. O altă modalitate este segmentarea semantică, care aduce o informație în plus, și anume sunt marcați pixelii din care sunt compuși obiectele căutate. Ulterior, s-a dezvoltat o modalitate care pe lângă confirmarea prezenței obiectului căutat, descoperă și numărul exact de obiect din aceeași clasă, numită detecția de obiecte. În final, metoda care le combină pe ultimile două metode enumerate, intitulată segmentarea instanțelor, detectează atât prezența cât și toți pixelii pentru fiecare obiect în parte din clasa căutată.

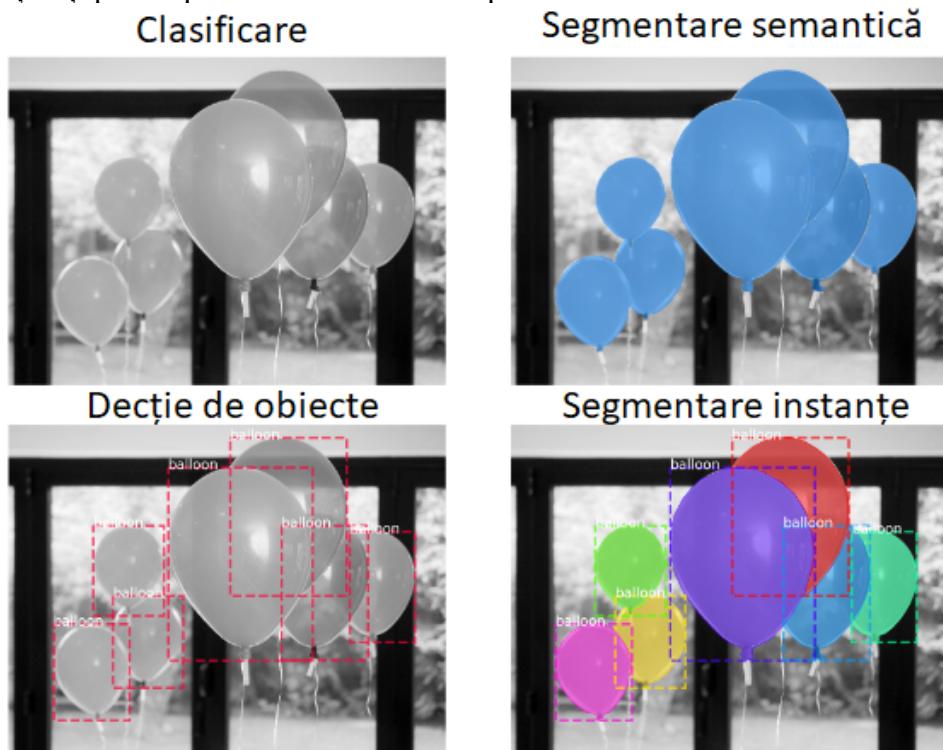


Figura 3.10 Metode pentru identificare obiecte [22].

În Figura 3.10 sunt exemplificate, pe un exemplu practic, cele patru metode explicate anterior.

Acest subcapitol va elabora metoda de segmentare a instațelor ce este utilizată pentru a identifica pixelii ce aparțin trecerilor de pietoni din imagini, folosindu-se de ajutorul algoritmilor de învățare automată.

3.1.2.1. Mask R-CNN

Mask R-CNN (rețea neuronală regională conoluțională) este un framework¹³ în două etape: prima etapă scanăza imaginea și generează propunerile (zone care pot conține un obiect) și cea de-a doua etapă clasifică propunerile și generează măști pentru încadrarea pixelilor.

Acesta fost introdus în anul 2017 prin articolul Mask R-CNN [23] pentru a-și extinde predecesorul, Faster R-CNN [24], de către aceiași autori. Faster R-CNN este un framework popular pentru detectarea obiectului, iar Mask R-CNN îl extinde cu segmentarea instațelor, printre altele. Astfel, extinderea are loc prin adăugarea unei ramuri pentru predicția tuturor pixelilor ce fac parte din obiecte, paralel ramurii ce realizează recunoașterea prin setarea unei casete de încadrare pentru obiect. Acest algoritm este simplu de antrenat și adaugă cost nesemnificativ față de metoda pe care o extinde. Mai mult decât atât, algoritmul Mask R-CNN este ușor de generalizat pentru alte sarcini (i.e., poate fi utilizat pentru a estima pozițiile persoanelor din același cadru).

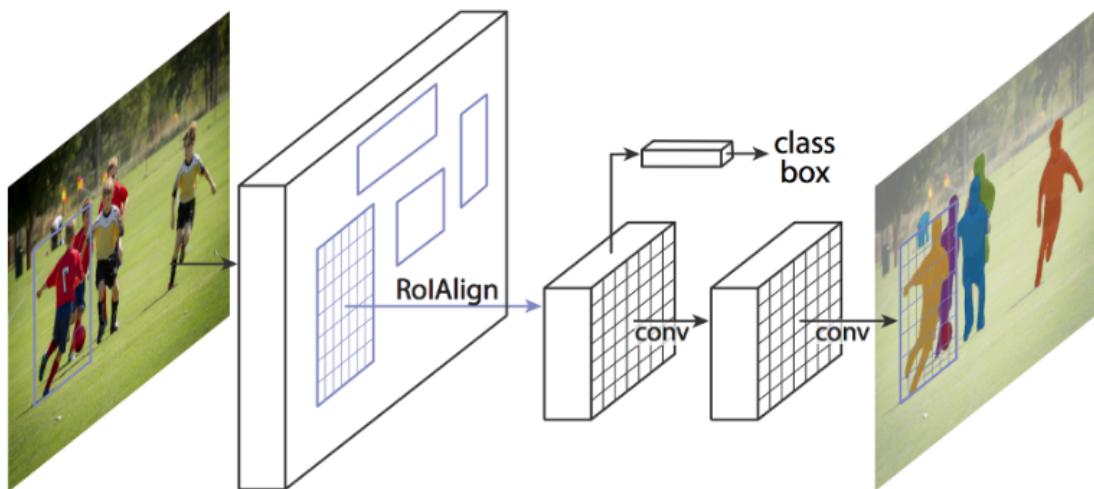


Figura 3.11: Framework-ul Mask R-CNN [23].

În mod general, Mask R-CNN este compus din patru module: rețeaua principală (în engleză backbone), RPN (Region Proposal Network), clasificatorul ROI (Region of Interest) împreună cu regresorul pentru casetele de încadrare ale obiectelor detectate și măștile pentru segmentare. Figura 3.11 prezintă cadrul general al acestei metode.

Backbone-ul este o rețea neuronală conoluțională clasică ce deservește ca un instrument pentru izolarea caracteristicilor existente.

În domeniul deep learning¹⁴, o rețea neuronală conoluțională este o clasă de rețele neuronale profunde, aplicate cel mai frecvent pentru a analiza imagini. Rețelele conoluționale au adus o serie de îmbunătățiri algoritmilor ce fac parte din domeniul învățării automate. Schema generală a unei rețele conoluționale se poate observa în Figura 3.12. Astfel, se pot observa două zone principale, cea formată din straturile de tip Conoluție și Unificare și cea de-a doua formată

13 Framework-ul reprezintă o modalitate generică, reutilizabilă de a rezolva o problemă.

14 Deep learning este un domeniu al inteligenței artificiale, bazat pe rețele neuronale artificiale.

din straturile FCN (Fully Connected Networks). Prima zonă are rolul de a extrage trăsături. Primele starturi convoluționale extrag informații de tip contururi ce devin mai complexe cu parcurgerea rețelei. A doua zonă este practic o rețea MLP (Multilayer perceptron), ce conține o serie de perceptri multistrat, aplicată trăsăturilor extrase anterior. Diferențele dintre modul de funcționare al MLP-ului și al stratului convoluțional oferă credibilitate aplicației [25].

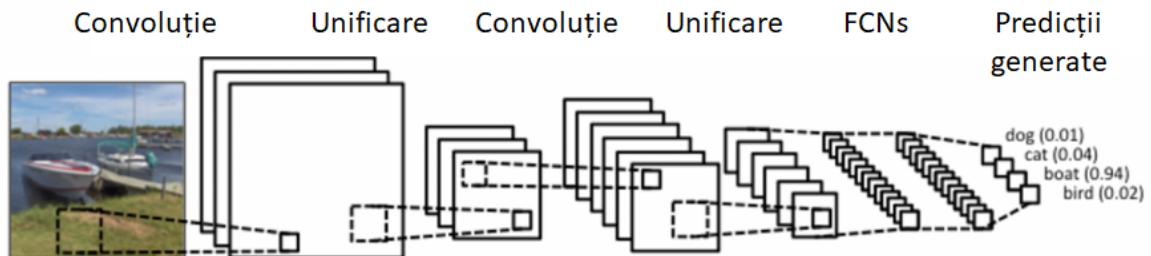


Figura 3.12: Schema generală a unei rețele convoluționale [25].

Straturile convoluționale sunt unitățile de bază ale acestei arhitecturi. Un strat convoluțional realizează o serie de operații de filtrare liniară pe matricea de la intrarea în stratul respectiv. Această matrice poate fi imaginea în sine sau rezultatul altui strat, denumit hartă de trăsături. Filtrarea ce utilizează tehnica ferestrei glisante poartă numele de convoluție bidimensională. Operația de filtrare liniară se realizează prin doi pași. Inițial se suprapune masca de filtrare peste fiecare pixel al imaginii originale astfel încât originea măștii să coincidă cu pixelul considerat. Ulterior, se calculează toate produsele dintre coeficienții măștii și valorile pixelilor peste care se suprapun acești coeficienți, iar suma acestor produse reprezintă noua valoare a pixelului considerat.

Precum se poate observa în Figura 3.13, se utilizează un nucleu deplasat asupra stratului anterior din doi în doi neuroni, atât pe orizontală, cât și pe verticală. Dimensiunea uzuală pentru nucleu este de 5×5 neuroni. Aceasta trebuie să fie impară pentru ca nucleul să poată fi centrat într-un pixel. Nucleul de convoluție este folosit pentru întreaga imagine, deci ponderile care conduc la un neuron specific din stratul următor sunt mereu aceleași.

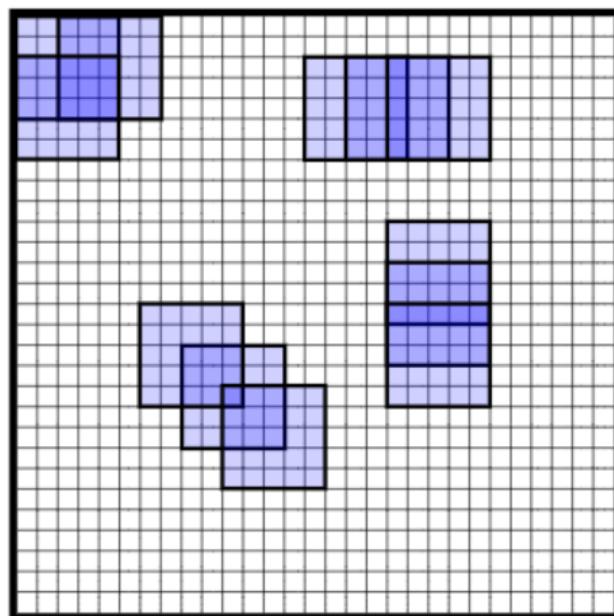


Figura 3.13: Convoluție bidimensională.

Toate straturile conoluționale sunt urmate de o funcție de activare (în general se aplică Unitatea Linear Rectificată – ReLU). Această funcție este definită ca partea pozitivă a propriului său argument:

$$f(x) = \max(0, x) . \quad (16)$$

Straturile de unificare conțin operații de grupare sau subeșantionare. Operațiile de acest tip înlocuiesc valoarea unei zone din imagine sau din harta de trăsături cu o statistică a aceleia zone. Funcția utilizată de obicei este cea care înlocuiește valoarea dintr-o zonă bine definită cu maximul acelei zone. Rezultatul obținut este o hartă de trăsături mai mică, dar care păstrează cea mai relevantă statistică. În acest mod se realizează atât reducerea dimensionalității, cât și o obținere de invariante la translații mici, păstrându-se necesarul informațional ce este utilizat în continuare.

A doua zonă, ce conține straturi FNC, este utilizată pentru a obține rezultatele finale, predicțiile generate pentru imaginea de intrare.

