



Universitatea
Transilvania
din Brașov
FACULTATEA DE MATEMATICĂ
ȘI INFORMATICĂ

Programul de studii:
Informatică Aplicată

Lucrare de licență
Sistem integrat de monitorizare și gestionare a unei parări auto
cu platformă web interactivă

Absolvent: Doloiu Mihai Alexandru
Coordonator științific: Lect. Dr. Vlad Monescu

Brasov, 2024

Cuprins

1	Introducere	5
1.1	Scopul și obiectivele lucrării	5
1.2	Motivația alegerii temei	6
1.3	Structura lucrării	6
2	Medii si concepte de programare	8
2.1	Limbajul de programare C++	8
2.2	Limbajul de programare Java	8
2.3	Limbajul SQL	9
2.4	Rețele neurale artificiale	10
2.5	Rețele neurale convoluționale	10
2.6	Biblioteca OpenCV	11
2.7	Biblioteca YOLOv5	11
2.8	Framework-ul Qt	12
2.9	Framework-ul Spring	13
2.10	Platforma CMake	13
3	Detecția plăcuțelor de înmatriculare	15
3.1	Preprocesarea imaginilor	15
3.1.1	Redimensionarea imaginii	15
3.1.2	Conversia imaginii în grayscale	15
3.1.3	Reducerea zgomotului	17
3.1.4	Operația morfologică de Opening și Thresholding	20
3.2	Extragerea conturilor	22
3.3	Extragerea plăcuței de înmatriculare	24
3.4	Metodă alternativă folosind rețele neurale convoluționale	24
3.4.1	Setul de date utilizat	25
3.4.2	Antrenarea modelului	26
3.4.3	Inferența în C++	27
3.5	Concluzie	28
4	Recunoașterea caracterelor din plăcuța de înmatriculare	29
4.1	Preprocesarea imaginii	29
4.1.1	Egalizarea adaptivă a histogramei	30
4.1.2	Undistortion	31
4.1.3	Skew Correction	34
4.1.4	Rezultatele celor două metode de corectare a distorsiunii	35
4.2	Detectarea caracterelor	36
4.3	Template Matching	37
5	Modelarea entităților	40
5.1	Entitatea "Camera"	40
5.2	Entitatea "CameraType"	41

5.3	Entitatea "CameraKey"	41
5.4	Entitatea "Session"	42
5.5	Entitatea "ParkingSpace"	42
5.6	Entitatea "ParkingSession"	43
5.7	Entitatea "Users"	43
5.8	Entitatea "Reports"	44
6	Prezentarea aplicațiilor	45
6.1	Aplicația C++	45
6.1.1	Arhitectura MVC	45
6.1.2	Arhitectura pe trei nivele pentru baza de date	46
6.1.3	Realizarea interfeței grafice	46
6.1.4	Camerele Video	48
6.1.5	Pagina principală	50
6.1.6	Meniul "Manage Cameras"	50
6.1.7	Pagina "Camera Management"	51
6.1.8	Pagina "Parking Spaces Management"	54
6.1.9	Meniul "Manage Entries"	56
6.1.10	Meniul "Settings"	57
6.2	Aplicația Java	58
6.2.1	Arhitectura	58
6.2.2	Realizarea interfeței grafice	59
6.2.3	Autentificarea (/login, /register, /passwordReset)	60
6.2.4	Pagina principală (/home)	62
6.2.5	Pagina și contul de administrator (/admin)	64
6.2.6	Pagina de raporturi (/report/list)	65
6.2.7	Pagina de întrebări și răspunsuri (/chat)	66
6.2.8	Părăsirea parcării folosind SecretID (/stuck)	67
7	Concluzii și dezvoltări ulterioare	68
7.1	Concluzii generale	68
7.2	Dezvoltări ulterioare	69

