UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

Procesarea imaginilor

Detecarea marcajelor de drum

Autori: Balint Catalin si Hruban Andrada
Grupa: 30236
Indrmator: Ana Rednic

Facultatea de Automatica
si Calculatoare

Cuprins

1 Introducere

1.1 Contextul temei

În contextul creșterii interconectării tehnologiilor de asistență a șoferilor și dezvoltării vehiculelor autonome, utilizarea tehnologiilor avansate de procesare a imaginilor pentru detectarea și interpretarea marcajelor rutiere a devenit esențială. Proiectul prezent abordează această necesitate prin implementarea unei soluții bazate pe procesarea imaginilor captate de camerele vehiculare, folosind algoritmi de vizualizare computerizată pentru a analiza și interpreta caracteristicile vizuale ale drumurilor. Soluția utilizează OpenCV, o bibliotecă specializată în procesarea digitală a imaginilor și vizualizarea computerizată, pentru a dezvolta un sistem capabil să asiste șoferii în navigarea pe drumuri și în detectarea marcajelor de circulație.

1.2 Problemele de rezolvat si obiective propuse

Obiectivele proiectului sunt multiple și vizează atât îmbunătățirea siguranței pe drumuri, cât și suportul pentru sistemele de navigare autonome:

Detectarea eficientă a marcajelor rutiere: Implementarea unui sistem capabil să identifice și să clasifice diferite tipuri de marcaje rutiere, de la linii continue și întrerupte, până la săgeți și alte simboluri specifice, utilizând tehnici de detectare a marginilor și transformări spațiale.

Clasificarea orientării și poziției marcajelor: Determinarea orientării și poziționării relaționale a marcajelor rutiere față de vehicul, ceea ce este crucial pentru interpretarea corectă a intențiilor de navigare.

Vizualizare clară și interpretare: Sistemul trebuie să ofere o reprezentare vizuală clară a rezultatelor procesării pentru utilizatori, arătând marcajele detectate direct pe imagini, facilitând astfel interpretarea și luarea deciziilor în timp real.

Adaptabilitate și robustețe: Asigurarea că sistemul funcționează eficient în diverse condiții de iluminare și vreme, precum și în diferite scenarii de trafic.

Interfața utilizator: Dezvoltarea unei interfețe utilizator prietenoase care să permită utilizatorilor să interacționeze ușor cu sistemul, să ajusteze setările și să vizualizeze datele procesate.

Aceste obiective sunt concepute pentru a asigura că sistemul nu doar că îmbunătățește siguranța prin asistență vizuală avansată, dar și că integrează tehnologia în mod eficient în infrastructura existentă și în comportamentul șoferilor, sprijinind tranziția spre o mai mare automatizare în domeniul transporturilor.

2 Fundamentare teoretica

2.1 Idee initiala

Idea initiala a fost sa folosim algoritmul de detectare al conturului.

Algoritmul de urmărire a conturului

Algoritmul de urmărire a conturului este o tehnică utilizată în procesarea imaginilor pentru a extrage conturul obiectelor. Este aplicabil în special pentru imagini binare sau pentru imagini în care obiectele au fost deja etichetate. Acest algoritm identifică și urmărește limita exterioară a unui obiect, permițând extragerea formei și dimensiunilor acestuia.

Pașii algoritmului:

1. **Identificarea pixelului de start:** Se începe scanarea imaginii de la colțul stânga sus până când se găsește un pixel care aparține conturului unui obiect. Acest pixel este considerat punctul de start (P0) al conturului.
2. **Inițializarea direcției:** Se definește o variabilă *dir* care menține direcția mutării anterioare de-a lungul conturului. Inițializarea se face diferit în funcție de tipul de vecinătate utilizat:
   - dir = 0 pentru vecinătate de 4.
   - dir = 7 pentru vecinătate de 8.
3. **Urmărirea conturului:** Se parcurge vecinătatea de 3x3 a pixelului curent în sens invers acelor de ceasornic, începând cu pixelul corespunzător direcției ajustate după formula $(dir + 3)4$ pentru vecinătatea de 4 sau $(dir + 7)8/(\text{dir} + 6)$ mod 8 pentru vecinătatea de 8. Primul pixel găsit care are aceeași valoare ca pixelul curent este considerat următorul element al conturului, iar *dir* se actualizează corespunzător.
4. **Verificarea completării conturului:** Procesul continuă până când elementul curent al conturului (Pn) se întoarce la al doilea element (P1) și elementul anterior (Pn-1) este egal cu primul element (P0), semn că întregul contur a fost urmărit.
5. **Reprezentarea conturului:** Conturul este reprezentat de secvența de pixeli identificați, de la P0 la Pn-2, evidențiind forma obiectului detectat.

Am gandit o strategie pentru implementarea codului. Aceasta se bazeaza pe liniile marcajelor indicatoare. Pentru inceput imaginea trebuie binarizata, pentru a se putea realiza conturul marcajului. Fiecare marcaj are linii in directii diferite, dar intr-o combinatie unica. Pe baza directiei liniilor acestui marcaj, se poate stabili o ordine pentru fiecare semn, ca un fel de cod, dupa care se poate diferentia fiecare marcaj.Pentru a intelege mai bine, voi atasa mai jos secvente din schita facuta pentru marcajele alese de noi.