Mask R-CNN utilizează o rețea neuronală conoluțională clasică (e.g., ResNet50 sau ResNet101) ce servește ca extractor de caracteristici. Straturile inițiale detectează muchii și colțuri, iar straturile următoare detectează caracteristici mai generale (e.g., mașină, persoană, cer etc.). Dimensiunea imaginii care trece prin backbone este micșorată și transformată într-o hartă de caracteristici ce este utilizată în următoarele etape.

Deși rețelele conoluționale clasice au rezultate bune, prin Mask R-CNN s-a introdus o îmbunătățire, numită FPN (Feature Pyramid Network). Aceasta se remarcă prin adăugarea unei noi piramide care preia caracteristicile detectate de către prima piramidă deja implementată și le pasează către straturile inferioare, așa cum este prezentat în Figura 3.14. În acest mod, fiecare nivel deține acces atât la caracteristicile generale, cât și la cele specifice, detectate anterior.

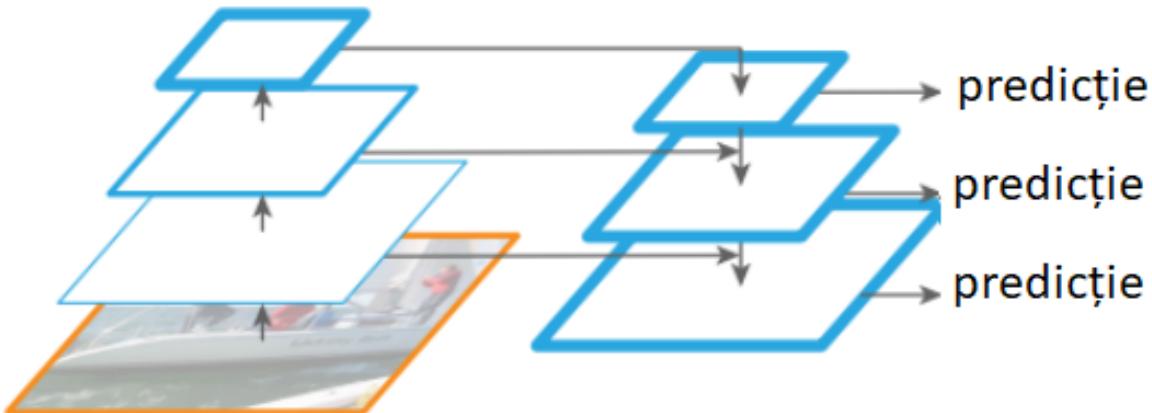


Figura 3.14: Utilizare FPN [22].

Următorul modul din care este compus Mask R-CNN este RPN-ul, ce reprezintă o rețea neuronală ce scană întreaga imagine într-o manieră ce utilizează o fereastră glisantă pentru a descoperi zone ce conțin obiecte. Regiunile care sunt scanate sunt numite ancore și practic sunt casete distribuite peste tot în imagine. Numărul lor variază în funcție de dimensiunea imaginii, cu mențiunea că acestea trebuie să se suprapună astfel încât să cuprindă cât mai mult din toată imaginea. Pe lângă acestea, se utilizează harta caracteristicilor determinată de către backbone ca intrare în locul imaginii, pentru a aduce un plus de optimizare. În acest mod RPN-ul poate reutiliza caracteristicile extrase deja pentru a evita efectuarea calculelor duplicate. Pentru fiecare ancoră, RPN-ul generează două ieșiri. Inițial se obține este clasa ancorei, care poate fi fie fundal,

fie parte din prim plan. Ulterior, se estimează un delta (calculat ca procent de schimbare pentru poziție și dimensiuni) pentru a se centra caseta generată peste obiectul detectat. Propunerile finale (regiunile de interes detectate) sunt pasate către etapele următoare [22].

Modulul următor, ROI, lucrează peste regiunile de interes propuse de către RPN. De asemenea, oferă două ieșiri. Inițial se stabilește o clasă specifică (e.g., persoană, scaun, mașină etc.), apoi se realizează o nouă rafinare a poziției casetelor de încadrare a obiectelor. Totodată, clasificatorii au nevoie de dimensiuni fixe pentru intrările gestionate. De aceea, se va realiza o decupare a hărții de caracteristici și va fi redimensionată către o mărire specifică, precum se poate observa în Figura 3.15.

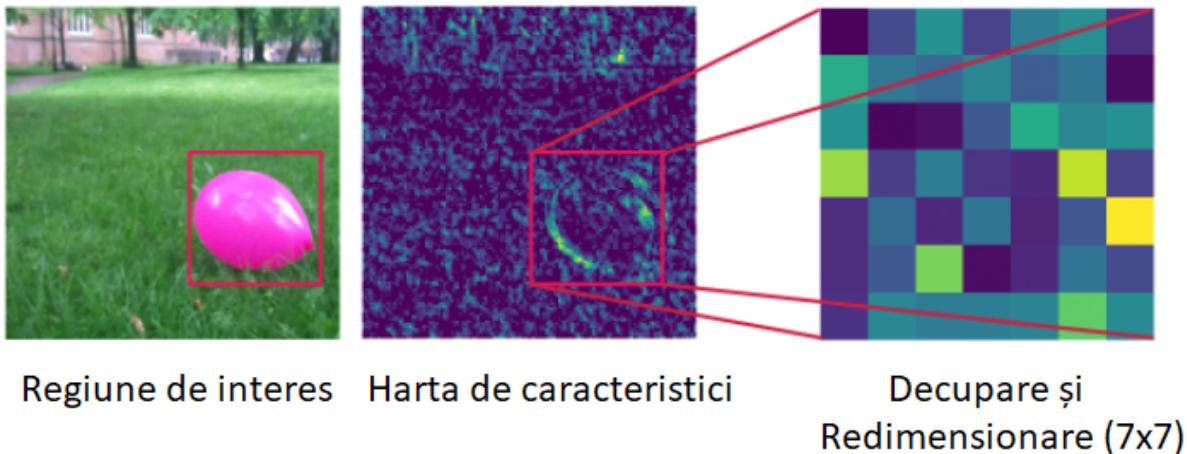


Figura 3.15: Exemplu de decupare și redimensionare a hărții de caracteristici [22].

Ultima secțiune a Mask R-CNN este cea care face diferența principală dintre aceasta și Faster R-CNN. Rețeaua ce adaugă prezența măștilor este completarea adusă în acest sens. Ramura pentru poziționarea măștilor este tot o rețea conoluțională ce preia regiunile pozitive selectate de clasificatorul anterior și generează măști pentru aceste regiuni. Măștile generate sunt reprezentate prin numere reale, astfel încât conțin mai multe detalii decât măștile binare. În perioada de antrenare măștile utilizate de punct de referință sunt micșorate și apoi sunt redimensionate pentru a se potrivi cu dimensiunea casetelor de încadrare obținute anterior. În acest mod, sunt obținute măștile finale, câte una pentru fiecare obiect [23].

3.2. Proiectarea și implementarea soluției

În cadrul acestui subcapitol este prezentată structura generală a aplicației implementate, iar pe parcurs sunt descrise și exemplificate toate modulele componente din punct de vedere arhitectural. De asemenea, sunt prezentate detalii de implementare pentru metodele utilizate, împreună cu explicații suplimentare pentru soluțiile introduse.

3.2.1. Arhitectura generală a sistemului de detecție

În Figura 3.16 este ilustrată arhitectura generală a sistemului construit, ce conține două module ce procesează imagini color și oferă ca rezultat imagini ce conțin marcaje ale prezenței trecerilor de pietoni. Rezultatele generate de cele două module sunt comparate atât din punct de vedere al acurateței, cât și din punct de vedere al timpului de execuție pentru a se putea decide ce metodă este mai potrivită pentru a îndeplini obiectivul de a utiliza aplicația în domeniul navigării vehiculelor autonome.

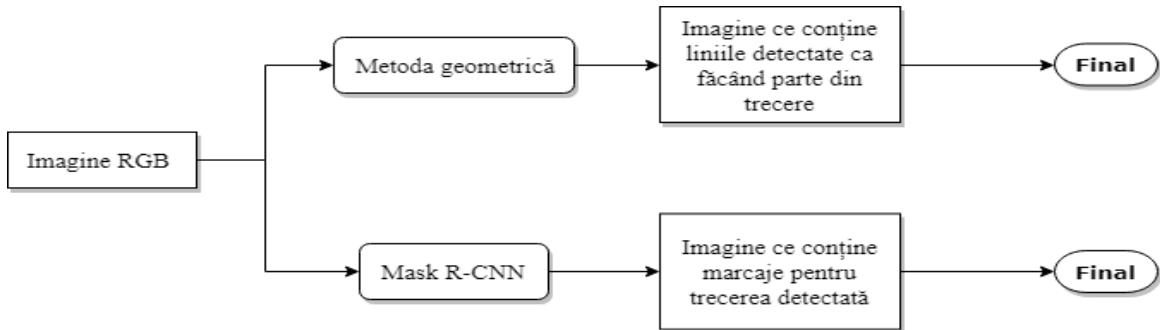


Figura 3.16: Arhitectura generală a sistemului.

3.2.2. Metoda geometrică

3.2.2.1. Diagrama de structură a metodei propuse

Arhitectura generală a metodei geometrice, prezentată în acest subcapitol este prezentată în Figura 3.17.

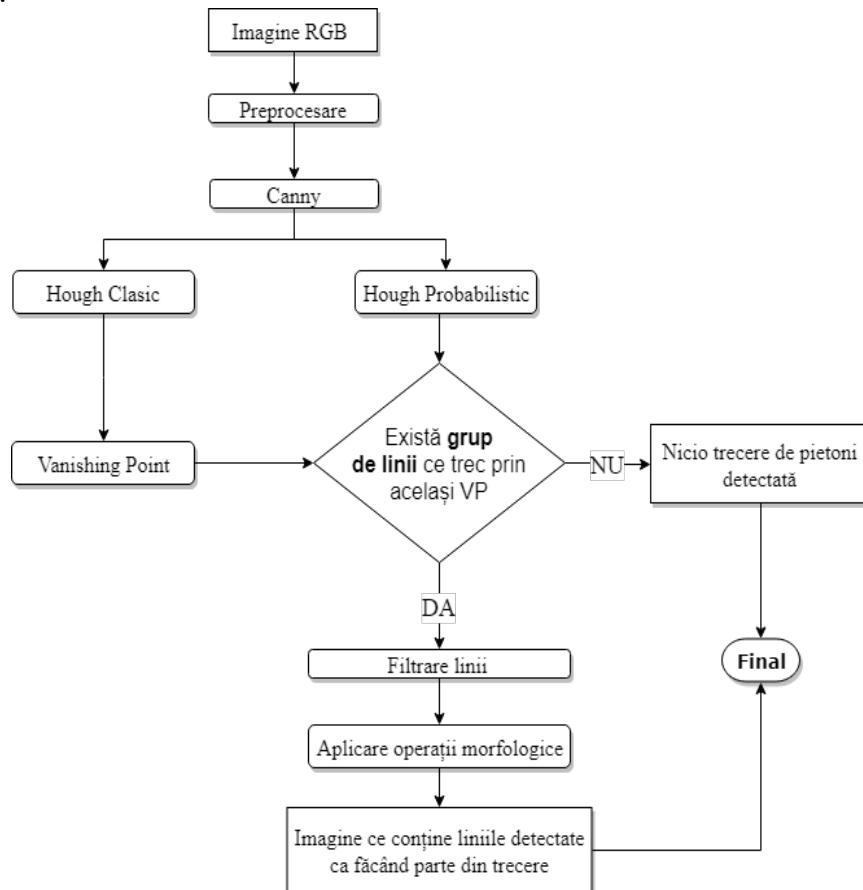


Figura 3.17: Diagrama de structură a metodei geometrice.

Pornind de la imaginea color, în format RGB, se realizează o preprocesare a imaginii pentru fi aliniată cu cerințele ulterioare ale algoritmului. În următorul pas se aplică metoda Canny pentru a extrage frontierele din imagine, apoi se calculează liniile din imagine, folosind două metode. Transformata Hough clasică este utilizată pentru a obține linii în formă polară pentru ca, ulterior, să fie calculat cu ajutorul acesteia, vanishing point-ul. Pe de altă parte, pentru a obține și o dimensiune exactă pentru liniile detectate, se utilizează Transformata Hough probabilistică.

