Detectarea semnelor de circulatie – Turcu Flavius

Rationale and task management:

Pentru acest proiect mi-am propus recunoasterea semnelor de circulatie cu semnificatie importanta pentru soferul auto. Acesta poate fii utilizat in diverse aplicatii, precum, sisteme de asistenta pentru soferi, vehicule autonome sau chiar pentru analiza automata a semnelor de circulatie in orase.

Problema de baza pe care si-o propune acest software sa o resolve este neatentia soferilor la drum lung prin capturarea semnelor de importanta la circularea in siguranta pe drumurile urbane sau nationale astfel solutia noastra ar contribui la reducerea accidentelor si imbunatatirea navigatiei.

Intuitia metodei: din ce intuiesc eu ca ar trebui sa faca acest proiect arf ii sa detecteze formele geometrice specific ale semnelor de circulatie . Acestea mai apoi ar trebui clasificate bazate pe caracteristici vizuale precum SIFT sau ORB. Inainte de toate acestea totusi v-a trebui sa prelucram imaginile in asa fel incat sa crestem luminozitatea cat si sa o uniformizam.

Planul proiectului:

* Ca o prima tinta a proiectului mi-am propus sa preiau imagini multiple cu semnele de circulatie in diferite unghiuri si poziti si sa le manipulez in asa fel incat sa iasa in vedere ceea ce ne intereseaza pe noi
* Ca o a doua tinta pentru acest proiect mi-am propus sa detectez regiunile din imagine care cotin semnele de circulatie. Se va utiliza o metodelor de detecție a formelor pentru a identifica contururi incomplete.
* Ca un al trei-lea pas mi-am propus sa extrag caracteristicile fiecarui semn de circulatie si sa le clasific in functie de semnificatie. Pentru aceasta se vor folosi descriptori vizuali avansati, precum mentionat mai sus SIFT sau ORB.

Implementation:

# Arhitectura proiectului de detecție a semnelor de circulație

Proiectul este structurat într-o arhitectură modulară, cu mai multe componente care lucrează împreună pentru a detecta semnele de circulație din imagini. Voi detalia fiecare modul, funcționalitățile sale și deciziile de design.

## Structura generală a proiectului

Proiectul este organizat în următoarele module principale:

1. **ORB Detector** - implementare proprie a algoritmului ORB pentru detectarea caracteristicilor
2. **Shape Detector** - detector de forme geometrice (cercuri, triunghiuri, dreptunghiuri)
3. **Sign Detector** - modul principal care integrează celelalte componente pentru a detecta semnele

## Detalii despre fiecare modul

### 1. ORB Detector (orb\_detector.h/cpp)

**Funcționalități principale:**

* Detectarea punctelor cheie (keypoints) folosind o implementare proprie a algoritmului FAST
* Calcularea orientării pentru fiecare punct cheie
* Calcularea descriptorilor BRIEF pentru punctele cheie
* Potrivirea (matching) descriptorilor între imagini

**Decizii de design:**

* Am implementat o clasă KeyPoint personalizată pentru a stoca informații despre punctele cheie
* Algoritmul FAST a fost implementat manual pentru a înțelege cum funcționează
* Calcularea orientării folosește momente centrate pentru a determina direcția gradientului
* Descriptorul BRIEF utilizează un model de perechi generate aleator, care sunt apoi rotite conform orientării punctului cheie

### 2. Shape Detector (shape\_detector.h/cpp)

**Funcționalități principale:**

* Detectarea contururilor în imagini folosind detecția de muchii
* Clasificarea contururilor în forme geometrice de bază (cerc, triunghi, dreptunghi)
* Calcularea unor metrici de similaritate pentru fiecare tip de formă

**Decizii de design:**

* Am implementat propria funcție de detecție a muchiilor folosind operatorul Sobel
* Am dezvoltat un algoritm personalizat de urmărire a contururilor
* Pentru clasificarea formelor am folosit metrici precum:
  + Circularitate pentru cercuri
  + Triangularitate pentru triunghiuri
  + Rectangularitate pentru dreptunghiuri
* Am implementat algoritmul Douglas-Peucker pentru aproximarea poligonală a contururilor