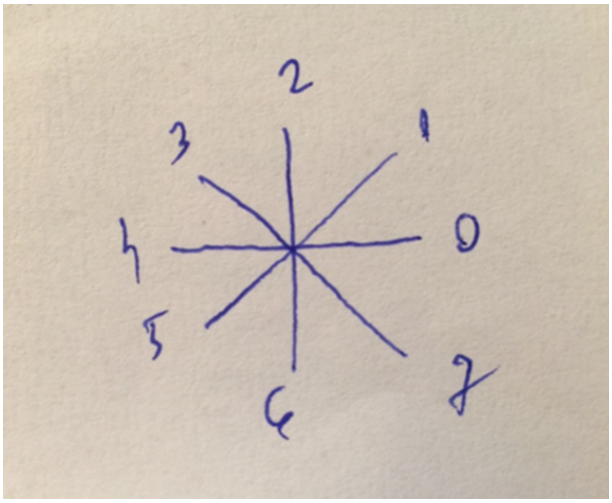


Figura 1: Directiile

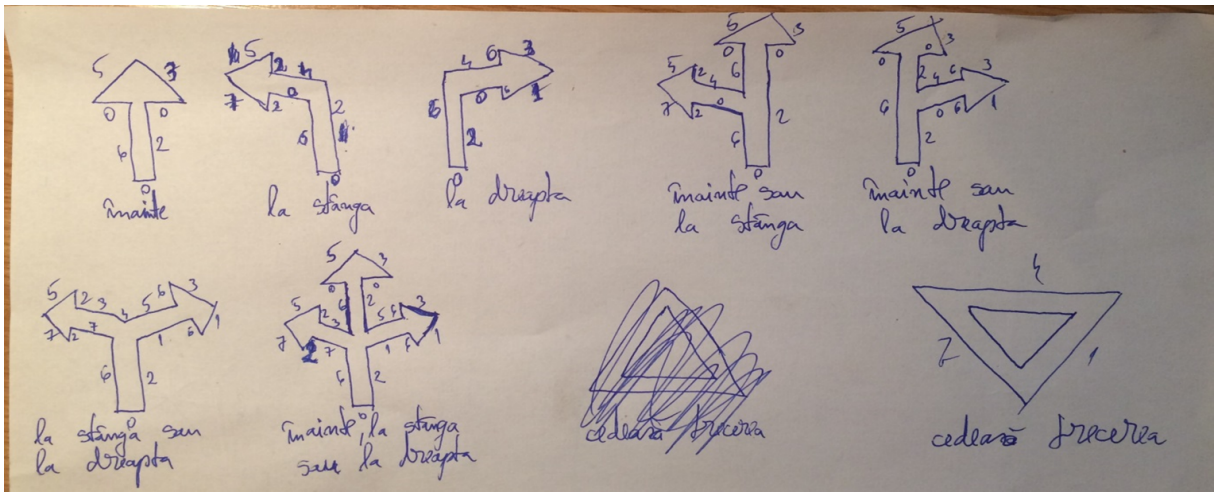


Figura 2: Marcajele cu directiile

Am implementat binarizarea unei imagini, bazandu-ne pe informatiile gasite in laboratoarele de PI. Am implementat si algoritmul pentru calcularea conturului marcajelor si am descoperit prima problema legata de rezultatul dat de directiile date de contur. Mai exact, algoritmul returneaza directia urmatorului pixel din contur, lucru care nu este deloc convenabil, pentru ca in functie de rezolutia imaginii, numarul de pixeli de pe o muchie pot varia. In plus, muchiile nu sunt netede, rezultand fluctuatii ale valorilor directiilor unei muchii. Astfel, am realizat un algoritm pentru a returna o singura valoare a directiei unei muchii, cea mai frecventa.Dupa ce algoritmii sunt pregatiti, am scris codul pentru recunoasterea fiecarui marcaj in functie de directiile muchiilor fiecaruia. Bineinteles, pot exista anomalii, facand totusi un compromis, am ales doar un model pentru fiecare tip de marcaj. Dar, fiindca nu mergea pe poze complexe din viata reala a trebuit sa mergem pe o alta idee.

2.2 Idee finala

Soluția propusă utilizează procesarea imaginii pentru a detecta și urmări marcajele de circulație de pe drum, în scopul asistării șoferilor sau sistemelor autonome. Procesul începe cu captarea unei imagini a drumului, transformarea acesteia în scala de gri pentru a simplifica analiza, reducerea zgomotului prin filtrare Gaussiană, detectarea marginilor cu ajutorul detectorului de margini Canny, și utilizarea Transformatei Hough pentru identificarea liniilor drepte care reprezintă marcajele de circulație.

3 Proiectare și implementare

3.1 Descrierea soluției

Proiectul nostru vizează dezvoltarea unei aplicații de procesare a imaginilor pentru detectarea și urmărirea marcajelor de circulație, ce poate fi utilizată în vehicule pentru a asista șoferii sau în sisteme de conducere autonomă. Soluția integrează mai multe tehnici de vizualizare computerizată pentru a analiza și interpreta caracteristicile vizuale ale drumurilor.

1. **Captarea imaginii:**
   - Gasirea unor imagini cu marcaje de circulatie din viata reala adecvate pentru testarea algoritmilor.
2. **Prelucrarea inițială:**

- **Conversia la scala de gri:** Transformarea imaginilor color în imagini alb-negru pentru a reduce complexitatea datelor și a îmbunătăți eficiența algoritmilor de detectare a marginilor.
- **Reducerea zgomotului:** Aplicarea unui filtru Gaussian pentru a elimina zgomotul de fond și a clarifica caracteristicile marcante, cum ar fi marginile.