Următorul pas este de a determina grupul de linii din setul obținut prin aplicarea transformatei Hough probabilistică ce îndeplinește condiția de a trece prin vanishing point. Dacă nu există un grup de cel puțin 5 linii, atunci nu există nicio trecere de pietoni în imagine. Dacă se obține un grup de cel puțin 5 linii se aplică operațiile morfologice pentru a îmbunătăți rezultatele și se construiește imaginea ce conține liniile detectate finale.

3.2.2.2. Detalii de implementare

Limbajul de programare folosit pentru dezvoltarea aplicației ce implementează metoda geometrică este C++, utilizându-se librăria OpenCV. Aplicația software (IDE) pentru care s-a optat este Microsoft Visual Studio deoarece conține un set complet de instrumente de dezvoltare necesare aplicației.

Aplicația este executată folosind procesorul Intel Core i7-6700HQ.

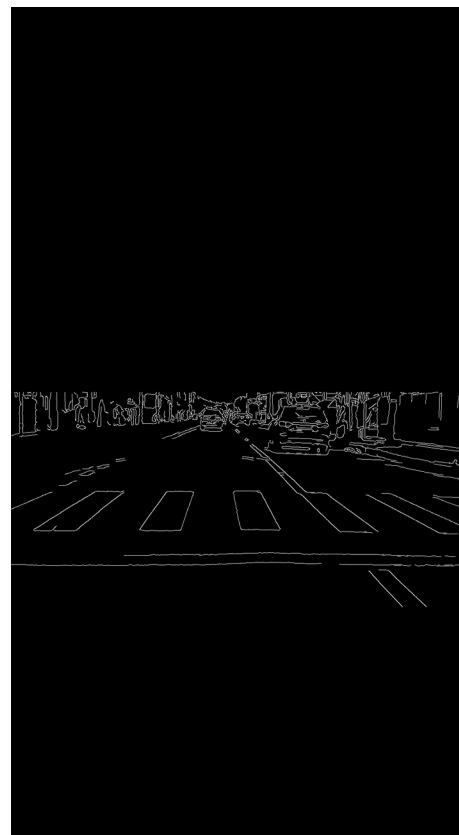
3.2.2.3. Canny

Primul pas realizat pentru detecția trecerilor de pietoni este de a extrage informațiile cu privire la frontierele din imagini. Frontierele sunt caracterizate prin existența variațiilor brusă ale intenității din imaginea ce urmează a fi prelucrată. În lucrarea de față se utilizează detectorul de frontiere Canny, deoarece acesta aduce îmbunătățiri față restul metodelor existente prin maximizarea raportului semnal-zgomot și eliminarea non-muchiilor.

Figura 3.18 prezintă un exemplu de frontiere obținute după aplicarea algoritmului Canny implementat. Imaginea (a) reprezintă imaginea neprelucrată preluată din setul de date inițial, iar imaginea (b) conține rezultatul obținut folosind algoritmul Canny.



(a)



(b)

Figura 3.18: Exemplu de aplicare al algoritmului Canny.

OpenCV pune la dispoziție o funcție care execută întreg algoritmul Canny pentru o imagine ce conține nivele de gri. Astfel, pentru a putea utiliza această funcție se aplică o conversie din imaginea color în imagine ce conține nivele de gri:

```
cv::cvtColor(m_image, houghLinesSrc, CV_BGR2GRAY);
```

Ulterior, se aplică un filtru blur¹⁵ pentru a netezi imaginea primită și a elimina zgomotul foarte puternic. Pentru ca imaginea să conțină doar informație utilă, se elimină pixelii ce se situează deasupra liniei orizontului și cei care se situează prea jos în imagine.

Ultimul pas este cel de aplicare a funcției Canny, din librăria openCV, ce conține ca parametri imaginea obținută din pasul anterior, imaginea rezultată după aplicarea algoritmului, pragul inferior, pragul superior și dimensiunea nucleului utilizat de filtru. Pragurile trebuie alese, îndeajuns de depărtate astfel încât operația histerezis să ofere rezultate clare, iar dimensiunea nucleului utilizat este 3.

```
cv::blur(m_image, image.blur, cv::Size(5, 5));
cv::Canny(image.blur, image.canny, 80, 150, 3);
```

3.2.2.4. Transformata Hough

Având la dispoziție frontierele din imagine, se calculează toate dreptele relevante pornind de la acestea. Transformata Hough este metoda ce rezolvă o problemă clasică din domeniul viziunii artificiale: găsirea tuturor dreptelor având la dispoziție o imagine ce conține puncte de interes. În aplicația construită se utilizează atât transformata Hough clasica, cât și cea probabilistică. Clasa *Hough* conține ambele metode ce sunt aplicate atunci când este necesar și oferă ca rezultat un vector de linii.

```
class Hough
{
public:
    inline Hough(cv::Mat image, std::vector<line::Line> lines);
    std::vector<line::Line> houghLines(char i);
    std::vector<line::Line> probabilisticHoughLines(char i);
private:
    cv::Mat m_image;
    std::vector<line::Line> m_lines;
};
```

Metoda clasica este implementată folosind funcția openCV *HoughLines()*. Motivația utilizării acestei metode constă în necesitatea utilizării rezultatelor oferite de aceasta. În următoarea etapă sunt necesari parametrii dreptei scrise sub formă polară (i.e., unghiul liniei, θ și distanța de la linie la origine, ρ) pentru a stabili poziția vanishing point-ului. Transformata Hough clasica generează ca rezultat un vector de linii exprimate cu ajutorul celor doi parametri menționați anterior. Argumentele funcției sunt: imaginea sursă, vectorul ce conține liniile în formatul menționat anterior, rezoluția parametrului ρ , rezoluția parametrului θ în radiani și un prag minim de intersecții utilizate pentru a detecta o linie.

```
HoughLines(houghLinesSrc, houghLines, 1, CV_PI / 180, 80);
```

¹⁵ Blur este un filtru ce estompează anumite caracteristici care sunt prea evidențiate în imagini.

Pentru implementarea transformatei Hough probabilistică este utilizată funcția openCV *HoughLinesP()*. Această funcție estimează și dimensiunea liniilor detectate, astfel încât oferă ca rezultat un vector ce conține puncte în sistemul cartezian, ce reprezintă capetele liniilor. Aceste rezultate sunt utilizate pentru a estima poziția finală a trecerilor de pietoni. Argumentele acestei funcții sunt: imaginea sursă, vectorul ce conține liniile detectate, rezoluția parametrului ρ , rezoluția parametrului θ în radiani, pragul minim de intersecții utilizate pentru a detecta o linie, numărul minim de puncte ce pot forma o linie și numărul maxim de spațiu dintre două puncte care să fie considerate ca făcând parte din aceeași linie.

```
HoughLinesP( houghLinesSrc, lines, 1, CV_PI / 180, 10, 20, 70);
```

În Figura 3.19 sunt prezentate rezultatele obținute prin aplicarea algoritmului Hough. Imaginea (a) conține rezultatul Transformatei Hough clasice, iar imaginea (b) conține rezultatul Transformatei Hough probabilistice.

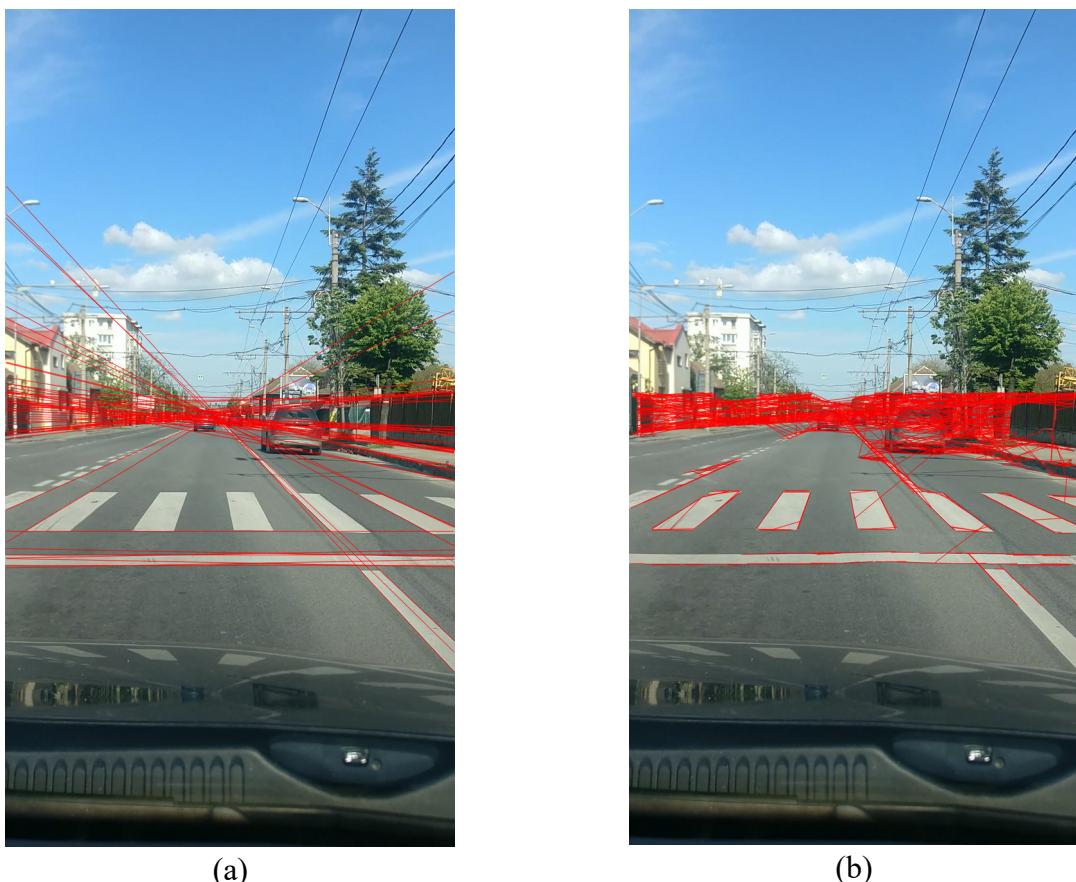


Figura 3.19: Exemplu de rezultate obținute folosind (a) transformata Hough și (b) transformata Hough probabilistică.

3.2.2.5. Vanishing point

Vanishing point-ul unei imagini este un punct din planul de proiecție în care liniile paralele reprezentate în perspectivă par să conveargă. Pentru detecția acestui punct, în lucrarea de față se utilizează o structură bidimensională în care se incrementează poziția tuturor punctelor ce fac parte din dreptele din imagine. Având în vedere condițiile determinate de direcția dreptelor, se parcurge imaginea pe lățime sau pe înălțime. Ulterior se parcurge structura obținută pentru a găsi poziția vanishing point-ului căutat.

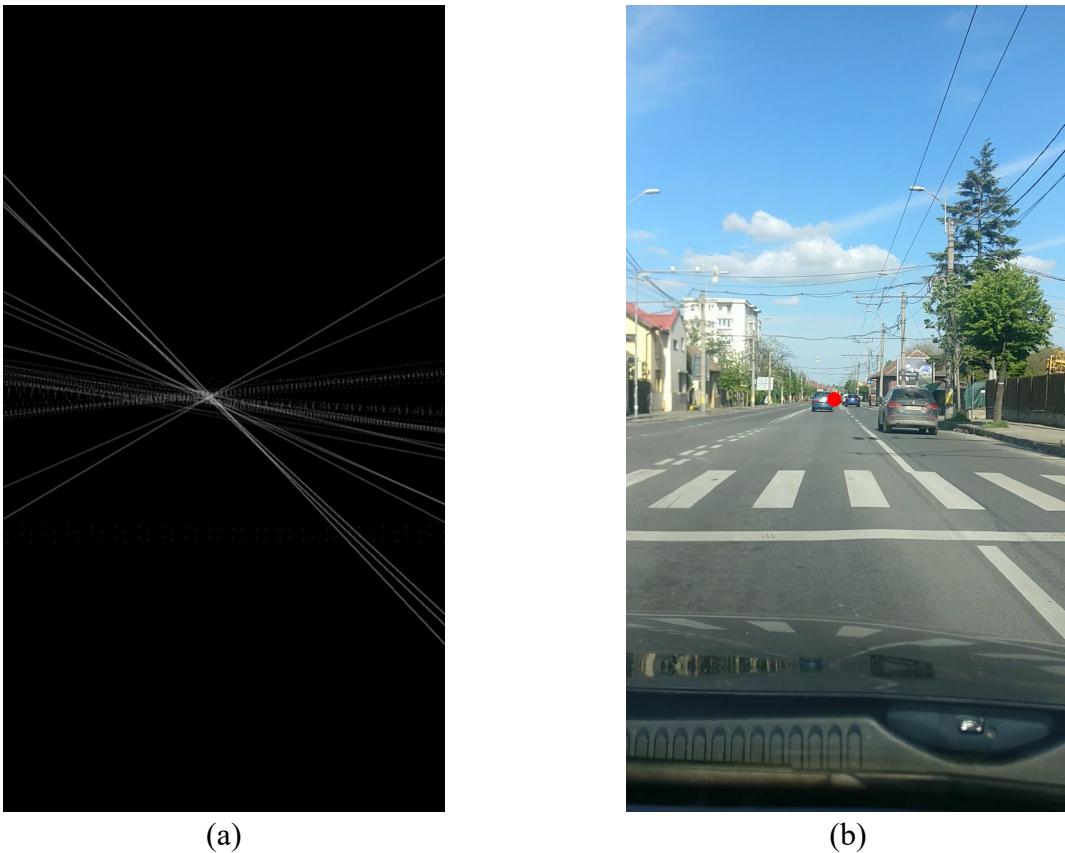


Figura 3.20: Rezultate obținute pentru determinarea vanishing point-ului.