### 3. Sign Detector (sign\_detector.h/cpp)

**Funcționalități principale:**

* Integrarea celor două metode de detecție (bazată pe forme și bazată pe ORB)
* Încărcarea de template-uri pentru potrivirea bazată pe caracteristici
* Filtrarea detecțiilor suprapuse

**Decizii de design:**

* Permite două abordări complementare de detecție:
  + Detectarea bazată pe forme geometrice
  + Detectarea bazată pe caracteristici ORB (deși aceasta nu funcționează optim)
* Implementarea metodei de filtrare a detecțiilor suprapuse folosind IoU (Intersection over Union)
* Prelucrarea imaginilor pentru îmbunătățirea detecției (egalizarea histogramei, blur gaussian)

## Fluxul de lucru al aplicației

1. Încărcarea imaginii de intrare
2. Încărcarea template-urilor din directorul specificat (dacă există)
3. Detectarea semnelor de circulație folosind metoda bazată pe forme
4. ~~Detectarea semnelor folosind metoda bazată pe ORB~~ (dezactivată în versiunea curentă)
5. Filtrarea detecțiilor suprapuse
6. Afișarea rezultatelor

## Încercări anterioare și probleme întâmpinate

Inițial, am încercat să folosesc metoda SIFT pentru detectarea semnelor de circulație, dar am întâmpinat dificultăți semnificative în implementare. Principala problemă a fost că pe o imagine, algoritmul detecta un singur semn de circulație ca fiind 4 semne diferite, iar în alte cazuri nu detecta nimic. Acest comportament inconsistent m-a determinat să caut abordări alternative.

După mai multe încercări, am decis să mă orientez spre o metodă bazată pe detectarea formelor geometrice, deoarece semnele de circulație au forme distinctive (cercuri, triunghiuri, dreptunghiuri). Această abordare a dat rezultate mai bune și mai consistente.

În paralel, am încercat să implementez și algoritmul ORB (Oriented FAST and Rotated BRIEF) pentru detecția bazată pe caracteristici, dar și această metodă a avut un succes limitat. Problemele principale au fost:

* Dificultatea de a găsi potriviri stabile între template-uri și imaginea de intrare
* Sensibilitatea mare la schimbările de iluminare și perspectivă
* Rata mare de false pozitive

În versiunea finală a codului, am păstrat implementarea ORB, dar am comentat apelul la funcția detectSignsByFeatures din metoda principală detectSigns, concentrându-mă mai mult pe detecția bazată pe forme geometrice, care s-a dovedit mai fiabilă pentru acest caz specific.

Testing:

**Metrici de evaluare utilizate**

1. **IoU (Intersection over Union)**  
   Reprezintă cât de mult se suprapune zona detectată față de zona reală. Se calculează ca raportul dintre aria intersecției și aria uniunii dintre două dreptunghiuri (bounding boxes).  
   Formula: IoU = aria\_intersectie / aria\_uniune  
   Dacă IoU este mai mare decât un prag (ex: 0.5), considerăm că semnul a fost detectat corect.
2. **Precision (Precizie)**  
   Măsoară proporția detecțiilor corecte (true positives) față de totalul detecțiilor realizate (true positives + false positives).  
   Formula: Precision = TP / (TP + FP)  
   Afișează cât de precise sunt detecțiile – adică câte dintre ele sunt cu adevărat corecte.
3. **Recall (Rata de detectare)**  
   Măsoară proporția semnelor corect detectate față de totalul semnelor reale din imagine (true positives + false negatives).  
   Formula: Recall = TP / (TP + FN)  
   Arată cât de multe dintre semnele reale au fost găsite.
4. **F1-Score**  
   Este media armonică dintre precision și recall. Este o metrică echilibrată care ia în calcul atât precizia, cât și capacitatea de a detecta toate semnele.  
   Formula: F1 = 2 \* (Precision \* Recall) / (Precision + Recall)
5. **mean IoU (IoU mediu)**  
   Este media valorilor IoU pentru toate detecțiile corecte.  
   Arată cât de bine se aliniază în medie dreptunghiurile detectate cu cele reale.
6. **AP (Average Precision)**  
   Reprezintă scorul mediu de precizie obținut la un anumit prag de IoU (ex: 0.5 sau 0.75).  
   Dacă toate detecțiile peste acel prag sunt corecte, AP va fi 1.0 (valoare perfectă).
7. **Miss Rate (Rata de semne ratate)**  
   Este proporția semnelor reale care nu au fost detectate.  
   Formula: Miss Rate = FN / (TP + FN)  
   Valoarea ideală este cât mai aproape de 0. Dacă este 1.0, înseamnă că nu s-a detectat nimic.
8. **Area Difference (Diferență de arie)**  
   Măsoară cât de diferită este dimensiunea zonei detectate față de zona reală.  
   Formula: Area Difference = |aria\_custom - aria\_opencv| / aria\_opencv  
   Indică dacă detecția este prea mică sau prea mare comparativ cu zona reală.