3. **Detectarea marginilor:**
   - **Detectorul de margini Canny:** Utilizarea acestui algoritm pentru a identifica zonele din imagine unde există tranziții semnificative de intensitate, care indică prezența marginilor. Detectorul de margini Canny este eficient pentru a distinge marginile relevante de zgomotul de fond.
4. **Identificarea structurilor liniare:**
   - **Transformata Hough pentru linii:** Această tehnică matematică este folosită pentru a detecta linii drepte în imagine, esențiale în identificarea și urmărirea marcajelor de circulație.
5. **Interpretarea și reprezentarea grafică:**
   - **Trasarea liniilor detectate:** După identificarea liniilor prin Transformata Hough, acestea sunt trasate înapoi pe imagine pentru a oferi o vizualizare clară a marcajelor de direcție detectate.
   - **Evaluarea și decizia:** Baza pentru interpretarea poziției și orientării vehiculului relativ la marcajele de circulație, contribuind la deciziile de navigare ale sistemului de conducere sau asistarea șoferului.

3.2 Schema bloc

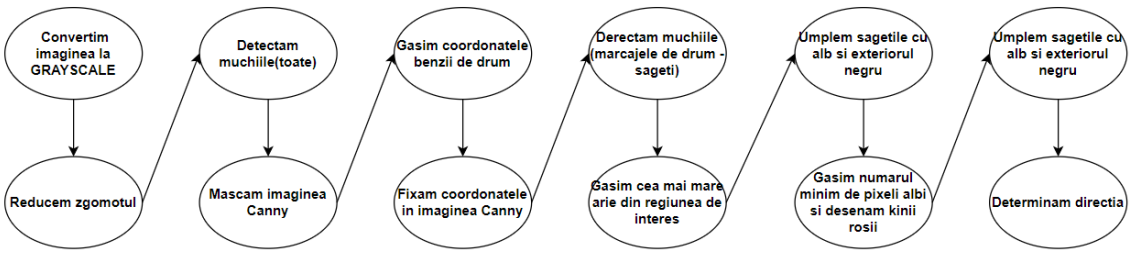


Figura 3: Schema bloc

3.3 Descrierea modului de implementare

Procesul implică următorii pași esențiali:

- **Capturarea imaginii:** Se obține o imagine a drumului și a marcajelor de direcție folosind o cameră sau alt dispozitiv de captare a imaginilor montat pe vehicul.
- **Conversia imaginii la scala de gri:** Imaginea color este transformată într-o imagine alb-negru pentru a facilita detectarea marginilor și analiza ulterioară a marcajelor de direcție.
- **Reducerea zgomotului:** Se elimină zgomotul din imagine pentru a evita detectarea eronată a marginilor marcajelor de direcție, folosindu-se un filtru de blurare, cum ar fi blurarea Gaussiană.

- **Detectorul de margini Canny:** Utilizat pentru a detecta marginile din imagine, identificând zonele în care apare o schimbare bruscă a intensității de lumină.
- **Delimitarea regiunii de interes:** Se selectează doar zona din imagine care conține marcajele de direcție, iar restul imaginii este eliminat pentru a concentra eforturile de detectare.
- **Transformarea Hough pentru linii:** Utilizată pentru a identifica liniile drepte din imagine, care reprezintă marcajele de direcție.
- **Trasarea liniilor pe imagine:** După detectarea liniilor, acestea sunt trasate pe imagine pentru a evidenția poziția și orientarea lor.
- **Identificarea obiectului din zona de interes cu cea mai mare arie:** După delimitarea regiunii de interes, se identifică marcajul de direcție care ocupă cea mai mare parte din această zonă.
- **Umplerea celei mai mari regiuni cu alb:** După identificarea marcajului de direcție cu cea mai mare arie, această regiune este umplută cu culoarea albă pentru a izola clar marcajul în imaginea originală.
- **Trasarea matematică a distanțelor:** Se calculează distanțele matematice între marcajele de direcție identificate și evidențiate în imagine pentru a determina poziția și orientarea vehiculului.

Average_Slope_Intercept: Această funcție este crucială pentru determinarea pantei și interceptării liniilor detectate prin transformata Hough.

Transformata Hough identifică linii drepte în imagine, iar această funcție ajută la clasificarea acestora în funcție de poziția lor relativă față de axa x (stânga sau dreapta).

În contextul detecției marcajelor de circulație pentru direcție, este esențial să distingem între linii care aparțin benzii de circulație din stânga și cele care aparțin benzii de circulație din dreapta, permițând clasificarea liniilor pe baza pantei lor și obținerea unei reprezentări mai precise a direcțiilor indicate de aceste linii.

Pixel_Points: Funcția Pixel_Points primește informații despre panta, interceptare și poziția pe axa y a unei linii și returnează coordonatele pixelilor care definesc linia pe imaginea detectată. Aceasta este crucială în detecția marcajelor de circulație, transformând informațiile despre linii în coordonate pixelilor pe imaginea originală, necesare pentru a desena linii vizibile care să reprezinte în mod precis direcțiile marcajelor de circulație detectate.

Lane_Lines: Funcția Lane_Lines integrează informațiile despre pante și interceptări obținute prin Average_Slope_Intercept cu coordonatele pixelilor calculate în Pixel_Points pentru a obține coordonatele complete ale benzilor de circulație stânga și dreapta.

În contextul detectării marcajelor de circulație, această funcție permite obținerea coordonatelor precise ale benzilor de circulație pe imaginea originală, esențiale pentru evidențierea și vizualizarea corespunzătoare a acestora.

Draw_Lane_Lines: Această funcție utilizează coordonatele obținute în Lane_Lines pentru a desena liniile corespunzătoare benzilor de circulație pe imaginea originală.

În procesul de detecție a marcajelor de circulație, această funcție este ultimul pas în evidențierea și vizualizarea direcțiilor marcajelor de circulație detectate pe imaginea drumului, creând o reprezentare vizuală clară a benzilor de circulație, care poate fi apoi utilizată pentru a ghida vehiculele pe drum.

find_all_min_white_pixels(image, axis): Această funcție primește o imagine și un ax specific (vertical sau orizontal).

