



**UNIVERSITATEA TEHNICĂ**  
DIN CLUJ-NAPOCA

**FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE**  
**DEPARTAMENTUL CALCULATOARE**

**DETECȚIA OBSTACOLELOR PRIN VIZIUNE MONOCULARĂ  
FOLOSIND UN TELEFON MOBIL INTELIGENT CU SISTEM  
DE OPERARE ANDROID**

LUCRARE DE LICENȚĂ

Absolvent: **Răzvan ITU**

Coordonator științific: **S. I. dr. ing. Radu Gabriel  
DĂNESCU**

**2013**



**UNIVERSITATEA TEHNICĂ**  
DIN CLUJ-NAPOCA

**FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE**  
**DEPARTAMENTUL CALCULATOARE**

DECAN,  
**Prof. dr. ing. Liviu MICLEA**

VIZAT,  
DIRECTOR DEPARTAMENT,  
**Prof. dr. ing. Rodica POTOLEA**

Absolvent: **Răzvan ITU**

**DETECȚIA OBSTACOLELOR PRIN VIZIUNE MONOCULARĂ FOLOSIND UN  
TELEFON MOBIL INTELIGENT CU SISTEM DE OPERARE ANDROID**

1. **Enunțul temei:** *Această lucrare de licență prezintă o soluție eficientă pentru problema detecției de obstacole din traficul rutier, bazată pe histogramme polare și radiale. Un algoritm de segmentare și eliminare a efectului de perspectivă a drumului este implementat. Detecția obstacolelor se bazează pe calcularea histogrammei polare, iar histogrammele radiale sunt folosite pentru determinarea distanței spre obstacol.*
2. **Conținutul lucrării:** *Pagina de prezentare, Aprecierile coordonatorului de lucrare, Cuprins, Introducere, Obiectivele proiectului și specificații, Studiu bibliografic, Analiză și fundamentare teoretică, Proiectare de detaliu și implementare, Testare și validare, Manual de instalare și de utilizare, Bibliografie.*
3. **Locul documentării:** Universitatea Tehnică din Cluj-Napoca, Catedra Calculatoare
4. **Consultanți:** S. I. dr. ing. Radu Gabriel Dănescu
5. **Data emiterii temei:** 5 noiembrie 2013
6. **Data predării:** 04 iulie 2013

Absolvent: \_\_\_\_\_

Coordonator științific: \_\_\_\_\_

**Declarație pe proprie răspundere privind  
autenticitatea lucrării de licență/diplomă/disertație**

Subsemnatul Itu Razvan

\_\_\_\_\_,  
legitimat cu C.I. \_\_\_\_\_ seria KX nr. 730888,

CNP 1900323125809 autorul lucrării

Detecția obstacolelor prin viziune monoculară folosind un telefon mobil  
inteligent cu sistem de operare Android

\_\_\_\_\_, elaborată în  
vederea susținerii examenului de finalizare a studiilor de licență

la Facultatea Automatică și Calculatoare,

Specializarea Calculatoare

din cadrul Universității Tehnice din Cluj-Napoca, sesiunea iulie a anului  
universitar 2013, declar pe proprie răspundere, că această lucrare este rezultatul  
propriei activități intelectuale, pe baza cercetărilor mele și pe baza informațiilor obținute din surse  
care au fost citate, în textul lucrării, și în bibliografie.

Declar, că această lucrare nu conține porțiuni plagiate, iar sursele bibliografice au fost folosite cu  
respectarea legislației române și a convențiilor internaționale privind drepturile de autor.

Declar, de asemenea, că aceasta lucrare nu a mai fost prezentată în fața unei alte comisii de examen  
de licență/diplomă/disertație.

În cazul constatării ulterioare a unor declarații false, voi suporta sancțiunile administrative,  
respectiv, *anularea examenului de licență/diplomă/disertație*.

Nume, prenume

Data

Semnătura

## Cuprins

1. Introducere .....	4
1.1. Context .....	4
1.2. Motivare .....	5
2. Obiectivele proiectului și specificații .....	7
2.1. Obiective .....	7
2.2. Componente .....	8
2.2.1. Componenta de detecție a obstacolelor .....	8
2.2.2. Componenta GPS .....	8
2.2.3. Componenta Bluetooth .....	9
2.3. Integrarea componentelor .....	9
3. Studiu bibliografic .....	10
3.1. Introducere .....	10
3.2. Detecția obstacolelor .....	10
3.3. Eliminarea efectului de perspectivă .....	10
3.4. Achiziția datelor .....	11
3.5. Eliminarea drumului din imagini .....	12
3.5.1. RGB .....	12
3.5.2. HSV, HSI, HSL .....	13
3.5.3. Segmentarea imaginilor .....	13
3.6. Dispozitive mobile .....	14
3.7. Aplicații existente pentru dispozitive mobile .....	14
3.7.1. Aplicația iOnRoad .....	14
3.7.2. Mobileye .....	15
3.7.3. Aplicația Augmented Driving .....	16
4. Analiză și fundamentare teoretică .....	17
4.1. Cerințe funcționale și non-funcționale .....	17
4.1.1. Cerințe funcționale .....	17
4.1.2. Cerințe non-funcționale .....	17
4.2. Cazuri de utilizare .....	18
4.3. Prezentare de ansamblu a sistemului .....	19

4.4.	Calibrarea camerei video.....	19
4.5.	Eliminarea efectului de perspectivă .....	20
4.6.	Segmentarea drumului.....	24
4.7.	Componenta Bluetooth.....	24
4.8.	Componenta GPS .....	25
4.9.	Sistemul de operare Android.....	25
4.9.1.	Nivelul kernel (Kernel Layer).....	26
4.9.2.	Nivelul de Librării Native (Native Libraries Layer).....	26
4.9.3.	Nivelul Framework-ului de Aplicații (Application Framework Layer) .....	26
4.9.4.	Nivelul de Aplicații (Application Layer).....	27
4.10.	Aplicații Android.....	27
5.	Proiectare de detaliu și implementare.....	29
5.1.	Arhitectura generală a sistemului.....	29
5.2.	Unelte de programare și dezvoltare.....	29
5.2.1.	Java .....	29
5.2.2.	C++ .....	30
5.2.3.	Eclipse IDE .....	30
5.2.4.	Android plugin pentru Eclipse .....	30
5.3.	Librăria OpenCV .....	30
5.4.	Prezentare de ansamblu a algoritmului .....	30
5.5.	Inițializare tabele .....	33
5.6.	Segmentare imagine și eliminarea efectului de perspectivă.....	34
5.7.	Calculare histogramă polară.....	35
5.8.	Calculare maxim și normalizarea histogramei polare .....	37
5.9.	Calcularea maximelor locale.....	38
5.10.	Calculare histogramă radială .....	38
5.11.	Comunicarea cu alte dispozitive.....	40
5.12.	Auto corecție imagini folosind accelerometrul .....	41
5.13.	Utilizarea modulului GPS.....	44
6.	Testare și validare.....	45
6.1.	Introducere .....	45
6.2.	Situații și scenarii pentru testare.....	45

6.3.	Performanța sistemului.....	48
6.4.	Comparație cu produse similare.....	50
7.	Manual de instalare și de utilizare .....	54
7.1.	Configurare și instalare mediu de dezvoltare.....	54
7.2.	Utilizarea aplicației – manual de utilizator .....	55
8.	Concluzii și dezvoltări ulterioare.....	59
9.	Bibliografie.....	61

# 1. Introducere

## 1.1. Context

Dezvoltarea vehiculelor inteligente reprezintă una din cele mai noi tendințe tehnologice din ultimii ani. Inovațiile și dezvoltarea accelerată a platformelor hardware și software au dus la un progres vizibil în acest domeniu. În ultimii ani s-au realizat inclusiv vehicule autonome care nu necesită conducător auto la volan. Cele mai multe sisteme propun vehicule semi-autonome, în care conducătorul auto și vehiculul colaborează împreună, fiecare având anumite roluri și responsabilități.

În ultimii ani, sisteme precum: asistarea la schimbarea benzii de circulație, asistent pentru semnele din trafic, tempomat adaptiv (cruise control) au fost introduse pe mașini de serie pentru a facilita șoferii în traficul rutier.

Una dintre tehnologiile care îmbunătățește siguranța autovehiculelor este detecția obstacolelor. Detecția obstacolelor se referă la analiza video a imaginilor preluate din trafic pentru identificarea tuturor obstacolelor care pot apărea în fața vehiculului: vehicule care circulă, vehicule staționate sau oprite, pietoni care trec strada, animale sau alte obstacole (de exemplu copaci căzuți în drum în urma unei furtuni). Pe lângă identificare, detecția obstacolelor presupune și urmărirea lor cadru cu cadru folosind camere video. Un conducător auto ar putea neglija unele obstacole în fața drumului datorită unei stări de oboseală sau din cauza neatenției. În aceste situații, un sistem de detecție a obstacolelor este util pentru a avertiza la timp șoferul de posibile obstacole și pentru a preveni situații neplăcute.

Cea mai simplă metodă de detecție este definită prin identificarea traiectoriei obiectului în planul imaginii. Acest lucru înseamnă ca în fiecare cadru, obiectul este identificat și marcat corespunzător. Unele sisteme de detecție mai avansate oferă informații suplimentare despre obiectul detectat, cum ar fi: viteza de deplasare a obiectului, distanța pana la obiect, orientarea, dimensiunea, forma etc.

Principalele dificultăți care pot apărea când se urmărește detecția de obstacole sunt următoarele:

- Zgomotul din imagini
- Iluminarea redusă
- Schimbări bruște ale luminozității
- Pierderea informației din scena 3D când se proiectează în imagini cu 2 dimensiuni
- Obstrucția obiectelor
- Complexitatea traiectoriei obiectului
- Cerințele hardware pentru procesarea în timp real

În literatura de specialitate există o mare diversitate de algoritmi care diferă prin abordarea unor anumite probleme. Majoritatea implementărilor existente folosesc sisteme de stereo-viziune,

bazate pe capturarea imaginilor folosind două camere video și folosirea celor două imagini corespondente pentru a detecta obstacole și pentru a calcula distanța până la ele. Sistemele bazate pe mono-viziune folosesc o singură cameră video pentru a înregistra informații din trafic. În general sistemele monoculare au o marjă de eroare mai ridicată, mai ales în cazul calcului distanței față de obiecte, deoarece depind foarte mult de calibrarea corectă a camerei.

## **1.2. Motivare**

Principalul motiv pentru dezvoltarea unor sisteme de asistare în trafic este siguranța. Numărul autovehiculelor aflate în trafic este în creștere de la an la an, dar numărul de accidente fatale are o tendință de scădere, datorită îmbunătățirii sistemelor de siguranță ale autovehiculelor. Conform unui studiu al “National Highway Traffic Safety Administration” din cadrul Departamentului de Transport al SUA, numărul de accidente fatale în Statele Unite în anul 2011 a fost 32.367 în scădere față de 2010 când s-au înregistrat 32.999 sau anul 2008 cu 37.423 de accidente [1]. O situație asemănătoare este și în România. Conform unui raport la Poliției Române, în anul 2012 s-au înregistrat 778 de accidente fatale, o ușoară scădere față de 2011 când au fost 846 [2]. Majoritatea accidentelor au fost provocate de eroarea umană și neatenția la volan, de exemplu: viteza prea mare, conducerea imprudentă a vehiculelor sau nerespectarea distanței în trafic. Aceste accidente rutiere pot fi reduse și prevenite prin folosirea unor sisteme de asistență la volan. Aceste sisteme au scopul de a preveni producerea accidentelor, spre deosebire de alte sisteme de siguranță care acționează după producerea accidentului.

Sistemele de asistență inteligente analizează mediul din fața mașinii și identifica marcajele de drum, benzile de circulație, vehicule. Pentru îmbunătățirea siguranței vehiculele asistate de camere video trebuie să reacționeze rapid în cazul unei situații periculoase. Acest lucru presupune o procesare rapidă și robustă a sistemului de viziune folosit.

Detecția și urmărirea obstacolelor reprezintă o funcție esențială în implementarea unui sistem de prevenire a coliziunilor. În traficul rutier urban, în majoritatea situațiilor, pe drum sunt prezente multe obstacole. Astfel, problema de rezolvat este reprezentată de identificarea obstacolelor multiple dar și determinarea distanței până la acestea.

Un algoritm robust și eficient este prezentat în [3]. Algoritmul prezentat folosește un sistem de stereo viziune și calcule bazate pe histograme polare și radiale ale imaginilor pentru determinarea distanței și a unghiului obiectului detectat. O altă soluție pentru identificarea obstacolelor și a distanței până la ele este prezentată în [4]. Această variantă folosește o singură cameră și implementează un algoritm simplu de segmentare a drumului și apoi de identificare a obstacolelor.

Lucrarea de față își propune implementarea unui sistem de detecție a obstacolelor prin viziune monoculară folosind un telefon mobil inteligent cu sistem de operare Android. Această lucrare de licență prezintă o soluție modificată pentru problema detecției de obstacole, bazată pe [3]. Un algoritm de segmentare a drumului și de calculare a histogramei polare și radiale este implementat. Pe parcursul acestei lucrări se vor prezenta concepte necesare detecției de obstacole,



dar și detalii legate de procesul de calibrare al camerei video, precum și informații despre programarea telefoanelor mobile.

În continuare, se vor detalia principalele obiective ale proiectului, algoritmi și tehnologiile folosite pentru implementarea acestui sistem de asistare în trafic.

## 2. Obiectivele proiectului și specificații

### 2.1. Obiective

Proiectul implementat în această lucrare are ca scop dezvoltarea unui sistem de detecție a obstacolelor în traficul rutier, prin viziune monoculară și folosind un telefon mobil inteligent cu sistem de operare Android. Detectarea obstacolelor din trafic se referă la detectarea mașinilor care circula, dar și a pietonilor care pot trece strada. Acest tip de proiect a mai fost implementat pe sisteme embedded sau laptop-uri folosind camere web sau camere externe, dar și pe telefoane mobile Android sau iOS.

Aplicația se adresează conducătorilor de autoturisme pentru a le facilita șofatul într-un mediu urban și nu numai. Utilizatorii aplicației vor fi șoferi care au nevoie de un mobil Android și un suport de telefon pentru parbrizul mașinii. Pentru a folosi aplicația, telefonul trebuie să stea într-o poziție fixă pe parbrizul autoturismului pentru a înregistra imaginile din trafic.

Siguranța în trafic reprezintă cea mai mare preocupare și este motivația din spatele acestui proiect. Din punctual de vedere al conducătorilor auto, se pot identifica mai multe nevoi legate de informare, asistare:

- Monitorizarea stării vehiculului: starea vehiculului
- Detecție timpurie: detecție din timp a problemelor legate de drum
- Detecția obstacolelor (pomi căzuți, stâlpi), a vehiculelor participante în trafic și a pietonilor care traversează strada
- Estimarea vitezei și a duratei: estimarea vitezei și distanței dintre vehiculul propriu și vehiculul din față.

Din punct de vedere al acestui proiect, aplicația dezvoltată va încerca să răspundă principalelor nevoi care ajută la creșterea siguranței:

1. Detecția și urmărirea obstacolelor: vehicule, pietoni, animale
2. Avertizarea la coliziuni: atenționarea timpurie a șoferului pentru evitarea unor situații neplăcute, accidente

Pentru a asista conducătorii auto, acest sistem trebuie să ia în considerare anumiți factori interni și externi. Factorii externi analizați sunt reprezentați de obstacolele de pe drum. Factorii interni sunt viteza de deplasare a autovehiculului. Dacă viteza se reduce, sistemul va putea stabili dacă șoferul a reacționat corespunzător în cazul unui pericol.

Utilitatea acestui sistem poate fi descrisă folosind următoarele exemple:

1. Componenta de detecție a obstacolelor detectează un obstacol la o distanță apropiată.
  - Modulul GPS al telefonului mobil oferă viteza actuală de deplasare.

- Dacă distanța este redusă sub un anumit prag, iar viteza rămâne constantă sau este în creștere, sistemul va avertiza conducătorul auto.
2. Componenta de detecție a obstacolelor detectează un pieton deplasându-se în fața vehiculului, iar vehiculul este oprit la semafor.
- În acest caz, viteza de deplasare citită din senzorul GPS este 0 km/h.
  - Conducătorul este avertizat în momentul în care viteza de deplasare este mărită și pietonul se află încă în fața vehiculului.
3. Componenta de detecție a obstacolelor detectează o zonă de culoare diferită pe suprafața drumului. În acest sistem, o zonă de culoare diferită poate reprezenta un alt tip de obstacol, de exemplu o groapă sau un obiect situat pe drum.
- Conducătorul auto este avertizat vizual de prezența acestui obstacol.
  - În situația în care distanța până la obstacol devine tot mai mică, iar obstacolul se află pe direcția de circulație a vehiculului, conducătorul auto va fi avertizat atât vizual, cât și sonor.

## 2.2. Componente

### 2.2.1. Componenta de detecție a obstacolelor

Această componentă trebuie să fie capabilă de a detecta și clasifica obstacolele care apar în imaginile (video) de intrare.

Camera video trebuie să fie într-o poziție fixă în autovehicul. Realizarea unei detecții cât mai robustă și mai corectă este foarte dificil de realizat, în special în cazul în care este vorba de un sistem monocular (cu o singură camera video). Unele situații și factori externi pot influența robustețea acestui sistem, cum ar fi: vremea nefavorabilă, luminozitatea redusă pe timp de seară, drumuri cu denivelări care afectează stabilitatea mașinii și implicit a camerei video.

### 2.2.2. Componenta GPS

A doua componentă a acestui sistem este modulul GPS. Acest modul este integrat în majoritatea telefoanelor mobile și este folosit pentru a determina viteza curentă de deplasare a vehiculului.

### 2.2.3. Componenta Bluetooth

O altă componentă a acestui sistem este reprezentată de modulul Bluetooth, integrat în telefoanele mobile moderne. Componenta de Bluetooth este utilizată pentru a trimite informații relevante din trafic unui sistem extern. Acest sistem extern poate fi conectat la magistrala de date a vehiculului și ar putea extinde acest sistem, prin oferirea unor informații esențiale computerului mașinii astfel încât vehiculul să reacționeze automat în cazul unui pericol iminent. De exemplu, implementarea unui sistem extern conectat la magistrala CAN a vehiculului, ar putea acționa sistemul de frânare automat, în cazul în care este detectat un obstacol în fața mașinii, iar distanța față de acesta devine din ce în ce mai mică, iar viteza de deplasare a vehiculului rămâne constantă, sau nu scade deloc.

## 2.3. Integrarea componentelor

Sistemul descris mai sus trebuie să fie precis, corect, robust și ușor de folosit de către orice utilizator. Pe lângă acestea, sistemul trebuie să fie accesibil și din punct de vedere financiar. Pentru a realiza acestea, am ales implementarea folosind telefoane mobile bazate pe sistemul de operare open source Android.

- S-a optat pentru telefoane mobile, deoarece folosirea unui calculator, computer în mașină pentru efectuarea de procesare nu favorizează mobilitatea și uneori un computer nu poate fi alimentat corespunzător
- Telefoanele mobile de ultima generație oferă suficientă putere de procesare și oferă camere video performante pentru a înregistra informațiile din traficul rutier în timp real
- Puterea de procesare a mobilelor este în creștere de la an la an

### 3. Studiu bibliografic

#### 3.1. Introducere

Deteția obstacolelor este bazată pe diverși algoritmi din domeniul procesării imaginilor. Pentru a identifica obstacole sunt necesari anumiți pași și au fost implementate diferite tehnici, care vor fi descrise în acest capitol.

#### 3.2. Deteția obstacolelor

Deteția de obstacole folosind procesarea imaginilor s-a dezvoltat folosind două direcții: deteție bazată pe o singură cameră (sistem monocular) sau deteție bazată pe două camere (sistem de stereo viziune). În general algoritmi de deteție bazați pe viziune monoculară au la bază următoarele tehnici: segmentarea pe culori [5], deteție unor caracteristici specifice (texturi) [6] sau simetrie [7], [8]. În sistemele bazate pe o singură cameră, măsurarea caracteristicilor 3D ale obiectelor este implementată după deteție și este facilitată de cunoașterea unor informații despre obiecte și parametrii camerei obținuți prin calibrare. Dacă prin obstacol ne referim la mașini, atunci partea de deteție este bazată pe căutarea unor tipare, modele specifice în imagini, de exemplu forma sau simetria. În aceste cazuri procesarea se poate baza doar pe analiza unei singure imagini.