Figura 3.20 conține în imaginea (a) structura determinată de dreptele din imaginea, iar în imaginea (b) se poate observa, marcat în culoare roșie, vanishing point-ul detectat.

Pseudocodul pentru algoritmul utilizat este descris în continuare.

```

dacă abs(tan(θ)) > 1 atunci
    pentru y = 0:img.înălțime
        x = (ρ - y * sin(θ)) / cos(θ)
        buff(y,x)++
    sfărșit
altfel
    pentru x = 0:img.lățime
        x = (ρ - y * cos(θ)) / sin(θ)
        buff(y,x)++
    sfărșit
pentru x = 0:img.înălțime
    pentru y = 0:img.lățime
        dacă max > buff(y,x)
            max = buff(x,y)
    sfărșit
sfărșit

```

3.2.2.6. Operații morfologice

Pentru a aduce îmbunătățiri asupra formei și structurii liniilor detectate anterior sunt necesare anumite operații de post-filtrare numite operații morfologice. În lucrarea de față se

utilizează deschiderea și închiderea. Deschiderea constă într-o eroziune urmată de o dilatare și este folosită pentru eliminarea pixelilor din regiunile care sunt prea mici pentru a conține informație utilă. Pe de altă parte închiderea este realizată prin aplicarea operației de dilatare urmată de o eroziune, și este folosită pentru a umple goluri și mici discontinuități.

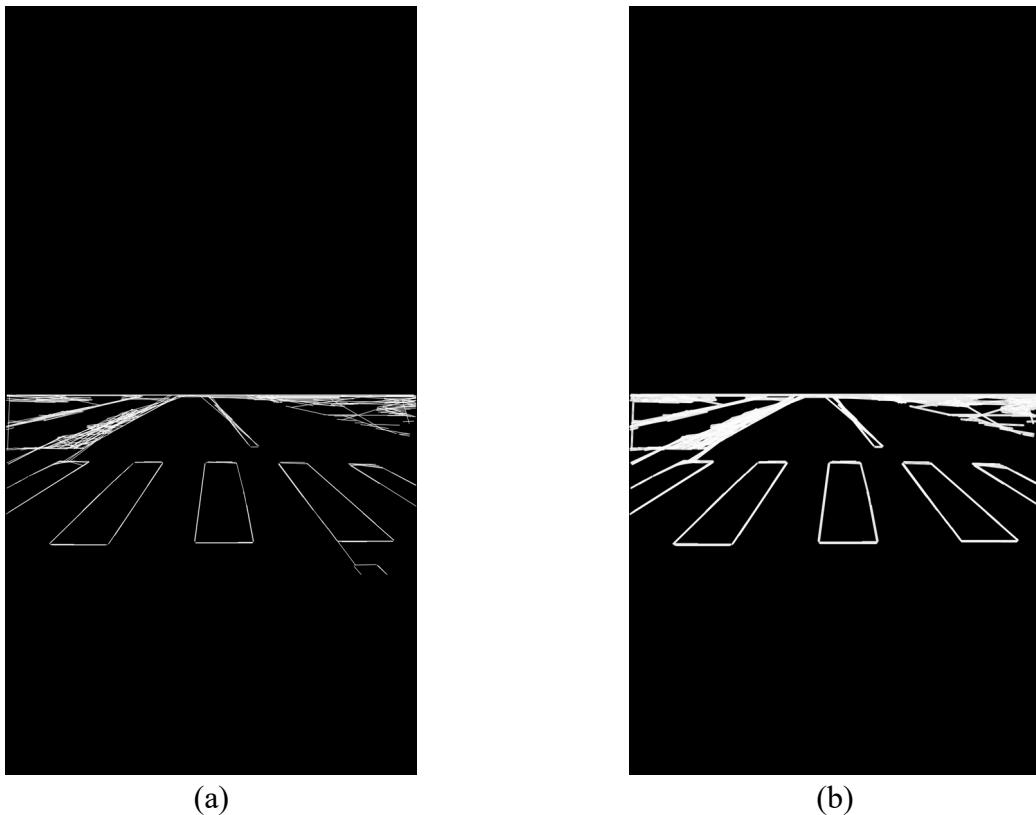


Figura 3.21: Exemplu de aplicare a operațiilor morfologice.

În Figura 3.21 sunt prezentate rezultatele obținute pentru: (a) înainte de aplicarea operațiilor morfologice și (b) după aplicarea operațiilor morfologice.

Pentru efectuarea operației de dilatare, în lucrarea de față, se utilizează funcția implementată de openCV: *dilate()*, ce primește ca parametri imaginea originală, imaginea destinație și elementul structural.

```
cv::Mat element = cv::getStructuringElement(cv::MORPH_CROSS,
                                             cv::Size(3, 3), cv::Point(3, 3));
cv::dilate(input, output, element);
```

Pentru efectuarea operației de eroziune, în lucrarea de față, se utilizează funcția implementată de openCV: *erode()*, ce primește ca parametri imaginea originală, imaginea destinație și elementul structural.

```
cv::Mat element = cv::getStructuringElement(cv::MORPH_CROSS,
                                             cv::Size(3, 3), cv::Point(3, 3));
cv::erode(input, output, element);
```

Elementul structural este de dimensiune 3x3 și are forma cruce având scopul de a elibera pixelii care nu conțin informație utilă pentru operația de eroziune sau de a umple goluri și discontinuități pentru operația de dilatare.

3.2.2.7. Filtrare linii după proprietăți

În final se dorește detectarea unui grup de linii paralele care fac parte din trecerile de pietoni. Liniile paralele în imagini au o proprietate comună: trec prin același punct, numit vanishing point, deoarece construcția în cazul cercetat este realizată ca perspectivă dintr-un punct. Având la dispoziție atât dreptele și cele două puncte care o mărginesc, cât și poziția vanishing point-ului se pot detecta acele drepte care trec prin acest punct.

```
std::vector<line::LineEquation>::iterator it;
for (it = m_lineEquation.begin(); it != m_lineEquation.end(); it++) {
    double euqation = it->get_a() * m_VP.x() + it->get_b() * m_VP.y() + it->get_c();
    double result = 0.0f;
    if (std::abs(euqation - result) < 30.0f)
    {
        m_CrossWalkLines.push_back(it->getLine());
    }
}
```

Dreptele detectate sunt apoi grupate pe baza unui scor bazat pe constrângeri de lungime și număr total. Numărul dreptelor detectate trebuie să fie de cel puțin 5, iar capetele acestora trebuie să fie în proporție de aproximativ 80% similară pe axa oY.



Figura 3.22: Trecere de pietoni detectată prin metoda geometrică.

Având la dispoziție poziția capetelor primei și ultimei drepte se construiește un patrulater care marchează trecerea de pietoni detectată, precum este reprezentat în Figura 3.22.

Codul C++ implementat pentru crearea patrulaterului ce încadrează trecerile de pietoni este prezentat în continuare.

```

Point points[1][4];
points[0][0] = Point(0, 0);
points[0][1] = Point(0, 0);
points[0][2] = Point(0, 0);
points[0][3] = Point(0, 0);
float min_x = lines.at(0).pointStart().x();
float max_x = lines.at(0).pointStart().x();
std::vector<line::Line>::iterator it;
for (it = lines.begin(); it != lines.end(); it++) {
    if (it->pointStart().x() >= max_x) {
        max_x = it->pointStart().x();
        points[0][2] = Point(it->pointStart().x(), it->pointStart().y());
        points[0][3] = Point(it->pointEnd().x(), it->pointEnd().y());
    }
    if (it->pointStart().x() <= min_x) {
        min_x = it->pointStart().x();
        points[0][0] = Point(it->pointStart().x(), it->pointStart().y());
        points[0][1] = Point(it->pointEnd().x(), it->pointEnd().y());
    }
}
const Point* ppt[1] = { points[0] };
int npt[] = { 4 };
cv::fillPoly(image,
    ppt,
    npt,
    1,
    Scalar(0, 0, 255),
    LINE_8);

```

3.2.3. Metoda ce folosește rețele neuronale

Următoarea metodă cercetată folosește algoritmul Mask R-CNN din domeniul inteligenței artificiale. În mod general acest algoritm este organizat în două etape executate secvențial. Inițial se scanează imaginea și se generează propunerii pentru zonele care pot conține obiectul căutat, iar apoi se clasifică propunerile și se generează măști pentru încadrarea cât mai precisă a pixelilor ce fac parte din trecerea de pietoni.

În lucrarea de față, se pornește de la o rețea antrenată anterior pe setul de date COCO cu un număr de clase deja existente și se realizează re-antrenarea cu setul de date propriu pentru a obține clasa dorită. În final, se construiește imaginea ce conține trecerea de pietoni încadrată corespunzător.

3.2.3.1. Diagrama de structură a metodei propuse

Arhitectura generală a metodei discutate în acest subcapitol este evidențiată și descrisă în Figura 3.23.

Metoda Mask R-CNN este compusă din patru module principale: backbone-ul, reprezentat printr-o rețea neuronală convezională; RPN-ul; clasificatorul ROI împreună cu regresorul pentru casetele de încadrare ale obiectelor și detecția măștilor utilizate pentru segmentare.

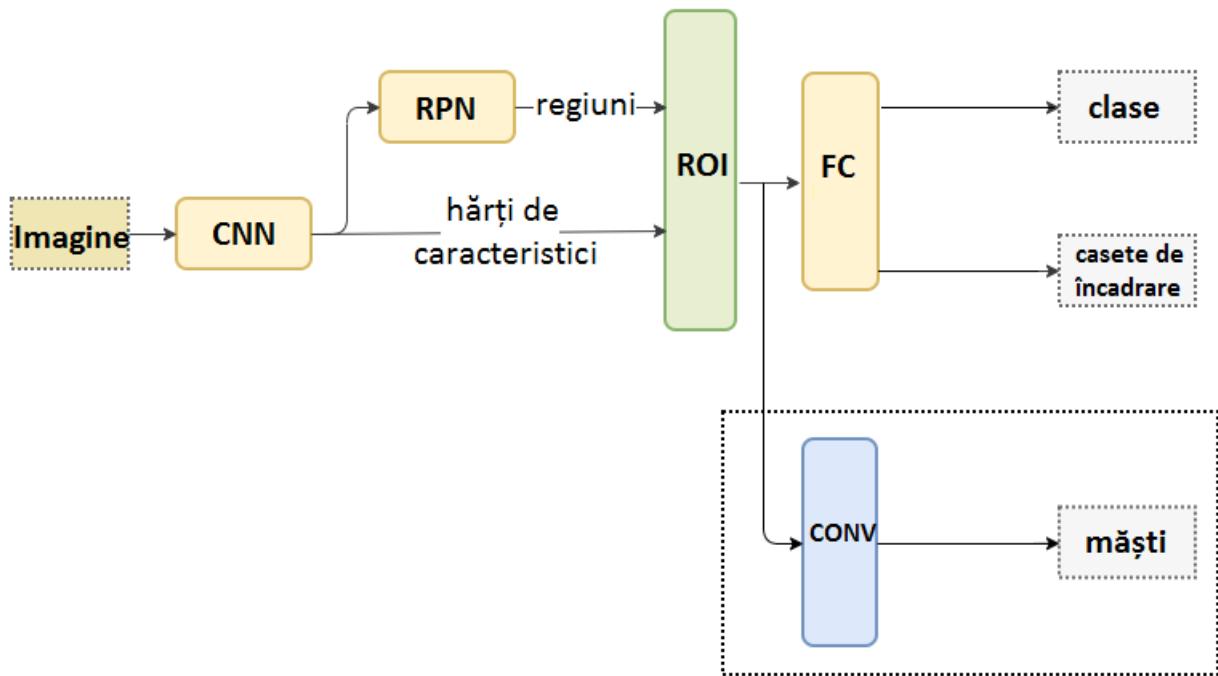


Figura 3.23: Structura Mask R-CNN [26].