Verifică dacă imaginea este color sau alb-negru și o convertește, dacă este necesar, în alb-negru pentru a facilita prelucrarea.

Numără pixelii albi de pe axa specificată și îi stochează într-un vector.

În cazul în care numărul de pixeli albi este mai mic sau egal cu 2, aceștia sunt considerați irelevanți și li se atribuie valoarea infinit. Funcția returnează o listă de indici ai pixelilor albi cu valoare minimă.

find_white_pixel_coordinates(image, column_indices) și find_white_pixel_coordinates_row(imag row_indices): Aceste funcții primesc o imagine și o listă de indici aferenți pixelilor cu valoare minimă pe coloane sau pe rânduri.

Pentru fiecare indice dat, funcțiile găsesc coordonatele pixelilor albi și calculează media aritmetică a acestora pentru axa specificată, asigurând o acuratețe mai ridicată asupra calculului distanței.

Rezultatul este o listă de coordonate grupate pe coloane sau pe rânduri, cu media pentru fiecare grup.

remove_duplicates(coordinates): Această funcție primește o listă de coordonate și folosește o structură de date de tip set pentru a ține evidența coordonatelor unice, eliminând duplicările de coordonate. Funcția returnează o listă de coordonate fără duplicate.

draw_lines(image, indices, color=(0, 0, 255), thickness=1, axis=0): Această funcție primește o imagine, o listă de indici și alte parametri opționali precum culoarea și grosimea liniilor. Pentru fiecare indice dat, funcția desenează o linie verticală sau orizontală, în funcție de axa specificată, creând o evidentiere mai bună a intersecției marginilor de pixeli albi cu paralele la axele Ox și Oy. Funcția returnează imaginea modificată cu liniile desenate.

calculate_distances(image, rows, columns): Această funcție primește o imagine, o listă de indici pentru rânduri și o listă de indici pentru coloane. Pentru fiecare indice de rând și de coloană dat, funcția extrage rândul și coloana corespunzătoare din imagine, identifică pixelii activi (albi) și calculează distanța orizontală de la linia roșie (de la indicele coloanei) până la cel mai apropiat pixel alb în rândul respectiv. Distanța minimă este găsită și afișată pentru fiecare rând și coloană.

find_largest_contour_area(image): Funcția primește o imagine și efectuează următorii pași pentru a găsi și a marca cea mai mare zonă de contur pe imagine.

Imaginea este prelucrată prin aplicarea unui filtru Gaussian pentru a reduce zgomotul și a îmbunătăți detectarea marginilor.

Se detectează marginile folosind detectorul de margini Canny.

Contururile sunt găsite în imaginea cu margini detectate, și se verifică dacă există contururi.

Se creează o mască pentru a desena cel mai mare contur, se identifică cel mai mare contur după suprafață, se desenează cel mai mare contur pe mască și pe imagine, și se aplică masca pe imaginea originală pentru a acoperi totul în afară de cea mai mare zonă.

Imaginea rezultată, care conține numai cea mai mare zonă de contur, este returnată.

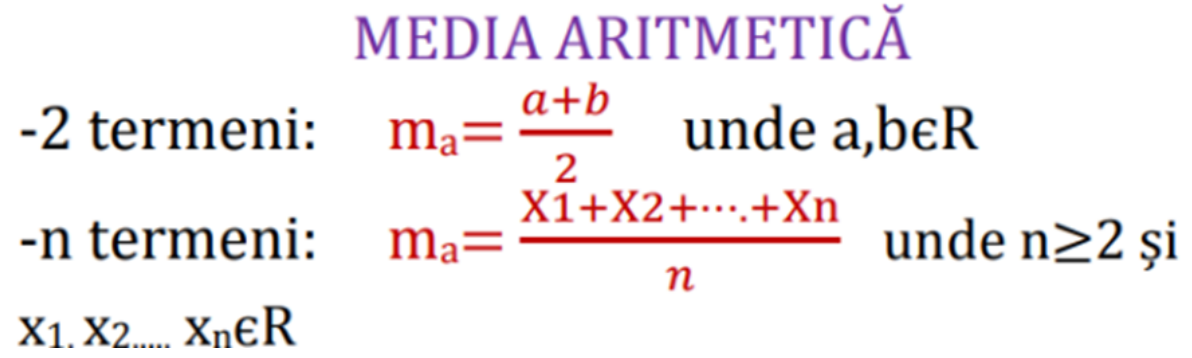


Figura 4: Media aritmetica

Am folosit formula de media aritmetica pentru a identifica o mai buna acuratete a distantei.Deoarece incercam sa ignoram posibilele greseli in reconstruirea pozei cum ar fi detectia unor pixeli albi singuri oarecum pierduti ,cautam doar minimul de pixeli de la 3 pe axa in sus.Din acest motiv media aritmetica este folositoare deoarece nu influenteaza negativ rezultatul ,ci din contra ,asigura calculul distantei din centrul segmentului de pixeli albi ,care reprezinta mai bine arrow shape-ul.