### Test 1 – Semnul STOP

Imagine (Custom vs OpenCV):

A screenshot of a computer screen

AI-generated content may be incorrect.

**Rezultate evaluare:**

| **Metrică** | **IoU Threshold = 0.5** | **IoU Threshold = 0.75** |
| --- | --- | --- |
| Precision | 1.0000 | 0.0000 |
| Recall | 1.0000 | 0.0000 |
| F1-Score | 1.0000 | 0.0000 |
| mean IoU | 0.5168 | 0.0000 |
| AP | 1.0000 | 0.0000 |
| Miss Rate | 0.0000 | 1.0000 |
| Area difference | 0.2391 | 0.2391 |

**Observație:**  
La pragul standard de 0.5, sistemul personalizat detectează semnul STOP corect, însă bounding box-ul este mai mic și poziționat slab. La un prag mai strict (0.75), diferențele de aliniere devin vizibile, iar detecția este considerată incorectă.

### Test 2 – Semn Avertizare (!)

Imagine (Custom vs OpenCV):  
A screenshot of a computer screen

AI-generated content may be incorrect.

**Rezultate evaluare:**

| **Metrică** | **IoU Threshold = 0.5** | **IoU Threshold = 0.75** |
| --- | --- | --- |
| Precision | 1.0000 | 1.0000 |
| Recall | 1.0000 | 1.0000 |
| F1-Score | 1.0000 | 1.0000 |
| mean IoU | 0.7910 | 0.7910 |
| AP | 1.0000 | 1.0000 |
| Miss Rate | 0.0000 | 0.0000 |
| Area difference | 0.2090 | 0.2090 |

**Observație:**  
Aceasta este o detecție ideală. Semnul este localizat corect atât ca poziție, cât și ca dimensiune. Sistemul funcționează excelent chiar și la praguri stricte.

### Test 3 – Semn Cedează Trecerea

Imagine (Custom vs OpenCV):  
A screenshot of a phone

AI-generated content may be incorrect.

**Rezultate evaluare:**

| **Metrică** | **IoU Threshold = 0.5** | **IoU Threshold = 0.75** |
| --- | --- | --- |
| Precision | 0.5000 | 0.0000 |
| Recall | 1.0000 | 0.0000 |
| F1-Score | 0.6667 | 0.0000 |
| mean IoU | 0.7423 | 0.0000 |
| AP | 0.5000 | 0.0000 |
| Miss Rate | 0.0000 | 1.0000 |
| Area difference | 0.1782 | 0.1782 |

**Observație:**  
Sistemul detectează semnul, însă poziționarea bounding box-ului nu este suficient de precisă pentru a trece un prag IoU de 0.75. La un prag moderat de 0.5, performanța este acceptabilă, dar trebuie îmbunătățită localizarea conturului.

Improvements:

Select two test cases that initially failed and explain how you resolved the issues. Describe the challenging process and any modifications or additions you made to the previous solution. Clearly justify your design decisions throughout this process.