În cazul în care prin obstacol se definește ca fiind orice obiect care poate obstrucționa, bloca vehiculul în calea sa, deteția obstacolelor este redusă la deteția spațiului liber (en. free space) și nu se referă la căutarea de modele predefinite. În acest caz se folosesc diferite tehnici, cele mai uzuale fiind bazate pe procesarea perechilor de imagini stereo. Problema principală în procesarea imaginilor stereo o reprezintă deteția de corespondențe dintre punctele celor două imagini. Această deteție conduce la o creștere semnificativă a complexității de calcul necesară, complexitate care va crește odată cu eliminarea zgomotului din imagini cauzat de mișcarea vehiculelor. Tehnicile de deteție a obstacolelor folosind stereo viziune folosesc abordări diferite pentru a reduce complexitatea de calcul, de exemplu [3] folosește stereo viziunea doar pentru a calcula distanța față de obiecte, după ce acestea au fost identificate din imagini monoculare. Un sistem care nu folosește deteția de corespondențe dintre imagini stereo este prezentat în [9].

O abordare simplă și eficientă a deteției obstacolelor este prezentată în [4], unde autorul folosește un dreptunghi pentru a parcurge imaginea și calculează diferențe de culoare.

#### 3.3. Eliminarea efectului de perspectivă

O mare parte a implementărilor existente se bazează pe schimbarea perspectivei din care se “privește” imaginea. Eliminarea efectului de perspectivă are la bază presupunerea că vehiculul se află pe o suprafață plată. Unghiul prin care o scenă este achiziționată de o camera video și distanțele până la obiectele din fața camerei contribuie la asocierea unor informații la fiecare pixel din imagine. Efectul de perspectivă din imaginile capturate trebuie luat în considerare când se

procesează imagini pentru a lua în calcul acele informații din pixeli. Transformarea proiecției și eliminarea perspectivei, cunoscută în domeniul procesării imaginilor sub numele de “inverse perspective mapping” (IPM), permite eliminarea efectului de perspectivă și o remapează într-un domeniu 2D în care informația este distribuită omogen în toți pixelii. Această tehnică a fost introdusă inițial în [18] și a fost rafinată de Bertozzi în [3] [9]. Figura 3.3 arată un exemplu de eliminare a perspectivei. În lucrarea [4] autorul folosește o transformare a proiecției pentru a determina distanțele până la obiectele detectate. Pentru a realiza o transformare a perspectivei, sunt necesare unele caracteristici specifice ale etapei de achiziție: poziția camerei, orientarea camerei, parametrii extrinseci și intrinseci. Determinarea parametrilor specifici camerei video este prezentată în detaliat în [13] și [14], iar în cazul în care se utilizează librăriile OpenCV, procesul de calibrare al camerei este prezentat în [15].

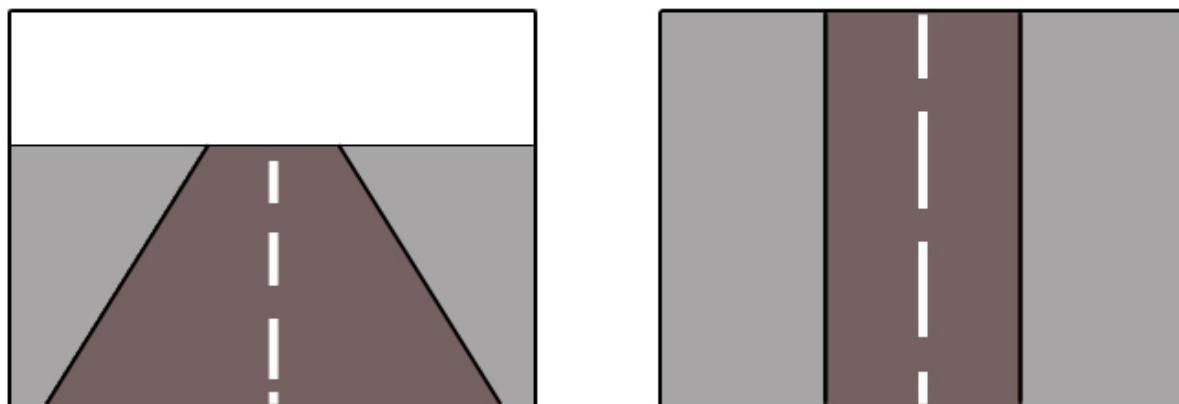


Figura 3.3 – Eliminarea efectului de perspectiva pe o suprafață de drum

În [11], este prezentat un algoritm de baza pentru a implementa eliminarea efectului de perspectiva (transformarea proiecției) folosind algoritmi de procesare a imaginilor disponibili în librăria OpenCV.

### 3.4. Achiziția datelor

Achiziția datelor este obținută de la camera video și de la senzori. În acest caz, senzorii folosiți sunt accelerometrul și camera video a telefonului mobil, iar datele sunt reprezentate de imaginile capturate.

În [10] autorii folosesc două camere video pentru a captura imaginile. Astfel, rezultatul achiziției datelor este o pereche de imagini cu informații despre poziția și orientarea camerelor. Aceste informații sunt utilizate împreună cu geometria epipolară pentru a determina coordonatele punctelor din lume. Lucrarea [4] prezintă o abordare în care achiziția imaginilor este făcută folosind doar o singură cameră. Se folosește o cameră video externă conectată la un laptop,

procesarea urmând să se efectueze pe computer. În aceasta lucrare de licență, se va folosi o singură cameră pentru a captura informațiile din trafic. Unele sisteme dezvoltate folosesc sisteme embedded în care atât achiziția, cât și procesarea datelor se efectuează pe anumite plăci de dezvoltare. Lucrarea [16] prezintă o astfel de abordare, în care nu se folosește un PC sau laptop pentru procesare. În [17] se prezintă un sistem în care procesarea este făcută pe un calculator cu procesor Pentium IV.

### 3.5. Eliminarea drumului din imagini

Pentru eliminarea pixelilor care conțin informații despre drum se folosește o segmentare bazată pe culoare a imaginilor. Algoritmii de segmentare color sunt în general implementați folosind sistemul de culoare RGB sau HSV.

#### 3.5.1. RGB

Spațiul de culoare RGB este un model de culori aditiv, în care cele 3 culori de baza: roșu, verde și albastru sunt folosite pentru a obține, reproduce alte culori. În modelul RGB, fiecare culoare este reprezentată ca o combinație a celor 3 culori. Culorile primare pot fi adunate pentru a produce culori secundare ale luminii: cyan (verde plus albastru), magenta (roșu plus albastru) și galben (roșu plus verde). Combinarea culorilor primare, roșu, verde și albastru produce culoarea albă (Figura 3.1).

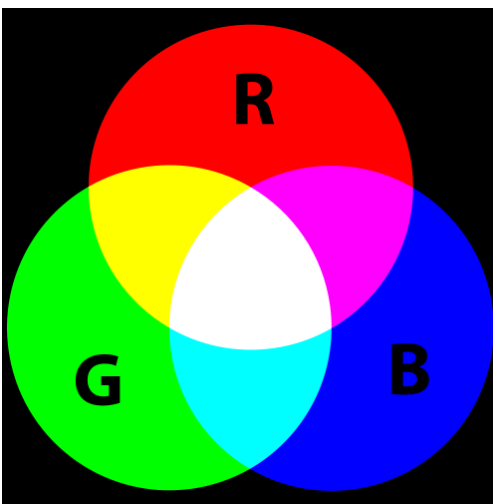


Figura 3.1 – spațiul de culoare RGB.

Modelul de reprezentare RGB este important deoarece este foarte apropiat de modul în care ochiul uman percepe noțiunea de culoare. De aceea, sistemul RGB a devenit foarte utilizat în domeniul calculatoarelor și acest sistem este un standard, fiind implementat și dezvoltat de ani de zile.

### 3.5.2. HSV, HSI, HSL

HSI, respectiv HSV reprezintă cele mai folosite sisteme de culori reprezentate sub forma de coordonate într-un cilindru (Figura 3.2). Aceste modele de culori au fost dezvoltate pentru a aproxima percepția umană asupra culorilor și pentru a fi mai intuitive în modificarea sau manipularea culorilor.

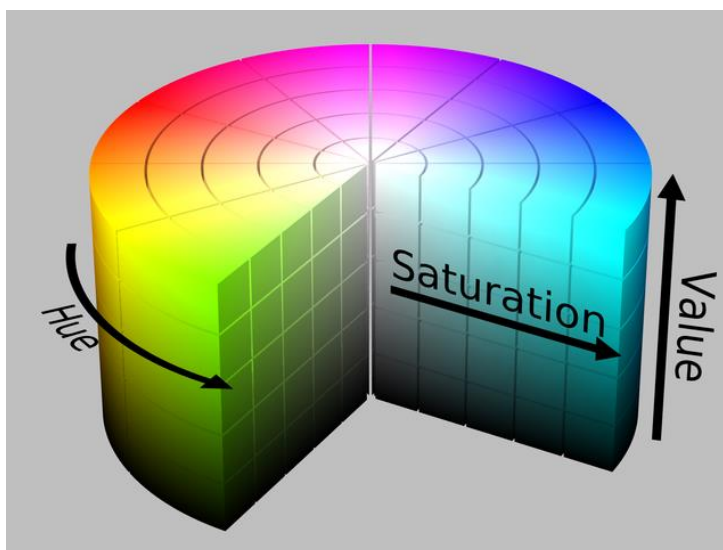


Figura 3.2 – spațiul de culoare HSV

- Hue (nuanța) – reprezintă informațiile despre culoare
- Saturation (saturația) – se referă la dominația nuanței (hue) în culoare, reprezintă puritatea culorii
- Intensity, Value (intensitatea) – reprezintă factorul de iluminare

Spațiul de culoare HSV are o variație mai mică a culorilor în diferite condiții de iluminare, comparativ cu spațiul RGB. De aceea, HSV este preferat pentru segmentarea imaginilor. În lucrarea [5] autorul folosește pentru segmentare imagini convertite în spațiul HSV.

### 3.5.3. Segmentarea imaginilor

În domeniul procesării imaginilor, segmentarea unei imagini se referă la partiționarea unei imagini în mai multe zone, segmente. Acest proces are ca scop schimbarea reprezentării unei imagini pentru a facilita analiza. Există mai multe tehnici de segmentare, cum ar fi: segmentare cu prag (en. thresholding), segmentare pe grupuri (en. clustering), segmentări bazate pe histograme, detecția muchiilor. Pentru domeniul de detecție în traficul rutier, segmentarea se face pe baza unui prag. Acest prag poate fi o valoare fixă, precălculată sau o valoare calculată automat. Acest tip de segmentare este cea mai simplă de implementat și în acest domeniu, una dintre cele mai eficiente



metode dacă este folosită în sistemul de culoare HSV. Prin alegerea unui prag, pixelii care sunt sub acel prag pot fi eliminați, în acest caz se aleg pixelii care aparțin suprafeței drumului. Un astfel de exemplu bazat pe spațiul de culoare HSV se regăsește în lucrarea [5], iar o variantă de segmentare folosind spațiul de culoare RGB este implementată în [4].

### **3.6. Dispozitive mobile**

Sistemul de detecție a obstacolelor trebuie să fie robust, precis, ușor de folosit și accesibil ca și costuri. Pentru aceasta, am ales să implementez sistemul de detecție folosind telefoane mobile inteligente cu sistem de operare Android. Etapele necesare implementării unui proiect Android sunt descrise în [12]. Pentru partea de procesare a imaginilor, sistemul folosește librăria OpenCV.

OpenCV (Open Computer Vision) este o librărie care conține algoritmi folosiți în procesarea imaginilor. Aceasta librărie a fost dezvoltată inițial de Intel și este open source. Librăria a fost dezvoltată pentru mai multe platforme, printre care Windows, Linux, iar variantele pentru telefoane mobile Android sau iOS au fost dezvoltate recent, unele funcții și algoritmi prezenți în versiunea pentru computere personale nefiind finalizate. O descriere a algoritmilor disponibili în OpenCV, dar și modul de utilizare al acestora se găsește în [11].

### **3.7. Aplicații existente pentru dispozitive mobile**

Alte produse software similare, disponibile atât pe platforma iOS și pe Android, oferă funcții de detecție a vehiculelor din fața mașinii curente. Aceste aplicații folosesc calcule bazate pe informațiile din GPS și oferă avertizări vizuale și sonore bazate pe calculul unui timp necesar pentru a ajunge la obstacolul din fața mașinii. Aplicația care a fost dezvoltată pentru această lucrare de licență se deosebește de produsele existente prin faptul că folosește o calibrare a camerei pentru a obține o proiecție diferită a drumului, iar avertizările emise în cazul unui pericol, se fac pe baza distanțelor calculate.

#### *3.7.1. Aplicația iOnRoad*

iOnRoad [19] folosește camera video și gps pentru a detecta vehicule aflate în fața mașinii și oferă avertizare conducătorului auto în cazul unei posibile coliziuni. Aplicația folosește informațiile din gps pentru a calcula viteza vehiculului curent și oferă avertizări în cazul în care timpul necesar de a ajunge la vehiculul din față este prea mic.

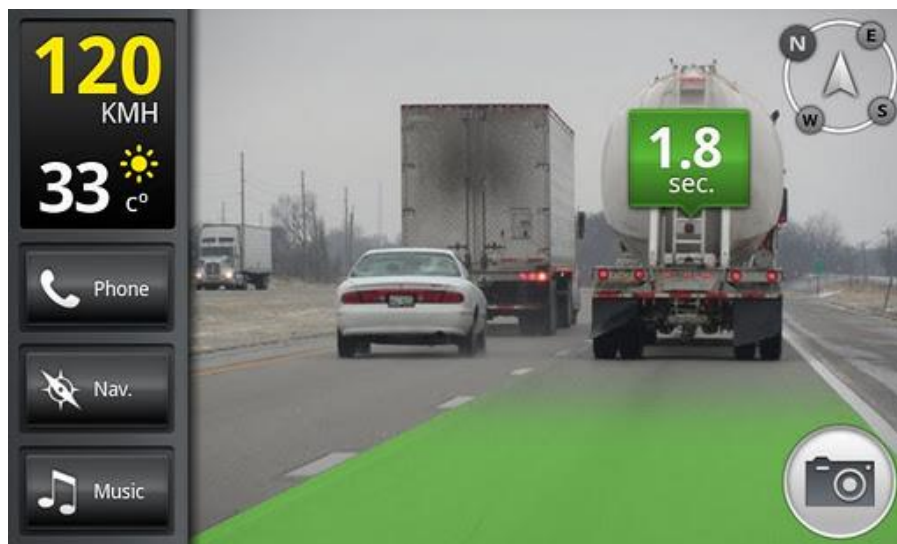


Figura 3.3 – Aplicația iOnRoad

### 3.7.2. Mobileye

Aplicația Mobileye [20], oferă o avertizare pentru posibile coliziuni cu obstacole aflate în fața vehiculului. Această avertizare este bazată pe calculul timpului rămas până la coliziune este prea mic. Conducătorul auto este alertat prin afișarea unei mașini roșii pe ecranul telefonului mobil. Aplicația Mobileye pentru Android, este dependentă de un dispozitiv extern produs de aceeași firmă. Acest dispozitiv oferă o camera video și un sistem de procesare a informațiilor din cameră, aplicația pentru telefoane mobile fiind folosită doar pentru afișarea informațiilor din hardware-ul extern.



Figura 3.4 – Aplicația Mobileye

### 3.7.3. Aplicația Augmented Driving

Aplicația Augmented Driving [21], disponibilă pe platforma iOS, oferă detecția vehiculelor și monitorizează distanța față de acestea. Vehiculele detectate pe banda curentă sunt încadrate în dreptunghiuri roșii, iar cele mai îndepărtate sunt încadrate în dreptunghiuri galbene. Aplicația folosește și modulul GPS din telefoanele mobile pentru a calcula intervalul de timp până la o posibilă coliziune. Informațiile utile sunt suprapuse peste imaginile din cameră, iar în cazul în care distanța față de obstacolul din față este prea mică, se emite un semnal de avertizare vizual și sonor.

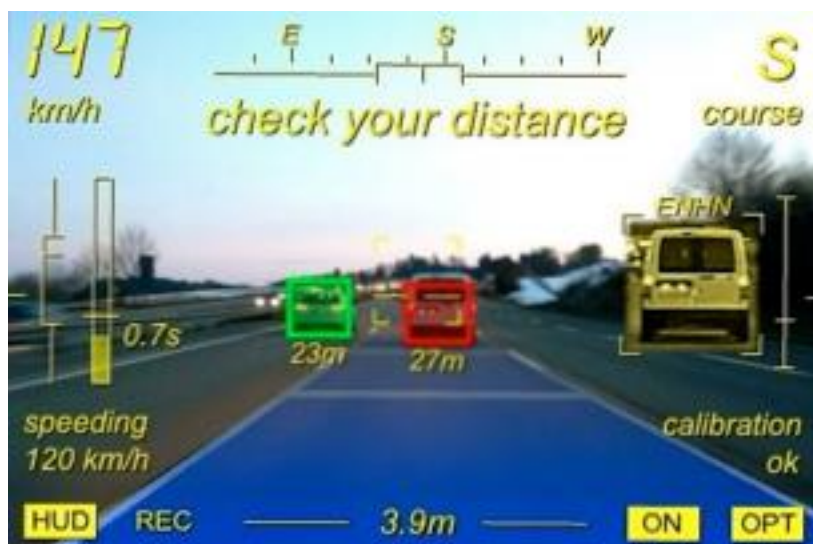


Figura 3.5 – Aplicația Augmented Driving

## 4. Analiză și fundamentare teoretică

### 4.1. Cerințe funcționale și non-funcționale

În cadrul acestui proiect, se identifică funcțiile de atenționare la coliziune, se va urmări evitarea coliziunii cu obstacole din fața vehiculului prin avertizarea, atenționarea vizuală și auditivă a conducătorului auto.

Mediul în care se va folosi aplicația este reprezentat de trafic și de participanții din traficul rutier.

- Aplicația va fi folosită de conducătorul auto
- Asistarea conducătorului depinde de mediul înconjurător, de starea drumului și de numărul de participanți din trafic
- Platformele folosite în prezent pentru aplicații similare sunt reprezentate de telefoane mobile folosind sisteme de operare bazate pe Android sau iOS
- Aplicația depinde de folosirea librăriilor oferite de OpenCV care sunt integrate în produsul final

#### 4.1.1. Cerințe funcționale

Scopul oricărui sistem de asistare a conducătorilor auto în traficul rutier este a oferi asistență și suport în condiții de trafic normal, dar sistemele pot și reacționa în cazul unor greșeli ale șoferului.

Sistemul implementat va trebui să atenționeze conducătorii auto despre posibile obstacole apărute pe șosea. Un șofer s-ar putea să nu observe și să nu reacționeze în timp util din cauza neatenției, lipsei de concentrare sau din cauza oboselii. Pentru aceste situații, sistemul va trebui să prevină situațiile neplăcute prin avertizarea conducătorului auto.

Pentru a realiza aceste funcționalități, sistemul trebuie să:

- Detecteze și să urmărească obstacolele din drum
- Avertizeze conducătorul autovehiculului

#### 4.1.2. Cerințe non-funcționale

Pentru a realiza detecția de obstacole în traficul rutier, sistemul implementat trebuie să fie robust. De aceea pentru partea de detecție a obstacolelor am implementat un algoritm bazat pe calculele unor histograme care s-a dovedit a fi foarte eficient. Datorită complexității algoritmului, în unele situații cu obstacole multiple, timpul de procesare este mai redus și astfel scade numărul de cadre pe secundă.