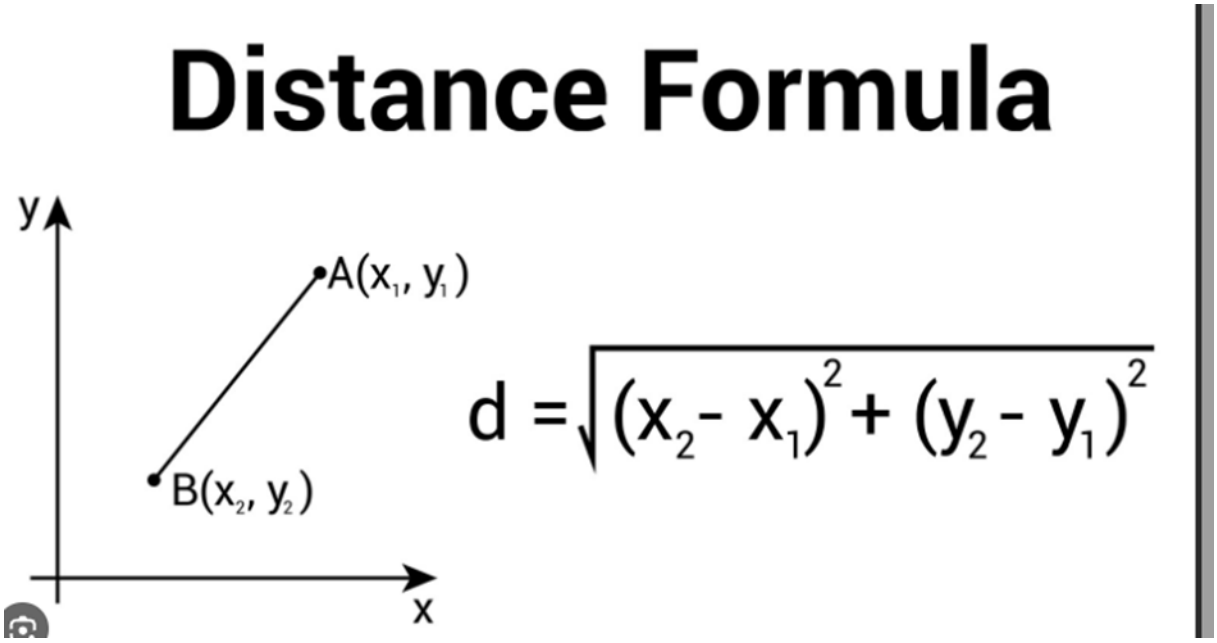


Figura 5: Formula distantei

Formula distantei dintre doua puncta este cea clasica in contextual bidimensional unde in functie de coordonatele celor punctelor , in urma eliminarii duplicatelor si a medii aritmetice ,se afla distanta corecta

3.4 Descrierea algoritmilor implementați

3.4.1 Conversia la Scala de Gri (Grayscale Conversion)

Implementare: Algoritmul de conversie la scala de gri calculează o medie ponderată a canalelor de culoare RGB (Roșu, Verde, Albastru), utilizând formula:

$$0.299 \times R + 0.587 \times G + 0.114 \times B$$

Această formulă ține cont de sensibilitatea ochiului uman la diferite culori, acordând mai multă importanță canalului verde.
Aplicație: În detecția marcajelor de circulație, conversia la scala de gri simplifică detectarea marginilor și a contrastului, facilitând algoritmii care urmează, cum ar fi Canny.

3.4.2 Filtrul Gaussian (Gaussian Blurring)

Implementare: Filtrul Gaussian aplică o matrice Gaussiană peste imagine, unde fiecare punct din output este calculat ca o medie ponderată a punctelor vecine, valorile ponderilor fiind determinate de o distribuție Gaussiană. Acest lucru rezultă într-o imagine estompată, cu reducerea detaliilor fine și a zgomotului.
Aplicație: Este util în prelucrarea preliminară pentru detectarea marginilor, pregătind imaginea pentru o analiză mai clară și reducând posibilitatea detectării falselor margini.

3.4.3 Detectorul de Margini Canny

Implementare: Algoritmul Canny pentru detectarea marginilor este unul dintre cei mai eficienți algoritmi utilizați în procesarea de imagini. Implementarea acestuia se desfășoară în următorii pași:

- **Reductia zgomotului:** Aplicarea unui filtru Gaussian pentru reducerea zgomotului și a detaliilor fine din imagine. Formula pentru filtrul Gaussian este:

  $$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

  unde G este valoarea pixelului filtrat, x și y sunt coordonatele pixelului în imagine, iar σ este deviația standard a distribuției Gaussiane, care controlează cât de mult se estompează imaginea.
- **Calculul gradientului imaginii:** Se folosesc filtre Sobel pentru a obține gradientul imaginii la fiecare punct. Ne salvam in doi vectori valorile din filtrele Sobel pentru X si pentru Y.
  Ne cream doua matrici de aceeasi dimensiune ca imaginea pe care vrem sa o filtram.
  Parcurgem imaginea si aplicam filtrele Sobel pe X si pe Y. Rezultatele le punem in cele doua matrici create anterior.
  Calculam in doua matrici separate modulul si direcția gradientului, aplicând pentru fiecare valoare din matrice,
  Gradientul și direcția sunt calculați ca:

  $$G = \sqrt{I_x^2 + I_y^2}, \quad \theta = \arctan\left(\frac{I_y}{I_x}\right)$$

  unde I_x și I_y sunt rezultatele convoluțiilor cu K_x și K_y, iar G și θ sunt magnitudinea și direcția gradientului.

Trebuie ulterior sa normalizam valorile din matricea in care am stocat modulul gradientului, adicăsa le aducem in intervalul [0-255] împărțindu-le la 4*sqrt(2), daca filtrul aplicat a fost Sobel.