Pornind de la imaginea color, în format RGB, se trece printr-o rețea neuronală conlovuțională clasă ce este folosită ca un instrument pentru izolarea caracteristicilor existente. Ulterior, se utilizează o rețea neuronală ce scană întreaga imagine pentru a descoperi zone ce conțin obiecte. Aceste zone sunt încadrate în case distribuite peste tot în imagine. Regiunile de interes detectate sunt transmise către etapele următoare. Modulul următor, ROI, lucrează peste regiunile de interes propuse de către RPN, rafinându-le prin ajustarea dimensiunilor. Apoi, folosind o rețea conectată complet (Fully Connected Network), sunt extrase clasele și casele de încadrare determinante. Ultima etapă, cea de aplicare a unei rețele conlovuționale, este cea care oferă măștile de încadrare ale pixelilor ce fac parte din obiectul detectat [26].

Prin acest procedeu se obține o imagine color ce conține toți pixelii ce fac parte din trecerea de pietoni, marcați folosindu-se culoarea roșie.

3.2.3.2. Detalii de implementare

Limbajul de programare folosit pentru dezvoltarea aplicației este python, utilizându-se librăria tensorflow și librăria Keras. Aplicația software pentru care s-a optat este Anaconda deoarece conține un set complet de instrumente de dezvoltare și testare necesare aplicației. De asemenea, pentru crearea setului de măști utilizate pentru antrenarea rețelei este utilizat via_Region_data care oferă ca rezultat un document de tip JSON ce este utilizat în aplicație în continuare.

Aplicația este executată folosind procesorul Intel Core i7-6700HQ, neavând la dispoziție un GPU puternic, perioada de antrenare a rețelei construite este mai mare. Pentru un set de 100 imagini și măști aferente, și 10 epoci, fiecare epocă cu câte un număr de 100 de pași, timpul de antrenare este de aproximativ 10 ore.

În Figura 3.24 este prezentat instrumentul ce este utilizat pentru adnotarea imaginilor pentru a se obține setul de măști ce este utilizat pentru antrenarea rețelei implementate. Acest

instrument poate fi executat folosind orice browser¹⁶.

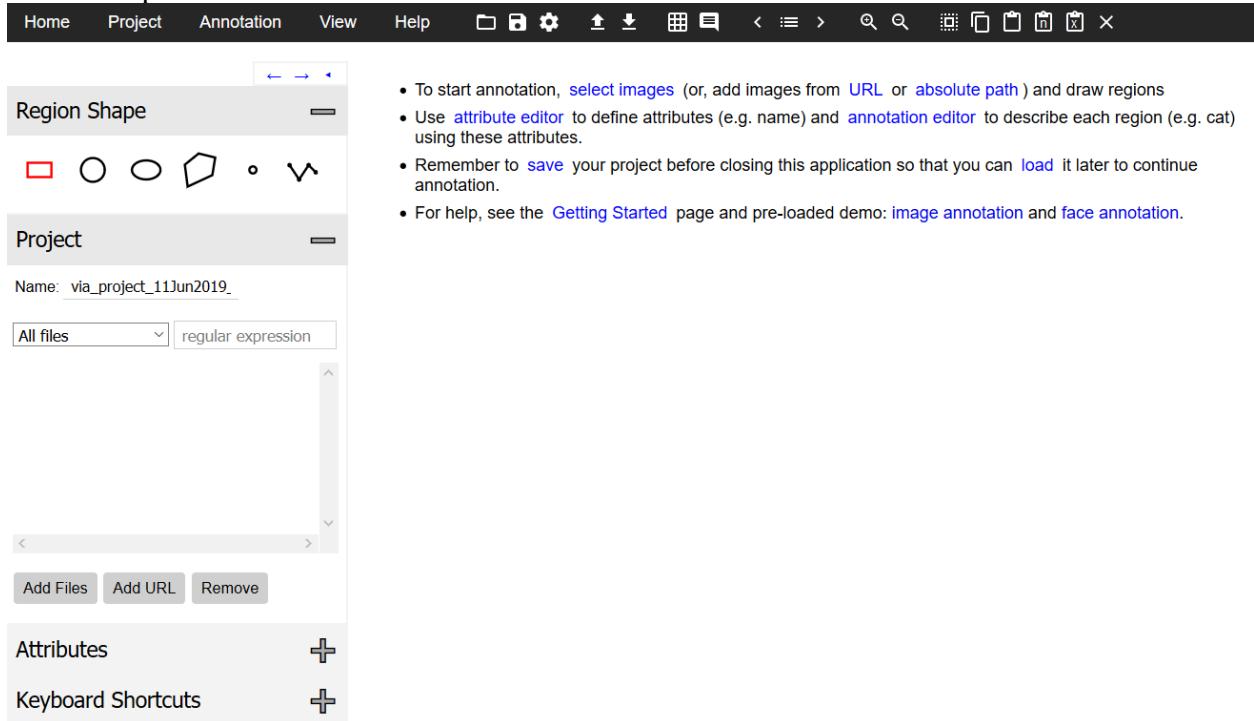


Figura 3.24: Instrumentul de adnotare via_Region_data.

3.2.3.3. Mask R-CNN

Algoritmul Mask R-CNN funcționează în două etape, inițial se scaneză întreaga imagine și se generează propuneri pentru ariile ce ar putea conține un obiect, iar ulterior se clasifică aceste propuneri, se generează casete de încadrare și măști definitorii pentru pixelii ce fac parte din obiectul identificat.

Cele patru module componente sunt implementate în modelul rețelei cu ajutorul librăriilor Keras și TensorFlow.

Pentru a obține date valide, dar fără a realiza o antrenare completă, se pornește de la modelul pre-antrenat folosit setul de date COCO. Aceasta include un număr de 81 de clase, considerând și clasa de fundal ce pot fi utilizate pentru a detecta și segmenta imagini primite ca intrare. Pe de altă parte, clasa dorită nu face parte din acest set de date, astfel încât modelul pre-antrenat nu primește și clasa trecere de pietoni. Tocmai de aceea, în lucrarea de față se realizează o nouă antrenare pentru a se detecta și prezența trecerilor de pietoni în imagini. Astfel, se utilizează un fișier ce conține ponderile modelului pre-antrenat.

Configurațiile utilizate pentru antrenare țin cont atât de scopul aplicației (i.e., detecția trecerii de pietoni), cât și de sistemul utilizat.

```

IMAGES_PER_GPU = 1
NAME = "crosswalk"
NUM_CLASSES = 1 + 1 # Background + crosswalk
STEPS_PER_EPOCH = 10
DETECTION_MIN_CONFIDENCE = 0.9
VALIDATION_STEPS = 50
  
```

¹⁶ Un browser este o aplicație software ce este folosită pentru a naviga pe internet.

```
BACKBONE = "resnet101"
```

Astfel, datorită lipsei unui GPU, numărul maxim de imagini pentru o procesare este 1. Numărul de clase utilizate este 1 deoarece vrem să identificăm doar obiectele de tip trecere de pietoni. Numele ales pentru configurație este sugestiv, trecere de pietoni (în engleză, crosswalk). Numărul de pași execuți pentru fiecare epocă este 10, iar detecțiile care nu obțin un nivel de încredere de cel puțin 90% sunt omise. De asemenea, numărul de pași pentru validare este 50. Validarea este executată la finalul fiecării epoci, de aceea un număr mare de astfel de pași va îmbunătăți rezultatele, dar va încetini timpul de antrenare. Arhitectura utilizată pentru backbone-ul acestei aplicații este resnet101, dar pentru alte aplicații poate fi utilizată și arhitectura resnet50.

Următorul pas este de a realiza antrenarea pentru propria clasă. Pentru realizarea antrenării este necesar un set de date ce conține măștile utilizate pentru setul de imagini pentru antrenare. Pentru crearea acestor măști se adnotează un număr de 100 de imagini color, cu prezența trecerilor de pietoni, folosind aplicația Region data.

Figura 3.25 conține un exemplu de adnotare a unei imagini din setul de date utilizat pentru antrenare.

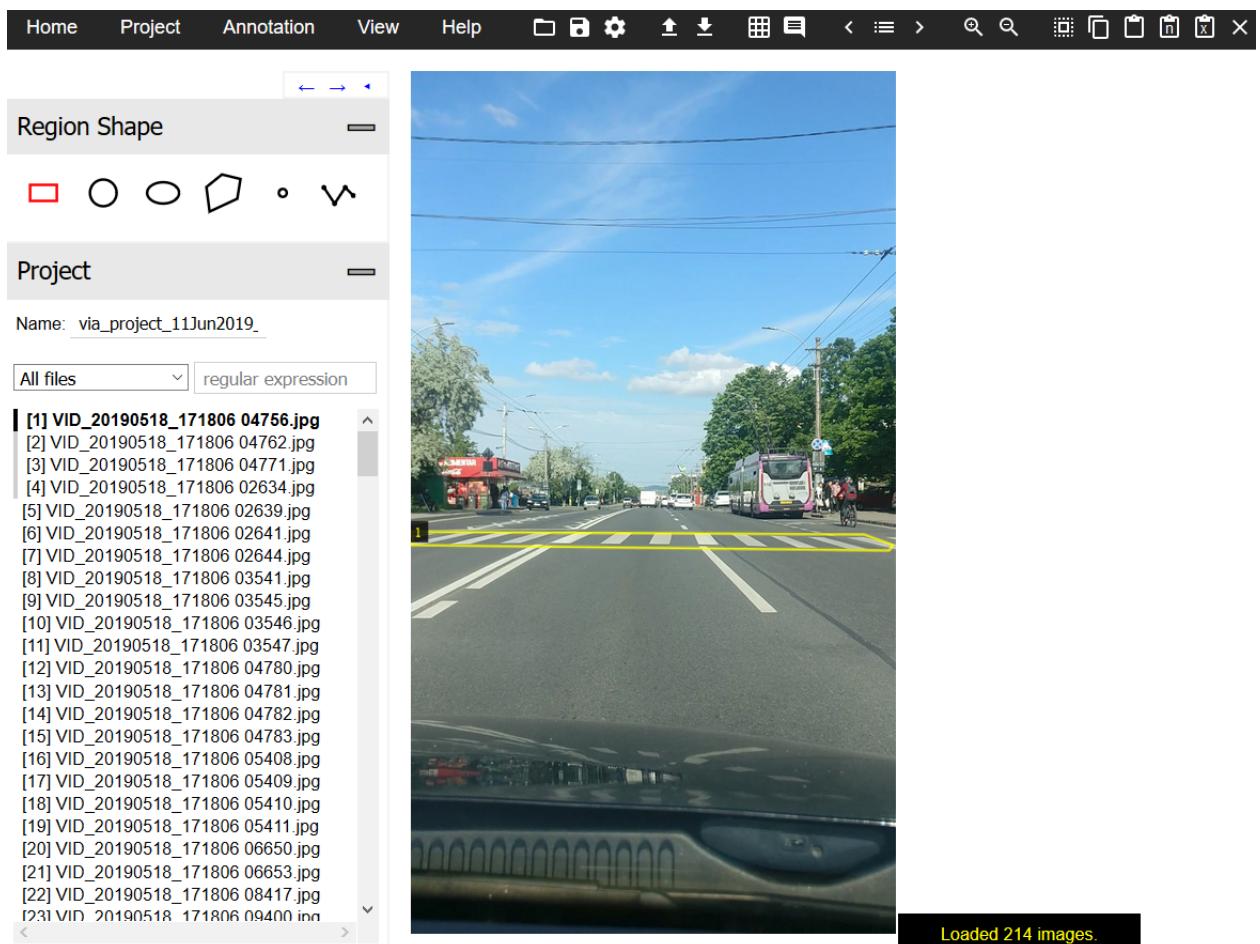


Figura 3.25: Exemplu de adnotare a prezenței trecerii de pietoni.