Alte cerințe non-funcționale importante sunt disponibilitatea și uzabilitatea sistemului. Aplicația dezvoltată va fi disponibilă tuturor conducătorilor auto care dețin un telefon mobil

inteligent cu sistem de operare Android. Gradul de uzabilitate va fi ridicat, deoarece interfața grafică a aplicației va fi intuitivă și ușor de folosit.

Performanța sistemului va fi măsurată luând în considerare timpii de răspuns ai sistemului, numărul de cadre pe secunda procesate (frames/second) și precizia distanței determinate până la obstacole.

## 4.2. Cazuri de utilizare

Cazul general de utilizare al proiectului dezvoltat în cadrul acestei lucrări de licență este ilustrat în figura următoare.

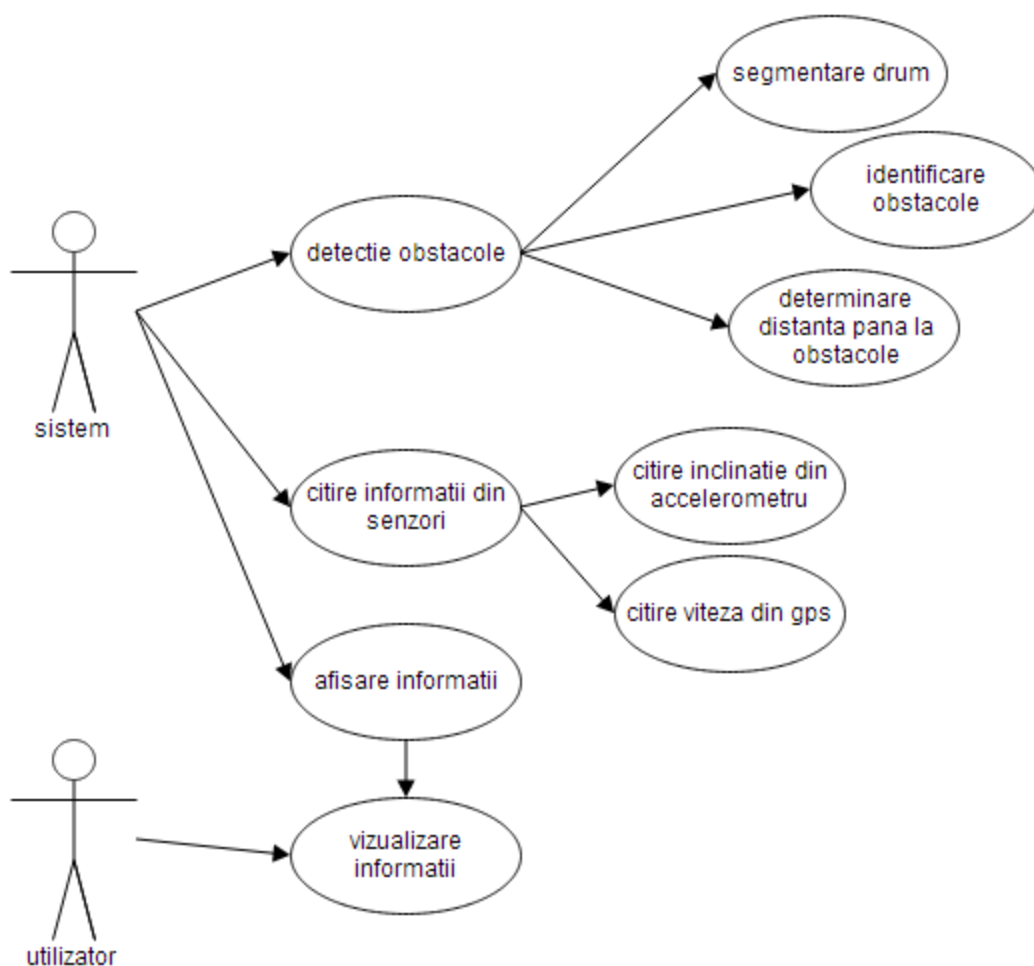


Figura 4.1 Diagrama cazului de utilizare a sistemului

### 4.3. Prezentare de ansamblu a sistemului

Sistemele existente de detecție a obstacolelor în trafic folosesc în general aceeași pași, indiferent de metodele de detecție folosite, sau numărul de camere video disponibile. Pașii sunt: achiziția datelor, transformarea perspectivei, eliminarea, segmentarea drumului, detecția obstacolelor și calcularea distanțelor și afișarea rezultatelor.

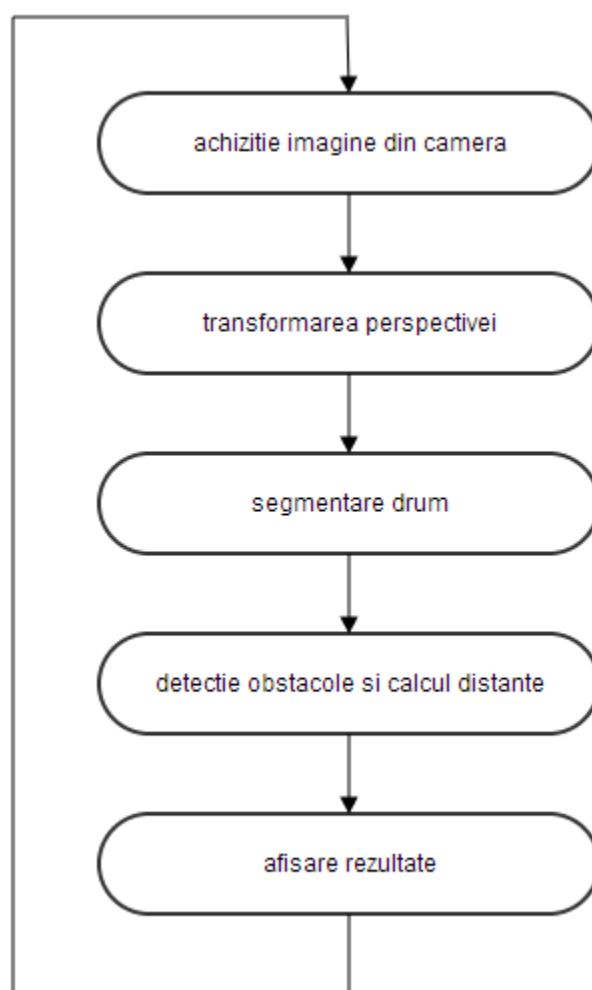


Figura 4.2 Prezentarea generală a sistemului de detecție a obstacolelor

### 4.4. Calibrarea camerei video

Primul pas pentru generarea imaginii cu perspectiva schimbată (IPM) este calibrarea camerei video folosite. În aceasta lucrare, am folosit o calibrare manuală a camerei, prin amplasarea unor marcaje cunoscute pe drum și măsurarea unor parametri. Figura 4.3 conține tipul

de marcaj utilizat pentru procesul de calibrare. Marcajele de pe drum (X-urile) oferă o referință dintre pixelii din imagine și distanța fizică din lumea reală.



Figura 4.3 – Tipul de marcaj folosit pentru calibrare.

Parametrii intrinseci reprezintă caracteristicile optice și geometrice ale camerei (caracteristici interne):

- Distanța focală
- Centrul optic
- Coeficienții de distorsiune radială și tangențială (ai lentilelor obiectivului)

Parametrii extrinseci sunt reprezentați de vectorul de translație și vectorul de rotație. Parametrii extrinseci și cei intrinseci vor fi folosiți pentru eliminarea distorsiunilor din imagini, și pentru a calcula matricile de transformare dintr-o perspectivă în alta.

#### **4.5. Eliminarea efectului de perspectivă**

Schimbarea perspectivei, cunoscută ca și “inverse perspective mapping” sau “bird’s eye view” în engleză, se referă la schimbarea poziției camerei în raport cu lumea reală. Folosind calcule matematice (o transformare geometrică), se poate transforma, modifica perspectiva din care se “privește” spre imagine. Pentru a obține o nouă perspectivă, sunt necesari mai mulți parametri ai camerei folosite.

Generarea unei proiecții de sus a drumului oferă următoarele avantaje:

- Reducerea efectului de perspectivă astfel încât drumul și benzile de circulație nu mai converg spre linia de orizont, ci sunt verticale și paralele
- Determinarea distanței se poate calcula mai ușor și mai precis având această imagine de sus a drumului, este necesară doar o corespondență între pixelii din imagine și metrul liniar din lumea reală

- Concentrarea doar pe o anumită zonă de interes din imagine, prin acest lucru obținându-se un bonus de performanță și rapiditate în execuție

Pentru a obține aceasta imagine IPM, se presupune că drumul este plat, neted și se utilizează parametri extrinseci (unghiurile de înclinare, care formează vectorul de translație și vectorul de rotație) și parametri intrinseci (distanța focală și centrul optic).

Prima modalitate de generare se bazează pe mai multe formule de calcul pentru a obține matricea de transformare.

Mai întâi, se definesc coordonatele lumii:  $F_w = \{ X_w, Y_w, Z_w \}$ , coordonatele camerei  $F_c = \{ X_c, Y_c, Z_c \}$  și coordonatele imaginii:  $F_i = \{ u, v \}$ .

Înălțimea camerei față de drum este notată cu  $h$ . Unghiul de înclinare al camerei față de drum este notat cu  $\alpha$  (pitch angle), iar unghiul de rotație al camerei în raport cu axul drumului este notat cu  $\beta$  (yaw angle).

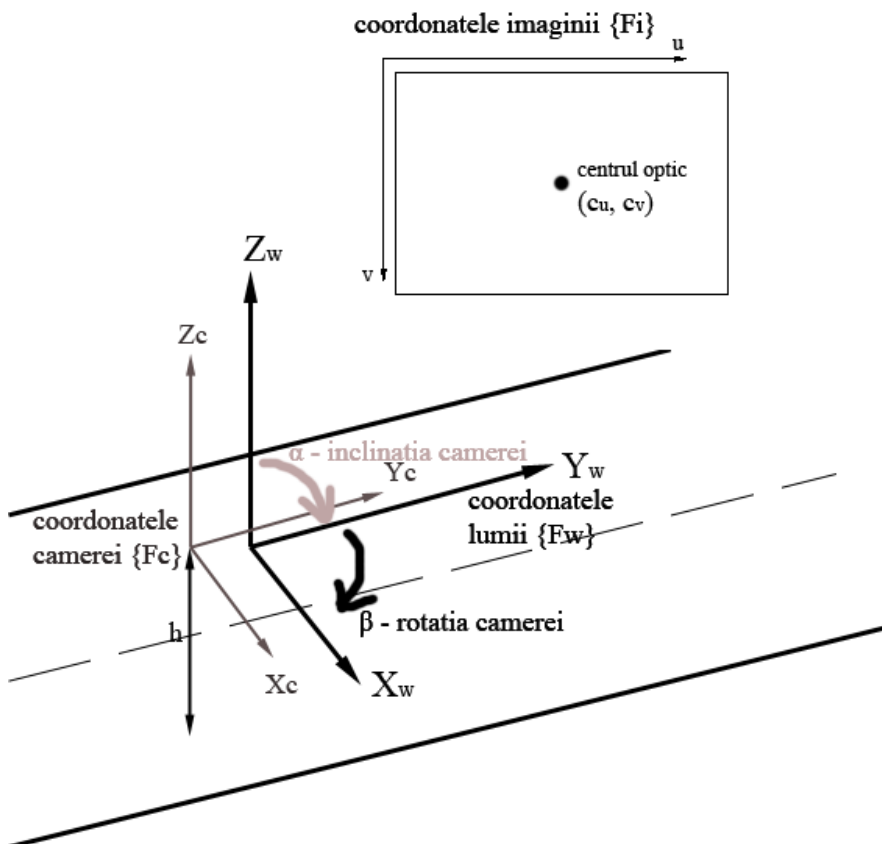


Figura 4.4 – Sisteme de coordonate ale lumii, camerei video și ale imaginii



Se folosesc următoarele notații:  $c_1 = \cos(\alpha)$ ,  $c_2 = \cos(\beta)$ ,  $s_1 = \sin(\alpha)$ ,  $s_2 = \sin(\beta)$  și centrul optic având coordonatele  $(c_u, c_v)$ . Astfel, începând cu orice punct din planul imaginii  $P = \{u, v, 1, 1\}$ , se poate calcula proiecția punctelor pe planul drumului  $(X_w, Y_w)$ , aplicând următoarea matrice de transformare  $T$ :

$$T = h \begin{bmatrix} -\frac{1}{f_u} c_2 & \frac{1}{f_v} s_1 s_2 & \frac{1}{f_u} c_u c_2 - \frac{1}{f_v} c_v s_1 s_2 - c_1 s_2 & 0 \\ \frac{1}{f_u} s_2 & \frac{1}{f_v} s_1 c_1 & -\frac{1}{f_u} c_u s_2 - \frac{1}{f_v} c_v s_1 c_2 - c_1 c_2 & 0 \\ 0 & \frac{1}{f_v} c_1 & -\frac{1}{f_v} c_v c_1 + s_1 & 0 \\ 0 & -\frac{1}{h f_v} c_1 & \frac{1}{h f_v} c_v c_1 - \frac{1}{h} s_1 & 0 \end{bmatrix}$$

$$c_1 = \cos(\alpha);$$

$$c_2 = \cos(\beta);$$

$$s_1 = \sin(\alpha);$$

$$s_2 = \sin(\beta);$$

Folosind un produs software de calibrare al camerei, oferit de OpenCV, se pot genera și determina vectorii de translație și matricea de rotație. Procedura de calibrare a camerei constă în amplasarea camerei într-un punct fix și capturarea unei set de imagini care conțin un model cunoscut. Capturând poze succesive cu camera fixă, iar modelul în poziții diferite, se pot calcula relațiile între punctele cheie ale modelului (pattern-ului) și corespondența lor în succesiunea de imagini și într-un final se obțin parametrii camerei: matricea de rotație și vectorul de translație.

Distanțele focale,  $F_x$  respectiv  $F_y$ , împreună cu centrul optic reprezentat de  $C_x$  și  $C_y$  formează matricea camerei (matricea intrinsecă), notată cu  $A$ :

$$A = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

Matricea de proiecție (transformare) este definită ca fiind:

$$P = A * [R | T],$$

unde  $R$  reprezintă matricea de rotație  $3 \times 3$ , iar  $T$  reprezintă vectorul de translație.

$[R | T]$  – matricea de rotație  $R$ , împreună cu vectorul de translație corespunzător  $T$ , formează o matrice  $3 \times 4$  care reprezintă parametrii extrinseci ai camerei. Această matrice definește orientarea și poziția camerei.

Transformarea unui punct se poate exprima:

$$m' = A * [R \mid T] M'$$

sau:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

O alta varianta de obținere a matricei de proiecție este folosind două seturi de puncte. Primul set conține 4 puncte care delimitează drumul și care formează un trapez. În cel de-al doilea set, sunt calculate manual pozițiile celor 4 puncte de intrare, care de această dată sunt sub forma unui dreptunghi, cele două linii laterale sunt paralele. Folosind funcția OpenCV “getPerspectiveTransform” se va obține o matrice de transformare, care aplicată pentru orice punct sursă, va determina poziția nouă în cadrul imaginii IPM.

Un exemplu cu transformarea perspectivei este prezentat în Figura 4.4 și Figura 4.5:



Figura 4.4 – Imaginea de intrare

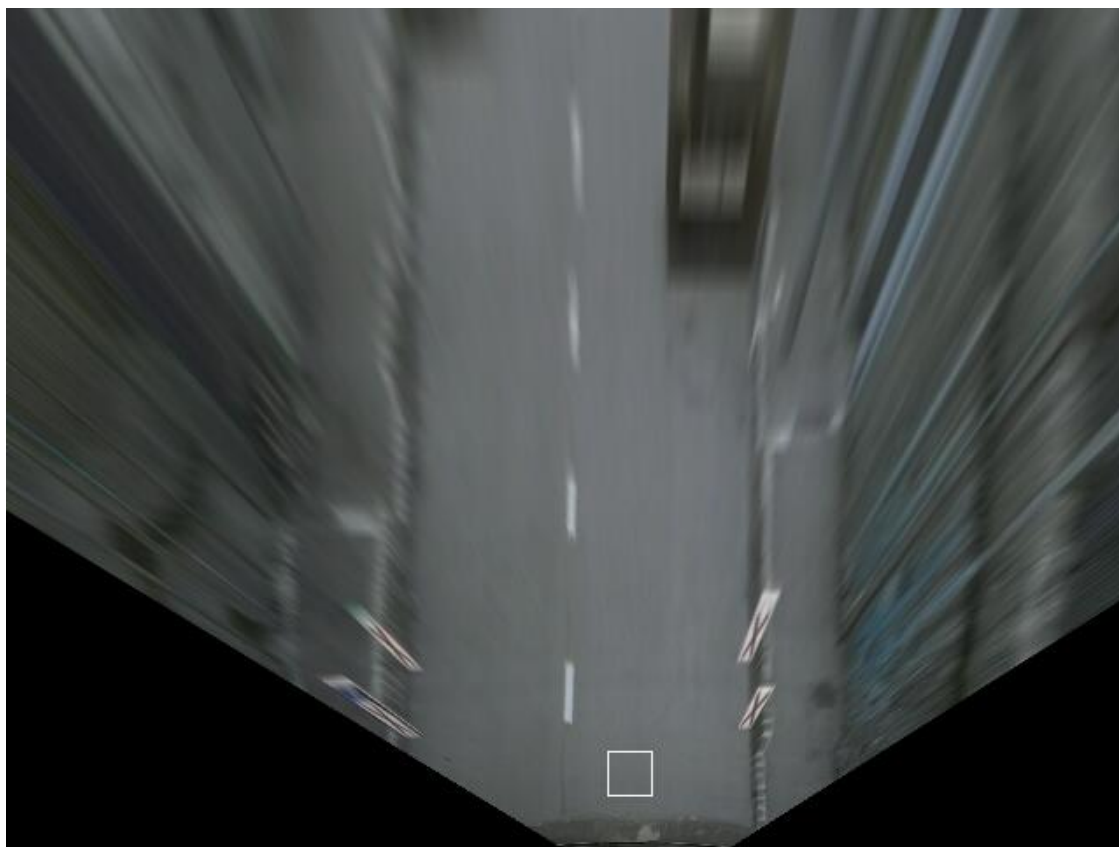


Figura 4.5 – Imaginea cu efectul de perspectivă eliminat

#### **4.6. Segmentarea drumului**

Pentru domeniul de detecție al obstacolelor din traficul rutier, segmentarea se face pe baza unui prag, care poate avea o valoare fixă, precalculeată sau o valoare calculată automat în fiecare cadru. Acest tip de segmentare este cea mai simplă de implementat și în acest domeniu, una dintre cele mai eficiente metode dacă este folosită în sistemul de culoare HSV. Prin alegerea unui prag, pixelii care sunt sub acel prag pot fi eliminați, în acest caz se aleg pixelii care aparțin suprafeței drumului. În această lucrare am implementat o segmentare pe culori în care pragul maxim de culoare este calculat automat în fiecare cadru, imagine. Deoarece condițiile de drum și de iluminare pot diferi, acest tip de calcul automat al pragului, bazat pe alegerea unei zone din fața vehiculului și calcularea unei culori medii, este de preferat.

#### **4.7. Componenta Bluetooth**

Bluetooth este o tehnologie standard pentru transmisie de date și comunicare pe distanțe mici, prin utilizarea undelor radio cu lungimi de undă cuprinse între 2400 - 2480 MHz. Bluetooth reprezintă un protocol bazat pe schimb de pachete folosind o arhitectura de tipul Client – Server

(sau Master – Slave). Aplicația dezvoltată în cadrul acestui proiect va fi extinsă pentru a putea transmite anumite informații relevante altor dispozitive conectate la mașină, transmiterea efectuându-se prin intermediul standardului Bluetooth.

Pentru testarea conexiunii Bluetooth și transmiterea informațiilor s-a folosit un laptop împreună cu un modul extern de Bluetooth prin USB. După conectarea telefonului și a laptop-ului, informațiile transmise sunt citite folosind un terminal, setat pe un port COM.