- **Suprimarea non-maximelor:** Se subțiază marginile eliminând pixelii care nu sunt considerați a fi parte a unei margini "adevărate". Acest pas asigură că marginile sunt cât mai subțiri posibil, aplicând o comparație între pixelul curent și vecinii săi în direcția gradientului.
  Trebuie sa cuantificam direcțiile gradientului, împărțindu-le in 4 zone. Deci trebuie sa parcurgem matricea care conține direcțiile gradientului si sa le încadrăm in cele 4 zone, comparându-le cu PI/8.
  Apoi in funcție de zona, trebuie sa comparam daca modulul gradientului de pe poziția pe care ne aflam este mai mare decât valorile vecinilor din zona detectata. De exemplu, daca detectam unghiul ca fiind din zona 3, vom verifica daca modulul gradientului de pe poziția curenta (i, j) este mai mare decât modulul gradientului de pe pozițiile (i-1, j-1) si (i+1, j+1). Daca este mai mare decât ambii vecini, îl păstrăm într-o matrice separata (rezultatul dupăsuprimarea maximelor), iar daca nu, punem 0 in matricea rezultat pe poziția (i, j).
- **Pragul dublu și urmărirea marginilor prin histerezis:** Se folosesc două praguri: unul joasă și unul înalt pentru a identifica pixelii puternici, slabi și irelevanți. Pixelii cu valori deasupra pragului înalt sunt considerați margini puternice, iar cei între pragurile joase și înalte sunt considerați slabi și sunt acceptați doar dacă sunt conectați la margini puternice.
  Vrem sa evidențiem muchiile tari.
  Daca într-o muchie exista si puncte tari si puncte slabe, le vom face pe cele slabe puncte tari, pentru a nu avea intermediari.
  Daca o muchie are doar puncte slabe si nu are nici un punct tare, va fi ștearsă complet.
  Putem sa folosim o coada in care sa stocam punctele tari pe care le găsim.
  Pentru fiecare punct tare căutăm daca exista puncte slabe printre vecinii lui, iar daca exista, le facem tari si le adăugăm in coada.
  Dupăce ne asiguram ca toate punctele tari au fost extinse in muchiile din care fac parte, parcurgem încăo data imaginea si suprimam (înlocuim cu 0) toate punctele slabe , pentru ca ele fac parte din muchii complet slabe si nu mai avem nevoie de ele.

Aplicație: Detectorul de margini Canny este esențial pentru extragerea precisă a contururilor și a marginilor din imagini, facilitând identificarea exactă a marcajelor rutiere. Prin extragerea acestor caracteristici, algoritmi suplimentari pot efectua recunoașterea și clasificarea obiectelor din cadrul scenei rutiere.

3.4.4 Selecția Regiunii de Interes (ROI)

Implementare: Selecția regiunii de interes (ROI) este efectuată pentru a focaliza analiza pe părți specifice ale imaginii, reducând astfel zgomotul de fond și îmbunătățind acuratețea detecțiilor ulterioare. Acest proces implică crearea unei măști binare care acoperă regiunea dorită. Măscarea imaginii se realizează apoi prin multiplicarea punct cu punct între imaginea originală și masca binară. Acest proces asigură că doar pixelii din regiunea de interes sunt considerați în pașii de procesare ulteriori.

Aplicație: Selecția eficientă a ROI-ului este crucială în aplicații precum detectarea și analiza traficului, unde este important să se izoleze vehiculele, marcajele rutiere sau alte elemente relevante dintr-un cadru larg. Acest pas simplifică problemele computaționale și îmbunătățește performanța algoritmilor de procesare vizuală.

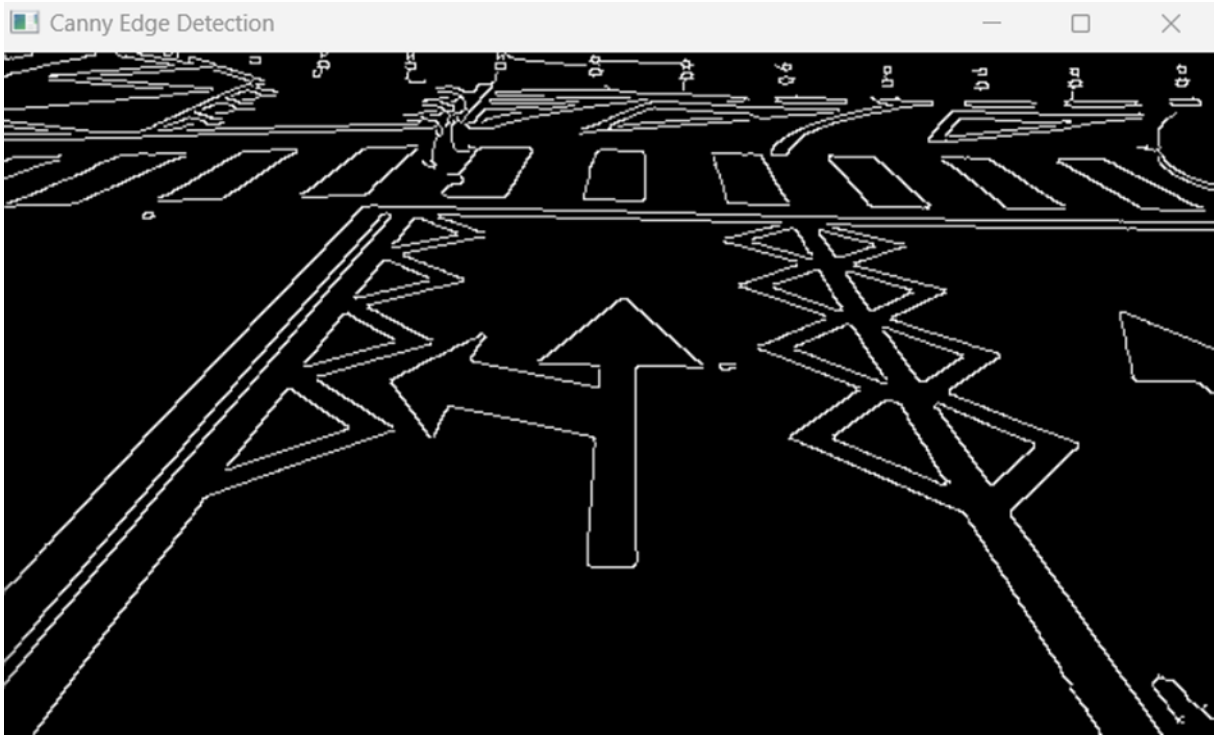
3.5 Modul de utilizare a aplicației implementate

1. **Captura și Pregătirea Imaginii:** Utilizatorul capturează o imagine a drumului care include diverse marcaje rutiere, cum ar fi săgeți și linii de circulație. Imaginea inițială este încărcată în aplicație, unde prima operație este conversia acesteia la scala de gri. Acest pas este esențial pentru simplificarea procesării ulterioare, reducând informația de culoare la intensități de gri.