Aplicația utilizată generează un fișier de tip JSON ce conține informația necesară pentru antrenare, anume punctele poligonului care reprezintă masca. Aceste puncte sunt preluate de către rețea implementată și procesate, astfel ca la rândul ei rețea să genereze pentru orice imagine de intrare acest tip de puncte care să reprezinte masca, pentru trecerea de pietoni detectată.

Punctele, în sistemul de coordonate cartezian, sunt reprezentate prin valorile atributelor: *all_points_x* și *all_points_y*. De asemenea, valoarea atributului *filename*, este utilizată ca un identificator pentru fiecare imagine. Un exemplu de format al unui obiect construit pentru o imagine este reprezentat astfel:

```
{
  "VID_20190518_171806 04756.jpg297818":
  {
    "filename": "VID_20190518_171806 04756.jpg",
    "size": 297818,
    "regions": [
      {
        "shape_attributes": {
          "name": "polygon",
          "all_points_x": [0, 1064, 1075, 1009, 3, 5],
          "all_points_y": [1055, 1069, 1055, 1033, 1025, 1047]
        },
        "region_attributes": {}
      }],
    "file_attributes": {}
  }
}
```

Prin interpretarea JSON-ului se reține, pentru fiecare imagine, un poligon ce reprezintă poziția trecerii de pietoni.

```
annotations = json.load(open(os.path.join(dir, "via_region_data.json")))
annotations = list(annotations.values())
for a in annotations:
  if type(a['regions']) is dict:
    polygons = [r['shape_attrib'] for r in a['regions'].values()]
  else:
    polygons = [r['shape_attrib'] for r in a['regions']]
  image_path = os.path.join(dataset_dir, a['filename'])
  image = skimage.io.imread(image_path)
  height, width = image.shape[:2]
  self.add_image("crosswalk",
    image_id=a['filename'],
    path=image_path,
    width=width, height=height,
    polygons=polygons)
```

Poligonul determinat este transformat într-o mască în format bitmap cu forma [*înălțime*, *lățime*, *număr de instanțe*]. Aceste informații sunt utilizate ulterior pentru antrenarea modelului.

```
info = self.image_info[image_id]
mask = np.zeros([info["height"], info["width"], len(info["polygons"])],
               dtype=np.uint8)
for i, p in enumerate(info["polygons"]):
  rr, cc = skimage.draw.polygon(p['all_points_y'], p['all_points_x'])
```

```

mask[rr, cc, i] = 1
return mask.astype(np.bool), np.ones([mask.shape[-1]], dtype=np.int32)

```

Pentru antrenarea modelului și validarea modelului se pornește de la ponderile pre-antrenate și se re-antrenează rețeaua folosind 20 de epoci și setul de date adnotat anterior.

```

dataset_train = CrosswalkDataset()
dataset_train.load_crosswalk(args.dataset, "train")
dataset_train.prepare()

dataset_val = crosswalkDataset()
dataset_val.load_crosswalk(args.dataset, "val")
dataset_val.prepare()

model.train(dataset_train, dataset_val,
            learning_rate=config.LEARNING_RATE,
            epochs=20,
            layers='heads')

```

Pe parcursul celor 20 de epoci eroare este minimizată, așa cum se observă în Figura 3.26, astfel încât după antrenare acuratețea crește.

```

Epoch 3/10
1/10 [==>.....] - ETA: 8:09 - loss: 2.3513
2/10 [=====>...] - ETA: 7:13 - loss: 2.8199
3/10 [======>...] - ETA: 6:21 - loss: 2.9159
4/10 [======>...] - ETA: 5:27 - loss: 2.8327
5/10 [======>...] - ETA: 4:32 - loss: 2.6530
6/10 [======>...] - ETA: 3:37 - loss: 2.4609
7/10 [======>...] - ETA: 2:43 - loss: 2.4270
8/10 [======>...] - ETA: 1:48 - loss: 2.4110
9/10 [======>...] - ETA: 54s - loss: 2.7143 -

```

Figura 3.26: Minimizare eroare pe parcursul antrenării.

Comanda utilizată pentru antrenarea rețelei implementate:

```
python crosswalk.py train --dataset=crosswalk --weights=mask_rcnn_crosswalk.h5
```

Pentru a relua antrenarea dintr-un anumit punct unde aceasta a fost întreruptă, se utilizează comanda:

```
python crosswalk.py train -dataset=crosswalk --weights=last
```

Pentru a marca prezența trecerii de pietoni în imagine, se coloarează cu roșu toți acei pixeli care fac parte din masca determinată anterior.

```

red_img = Image.new('RGB', (1080, 1920), "red")
# Copiază pixelii de culoare roșie din imaginea ce conține doar pixeli roșii în
# imaginea de intrare, acolo unde este alocată masca
if mask.shape[-1] > 0:
    mask = (np.sum(mask, -1, keepdims=True) >= 1)

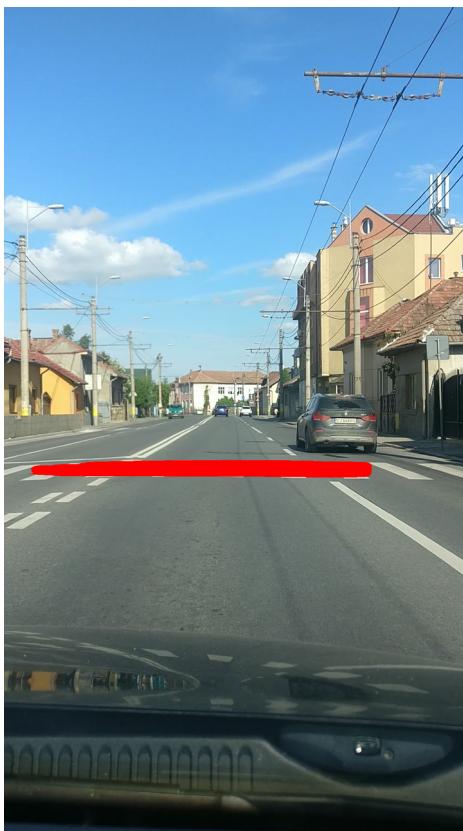
```

```
final_image = np.where(mask, red_img, image).astype(np.uint8)
else:
    final_image = image.astype(np.uint8)
return final_image
```

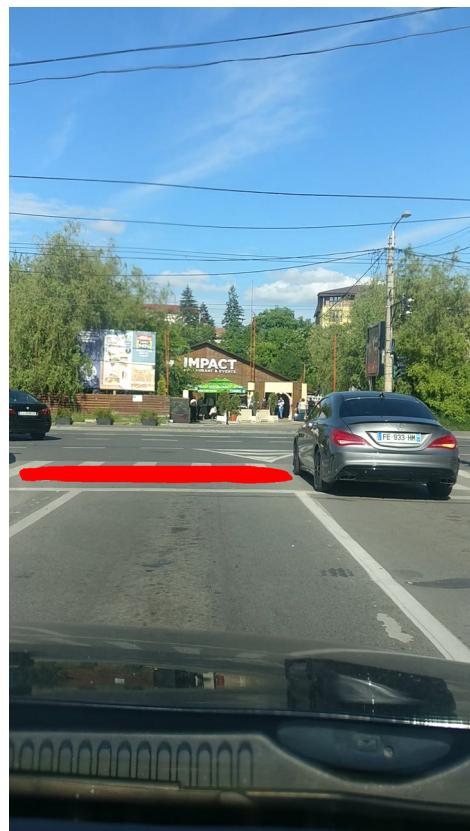
Comanda folosită pentru aplicarea rețelei și detectarea trecerii de pietoni din imaginea procesată:

```
python crosswalk.py splash --weights=mask_rcnn_crosswalk_0020.h5
--image=crosswalk/image.jpg
```

Figura 3.27 conține două exemple de rezultate obținute prin aplicarea metodei descrise în acest subcapitol.



(a)



(b)

Figura 3.27: Exemplu de rezultate obținute folosind metoda ce utilizează Mask R-CNN.

Funcția principală conține următoarele instrucțiuni pentru a interpreta comenziile din terminal.

```
if __name__ == '__main__':
    import argparse

    parser = argparse.ArgumentParser( description='Train Mask R-CNN.' )
    parser.add_argument('--dataset', required=False,
                        metavar="/path/to/crosswalk/dataset/")
    parser.add_argument('--weights', required=True,
                        metavar="/path/to/weights.h5")
```

```
parser.add_argument('--logs', required=False,
                    metavar="/path/to/logs/")
parser.add_argument('--image', required=False,
                    metavar="path or URL to image")
args = parser.parse_args()
```

Capitolul 4. Rezultate experimentale

În această secțiune sunt descrise rezultatele experimentale obținute pentru evaluarea metodelor propuse pentru detecția trecerii de pietoni. Rezultatele obținute sunt testate pentru ambele modalități de detecție propuse, atât pentru metoda prezentată în subcapitolul 3.1.1 Metoda geometrică de detecție, cât și pentru metoda descrisă în subcapitolul 3.1.2 Metoda ce folosește rețelele neuronale.

4.1. Descrierea setului de date de test

Pentru evaluarea modulelor implementate se utilizează un set de date ce conține 20 imagini ce sunt utilizate pentru testare. Acest set de date este diferit față de setul de date, ce conține 100 de imagini, utilizat pentru antrenarea rețelei neuronale din cea de-a doua metodă.

Cele 20 de imagini de test sunt achiziționate cu ajutorul camerei din spate a telefonului One Plus 3. Dimensiunea imaginilor este de 1080x1920 pixeli.

Aplicația este executată pentru fiecare imagine din setul de date, iar rezultatele fiecărui modul sunt comparate pentru a alege modalitatea cea mai fezabilă pentru a îndeplini obiectivul propus inițial.

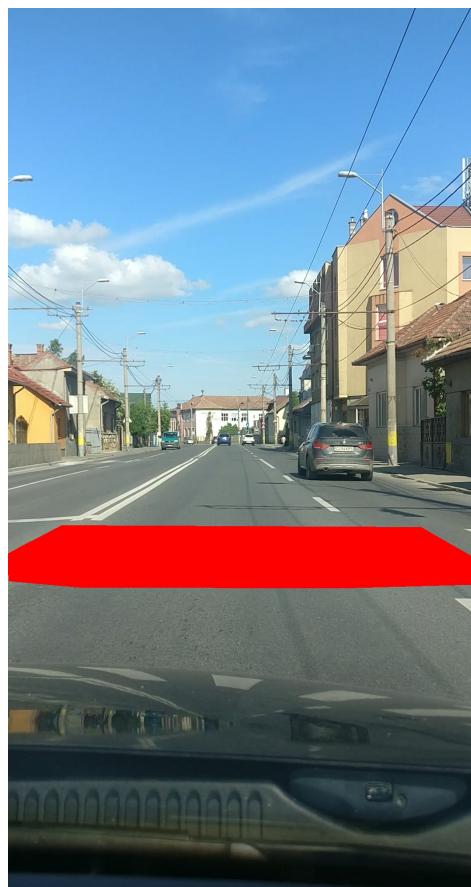


Figura 4.1: Imagine adnotată manual.

Pentru a putea evalua acuratețea detecției, descrisă pe larg în secțiunea 4.3 Acuratețea detecției, s-au adnotat manual cele 20 de imagini de test pentru fiecare metodă separat. În Figura 4.1 este ilustrat un exemplu de imagine de test ce conține o trecere de pietoni, adnotată manual pentru evaluarea ambelor metode.

4.2. Timpi de execuție

Pentru fiecare modul se calculează timpul mediu de execuție. Astfel, aceste rezultate dețin un impact major în alegerea metodei care se aplică cel mai eficient în funcție de situație.

Pentru a testa timpul de execuție al metodei geometrice se utilizează libraria *time.h* ce conține funcțiile necesare calculului propus.

```
const clock_t begin_time = std::clock();
//compute the crosswalk detection
std::cout << "\nTime in ms: " << 1000.0 * float(std::clock() - begin_time) /
CLOCKS_PER_SEC << endl;
```

Pentru a calcula timpul de detecție prin metoda ce folosește rețelele neutronale se utilizează librăria *time*.

```
import time
begin_time = time.time()
//compute the crosswalk detection
end_time = time.time()
print("Time in ms: ", end_time - begin_time)
```

Tabelul 4.1 conține atât media timpilor de execuție obținuți prin execuția modulelor componente ale metodei geometrice, cât și media timpilor de execuție înregistrati pentru metoda ce folosește rețelele neuronale.