#### 4.8. Componenta GPS

În acest proiect s-a folosit modulul de gps intern al telefonului mobil cu scopul de a afișa vizual viteza curentă de deplasare a autovehiculului. Pentru a facilita achiziția de date din modulul gps, sistemul de operare Android oferă mai multe funcții și metode publice.

#### 4.9. Sistemul de operare Android

Android este un sistem de operare open source disponibil pentru telefoane mobile și tablete. Sistemul de operare Android folosește kernel-ul de Linux 2.6 pentru abstractizarea hardware-ului. Android este structurat pe mai multe nivele: Applications, Application Framework, Native (Libraries + Android Runtime, Hardware Abstraction Layer), Linux Kernel. Figura următoare prezintă structura sistemului de operare Android:

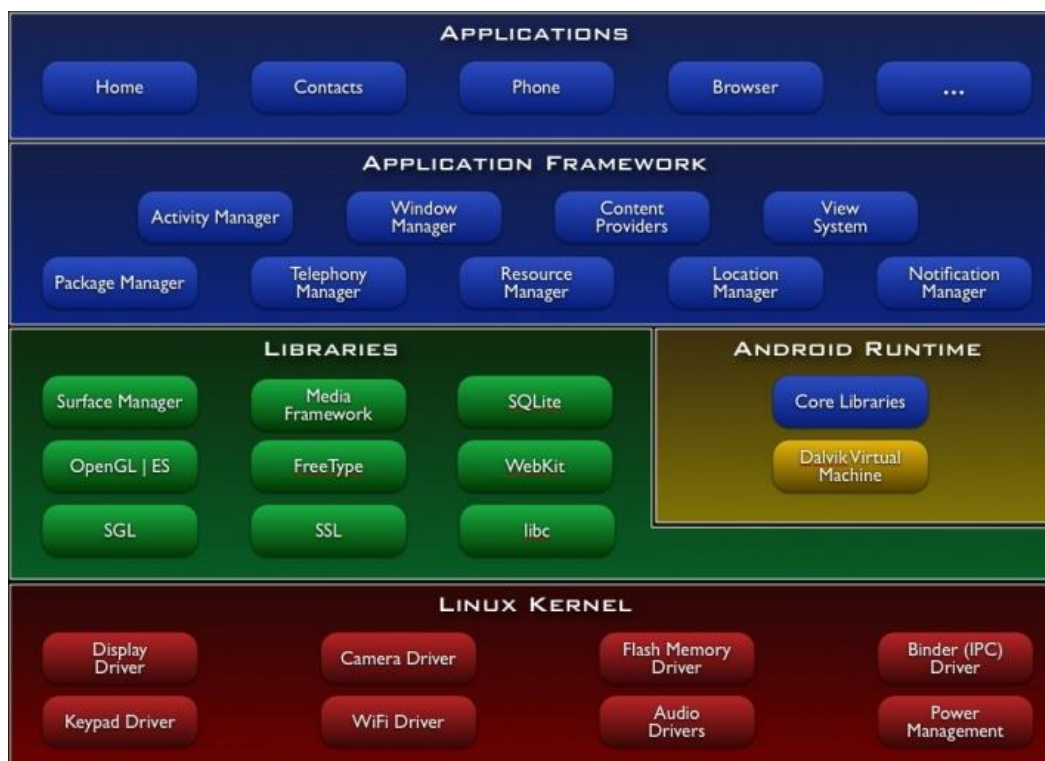


Figura 4.6 – Arhitectura sistemului de operare Android

#### 4.9.1. Nivelul kernel (*Kernel Layer*)

Nivelul de kernel, este nivelul cel mai de jos al arhitecturii Android. Oferă următoarele funcționalități pentru sistemul de operare:

- Abstractizarea hardware-ului
- Programe de management al memoriei
- Setări de securitate
- Programe de management al bateriei
- Drivere hardware
- Suport pentru librării partajate
- Conexiune la rețele (GSM, WiFi, etc.)

Sistemul de operare Android folosește un framework pentru realizarea comunicării dintre procese (IPC Inter Process Communication).

#### 4.9.2. Nivelul de Librării Native (*Native Libraries Layer*)

Următorul nivel al arhitecturii sistemului Android este reprezentat de librăriile native. Aceste librării conțin algoritmi, instrucțiuni pentru a realiza diferite task-uri (exemplu: înregistrarea audio și video este realizată de Media Framework Library).

Librării native:

- Surface Manager
- SGL: grafică 2D
- Open GL|ES: grafică 3D
- Media Framework: audio, video & poze
- Free Type: randare/afișare font-uri și text
- WebKit: browser engine
- Libc (System C libraries)
- SQLite
- OpenSSL

Pe acest nivel se mai găsește și layer-ul Android runtime, care include funcționalități Java (core libraries) și mașina virtuală Dalvik.

#### 4.9.3. Nivelul Framework-ului de Aplicații (*Application Framework Layer*)

Acest nivel conține anumite module și blocuri cu care aplicațiile dezvoltate de programatori vor interacționa în mod direct. Aceste module și programe gestionează funcțiile de baza ale telefonului, cum ar fi: managementul resurselor, apelurilor telefonice, etc. Programatorii pot utiliza aceste unelte de bază în procesul de dezvoltare al aplicațiilor Android.

Principalele module disponibile în acest nivel sunt:

- Activity Manager: gestionează ciclul de viață al fiecărei aplicații
- Content Providers: gestionarea schimbului de date între aplicații
- Telephony Manager: administrează toate apelurile vocale
- Location Manager: managementul locației folosind GPS sau rețeaua GSM
- Resource Manager: gestionează diferite tipuri de resurse utilizate în aplicații

#### *4.9.4. Nivelul de Aplicații (Application Layer)*

“Application Layer” reprezintă nivelul cel mai de sus din arhitectura Android. Acest nivel oferă aplicații cu care utilizatorul va interacționa, de exemplu: contacte, telefon, browser internet, etc.

Librăria de aplicații (en. Application library): Android folosește o versiune specială de libc, numită Bionic libc. Spre deosebire de varianta GNU libs (glibs), varianta implementată pe Android ocupă un spațiu mai redus (aproximativ 200KB, glibs are peste 400KB). Reducerea dimensiunii s-a realizat prin renunțarea la unele funcționalități și funcții C++ prea complexe sau complicate: de exemplu s-a renunțat la partea de excepții C++. Bionic libc oferă o implementare parțială pentru aplicații multi-thread folosind “pthread”, nu suportă în totalitate POSIX și nu este compatibilă cu glibc.

Kernel-ul Linux se ocupă de managementul memoriei telefonului sau a tabletei cu sistem de operare Android. Este bazat pe pagini, iar adresele virtuale sunt mapate pe adrese fizice. Android nu oferă memorie virtuală.

### **4.10. Aplicații Android**

Aplicațiile dezvoltate pentru Android vor fi executate prin intermediul Dalvik Virtual Machine (Dalvik VM), care este o mașină virtuală Java, special dezvoltată pentru Android și concepută pentru a funcționa pe sisteme cu resurse limitate. Fiecare aplicație va avea un proces independent și va rula până când procesul va fi terminat de sistem în cazul în care memoria este plină. O aplicație poate avea mai multe componente: activități, servicii, anunțuri (en. Broadcasts), furnizori de conținut (en. Content Providers).

Principala componentă a unei aplicații o reprezintă activitățile (en. Activities) și reprezintă fereastra vizibilă pe ecranul telefonului sau a tabletei. Prin intermediul unei activități se pot afișa informații utile și se pot citi alte informații noi de la utilizator.

O altă componentă o reprezintă serviciile, care sunt transparente din punctul de vedere al utilizatorilor, deoarece ele rulează în fundal. Folosind servicii nu se vor afișa mesaje pentru utilizatori și nici nu se implementează o interfață grafică. Un exemplu pentru servicii îl reprezintă un client de email care folosește un serviciu pentru a rula în fundal și pentru verificarea căsuței de email.

Următoarea componentă a unei aplicații este reprezentată de anunțuri (en. Broadcasts). Aceste componente răspund la orice eveniment transmis de sistemul de operare sau de la alte componente. Majoritatea anunțurilor provin de la sistemul de operare. De exemplu Android va emite un “broadcast” când ecranul a fost stins sau când bateria este la un nivel prea scăzut.

Furnizorii de conținut (en. Content Providers) reprezintă o alta componenta a unei aplicații Android. Acești furnizori permit accesul aplicațiilor la datele și informațiile partajate, publice disponibile în telefon. De exemplu, anumite informații pot fi salvate pe platforma Android folosind memoria internă sau SQLite, iar furnizorii de conținut facilitează accesul din aplicații la aceste date.

Integrarea și activarea acestor componente ale unei aplicații se face prin intermediul unor mesaje asincrone, numite “Intents”. La Android, un “Intent” face conexiunea între mai multe componente. De exemplu, pentru pornirea unei activități noi se va trimite un nou “intent” la metoda “startActivity()”. Toate componentele trebuie declarate într-un fișier XML numit AndroidManifest. Acest fișier conține și alte informații despre aplicație, cum ar fi: versiunea minimă de Android suportată de aplicație, nivelul de acces necesar aplicației (de exemplu: aplicația dezvoltată are acces sa utilizeze serviciul de date, să citească contactele existente în agenda telefonică, etc.).

## 5. Proiectare de detaliu și implementare

### 5.1. Arhitectura generală a sistemului

Arhitectura sistemului este compusă din mai multe părți și este prezentată în Figura 5.1

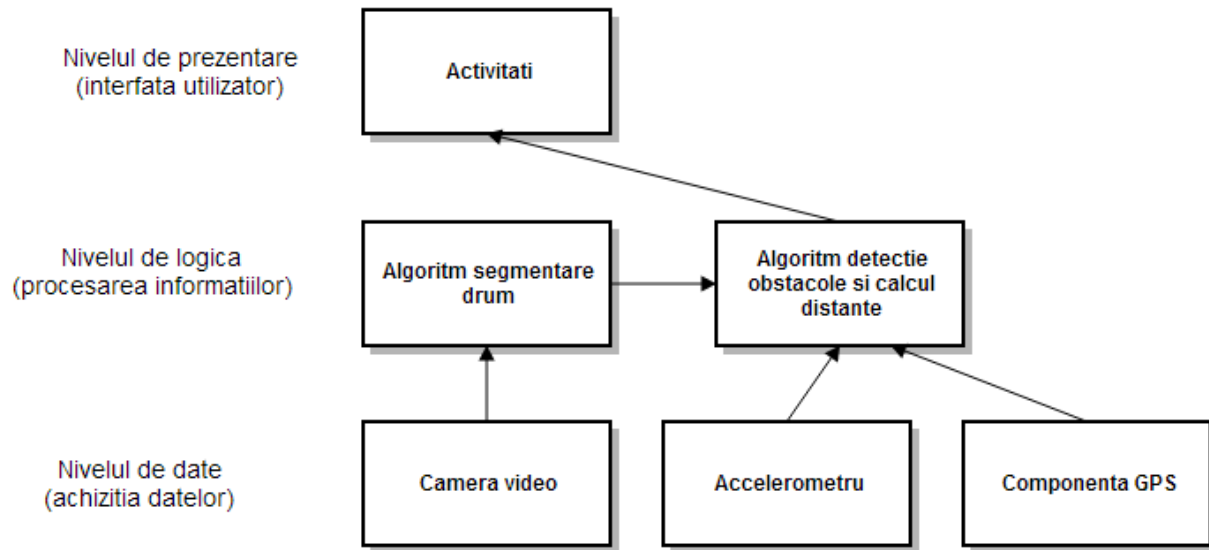


Figura 5.1 – diagrama de arhitectură conceptuală a sistemului

### 5.2. Unelte de programare și dezvoltare

Acest proiect a fost implementat folosind două limbaje de programare: Java și C++ într-un mediu de dezvoltare integrat (Eclipse).

#### 5.2.1. Java

Java este un limbaj de programare orientat pe obiecte dezvoltat la începutul anilor 90 de un inginer din cadrul companiei Sun Microsystems. În ziua de azi, o mare parte a aplicațiilor distribuite sunt scrise în Java. Limbajul împrumută o parte din sintaxa specifică limbajului C și C++, dar prezintă un model mai simplu al obiectelor. Principalul avantaj al Java este acela că un program compilat corect și fără erori poate rula pe orice platformă care suportă mașină virtuală Java (Java Virtual Machine). În limbajele tradiționale, această portabilitate nu este posibilă.



### 5.2.2. C++

C++ este un limbaj de programare care a fost dezvoltat în anii 1980 și a devenit popular la începutul anilor 90. Acest limbaj a fost conceput ca o îmbunătățire a limbajului C, prin adăugarea de clase, suprascrierea metodelor, moștenire, funcții virtuale și excepții. Spre deosebire de Java, limbajul C++ necesită alocarea și eliberarea memoriei să fie făcută explicit de către programator. În limbajul Java, alocarea de memorie și eliberarea (“garbage collection”) este făcută automat.

### 5.2.3. Eclipse IDE

Dezvoltarea de aplicații Java și C++ este suportată de mai multe medii de dezvoltare. O unealta foarte importantă este Eclipse IDE – Integrated Development Kit. Eclipse este aplicația preferată pentru dezvoltarea de aplicații Android.

### 5.2.4. Android plugin pentru Eclipse

Android Development Toolkit (ADT) reprezintă un modul (plugin) pentru Eclipse, recomandat de echipa Android pentru dezvoltarea de aplicații. Acest plugin oferă toate facilitățile pentru a dezvolta aplicații pentru toate platformele existente de Android, oferind suport pentru crearea rapidă de proiecte, pentru dezvoltarea interfeței de utilizator (împreună cu un editor grafic), oferă suport pentru depanare (debug) la aplicații și oferă posibilitatea de a exporta aplicația în format “.apk” pentru a putea fi ușor distribuită.

## 5.3. Librăria OpenCV

OpenCV este o librărie de procesare de imagini (en. Computer Vision) open source, dezvoltată de Intel și suportată de Willow Garage. Funcțiile și algoritmi principali disponibili sunt descriși în cartea [11]. Librăria a fost implementată pe mai multe platforme, printre care și Android. Scopul acestei librării este de a oferi o colecție a celor mai eficienți algoritmi din domeniul procesării de imagini.

## 5.4. Prezentare de ansamblu a algoritmului

Algoritmul de detecție a obstacolelor din traficul rutier folosește funcții predefinite ale librăriei de procesare de imagini OpenCV. Aplicația folosește modulul video al camerei de filmat a telefonului mobil. Fiecare imagine surprinsă în trafic, este preluată de algoritm și salvată într-o structură de tip matrice. Imaginile preluate folosind camera video sunt în format RGB (Red, Green, Blue) și sunt setate la o rezoluție fixă de 640x480 pixeli. S-a ales acest format 4:3 standard, pentru a reduce timpii de procesare și a facilita aplicația să ruleze cursiv și fără întârzieri.

În acest proiect, am implementat un algoritm de detecție a obstacolelor din traficul rutier folosind anumite concepte și idei prevăzute în lucrarea [3]. Spre deosebire de implementarea prezentată în articol, acest proiect folosește un sistem monocular, o singură camera video – cea a telefonului mobil. Fiind un articol de referință în domeniul procesării de imagini, am ales să

implementez o variantă asemănătoare și adaptată la resursele disponibile și oferite de telefoanele mobile din ultimii ani.

Fluxul general al algoritmului implementat în această lucrare este descris în figura 5.2. Proiectul de față folosește o soluție implementată în mediul de dezvoltare Eclipse folosind unelte oferite de echipa Android. Pentru optimizarea aplicației, algoritmul de detecție al obstacolelor a fost implementat folosind limbajul de programare C++ și folosind uneltele Android NDK (Native Development Kit). Utilizând dezvoltarea aplicațiilor în cod nativ, se obține o îmbunătățire semnificativă a timpilor de execuție și de procesare, deoarece aplicația nu va executa algoritmi de detecție folosind mașina virtuală Dalvik. Aplicațiile dezvoltate folosind NDK se execută într-o zonă de memorie separată care necesită alocarea de zone și eliberarea memorie să fie făcută explicit de către programator.

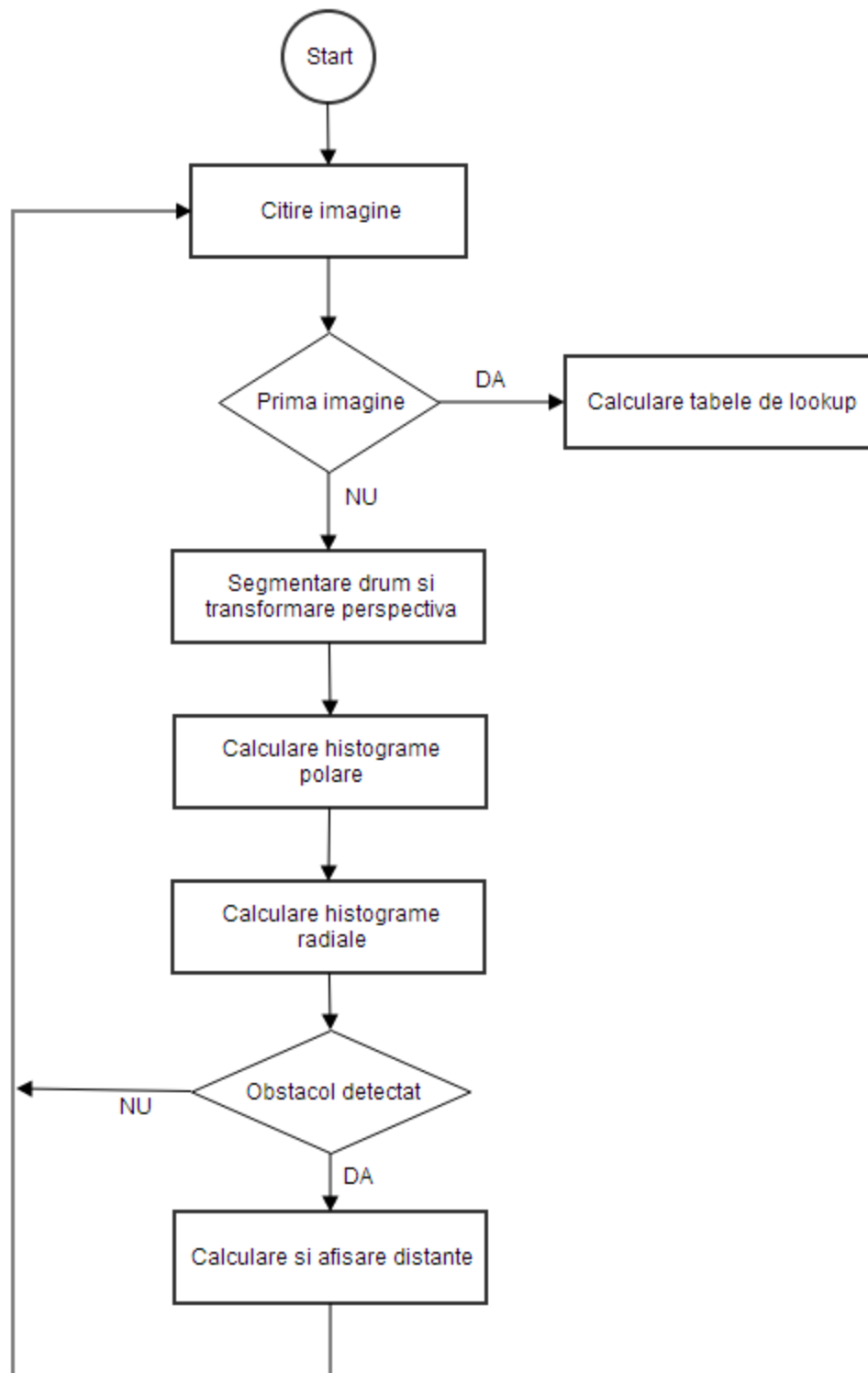


Figura 5.2 – Fluxul general al algoritmului de detecție a obstacolelor

### 5.5. Inițializare tabele

Înainte de procesarea fiecărui frame, algoritmul va calcula anumite valori constante care vor fi refolosite și apelate pe parcursul procesării. În prima etapă, se va calcula tabelul care va conține corespondențe între pixelii din imaginea sursă și pixelii din imaginea cu efectul de perspectivă eliminat. Imaginea este transformată într-o vedere de tipul “bird’s eye view”, o proiecție de sus a drumului folosind funcția *warpPerspective* din librăria OpenCV. Apoi se va parcurge imaginea pe înălțime și lățime, iar pentru fiecare punct din imaginea cu perspectiva eliminată se va calcula punctul corespondent din imaginea sursă (de intrare). Valorile pentru coordonatele x sunt salvate într-un tabel, iar valorile pentru coordonatele y sunt salvate în alt tabel lookup („map\_x”, respectiv „map\_y”).