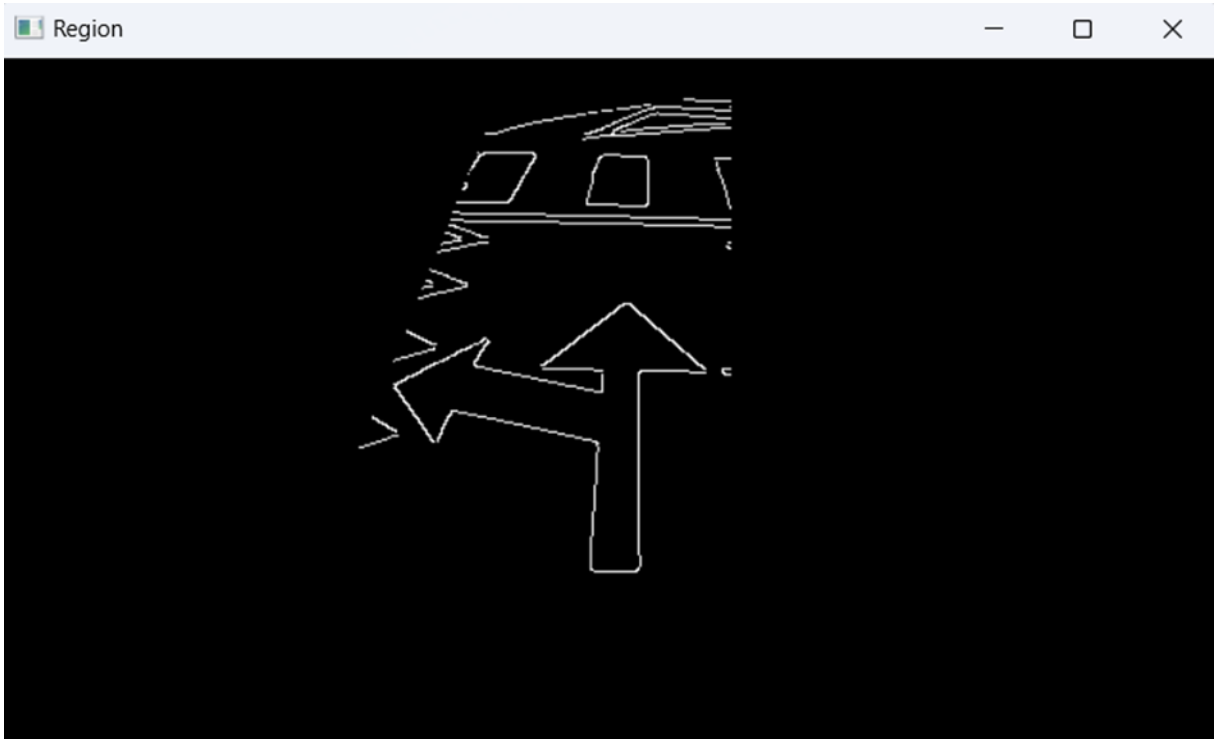
2. **Aplicarea Filtrului Gaussian:** Se aplică un filtru Gaussian pentru a reduce zgomotul din imagine. Aceasta este o pregătire standard pentru detectarea marginilor, deoarece

estomparea contribuie la eliminarea detaliilor fine și a zgomotului care ar putea interfera cu detectarea eficientă a marginilor.

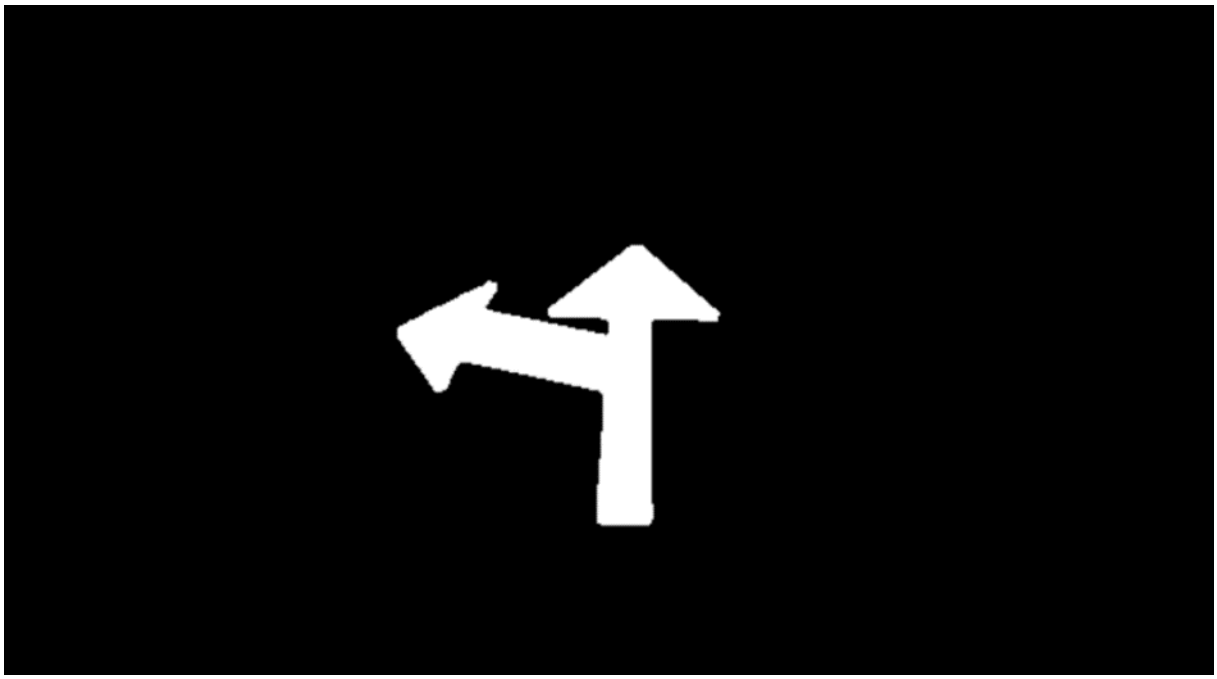
3. **Detectarea Marginilor cu Canny:** Algoritmul Canny este utilizat pentru a detecta marginile în imaginea pregătită. Acesta identifică marginile prin urmărirea schimbărilor intense de intensitate, rezultând într-o imagine binară care afișează contururile clare ale obiectelor și marcajelor din imagine.