Tabelul 4.1: Media timpilor de execuție a modulelor componente.

Metoda testată	Timp execuție (secunde)
Metoda geometrică	1.9117s
Metoda ce folosește rețelele neuronale	18.4573s

Rezultatele obținute pentru primele 10 imagini din setul de date utilizat se regăsesc în Tabelul 4.2 ce conține timpii de execuție pentru datele prelucrate prin metoda geometrică.

Tabelul 4.2: Timpul de execuție pentru metoda geometrică pentru primele 10 imagini din setul de date.

Metoda geometrică	Timp execuție
Img1	1.947s
Img2	2.423s
Img3	1.661s
Img4	2.703s
Img5	1.669s
Img6	1.868s
Img7	1.622s
Img8	1.671s
Img9	1.963s
Img10	2.365s

Rezultatele obținute pentru fiecare imagine se regăsesc în Tabelul 4.3 ce conține timpii de execuție pentru datele prelucrate prin metoda ce utilizează rețele neuronale. Pe de altă parte, timpul de antrenare al rețelei este de aproximativ 17 ore și 46 de minute. Pentru antrenarea este folosind un sistem ce nu are GPU, iar ca parametri de antrenare sunt utilizati: 20 de epoci, 100 etape de antrenare și 50 de etape de validare.

Tabelul 4.3: Timpul de execuție pentru metoda ce folosește rețele neuronale pentru primele 10 imagini din setul de date.

Metoda ce folosește rețele neuronale	Timp execuție
Img1	17.987s
Img2	18.792s
Img3	18.183s
Img4	19.003s
Img5	18.183s
Img6	17.594s
Img7	19.039s
Img8	18.776s
Img9	18.779s
Img10	18.273s

Având în vedere aceste rezultate obținute se poate concura că metoda geometrică este mai rapidă.

4.3. Acuratețea detecției

Fiecare metodă implementată a fost evaluată din punct de vedere calitativ folosind același set de date ce conține câte 20 imagini. S-au comparat rezultatele obținute cu imagini adnotate manual.

Pentru a evalua calitatea rezultatelor obținute, se folosește matricea de confuzie. Această matrice conține [27]:

- Numărul de date clasificate corect ca aparținând clasei de interes: (TP),
- Numărul de date clasificate corect ca neaparținând clasei de interes: (TN),
- Numărul de date clasificate incorect ca aparținând clasei de interes: (FP),
- Numărul de date clasificate incorect ca neaparținând clasei de interes: (FN).

Pe baza acestei matrice de confuzie se pot deriva diferite măsurători. Pentru a obține informațiile necesare pentru evaluarea calității rezultatelor obținute se calculează **senzitivitatea sau TPR (True Positive Rate)**, **specificitatea sau TNR (True Negative Rate)** și **FPR-ul (False Positive Rate)**. Senzitivitatea este capacitatea clasificatorului de a identifica toate cazurile care sunt pozitive. Specificitatea reprezintă capacitatea clasificatorului de a nu identifica ca fiind pozitive toate cazurile care sunt, în realitate, negative. Pe de altă parte, FPR-ul constituie capacitatea clasificatorului de a identifica ca fiind negative cazurile care sunt, în realitate, pozitive.

Pentru ca rezultatele să fie cât mai precise, trebuie ca valorile TPR și TNR să fie apropriate de 1.0, iar valoarea FPR trebuie să fie apropiată de 0.0 [27]:

$$\begin{aligned}
 TPR &= \frac{TP}{TP+FN} \\
 TNR &= \frac{TN}{TN+FP} \\
 FPR &= \frac{FP}{TN+FP}
 \end{aligned} \tag{17}$$

Formula (18) conține modalitatea de calcul a mediei metricilor utilizate pentru evaluarea calitativă a rezultatelor.

$$\begin{aligned}
 TPR_m &= \frac{\sum TPR_i}{10} \\
 TNR_m &= \frac{\sum TNR_i}{10} \\
 FPR_m &= \frac{\sum FPR_i}{10}
 \end{aligned} \tag{18}$$

Astfel, Tabelul 4.4 conține mediile rezultatelor obținute pentru fiecare dintre metodele implementate.

Tabelul 4.4 Mediile rezultatelor obținute pentru testarea acurateței metodelor implementate.

Metoda testată	TPR_m	FPR_m	TNR_m
Metoda geometrică	0.8645	0.0079	0.9919
Metoda ce folosește rețelele neuronale	0.8101	0.0068	0.9964

Tabelul 4.5 conține rezultatele obținute pentru testarea acurateței metodei geometrice pentru primele 10 imagini din setul de date.

Tabelul 4.5 Rezultatele obținute pentru testarea acurateței metodei geometrice pentru primele 10 imagini din setul de date.

Metoda geometrică	TPR_m	FPR_m	TNR_m
Img1	0.7546	0.0033	0.9966
Img2	0.6366	0.0027	0.9971
Img3	0.6893	0.0002	0.9997
Img4	0.7589	0.0023	0.9976
Img5	0.8223	0.0061	0.9938
Img6	0.7119	0.0012	0.9987
Img7	0.8989	0.0041	0.9958
Img8	0.9032	0.0066	0.9933
Img9	0.8666	0.0024	0.9975
Img10	0.71006	0.0009	0.9990

Tabelul 4.6 conține rezultatele obținute pentru testarea acurateței metodei geometrice pentru fiecare imagine din setul de intrare.

Tabelul 4.6 Rezultatele obținute pentru testarea acurateței metodei ce folosește rețelele neuronale pentru primele 10 imagini din setul de date.

Metoda ce folosește rețelele neuronale	TPR_m	FPR_m	TNR_m
Img1	0.6316	0.0036	0.9963
Img2	0.7634	0.0020	0.9979
Img3	0.6691	0.0018	0.9981
Img4	0.6316	0.0017	0.9982
Img5	0.6389	0.0066	0.9933
Img6	0.8001	0.0143	0.9856
Img7	0.6773	0.0063	0.9936
Img8	0.8994	0.0012	0.9987
Img9	0.8078	0.0049	0.9950
Img10	0.9206	0.0105	0.9894

Pentru a calcula acuratețea rezultatelor s-a utilizat următoarea implementare, având în vedere faptul că pentru reprezentarea trecerii de pietoni în imagini s-a utilizat culoarea cu modelul BGR (0, 0, 255):

```

int TP = 0, TN = 0, FP = 0, FN = 0;
for (int x = 0; x < img1.size().height; x++) {
    for (int y = 0; y < img1.size().width; y++) {
        Vec3b color = img1.at<Vec3b>(x, y);
        Vec3b color_adno = img2.at<Vec3b>(x, y);
        if (color[0] == 0 && color[1] == 0 && color[2] == 255 && color_adno[0] == 0
            && color_adno[1] == 0 && color_adno[2] == 255) {
            TP++;
        }
        else if (color[0] == 0 && color[1] == 0 && color[2] == 255) {
            FP++;
        }
        else if (color_adno[0] == 0 && color_adno[1] == 0 && color_adno[2] == 255) {
            FN++;
        }
        else {
            TN++;
        }
    }
}
cout << "\nTPR = " << (double)TP / (TP + FN) << endl;
cout << "\nTNR = " << (double)TN / (TN + FP) << endl;
cout << "\nFPR = " << (double)FP / (TN + FP) << endl;

```

În Figura 4.2 este reprezentat un exemplu de comparare, pentru două imagini din setul de date, a rezultatelor obținute prin (a) și (d) adnotare manuală, (b) și (e) metoda ce folosește rețele neuronale și (c) și (f) metoda geometrică.

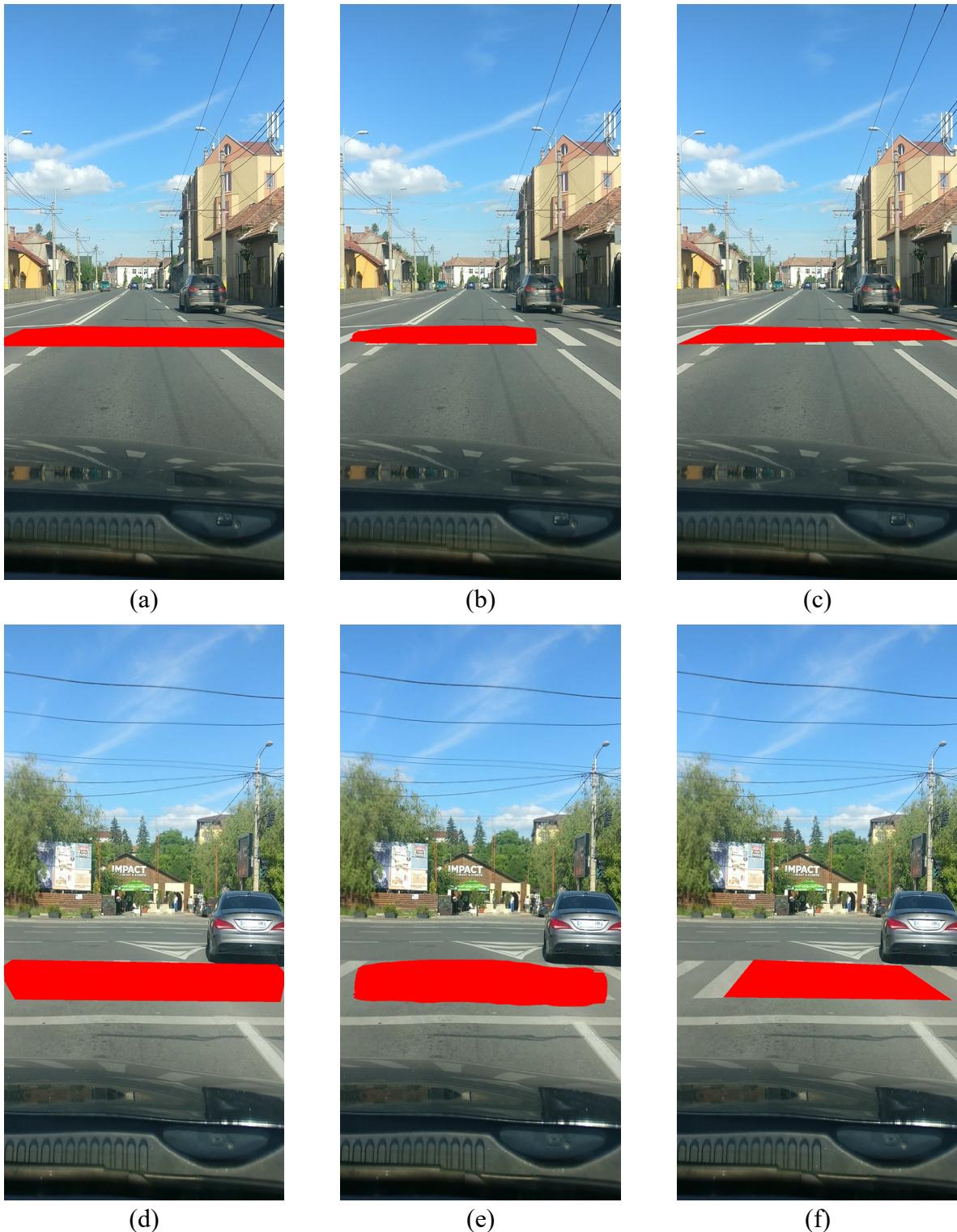


Figura 4.2: Două exemple de comparare a rezultatelor obținute pentru cele două metode propuse.

Având în vedere obiectivul propus, anume cel de utilizare al aplicației în domeniul navigării vehiculelor autonome, se dorește minimizarea FPR-ului ce reprezintă capacitatea

clasificatorului de a identifica ca fiind negative cazurile care sunt în realitate pozitive. Acest fapt este argumentat prin asigurarea siguranței pietonilor. Se preferă ca un vehicul autonom să oprească dacă se detectează o trecere de pietoni chiar și în locurile unde în realitate nu există, decât să treacă fără a opri la o trecere de pietoni. Cazul al doilea ar putea duce la grave accidente. Pentru a minimiza FPR-ul se măresc limitele utilizate în cadrul algoritmilor de detectie a trecerilor de pietoni, chiar dacă aceasta ar putea mări expunerea la riscul detecției incorecte a unor alte obiecte ca fiind treceri de pietoni.