Deoarece algoritmul implementat va folosi sistemul de coordonate polar, sunt necesare anumite valori ale arc tangentei pentru puncte (pixeli) din imagine. Am calculat toate valorile pentru arc tangente la începutul execuției programului și aceste valori sunt salvate într-un tabel (atanMat). Este necesar calculul arc tangentei pentru anumiți pixeli, deoarece se va calcula o histogramă polară, iar adăugarea valorilor în aceasta histogramă se va face pe baza de votare. Arc tangenta este folosită pentru a calcula unghiul, respectiv index-ul fiecărui pixel care aparține unei linii drepte proiectate pe imagine care are originea dintr-un punct de focus F (368, 480).

Folosind valori pre-calculate la lansarea programului, s-a obținut o îmbunătățire semnificativă a sistemului și a timpului de execuție necesar fiecărei imagini. Figura 5.3 reprezintă imaginea de intrare obținută din camera video a telefonului mobil.

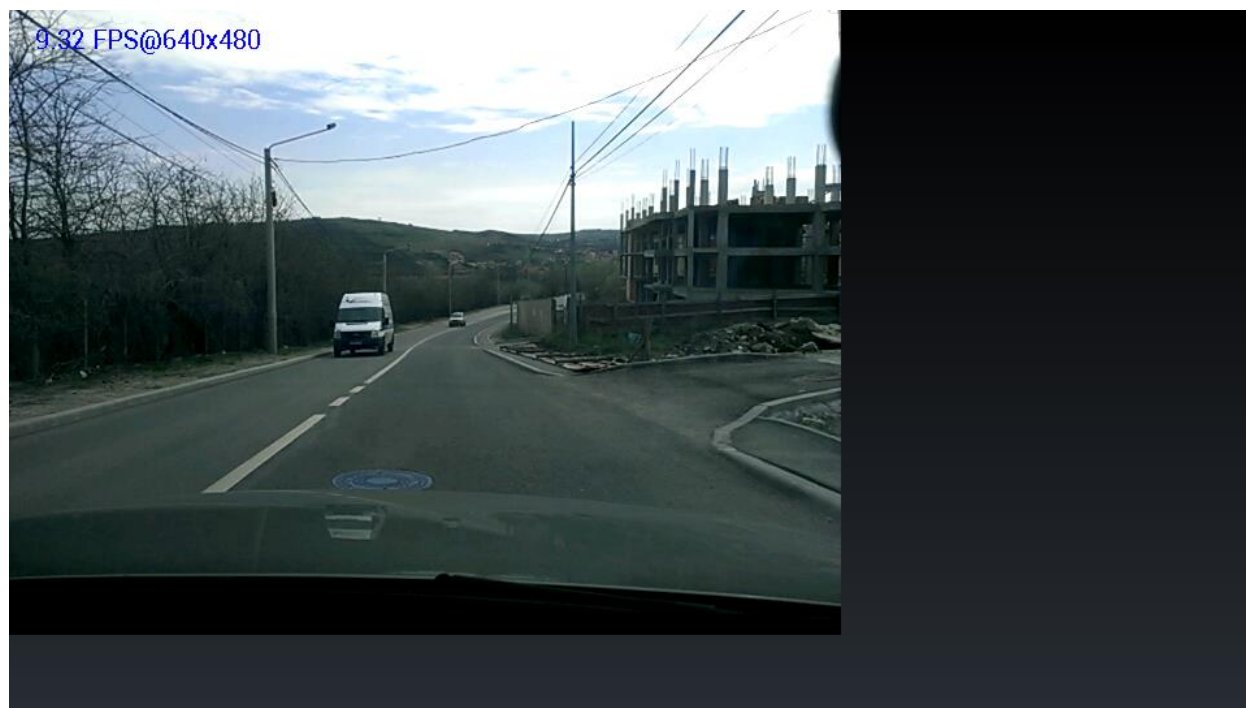


Figura 5.3 - Imagine din trafic capturată de camera video (în formatul RGB)

### 5.6. Segmentare imagine și eliminarea efectului de perspectivă

Pentru detecția posibilelor obstacole din fața mașinii, trebuie să eliminăm pixelii care aparțin drumului. Pentru a realiza acest lucru, se alege o zonă, dreptunghi din fața mașinii și se calculează culoarea medie din zona respectivă. Imaginea de intrare este convertită în format gri (Gray scale) folosind funcția din OpenCV: *cvtColor*.

Eliminarea pixelilor care aparțin drumului, se va face folosind o segmentare cu prag fix, pragul fiind culoarea calculată precedent, incrementat cu o valoare determinată din mai multe teste. Odată determinat pragul de segmentare, eliminarea pixelilor care aparțin suprafeței drumului se face prin parcurgerea imaginii, identificarea culorii fiecărui pixel și compararea cu pragul. Astfel, dacă pixelul curent are o culoare mai mică decât pragul, pixelul va fi setat pe culoarea 0 – alb (aparține intervalului de culoare specific drumului), iar dacă este peste acest prag, pixelul curent va lua valoarea 255 - negru, pixelul nu reprezintă drumul, ci este un candidat pentru un posibil obstacol. Pe imaginea rezultată după binarizare (segmentare) se elimină efectul de perspectivă (imaginea se proiectează “top-view” - inverse perspective mapping). Aceasta proiecție va facilita calcularea distanței până la obiectele detectate. Pentru îmbunătățirea timpilor de execuție, reproiectarea imaginii se face folosind tabele de look-up care conțin toate corespondențele dintre punctele imaginii sursă, respectiv puncte ale imaginii cu perspectiva eliminată. Schimbarea proiecției se face folosind funcția din OpenCV: *remap*, având ca parametrii cele două tabele cu coordonatele pre-calculate: „map\_x” și „map\_y”. Cunoscând aceste corespondențe dintre pozițiile fiecărui pixel din imaginea sursă și imaginea reproiectată, s-a redus considerabil timpul de execuție pentru fiecare imagine, deoarece căutarea într-un tabel este întotdeauna mai rapidă decât calcularea unei valori (care este constantă) în fiecare pas. Un exemplu de segmentare este ilustrat în figura următoare:



Figura 5.4 – În stânga este imaginea de intrare, în dreapta imaginea segmentată și cu efectul de perspectivă eliminat

### 5.7. Calculare histogramă polară

Algoritmul implementat folosește calcule pe baza histogramelor. Folosirea acestui tip de histogramă este justificată prin faptul că, în imaginea cu efectul de perspectivă eliminat, trăsăturile obstacolelor sunt dispuse radial, se alungesc spre exteriorul imaginii și astfel se urmărește determinarea intervalului angular corespunzător obstacolului.

Pentru a construi o histogramă polară, se începe prin alegerea unui punct de focus F. În cazul acesta punctul F este la coordonatele 368, 480 în centrul imaginii reproiectate și segmentate (Figura 5.5). Histograma polară este bazată pe sistemul de coordonate polar.

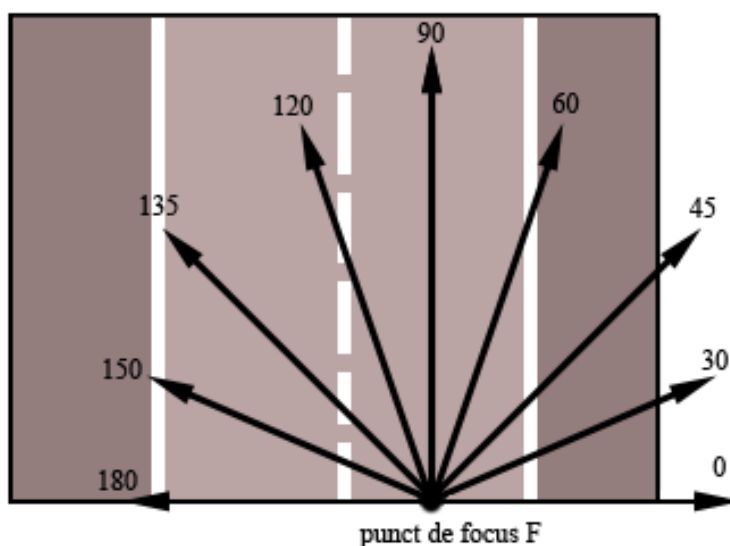


Figura 5.5 – Suprafața de drum cu punctul de focus și coordonate polare

În pasul următor se parcurge imaginea pe zona de interes, parcurgere care reduce timpul de procesare, și se calculează numărul de pixeli pentru fiecare linie dreaptă cu originea din punctul de focus. Pentru fiecare linie având originea din punctul F, se va calcula numărul de pixeli negri, care reprezintă posibile obstacole, astfel se obțin valorile pentru histograma polară care sunt salvate folosind o structură de date de tip vector.

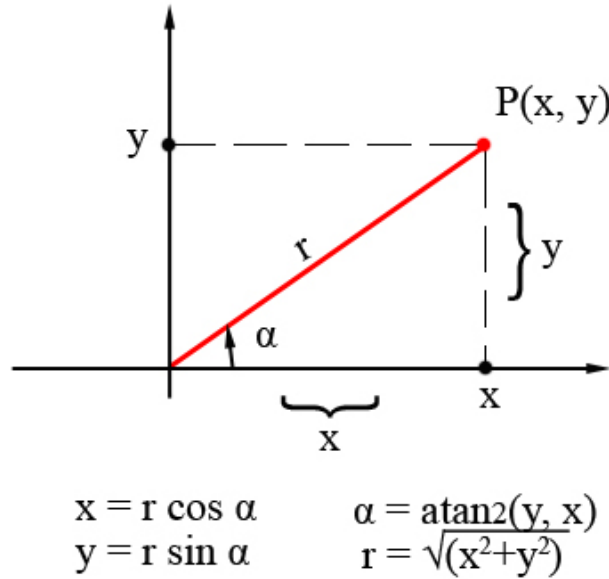


Figura 5.6 – Formule de conversie a coordonatelor din sistemul cartezian în sistemul de coordonate polar

Pentru crearea histogramei polare se folosește sistemul de coordonate polar. Folosind un sistem de coordonate polar, se facilitează și detecția unghiului la care este situat obstacolul în fața mașinii. Formulele de conversie din sistemul clasic folosit pentru procesarea de imagini (sistemul de coordonate cartezian), în sistemul de coordonate polar sunt ilustrate în figura 5.6. Astfel, pentru implementarea algoritmului s-au definit următoarele variabile:

- $\alpha_{\text{maxim}} = 180$  și reprezintă unghiul maxim (intervalul 0 – 180 grade)
- $\Delta\alpha$  – delta Alpha, utilizat pentru conversia din grade în radiani
- $\alpha_{\text{pixel}}$ , care reprezintă unghiul format de linia determinată de pixelul curent din imagine și punctul de origine F (punctul de focus)
- $\alpha_{\text{index}} = (\alpha_{\text{pixel}} / \Delta\alpha)$  – reprezintă unghiul pixelului exprimat în radiani

Mai jos este prezentată o secvență de pseudocod pentru implementarea histogramelor polare:

```

double  $\alpha_{\text{index}} = -1$ ,  $\alpha_{\text{pixel}} = -1$ ,  $\Delta\alpha = (3 * \pi) / 180$ 
for int j = 0; j < 480; j++
    for int i = 180; i < 450; i++
    {
        int color = getPixel.at( j , i )
        if ( color == 0 )
        {
            int xnew = i - 368
            int ynew = j - 480
             $\alpha_{\text{pixel}} = \text{atan2}(y_{\text{new}}, x_{\text{new}})$ 
             $\alpha_{\text{index}} = \text{round}(\alpha_{\text{pixel}} / \Delta\alpha)$ 
            if (  $\alpha_{\text{index}} > 0 \ \&\& \ \alpha_{\text{index}} < \alpha_{\text{maxim}}$  )

```

```

        }
        histograma[a index] ++
    }
}

```

În figura 5.7 este ilustrat un exemplu cu o imagine segmentată și cu efectul de perspectivă eliminat. Pe aceasta imagine s-a suprapus histograma polară folosind linii gri având originea în punctul de focus F.

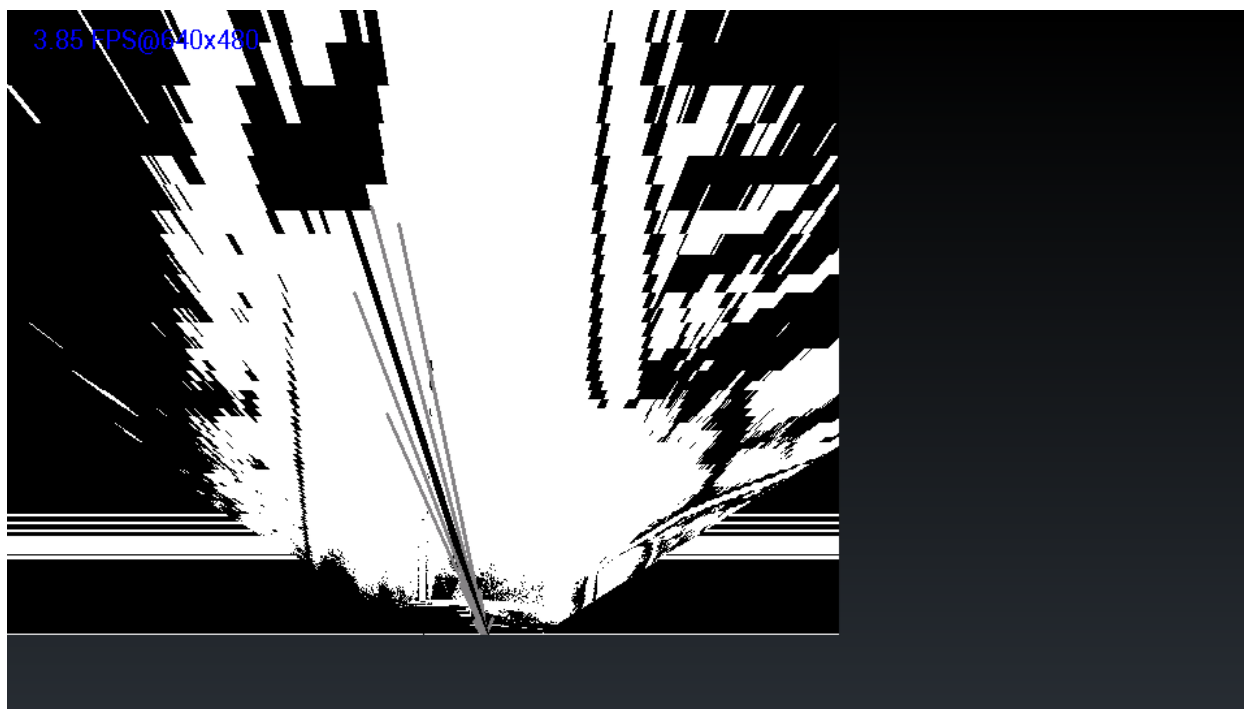


Figura 5.7 - Imagine cu perspectiva transformată și drumul segmentat; histograma polară este desenată cu linii gri, iar linia neagră reprezintă maximul local detectat.

### 5.8. Calculare maxim și normalizarea histogramei polare

În pasul următor, se normalizează histograma polară în funcție de valoarea maximă din histogramă. Pentru determinarea maximului din histogramă este necesară o parcurgere a structurii de date și salvarea valorii maxime întâlnite. Pe histograma polară, se va aplica un filtru de tipul low-pass pentru a reduce posibilele zgomote care pot să apară în imagini. Filtrul este aplicat folosind o parcurgere a histogramei și prin eliminarea unor valori asemănătoare (devin 0). Figura 5.7 reprezintă un exemplu în care este calculat și afișat maximul local împreună cu histograma polară. În exemplul dat, maximul local detectat este desenat cu o linie neagră, iar histograma cu linii gri, fiecare linie având originea în punctul de focus ales în prima etapă a algoritmului.



### 5.9. Calcularea maximelor locale

Se calculează maxime locale ale histogramei. Pentru estimarea distanței față de un obiect este necesară încadrarea fiecărui obstacol într-o anumită zonă. Maximele locale (vârfurile) din histogramă facilitează detecția acestor zone, pixeli care conțin fiecare obstacol detectat. După detecția vârfurilor din histogramă, se va scana fiecare zonă din histogramă pentru a determina obstacolele. Pentru fiecare vârf, maxim local al histogramei polare, se va calcula o histogramă radială definită în zona determinată de  $\alpha-1$  și  $\alpha+1$ , care reprezintă 50% din înălțimea vârfului (Figura 5.8). Odată determinate maximele locale, se folosesc două liste pentru a salva valorile fiecărui sector, zone: lista min, respectiv lista max. Aceste două liste vor salva informațiile necesare pentru liniile care încadrează un posibil obstacol detectat. În figura 5.9 este reprezentată o imagine care conține un obstacol detectat care este încadrat corect de cele două linii salvate în lista min și lista max. Cele două linii cu originea din punctul de focus F formează un triunghi, iar latura de sus a triunghiului fiind exact obstacolul identificat.

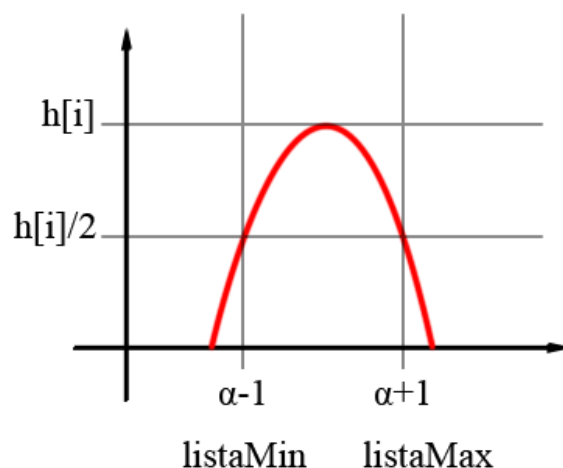


Figura 5.8 – Vârful unei histograme

### 5.10. Calculare histogramă radială

Se parcurge imaginea din nou, o parcurgere doar pe zona de interes, iar fiecare linie cu originea din punctul de focus F(368, 480) care se află într-un “triunghi”, într-o zonă încadrată de liniile salvate anterior în cele două liste (lista min și lista max), se va adăuga într-un vector de date (distHistNorm), iar dacă culoarea pixelului curent este 0 (reprezintă un posibil obstacol), se adaugă și în vectorul distHist. Acest tip de histogramă este calculat pentru fiecare obstacol și facilitează calcularea distanței până la obiecte.

În pasul următor, se normalizează histograma radială prin împărțirea valorilor corespundente din cei doi vectori de date ( $\text{distHist} = \text{distHist} / \text{distHistNorm}$ ). Partea finală a detecției de obstacole, anume calcularea distanței, este astfel redusă la o simplă verificare (en. „threshold”) a

valorilor din histograma radială în funcție de un anumit prag. Acest prag a fost determinat experimental după mai multe testări și verificări în diferite condiții. În figura 5.10 este prezentată o imagine care conține în partea de sus o histogramă radială a obstacolului detectat.