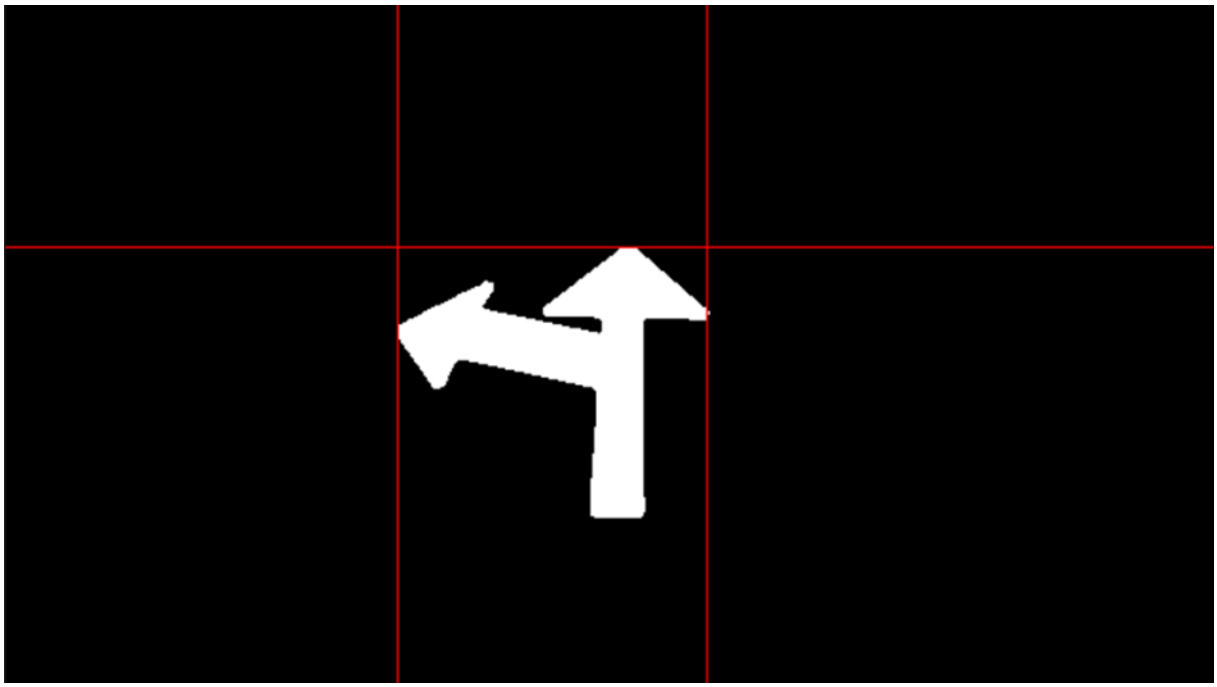
4. **Selectarea Regiunii de Interes (ROI):** După detectarea marginilor, utilizatorul sau algoritmul poate defini o regiune de interes, concentrându-se pe zonele imaginii care conțin marcaje rutiere relevante. Acest lucru se face pentru a limita analiza la părțile esențiale ale imaginii, îmbunătățind astfel viteza și acuratețea procesării.



5. **Identificarea și Analiza Marcajelor Rutiere:** Aplicația analizează contururile detectate pentru a identifica și clasifica diferite tipuri de marcaje rutiere, cum ar fi săgețile. Se utilizează diverse metode de analiză geometrică și de recunoaștere a formelor pentru a interpreta și a distinge între diferite tipuri de marcaje.



6. **Vizualizarea și Interpretația Rezultatelor:** Rezultatele sunt vizualizate pe ecran, unde marcajele detectate sunt afișate, uneori cu adăugarea de linii sau alte marcaje pentru a evidenția direcția indicată de săgeți sau configurarea benzilor de circulație. Utilizatorul poate interacționa cu rezultatele pentru a obține mai multe detalii sau pentru a ajusta parametrii de procesare dacă este necesar.



Exemplu Specific

Bazându-ne pe imaginile trimise, se începe cu o imagine a unei intersecții, după care se aplică secvențial filtrele descrise pentru a extrage informații despre direcțiile indicatoare de săgeți de pe carosabil. Finalul procesului arată cum din imaginea originală se extrag informații relevante pentru navigație sau pentru sisteme autonome de conducere, cu clarificarea marcajelor rutiere.