Totuși, pentru obținerea unor rezultate cât mai stabile, se realizează un trade-off¹⁷ între metricile TPR și FPR și în acest mod se impune un prag pentru TPR, ulterior minimizându-se FPR-ul cât mai mult.

17 Trade-off-ul este un echilibru realizat între două proprietăți dorite, dar incompatibile; un compromis.

Capitolul 5. Concluzii

Detectia trecerilor de pietoni se numara printre subiectele principale ce apar in domeniul viziunii artificiale. Datorita importantei sigurantei participantilor la trafic, acest subiect este tratat in diferite lucrari de specialitate. In acest mod sunt propuse o serie de metode pentru a realiza cat mai eficient si precis detectia trecerilor de pietoni.

Focusul principal al lucrarii de facta este compararea a doua solutii de detectare a prezentei trecerilor de pietoni in imagini si de a decide care este metoda cea mai potrivita din perspectiva unui studiu de caz aplicativ **in domeniul navigarii vehiculelor autonome**. In acest context sunt definite o serie de solutii ce vin in sprijinul celor ce doresc sa propuna o noua modalitate de detectare a trecerilor de pietoni, dar si care pot fi folosite ca suport pentru imbunatatirea metodelor existente.

Pe langa acestea s-a dorit si conturarea unor solutii ce asigura solutarea unor probleme clasice din domeniul viziunii artificiale (e.g., detectia frontierelor in imagini, determinarea liniilor din imagini, localizarea vanishing point-ului, adnotarea unui set de imagini cu casete de incadrare corespunzatoare scopului propus etc).

Prima metoda propusa isi indreapta focusul catre utilizarea geometriei analitice ca pilon principal in rezolvarea sarcinii propuse. Ca prim pas se realizeaza o preprocesare a imaginii pentru fi aliniata cu cerintele aplicarii algoritmului Canny pentru a extrage frontierele din imagine. Ulterior, transformata Hough clasică este utilizata pentru a obtine linii in forma polară si pentru a calcula cu ajutorul acestora vanishing point-ul imaginii. Pe de alta parte, transformata Hough probabilistica se foloseste pentru a obtine setul de linii din imagine, impreună cu dimensiunea acestora. Ulterior, se determina grupul de linii din setul obtinut ce indeplineste conditia de a trece prin vanishing point. Daca se obtin un grup de cel putin 5 linii, ce indeplinesc conditiile de filtrare, se aplică operațiile morfologice pentru a imbunatati rezultatele si se construiește imaginea ce conține liniile detectate finale si trecerea de pietoni incadrata corespunzator.

Cea de-a doua metoda cercetata face parte din domeniul retelelor neuronale si foloseste algoritmul Mask R-CNN. Acest framework este organizat in doua etape: prima etapa scanazeaza imaginea si genereaza propunerii pentru zonele care pot contine obiectul cautat si cea de-a doua etapa clasifica propunerile si genereaza masuri pentru incadrarea cat mai corecta a pixelilor. Astfel, se porneste de la o retea antrenata anterior pe setul de date COCO, cu un numar de clase deja existent si se realizeaza re-antrenarea cu setul de date propriu pentru a obtine clasa noua, „trecere de pietoni”. Ulterior se construieste imaginea ce conține trecerea de pietoni incadrata corespunzator.

Astfel, tehniciile analizate in continutul acestei lucrari pot fi utilizate in continuare pentru a optimiza algoritmi si metode deja existente in literatura sau pentru a dezvolta noi sisteme utilizabile in domeniul navigarii vehiculelor autonome.

Avantajele utilizarii metodei geometrice constau in: usurinta de inteleghere a conceptelor matematice utilizate, gradul mare de adaptabilitate la limbajul de programare folosit si raportul convenabil dintre calitatea rezultatelor si timpul de executie. Pe de alta parte un dezavantaj major consta in rigiditatea rezultatelor obtinute in functie de conditiile de mediu. Odata cu aparitia curbelor, a marcapunctelor neclare, a umbrelor si chiar a vremii nefavorabile, apar si rezultate influente negativ.

Pentru a obtine avantajul de adaptare la conditii dificile de mediu se recomanda utilizarea metodei ce foloseste algoritmul mask R-CNN. Totusi, aceasta metoda are si o serie de dezavantaje precum necesitatea unui timp mare de antrenare a retelei si complexitatea

algoritmilor utilizați.

Având în vedere rezultatele obținute atât din punct de vedere calitativ, cât și al timpului necesar pentru a aplica metoda în cauză, putem conchide că metoda geometrică este mai rapidă, însă oferă rezultate mai puțin calitative față de metoda ce folosește rețele neuronale. De asemenea, pentru metoda ce folosește mask R-CNN este necesar un timp de antrenare ridicat pentru obținerea unor rezultate satisfăcătoare. Pe de altă parte, timpul de antrenarea poate fi scăzut prin adăugarea unor noi resurse computaționale (i.e., GPU cu putere mare).

În final, putem conchide că pentru obiectivul propus, este necesară utilizarea unor soluții care să ofere rezultate cât mai corecte, dar în același timp și să ofere posibilitatea execuției în sisteme de timp real. Tocmai de aceea, propunem utilizarea soluțiilor ce realizează cât mai puțini pași computaționali și operații cât mai simple. În acest context, dintre modalitățile de a detecta treceri de pietoni enumerate, considerăm implementarea metodei geometrice ca fiind cea mai potrivită.

5.1. Direcții de dezvoltare

Un principal obiectiv pentru a obține rezultate cât mai corecte, într-un mod cât mai eficient, este de a îmbunătăți ambele metode atât din punct de vedere al implementării algoritmilor utilizați, cât și prin adăugarea unor noi concepte pentru a aduce un plus de calitate.

Ulterior, se poate utiliza o imagine de adâncime pentru a aduce un plus de informație ce poate fi utilizată pentru detecția trecerilor de pietoni. De asemenea, pentru determinarea vanishing point-ului poate fi utilizat algoritmul RANSAC. Pe de altă parte, din punct de vedere al rețelei neuronale implementate se poate investiga care este numărul optim de epoci și de pași pentru antrenare.

Având în vedere faptul că rezoluția imaginilor utilizate este mare, timpul de procesare crește la rândul său. Totuși, timpul determinat de aproximativ două secunde pentru procesare unei imagini și oferire a rezultatelor, aplicația ce folosește metoda geometrică poate fi extinsă astfel încât să fie executată în sisteme de timp real prin achiziție continuă de frame-uri RGB cu ajutorul telefonului la fiecare două secunde. Pe de altă parte, metoda ce utilizează rețele neuronale nu poate fi utilizată într-un sistem de timp real, la rezoluția utilizată. Datorită timpului de 20 secunde necesar procesării unei imagini, un sistem de timp real ce ar folosi această aplicație nu ar fi fezabil. Totuși pentru a scădea timpul de procesare pentru ambele metode, s-ar putea utiliza un set de date de intrare cu o rezoluție semnificativ mai mică.

Se consideră importantă și posibilitatea de a executa ambele metode folosind procesoare CPU și GPU mai performante.

Bibliografie

- [1] Jason Banich, „ZEBRA CROSSWALK DETECTION ASSISTED BY NEURAL NETWORKS”, Faculty of California Polytechnic State University, San Luis Obispo, , pp. -, 2016.
- [2] Stephen Se, „Zebra-crossing Detection for the Partially Sighted”, IEEE, 2, pp. 211-217, 2000.
- [3] Shuihua Wang, Hangrong Pan, Chenyang Zhang, Yingli Tian, „RGB-D Image-Based Detection of Stairs, Pedestrian Crosswalks and Traffic Signs”, Journal of Visual Communication and Image Representation, 25, pp. 263-272, 2014.
- [4] James M. Coughlan, Huiying Shen, „A Fast Algorithm for Finding Crosswalks using Figure-Ground Segmentation”, Smith-Kettlewell Eye Research Institute San Francisco,, pp. -, 2006.
- [5] Volodymyr Ivanchenko, James Coughlan, Huiying Shen, „Detecting and Locating Crosswalks using a Camera Phone”, IEEE, 5105, pp. 1122-1128, 2008.
- [6] Mihály Radványi, Balázs Varga, Kristóf Karacs, „Advanced crosswalk detection for the Bionic Eyeglass”, IEEE, 10, pp. -, 2010.
- [7] John Canny, „A Computational Approach to Edge Detection”, IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, PAMI-8 , pp. 679, 1986.
- [8] Florica Moldoveanu, Detectia frontierelor din imagini [Online], Disponibil la adresa: andrei.clubcisco.ro/cursuri/.../9.detectia.frontierelor.in.imagini.PDF, Accesat: 2019.
- [9] R. O. Duda, P. E. Hart, „Use of the Hough Transformation to Detect Lines and Curves in Pictures”, Comm. ACM, , pp. 11-15, 1972.
- [10] P.V.C. Hough, „Method and means for recognizing complex patterns”, u.s. patent 3069654, , pp. -, 1962.
- [11] Ghassan Hamarneh, Karin Althoff, Rafeef Abu-Charbieh, „Automatic Line Detection”, Image Analysis Group Department of Signals and Systems Chalmers University of Technology, , pp. 7-11, 1999.
- [12] C. Galambos, J. Kittler, J. Matas, „Progressive probabilistic hough transform for line detection”, IEEE, , pp. 1:1554, 1999.
- [13] Robert P., CALCULATING VANISHING POINTS [Online], Disponibil la adresa: http://robert-pfeffer.net/kunst/englisch/nachgeladener_rahmen.html?fluchtpunkte.html, Accesat: 2019.
- [14] B. Caprile, V. Torre, „Using Vanishing Points for Camera Calibration”, International Journal of Computer Vision, 4, pp. 127-139, 1990.
- [15] Robert C. Bolles, Martin A. Fischle, „A RANSAC-Based Approach to Model Fitting and Its Application to Finding Cylinders in Range Dat”, IJCAI'81, -, pp. 637-643, 1981.
- [16] P.H.S. Torr and A. Zisserman, „MLESAC: A new robust estimator with application to estimating image geometry”, Journal of Computer Vision and Image Understanding 78 , , pp. 138-156, 2000.
- [17] O. Chum, J. Matas, „Matching with PROSAC - progressive sample consensus”, IEEE, 1, pp. 220-226, 2005.
- [18] O. Chum, J. Matas, „Randomized RANSAC with Td,d test”, 13th British Machine Vision Conference, , pp. 837-842, 2004.
- [19] Florin Oniga, RANSAC – potrivirea unei linii la un set de puncte [Online], Disponibil la adresa: <http://users.utcluj.ro/~onigaf/files/PRS.html>, Accesat: 2019.
- [20] Steven W. Smith, „The Scientist and Engineer's Guide to Digital Signal Processing”, California Technical Publishing, 1999.
- [21] *, Dreapta în plan [Online], Disponibil la adresa: [https://www.mateonline.net/matricea/80/s/Dreapta_\(R\)n_plan.htm](https://www.mateonline.net/matricea/80/s/Dreapta_(R)n_plan.htm), Accesat: 2019.

- [22] Waleed Abdulla, Instance Segmentation with Mask R-CNN and TensorFlow [Online], Disponibil la adresa: <https://engineering.matterport.com/splash-of-color-instance-segmentation-with-mask-r-cnn-and-tensorflow-7c761e238b46>, Accesat: 2019.
- [23] Kaiming He, Georgia Gkioxari, Piotr Dollár, Ross Girshick, „Mask R-CNN”, Facebook AI Research, , pp. , 2017.
- [24] Ross Girshick, „Fast R-CNN”, IEEE, , pp. 2380-7504, 2015.
- [25] Marco Jacobs, Deep Learning in Five and a Half Minutes [Online], Disponibil la adresa: <https://www.embedded-vision.com/industry-analysis/blog/deep-learning-five-and-half-minutes>, Accesat: 2019.
- [26] Jonathan Hui, Image segmentation with Mask R-CNN [Online], Disponibil la adresa: https://medium.com/@jonathan_hui/image-segmentation-with-mask-r-cnn-ebe6d793272, Accesat: 2019.
- [27] Takaya Saito, Marc Rehmsmeier, Basic evaluation measures from the confusion matrix [Online], Disponibil la adresa: <https://classeval.wordpress.com/introduction/basic-evaluation-measures/>, Accesat: 2019.