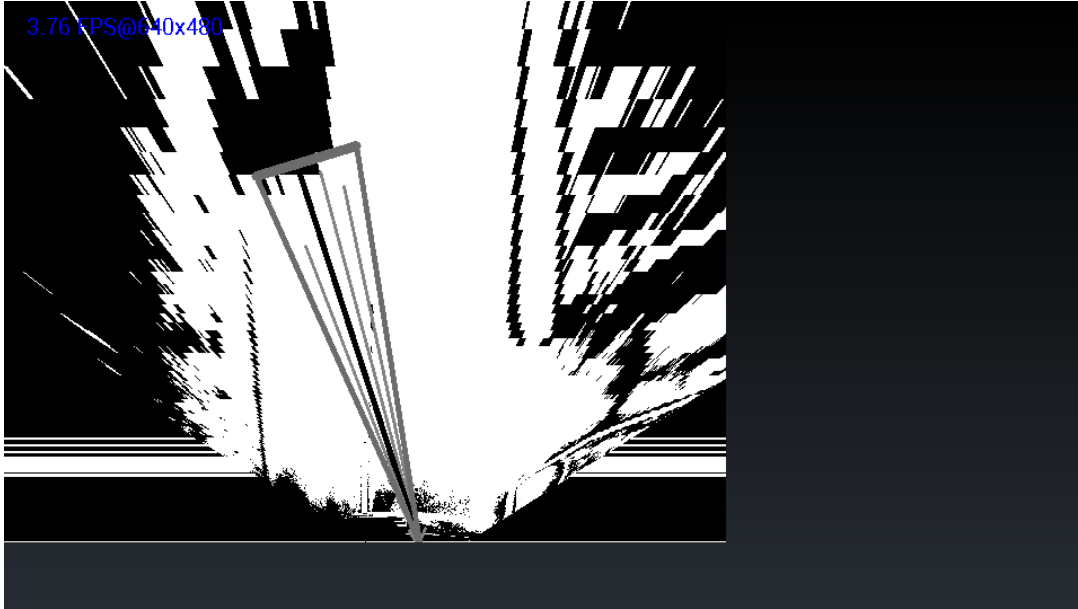


Figura 5.9 - Imaginea cu obstacolul detectat și delimitat corect. Latura de sus a triunghiului reprezintă obstacolul detectat.



Figura 5.10 - Imaginea cu obstacolul detectat și cu histograma radială afișată în partea de sus a imaginii.

Ultima etapă a algoritmului presupune desenarea unei linii în dreptul obstacolului identificat și afișarea distanței calculate (distanța este exprimată în metri). Figura 5.11 reprezintă rezultatul final al algoritmului de detecție a obstacolelor.



Figura 5.11 - Rezultatul final al procesării, pe imaginea de intrare s-a afișat obstacolul detectat, dar și distanța până la acesta, exprimată în metri.

### 5.11. Comunicarea cu alte dispozitive

Pentru posibile integrări cu alte sisteme sau dispozitive de asistare în traficul rutier, sistemul a fost extins prin implementarea unei comunicări prin standardul Bluetooth. Pentru testarea comunicării, s-a folosit o conexiune către un laptop de unde s-au citit informațiile folosind un terminal. Figura 5.12 reprezintă consola de terminal cu informațiile recepționate pe laptop, din aplicația de pe telefonul mobil. Pentru implementarea unei transmisii Bluetooth, Android oferă mai multe posibilități de implementare. Cea mai ușoară metodă este de a face conexiunea (en. Pairing) între cele două dispozitive și apoi crearea unei conexiuni de tip „socket” prin care se transmit datele. Un exemplu de transmisie prin socket-uri este prezentat mai jos:

```
socket = getBluetoothSocket();
try {
    socket.connect();
} catch (IOException e) {
    socket = null;
}
```

```

if(socket != null){
    try {
        DataOutputStream dos = new DataOutputStream(socket.getOutputStream());
        dos.writeUTF(status);
        socket.close();
        return true;
    } catch (IOException e) {
        socket = null;
        return false;
    }
}

```

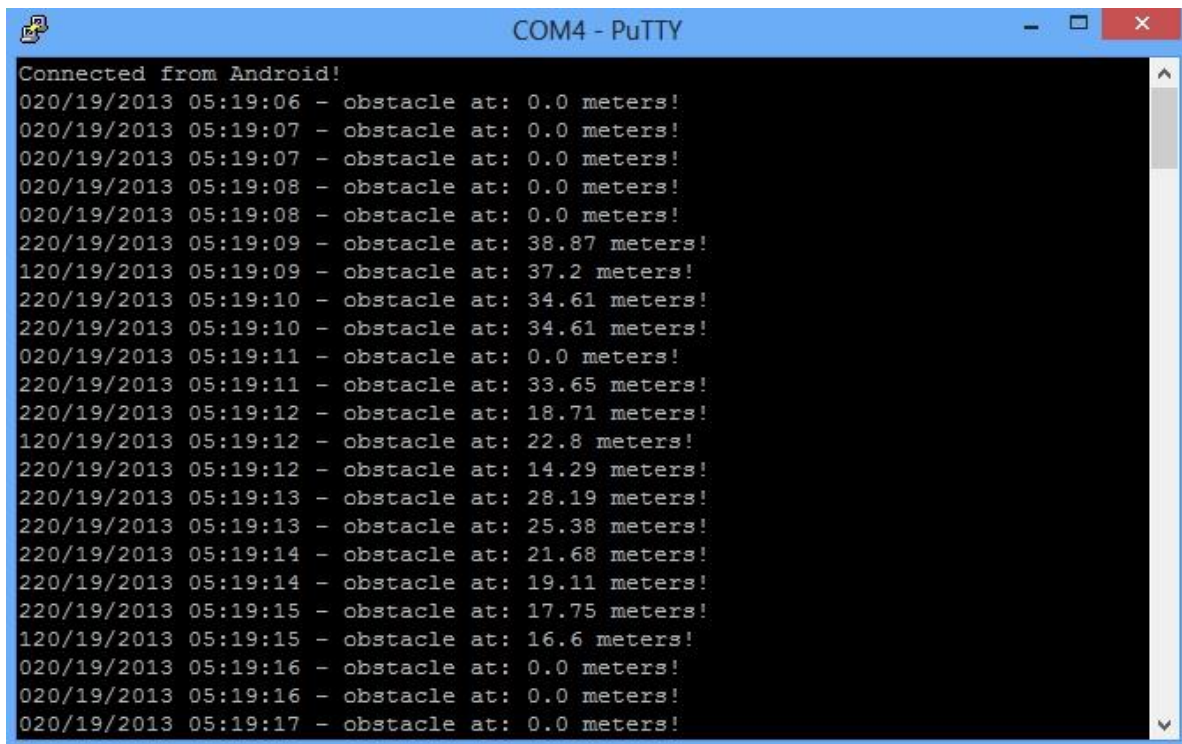


Figura 5.12 - Exemplu de transmisie informații prin intermediul Bluetooth. Informațiile citite din terminal de pe laptop.

### 5.12. Auto corecție imagini folosind accelerometrul

Pentru determinarea corespondențelor dintre punctele din lumea reală și pixelii din imaginile capturate, în primul pas am calibrat camera foto a telefonului. Am ales să fac o calibrare manuală prin calcularea unor anumite puncte relevante din imagini. Pe un drum am poziționat niște marcaje sub forma de “X”-uri (Figura 4.3) și am realizat un set de poze. Pozele au fost făcute din autovehicul, telefonul fiind așezat într-un suport de parbriz fix.

Pentru a elimina anumite erori de calcul și a preveni cazuri extreme, s-au făcut poze din diferite înclinații ale telefonului pe axa Z - Figura 5.13. Verificând pozițiile marcajelor din drum și calculând posibilele lor poziții din imaginile cu efectul de perspectivă eliminat, s-a obținut o relație

între cele două seturi de puncte. Pe baza acestor puncte se poate calcula o matrice de proiecție care va fi folosită pentru calcularea tuturor punctelor din imagine. Având  $n$  seturi de puncte diferite pentru mai multe înclinații pe axa Z a telefonului, s-au obținut  $n$  matrici de proiecție.

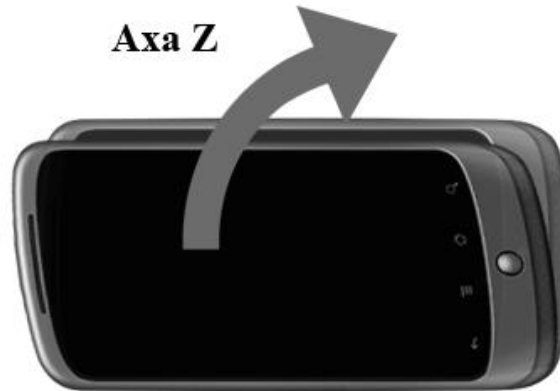


Figura 5.13 – Axa Z a înclinației telefonului mobil

În acest proiect am calculat un set de 5 matrici folosite pentru auto corecție. Intervalele relevante pentru unghiurile de înclinație pe axa Z sunt:  $[-6.0, -0.5]$ ,  $(-0.5, 0.0]$ ,  $(0.0, 1.6]$ ,  $(1.6, 2.6]$ ,  $(2.6, \infty)$ . Astfel, pentru fiecare din intervalele definite anterior, se vor utiliza diferite matrici. Figura 5.14 ilustrează mai bine ideile descrise mai sus.

Un exemplu de calculare al unei matrici folosite pentru procesul de autocorecție este descris mai jos:

```
// Z axis -6.0, -0.5
src1 = new Mat(4, 1, CvType.CV_32FC2);
dst1 = new Mat(4, 1, CvType.CV_32FC2);
src1.put(0, 0, 215, 286, 361, 286, 425, 339, 0, 339);
dst1.put(0, 0, 213, 180, 426, 180, 426, 401, 213, 401);
m1 = new Mat(3, 3, CvType.CV_32FC2);
m1 = Imgproc.getPerspectiveTransform(src1, dst1);
invM1 = new Mat(3, 3, CvType.CV_32FC2);
invM1 = Imgproc.getPerspectiveTransform(dst1, src1);
src1.release();
dst1.release();
```

În secvența de cod de mai sus este prezentat modul efectiv de calculare a matricii de transformare notată cu „m1” pentru cazul în care înclinația telefonului este cuprinsă în intervalul  $[-6.0, -0.5]$ . Matricile notate cu „src1” și „dst1” conțin cele două seturi de puncte corespondente din imagini.






Orientarea telefonului	Valoarea înclinației pentru axa Z	Matrice folosită
	$[-6.0, -0.5]$	[ m1 ]
	$(-0.5, 0.0]$	[ m2 ]
	$(0.0, 1.6]$	[ m ]
	$(1.6, 2.6]$	[ m3 ]
	$(2.6, \infty)$	[ m4 ]

Figura 5.14 – Valorile utilizate pentru procesul de auto corecție

Prin calcularea acestor matrici, s-a implementat o corecție a imaginii automată, bazată pe înclinația mașinii sau a telefonului. Modificarea înclinației este obținută folosind senzorul de accelerometru al telefonului mobil. Majoritatea telefoanelor mobile inteligente oferă un senzor de tip accelerometru pentru a roti automat imaginea de pe ecran. Un dezvoltator de aplicații pentru sistemul de operare Android, poate accesa și folosi valorile înregistrate de accelerometru prin intermediul clasei *SensorManager* care oferă un mecanism automat de tip *listener* pentru oferirea valorilor actualizate în timp real. Mai jos este prezentată o secvență de cod pentru citirea valorilor din senzorul de accelerare:

```
public void onSensorChanged(SensorEvent event) {
    if (event.sensor.getType() == Sensor.TYPE_ACCELEROMETER) {

        System.arraycopy(event.values, 0, m_lastAccels, 0, 3);    }
}
```

S-a optat pentru corecția imaginilor pentru a preveni situația în care un autovehicul trece peste o denivelare sau o groapă, iar având o matrice de proiecție fixă pentru fiecare situație, se obțineau erori de detecție a obstacolelor. Mișcarea mașinii provoacă o modificare a unghiului dintre cameră și drum, astfel încât matricea de proiecție calibrată în repaus nu este întotdeauna adecvată pentru generarea imaginii cu efectul de perspectivă eliminat, de aceea e nevoie de o ajustare dinamică.

### 5.13. Utilizarea modului GPS

În cadrul acestui proiect s-a utilizat modulul de gps intern al telefonului mobil pentru a afișa viteza curentă de deplasare a autovehiculului. Obținerea locației curente se face folosind serviciile de localizare din Android. Folosind obiecte de tip „listener”, kit-ul de dezvoltare software al sistemului de operare Android oferă în timp real informații despre locația curentă a terminalului mobil. Pentru determinarea vitezei de deplasare, mai întâi se verifică dacă telefonul se află în mișcare, iar apoi se apelează funcția *getSpeed*. Deoarece în Europa se folosește sistemul de măsurare metric, valoarea obținută trebuie convertită în kilometri pe oră.

```
LocationListener locationListener = new LocationListener() {  
    public void onLocationChanged(Location location) {  
        if(location.hasSpeed()) {  
            float sp = location.getSpeed();  
            if(sp != 0) {  
                speedTextView.setText((location.getSpeed()*3600/1000 + " km/h"));  
            } else {  
                speedTextView.setText("0 km/h");  
            }  
        } else {  
            speedTextView.setText("0 km/h");  
        }  
    }  
};
```

Afișarea informației despre viteză este disponibilă utilizatorului doar în cazul în care modulul gps este activat. Pentru partea de afișare se utilizează o componentă de interfață grafică de tipul *TextView*.



## 6. Testare și validare

### 6.1. Introducere

În cadrul acestui capitol sunt prezentate metodele de testare și de validare utilizate pentru acest proiect. S-a încercat obținerea unei produs software rapid fără a compromite precizia detecției. Prin testarea în mod continuu și succesivă a mai multor situații relevante din traficul rutier, am ajuns la o îmbunătățire a sistemului și în unele cazuri la reducerea timpilor de procesare.

Pentru testare și validare s-au urmărit în special performanța ridicată (frame rate cât mai mare) și acuratețea, precizia detecției.

### 6.2. Situații și scenarii pentru testare

Acuratețea detecției de obstacole se referă la identificarea corectă a obstacolelor în condiții de drum iluminat neuniform, de exemplu prezența umbrelor pe drum. De asemenea, s-a testat și pe timp de noapte în condiții de luminozitate redusă și s-au obținut rezultate bune (Figura 6.4). În majoritatea situațiilor s-a reușit obținerea unei rate de 8-10 cadre/secunda (frames per second).

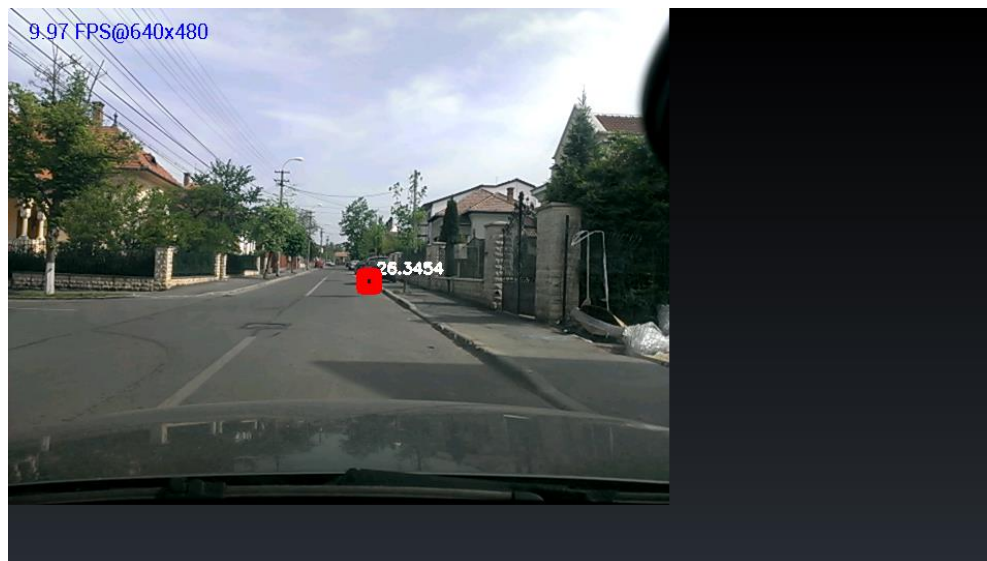


Figura 6.1 – Testare pe un drum iluminat neuniform (umbre ale copacilor sau caselor sunt vizibile pe șosea în fața vehiculului)



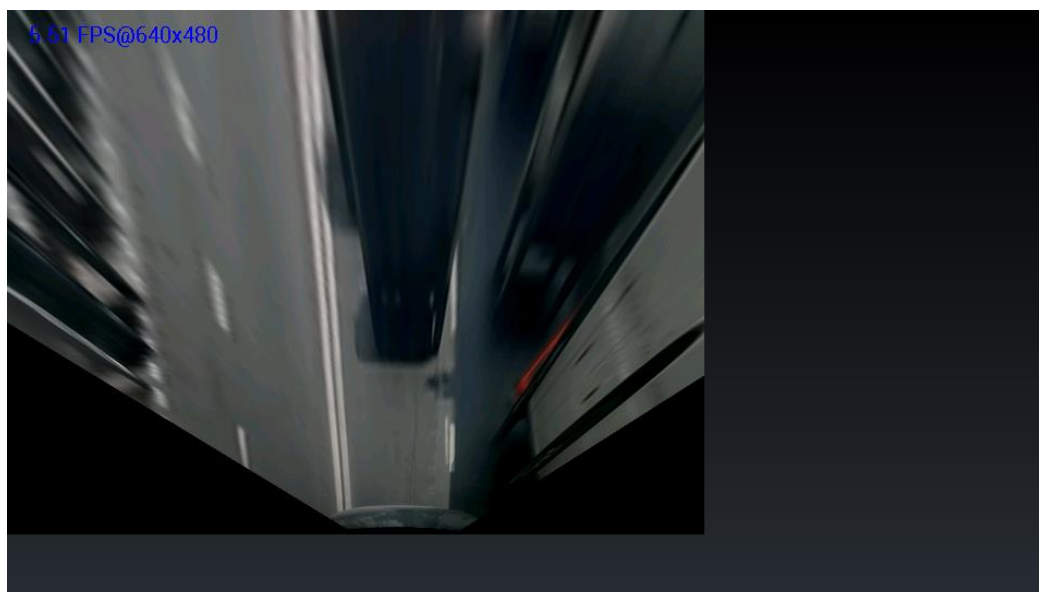


Figura 6.2 – Testare în trafic aglomerat, imagine cu efectul de perspectivă eliminat



Figura 6.3 – Testare în trafic aglomerat, imagine cu rezultatul final al procesării

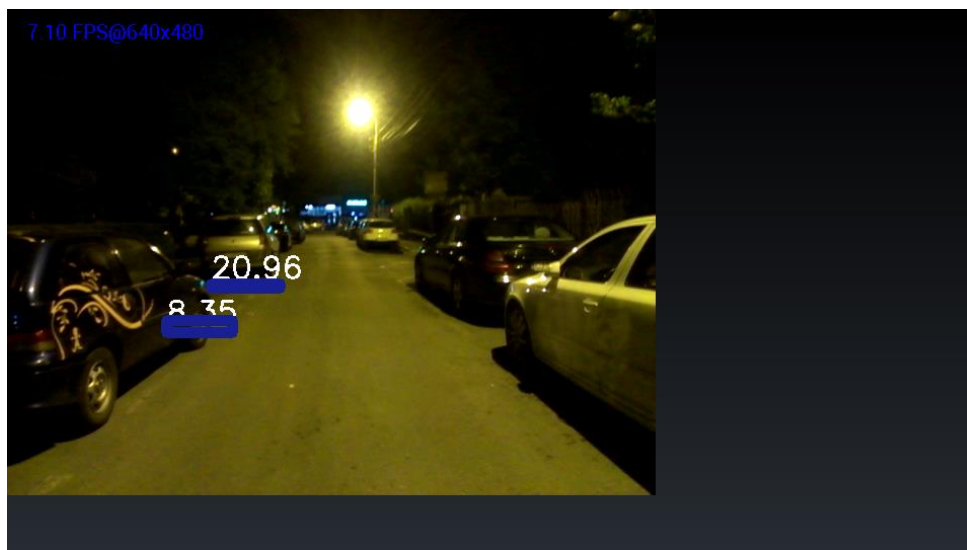


Figura 6.4 – Testare pe timp de noapte, cu luminozitatea redusă

Un alt aspect luat în considerare pentru testarea algoritmului este detecția pietonilor sau a bicicliștilor, respectiv a motocicliștilor. Deoarece pietonii sau bicicliștii reprezintă obstacole mai mici, s-a urmărit ca algoritmul să funcționeze corect și în aceste situații. În figura 6.5 este ilustrat rezultatul unei detecții de pieton, iar figura 6.6 reprezintă rezultatul detecției unui motociclist.



Figura 6.5 – Exemplu detecție pieton



Figura 6.6 – Exemplu detecție a unui motociclist

### 6.3. Performanța sistemului

Deși aplicația nu este dezvoltată pentru a folosi mai multe procesoare, s-a observat o diferență semnificativă între telefoane mobile cu sisteme single-core, dual-core, respectiv quad-core. Hardware-ul mai nou va facilita întotdeauna o execuție cursivă și fluidă a aplicațiilor care procesează imagini. Diferențele de calcul a fiecărei imagini se observă în graficele următoare.

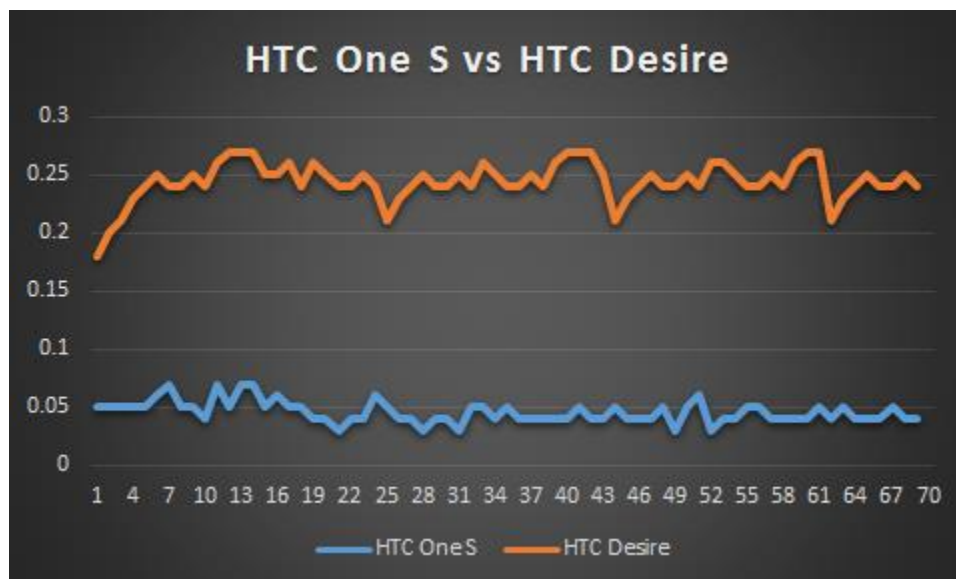


Figura 6.7 - Grafic comparativ între HTC One S (dual-core) și HTC Desire (single core) – axa X reprezintă index-ul imaginii procesate, iar axa Y reprezintă timpul de procesare necesar pentru imagine (timp exprimat în secunde)

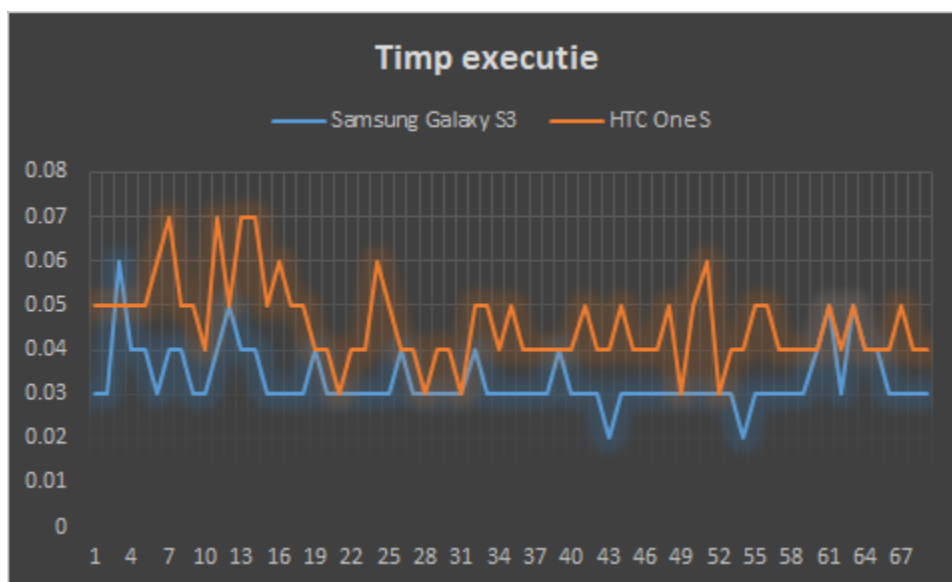


Figura 6.8 - Grafic comparativ între HTC One S (dual-core) și Samsung Galaxy S3 (quad core) – axa X reprezintă index-ul imaginii procesate, iar axa Y reprezintă timpul de procesare necesar pentru imaginea respectivă (timp exprimat în secunde)

Pentru obținerea graficelor de mai sus, s-au introdus niște instrucțiuni speciale în codul sursa al aplicației și s-a memorat timpul de procesare (care este exprimat în secunde) pentru fiecare imagine. Pentru comparația dintre cele trei telefoane mobile am utilizat același set de 70 de imagini.

Sistemul de detecție al obstacolelor are o rază de acțiune efectivă cuprinsă între 4 și 40 metri liniari, acest interval fiind ales pentru a reprezenta o acuratețe cât mai ridicată a distanței. Camera video a fost calibrată pentru a oferi valori precise pe această zonă de interes, care are o lățime de aproximativ 13 metri (echivalentul a două benzi standard de circulație). Figura următoare ilustrează raza de acțiune a sistemului precum și zona de interes.

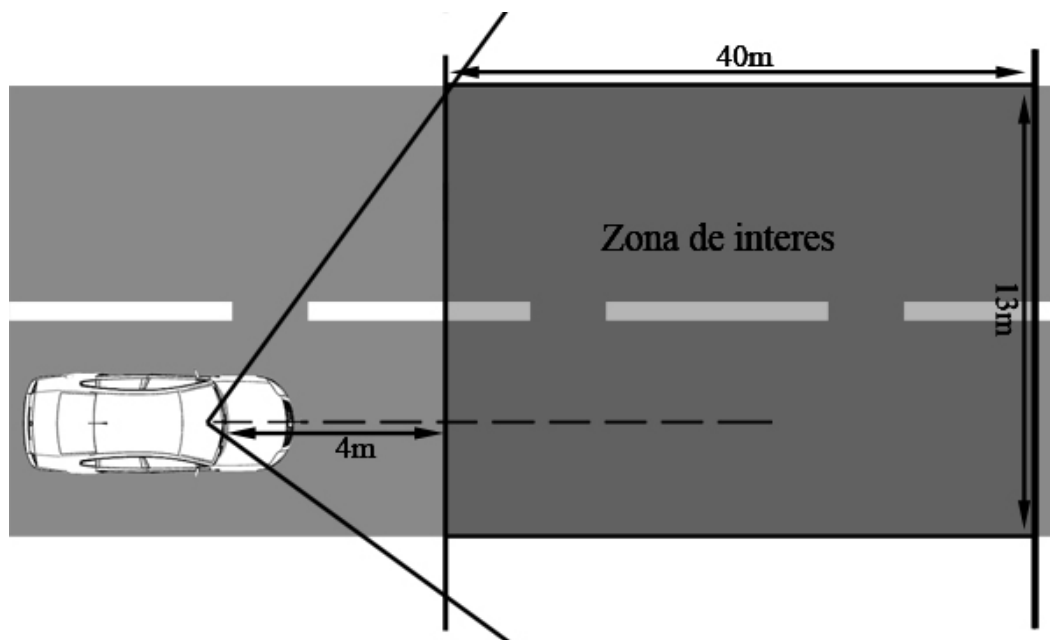


Figura 6.9 – Zona de interes și raza de acțiune a sistemului

#### 6.4. Comparație cu produse similare

Pentru a avea o referință asupra performanței sistemului și pentru verificarea detecției în situații de trafic real, am realizat o comparație cu alte produse similare disponibile pentru platforma Android. Comparația am realizat-o pe un drum cu trafic redus, autovehiculul fiind în staționare, pentru a avea situații asemănătoare în momentul testării fiecărei aplicații. Cele 2 aplicații la care m-am raportat sunt iOnRoad și Drivea.

Rezultatele acestei comparații se pot observa în figurile care urmează. În prima imagine (Figura 6.10) este prezentat un screenshot al aplicației [19], în care se observă faptul că nu s-a identificat autovehiculul din fața mașinii. Figura 6.11 ilustrează o situație asemănătoare, diferența fiind dată de amplasarea obstacolului pe contrasens. Nici în această situație aplicația iOnRoad nu a afișat nimic.



Figura 6.10 – Aplicația iOnRoad – primul scenariu de test



Figura 6.11 – Aplicația iOnRoad – scenariul de test 2

În următoarele două figuri este prezentat rezultatul obținut de aplicația Drivea [24]. În prima situație (figura 6.12), obstacolul este în fața ego-vehiculului și este identificat și încadrat corect, dar la un moment foarte întârziat când obstacolul este deja la o distanță mare de ego-vehicul. A doua imagine (figura 6.13) reprezintă situația în care un obstacol se deplasează pe sensul opus înspre ego-vehicul. La fel ca în cazul aplicației iOnRoad, obstacolul nu este detectat.





Figura 6.12 – Aplicația Drivea scenariul de test 1



Figura 6.13 – Aplicația Drivea, scenariul al doilea de testare

Ultimele două seturi de imagini (figura 6.14 și 6.15), reprezintă aceleași două situații de test luate în calcul pentru comparația cu produsele concurente. În ambele scenarii, sistemul dezvoltat în cadrul acestei lucrări de licență a detectat obstacolele corespunzător.



Figura 6.14 – Sistemul implementat si descris în această lucrare, scenariul de test 1



Figura 6.15 – Aplicația dezvoltată în cadrul acestei lucrări de licență – detecție obstacol de pe banda opusă



## 7. Manual de instalare și de utilizare

Acest capitol va cuprinde pașii necesari instalării și utilizării aplicației dezvoltate în cadrul acestui proiect. Prima parte a capitolului descrie etapele necesare instalării și configurării mediului de dezvoltare și al aplicațiilor utilizate în cadrul proiectului. A doua parte conține manualul de utilizare al aplicației de pe telefonul mobil.

### 7.1. Configurare și instalare mediu de dezvoltare

Înainte de executarea aplicației, trebuie configurate și instalate uneltele necesare dezvoltării de aplicații Android. Asemănător cu alte sisteme de operare pentru telefoane mobile, aplicațiile și programele Android sunt dezvoltate pe un computer („host machine”) și după aceea transferate pe terminalul mobil („target machine”).

Pentru a începe dezvoltarea de aplicații mobile, sunt necesare următoarele unelte:

- Java SDK
- Mediul de dezvoltare Eclipse
- Plugin-ul Android pentru Eclipse: ADT (Android Developer Tools)
- Uneltele Android: Android SDK

Echipa din spatele sistemului de operare Android oferă o arhivă de tipul „zip” cu mediul de dezvoltare Eclipse preconfigurat pentru dezvoltarea de aplicații mobile. Această arhivă conține și plugin-ul ADT (Android Developer Tools) care este gata configurat împreună cu Eclipse. Pentru a începe dezvoltarea de aplicații este necesară arhiva cu toate programele, disponibilă la [22]. Depinzând de sistemul de operare al computerului, utilizatorul acestei aplicații va avea nevoie și de pachetul Java SDK instalat.

După descărcarea Android SDK, este necesar să dezarhivăm pachetul, și după aceea aplicația poate fi importată și executată. Deoarece aplicația dezvoltată în cadrul acestei lucrări de licență folosește un algoritm scris în limbajul de programare C++, este necesar să instalăm și pachetul Android NDK (Native Development Kit). Acest pachet de aplicații suplimentare este disponibil la [23]. Odată instalat și pachetul NDK, aplicația poate fi compilată în Eclipse.

Pentru testarea aplicației este necesar un terminal mobil cu sistem de operare Android 2.2 sau mai mare. Aplicația se poate testa și prin intermediul emulatorului Android care este integrat în pachetul SDK. Deoarece proiectul dezvoltat în cadrul acestei lucrări de licență folosește camera video, nu se poate verifica aceasta funcționalitate folosind emulatorul Android.

Importarea proiectului în mediul Eclipse se face prin apăsarea butonului „File > Import” și după aceea se alege opțiunea „Existing Android Code Into Workspace”.

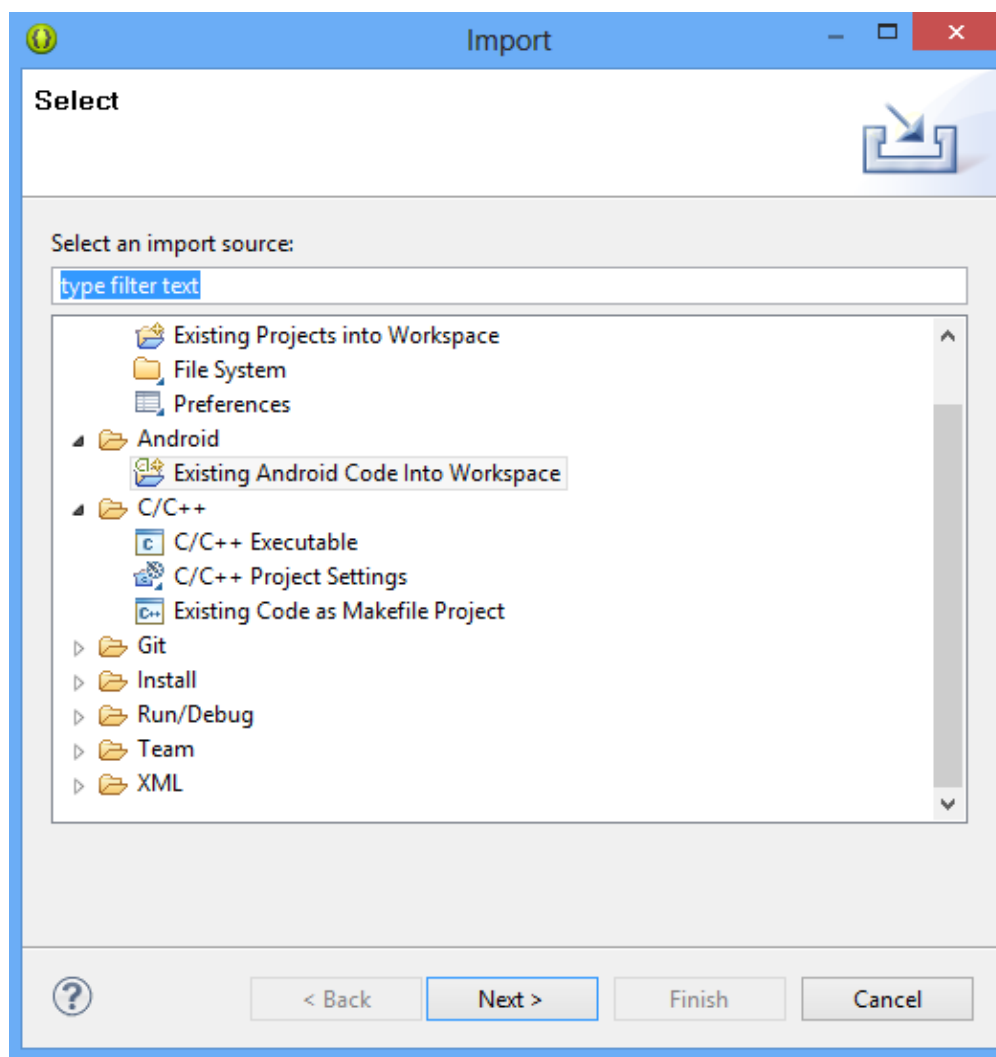


Figura 7.1 – Importarea proiectului în Eclipse

## 7.2. Utilizarea aplicației – manual de utilizator

Instalarea aplicației pe telefonul mobil poate fi făcută în două moduri. Cea mai simplă variantă este conectarea telefonului la un port USB și după aceea se rulează aplicația din mediul Eclipse. Acest lucru se face prin click dreapta pe proiectul importat și apăsarea butonului „Run as > Android application”. Prin aceasta metodă, aplicația este instalată automat pe telefonul mobil și după aceea va fi executată.

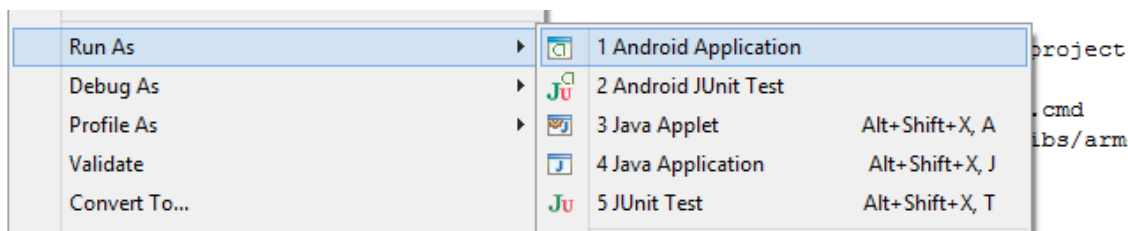


Figura 7.2 – Instalare și executare automată a aplicației folosind Eclipse

O altă variantă de instalare a aplicației este prin exportarea proiectului din Eclipse sub formă de pachet „apk”, care este echivalentul aplicațiilor executabile „.exe” din Windows. Așadar, proiectul salvat sub forma de executabil va trebui copiat pe memoria internă a telefonului și după aceea instalat.

La deschiderea aplicației, utilizatorul are la dispoziție o interfață grafică simplă care conține trei butoane:

- Start – pentru pornirea vizualizării din camera video
- Bluetooth – pentru activarea conexiunii de tip Bluetooth
- GPS – buton folosit pentru activarea modulului de GPS

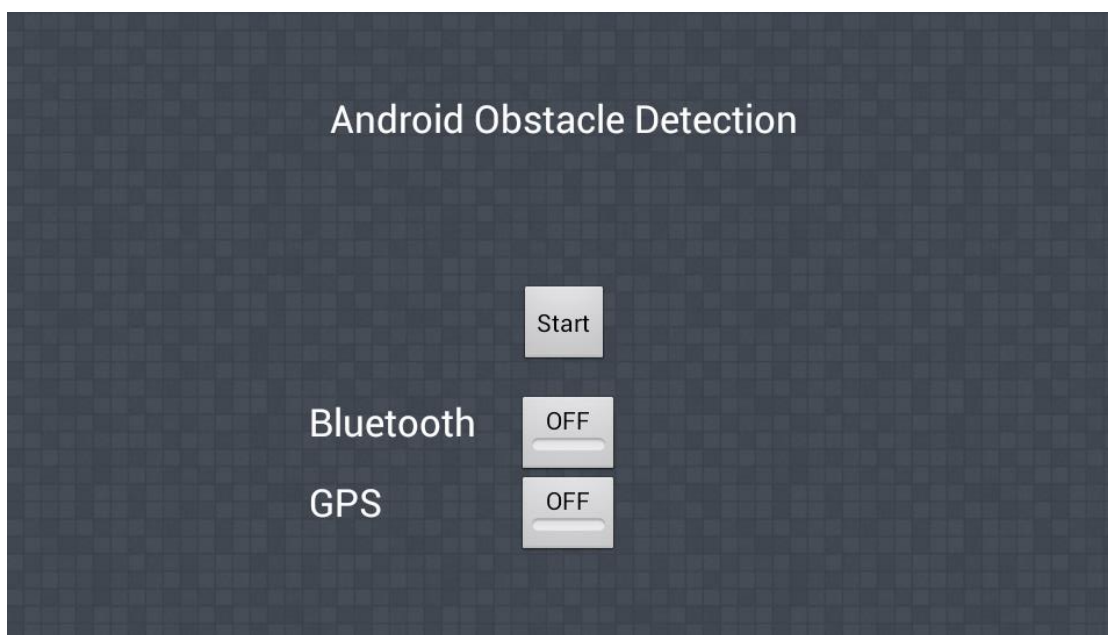


Figura 7.3 – Ecranul inițial al aplicației

Prin apăsarea butonului de start, se pornește o nouă activitate Android care conține ecranul principal al aplicației. În partea centrală a acestei interfețe se află imaginile surprinse de camera video. Obstacolele detectate vor fi indicate printr-o linie care va urmări obstacolul. Deasupra liniei va fi scrisă și informația despre distanța până la acest obiect.

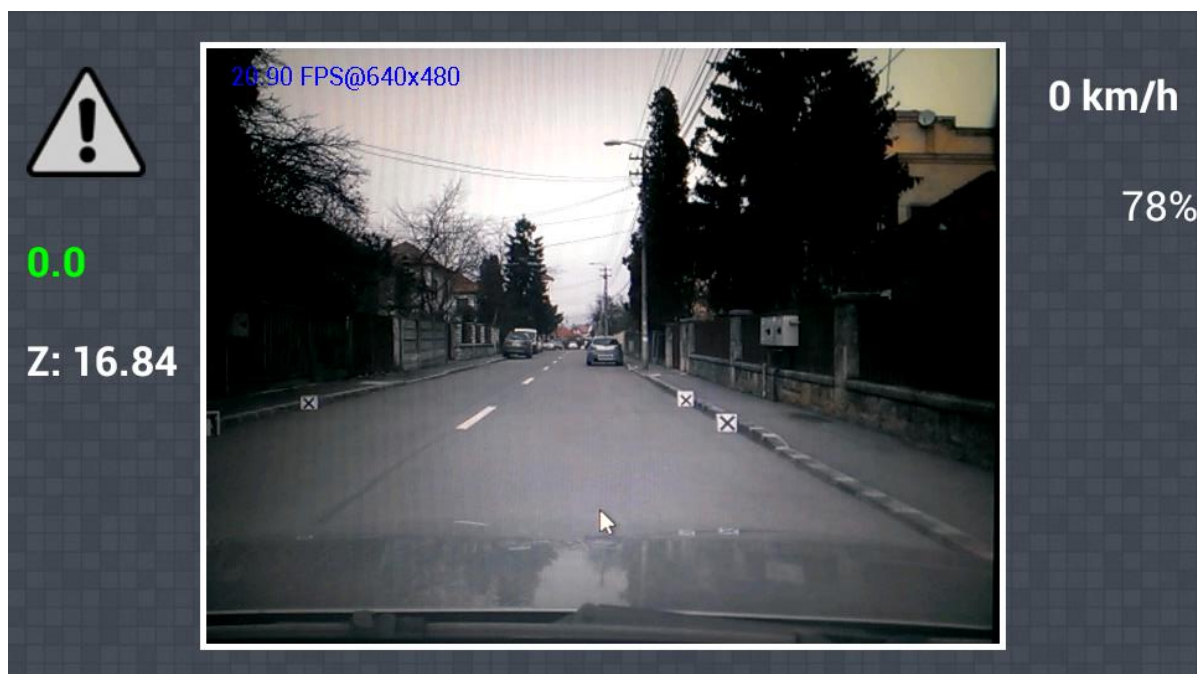


Figura 7.4 – Ecranul principal al aplicației (în centru imaginea din camera video)

La activarea modulului Bluetooth, utilizatorul va fi alertat printr-un dialog de confirmare, ilustrat în figura 7.5. Dacă se apasă pe „allow”, conexiunea Bluetooth la adresa mac a laptopului este inițiată.

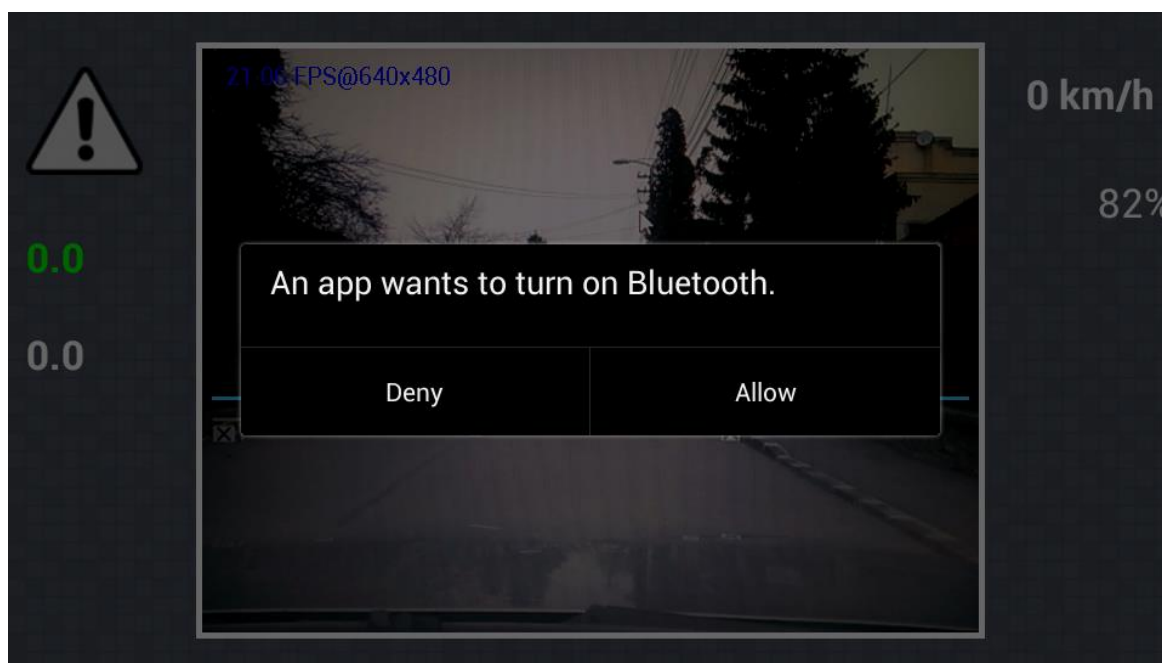


Figura 7.5 – Alertă la activarea transmisiei prin Bluetooth

La apăsarea butonului GPS, la pornirea aplicației, utilizatorul este trimis în meniul de activare a poziționării folosind sateliți gps. După apăsarea butonului „GPS satellites”, aplicația va reveni la ecranul principal, iar în momentul deplasării viteza va fi actualizată și afișată în partea de dreapta sus a interfeței grafice.

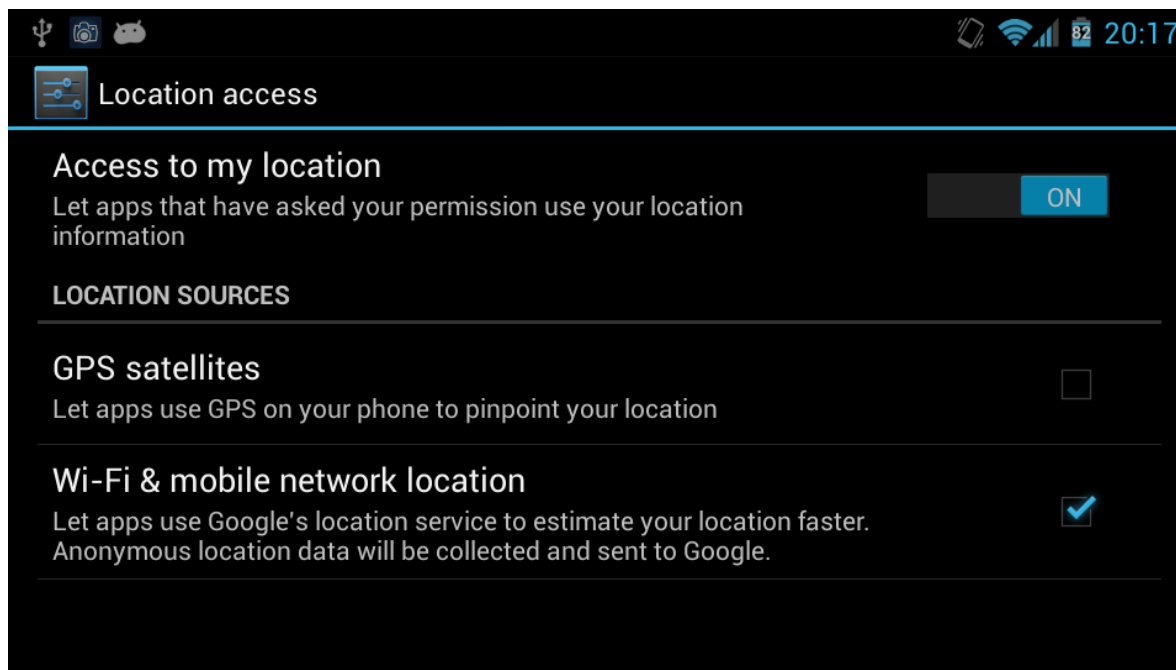


Figura 7.6 – Meniul de activare al poziționării folosind sateliți GPS

În cazul în care un obstacol detectat se afla în fața vehiculului, iar distanța este prea mică sau scade sub un anumit prag, utilizatorul va fi alertat atât vizual, cât și sonor. Pentru alerta vizuală am folosit un triunghi care va deveni colorat în momentul unui pericol, iar alerta sonoră folosește sunetul standard de alarmă al telefonului mobil utilizat. Partea dreaptă a interfeței grafice prezintă informații despre viteza de deplasare (doar în cazul în care modulul GPS este activat) și informații despre nivelul de baterie al telefonului, nivel exprimat folosind procente (100% va reprezenta o baterie încărcată la nivelul maxim). Figura 7.4 ilustrează informațiile descrise mai sus, cu alte cuvinte ecranul principal al aplicației mobile.

Pentru utilizarea acestui sistem, utilizatorii vor avea nevoie și de un suport de parbriz pentru mașină. Telefonul trebuie amplasat într-o poziție cât mai fixă în parbriz pentru a captura imagini din traficul rutier.

## 8. Concluzii și dezvoltări ulterioare

Proiectul dezvoltat în cadrul acestei lucrări de licență prezintă o soluție pentru detecția obstacolelor în traficul rutier. Rezultatul obținut este eficient din punct de vedere computațional.

Acest proiect a fost implementat pe telefoane mobile inteligente cu sistem de operare Android, făcând acest sistem accesibil ca și costuri dar și foarte portabil. Terminalele mobile au susținut o continuă dezvoltare din punct de vedere al puterii de calcul și de aceea au reprezentat un candidat ideal pentru acest tip de proiect.

Metoda de detecție a fost testată intens în mai multe situații de trafic și s-a dovedit a fi o variantă robustă. Rezultatele experimentale indică faptul că proiectul de față implementează cu succes o detecție în timp real a obstacolelor. Principalele etape necesare implementării sunt: achiziția imaginilor, segmentarea drumului și eliminarea efectului de perspectivă, calcularea histogramelor polare, calcularea histogramelor radiale și determinarea distanței. Achiziția imaginilor se referă la capturarea informațiilor din trafic folosind camera video a terminalului mobil. A doua etapă, segmentarea drumului și eliminarea efectului de perspectiva presupune eliminarea pixelilor care aparțin suprafeței drumului și apoi transformarea imaginii într-o nouă proiecție. Rezultatul acestei etape este o imagine binară (alb – negru) a drumului privit de sus, în care drumul va fi indicat de culoarea albă, iar obstacolele sunt marcate cu culoarea neagră. Din această imagine, se remarcă faptul că trăsăturile obstacolelor sunt dispuse radial, alungindu-se spre exteriorul imaginii. Astfel, se urmărește identificarea intervalului angular corespunzător obstacolului folosind histograma polară și calcularea distanței până la obstacole, folosind histograma radială. În următoarea etapă, am calculat o histogramă polară a imaginii rezultate în urma pasului precedent. Histogramele polare sunt calculate prin proiectarea unor linii imaginare dintr-un punct de focus (origine) și prin numărarea pixelilor de o anumită culoare, în cazul acesta am memorat numărul de pixeli negri. În această etapă am calculat și maximele locale ale acestei histograme. Aceste maxime sunt necesare pentru a determina obstacolele și pentru a le încadra corespunzător în anumite zone. În următorul pas, am calculat histogramele radiale pentru fiecare obiect prezent în imagine. Calcularea acestei histograme se face în mod analog cu histograma polară, diferența fiind în cazul histogramei radiale se folosesc informațiile obținute din listele care conțin maximele locale, pentru a calcula distanțele până la obstacole. Așadar, o histogramă radială se calculează pentru un posibil obstacol detectat pentru determinarea distanței, în timp ce histograma polară este determinată pentru toată imaginea pentru identificarea și încadrarea obstacolelor. În ultima fază a algoritmului se calculează distanța până la obiect folosind o corespondență între pixelii din imagine și unitatea de 1 metru din lumea reală. Această corespondență a fost calculată în momentul calibrării camerei video a telefonului mobil. Pentru a obține timpi de execuție mai reduși și pentru a realiza un sistem în timp real, am folosit structuri de date de forma unor tabele pentru stocarea informațiilor care sunt constante și necesare algoritmului pentru fiecare imagine în parte. Utilizarea acestor tabele (en. „lookup tables”) a dus la o îmbunătățire semnificativă a numărului de cadre pe secunda procesate. Totodată, am folosit și parcurgeri ale imaginilor doar pe anumite zone de interes. Efectiv, am procesat doar zonele din imagini care reprezintă suprafața de drum.

Pe lângă implementarea algoritmului propriu zis, a fost necesar să calibrez camera video a telefonului mobil. Calibrarea am realizat-o folosind informațiile existente în literatura de specialitate și cu ajutorul unor unelte de calibrare disponibile gratis pe internet. În urma acestei etape am obținut parametrii specifici ai camerei video: parametrii intrinseci și parametrii extrinseci. Aceste informații sunt vitale pentru procesul de eliminare al efectului de perspectivă din imagini.

Platformele și uneltele de programare (software) folosite în cadrul acestei lucrări de licență sunt de tipul open source. Atât sistemul de operare folosit (Android), cât și librăriile (OpenCV) și mediile de dezvoltare (Eclipse IDE) sunt disponibile fără plată și oferă o documentație bogată pentru folosirea lor.

Deși algoritmul implementat s-a dovedit a fi foarte eficient, există o serie de îmbunătățiri care pot fi aduse pe viitor. Principala îmbunătățire care poate fi adusă este o variantă mai complexă a algoritmului de segmentare a drumului. Deși în majoritatea situațiilor drumul este eliminat complet din imaginile sursă, în unele cazuri excepționale, datorită prezenței umbrelor pe asfalt, pixelii care aparțin acestor suprafețe de intensitate diferită nu sunt eliminați și algoritmul uneori identifică umbrele de pe drum ca fiind posibile obstacole. O altă optimizare care poate fi adusă este modificarea algoritmului curent pentru a permite funcționarea pe mai multe procesoare (multi-thread). Pe partea de extinderi posibile, se poate implementa și o detecție a benzilor de circulație, deoarece imaginea cu perspectiva eliminată permite acest lucru.

Pe partea de interfață de utilizator se poate extinde sistemul și se poate modifica pentru a fi mai ușor de folosit și pentru a afișa mai lizibil informațiile despre obstacolele identificate.

## 9. Bibliografie

- [1] – National Highway Traffic Safety Administration, "Early Estimate of Motor Vehicle Traffic Fatalities," U.S. Department of Transportation, Washington, 2012
- [2] – Politia Română, statistici accidente rutiere, accesibile online la pagina:  
[http://www.politiaromana.ro/dpr/dinamica\\_accidentelor\\_circulatie.htm](http://www.politiaromana.ro/dpr/dinamica_accidentelor_circulatie.htm)
- [3] - M. Bertozzi and A. Broggi, "GOLD: a Parallel Real-Time Stereo Vision System for Generic Obstacle and Lane Detection", IEEE Trans. on Image Processing, paginile 62-81, 1998.
- [4] – S. Tuohy, "Distance determination for an automobile environment using Inverse Perspective Mapping în OpenCV", Signals and Systems Conference, 2010.
- [5] - I. Ulrich, I. Nourbakhsh, "Appearance-Based Obstacle Detection with Monocular Color Vision", Proc. of the AAAI National Conference on Artificial Intelligence, Austin, TX, 2000.
- [6] - T. Kalinke, C. Tzomakas, W. von Seelen, "A Texture based Object Detection and an Adaptive Model-based Classification", în Procs. IEEE Intelligent Vehicles Symposium'98, Stuttgart, Germany), paginile 341–346, 1998.
- [7] - A. Kuehnle, "Symmetry-based vehicle location for AHS", Procs. SPIE Transportation Sensors and Controls: Collision Avoidance, Traffic Management, și ITS, vol. 2902, (Orlando, FL), paginile 19–27, 1998.
- [8] - G. Marola, "Using Symmetry for Detecting and Locating Objects în a Picture", Computer Vision Graphics and Image Processing, vol. 46, paginile 179–195, 1989.
- [9] - M. Bertozzi, A. Broggi, A. Fascioli, and S. Nichele, "Stereo Vision-based Vehicle Detection", Procs. IEEE Intelligent Vehicles Symposium 2000, paginile 39-44, Detroit, USA, 2000.
- [10] - A. Broggi, "Obstacle Detection with Stereo Vision for Off-Road Vehicle Navigation", Computer Vision and Pattern Recognition, 2005.
- [11] - Gary Bradski, Adrian Kaehler, "Learning OpenCV: Computer Vision with the OpenCV Library", O'Reilly, 2008.
- [12] – Reto Meier, "Professional Android 4 Application Development", Wrox, 2012.
- [13] – J. Weng, "Camera calibration with distortion models and accuracy evaluation", Pattern Analysis and Machine Intelligence, vol. 14, paginile 965-980, Oct. 1992
- [14] – Universitatea Tehnică din Cluj-Napoca, Cursuri Procesarea Imaginilor, accesibile la adresa: [http://users.utcluj.ro/~rdanescu/teaching\\_pi.html](http://users.utcluj.ro/~rdanescu/teaching_pi.html)
- [15] – Calibrarea Camerei cu OpenCV, articol accesibil online la pagina:  
[http://docs.opencv.org/trunk/doc/tutorials/calib3d/camera\\_calibration/camera\\_calibration.html](http://docs.opencv.org/trunk/doc/tutorials/calib3d/camera_calibration/camera_calibration.html)



[16] – G. Trinidad-Garcia, “Embedded system applied to obstacles detection în a mobile controlled by a PC via RF”, Electrical Communications and Computers, 27-29 Feb 2012

[17] – U. Franke, A. Wedel, “Realtime Depth Estimation and Obstacle Detection from Monocular Video”, 28th DAGM Symposium, Berlin, Germany, September 12-14, 2006. Proceedings, pp 475-484. 2006

[18] - H.H.Little, J.J. Bohrer S. Mallot, H.A. Bulthoff, “Inverse perspective mapping simplifies optical flow computations and obstacle detection”, Biological Cybernetics, volumul 64, paginile 177–185, 1991.

[19] – Aplicația iOnRoad, disponibilă online la adresa:  
<https://play.google.com/store/apps/details?id=com.picitup.iOnRoad&hl=en>

[20] – Aplicația Mobileye, disponibilă online la adresa:  
<http://www.mobileye.com/products/mobileye-5-series/mobilapp/>

[21] – Aplicația Augmented Driving, disponibilă online la adresa:  
<http://www.imaginyze.com/Site/Welcome.html>

[22] – Unelte de dezvoltare aplicații Android, disponibile la adresa:  
<http://developer.android.com/sdk/index.html>

[23] – Kitul de dezvoltare aplicații folosind Android NDK, accesibil la pagina:  
<http://developer.android.com/tools/sdk/ndk/index.html>

[24] – Aplicația Drivea, disponibilă online la adresa:  
<https://play.google.com/store/apps/details?id=com.driveassist.experimental&hl=en>