Listă de figuri

1	Structura directorului sursă al proiectului principal.	14
2	Imaginea inițială (stânga); Imaginea convertită în grayscale folosind "Luminosity" (dreapta)	16
3	Imaginea convertită în grayscale folosind metoda "Shades of Gray", în 4 nuanțe de gri (stânga); Imaginea convertită în grayscale folosind metoda "Shades of Gray", în 16 nuanțe de gri (dreapta)	16
4	Imaginea inițială (stânga); Imaginea după aplicarea filtrului "Average" cu un kernel de dimensiune 5x5 (dreapta)	18
5	Imaginea inițială (stânga); Imaginea după aplicarea filtrului median cu un kernel de dimensiune 5x5 (dreapta)	19
6	Imaginea inițială (stânga); Imaginea după aplicarea filtrului Gaussian cu un kernel de dimensiune 5x5 (dreapta)	19
7	Imaginea inițială (stânga); Imaginea după aplicarea filtrului bilateral cu un kernel de dimensiune 5x5 (dreapta)	20
8	Acuratețea pentru cele 4 filtre încercate, atât pe setul de testare, cât și într-un scenariu real.	20
9	Imaginea rezultată în urma aplicării filtrului bilateral (stânga); Imaginea rezultată după efectuarea operației morfologice de opening cu un kernel de dimensiune 13x13 (dreapta)	21
10	Imaginea rezultată în urma opening-ului (stânga); Imaginea rezultată în urma scăderii (mijloc); Imaginea rezultată în urma binarizării cu metoda Otsu (dreapta) . . .	22
11	Toate contururile din imaginea candidat, colorate în verde.	23
12	Contururile valide din imaginea candidat, colorate în verde.	24
13	Plăcuța de înmatriculare extrasă folosind metoda IMAGE_PROCESSING.	24
14	Imagini din setul de date.	26
15	Acuratețea la începutul fiecărei epoci.	27
16	Plăcuța de înmatriculare extrasă folosind metoda IMAGE_PROCESSING (stânga); Plăcuța de înmatriculare extrasă folosind metoda DNN (dreapta).	28
17	Plăcuța de înmatriculare detectată (stânga); Plăcuța de înmatriculare după preprocesare, fără corectarea distorsiunii (dreapta).	29
18	Plăcuța de înmatriculare detectată (stânga); Plăcuța de înmatriculare după aplicarea algoritmului CLAHE (dreapta).	31
19	Plăcuța de înmatriculare preprocesată (stânga); Plăcuța de înmatriculare după eliminarea caracterelor și opening (dreapta)	32
20	Colțurile identificate în plăcuța de înmatriculare	33
21	Plăcuța de înmatriculare după eliminarea caracterelor și opening (stânga); Plăcuța de înmatriculare originală, preprocesată (mijloc); Caracterele din interiorul plăcuței (dreapta)	34
22	Caracterele din interiorul plăcuței (stânga); Caracterele din interiorul plăcuței după corectarea distorsiunii folosind Undistortion (dreapta)	34
23	Caracterele din interiorul plăcuței (stânga); Caracterele din interiorul plăcuței după corectarea distorsiunii folosind SkewCorrection (dreapta)	35

24	Plăcuța de înmatriculare (stânga); Corectarea distorsiunii folosind SkewCorrection (mijloc); Corectarea distorsiunii folosind Undistortion (dreapta)	36
25	Toate caracterele posibile pe o plăcuță din România.	38
26	Rezultatele recunoașterii pe diferite plăcuțe.	39
27	Pagina principală a aplicației C++.	50
28	Meniul "Manage Cameras".	51
29	Pagina "Camera Management".	52
30	Pagina "Parking Spaces Management".	55
31	Meniul "Manage Entries".	56
32	Meniul opțiunii "Force exit for an entry".	57
33	Meniul opțiunii "Force action by photo".	57
34	Meniul "Settings".	57
35	Meniul opțiunii "View Specific Camera".	58
36	Meniul opțiunii "Change Detection Type".	58
37	Meniul de login din pagina de autentificare.	61
38	Meniul de register din pagina de creare a unui cont de utilizator.	62
39	Pagina principală (/home).	63
40	Pagina pentru vizualizarea locurilor de parcare în care numărul de înmatriculare pentru sesiunea respectivă a fost detectat. (/home/spaces/{ID sesiune}).	63
41	Accesarea unei pagini la care utilizatorul nu are acces.	64
42	Pagina de administrator (/admin).	65
43	Pagina de raporturi (/report/list).	66
44	Pagina de întrebări și răspunsuri (/chat).	67

1 Introducere

1.1 Scopul și obiectivele lucrării

Scopul acestei lucrări de diplomă este de a dezvolta un proiect care să vină atât în ajutorul șoferilor de autovehicule atunci când sunt nevoiți să își lase mașina în interiorul unei parări, cât și a administratorilor acestor parări. Pentru realizarea acestui proiect, a fost necesară dezvoltarea a două aplicații, una implementată în limbajul de programare C++ ce se ocupă de detectarea, recunoașterea și salvarea numerelor de pe plăcuțele de înmatriculare ale autovehiculelor într-o bază de date, aceasta fiind sistemul de monitorizare și gestionare în sine, dar și aplicația destinată administratorului parării, iar cealaltă aplicație, destinată șoferilor, implementată în limbajul de programare Java și constă într-o pagină web unde aceștia vor putea vizualiza diferite detalii legate de autovehiculul lor, precum timpul petrecut, data și ora la care au intrat/părăsit parcare pentru fiecare sesiune în parte, posibilitatea raportării unor probleme dacă este cazul și a unui mini-chat unde aceștia pot adresa diferite întrebări sau pot afla noutăți legate de parcare.

De asemenea, necesitatea unui proiect de acest fel este evidentă, numărul de mașini în România crește de la an la an iar nevoia șoferilor de a avea un loc de parcare pentru propriul autovehicul în timpul unei deplasări este absolut necesară. Acest proiect ar putea fi utilizat în mai multe scenarii: într-o parcare supraetajată, în parcare a unui mall, în parcare a unui aeroport sau a parcarilor din preajma acestuia, și așa mai departe. Spre exemplu, pe durata vizitării unui mall, tot procesul din interiorul parării ar fi mult mai ușor atât pentru șofer, cât și pentru deținătorul parării deoarece numărul de înmatriculare al autovehiculului este detectat, recunoscut și adăugat automat într-o bază de date la intrarea, părăsirea sau schimbarea unui loc din interiorul parării, iar șoferul va putea vedea timpul petrecut în parcare și costul acesteia, dar și locul pe care a parcat, în cazul uitării acestuia, folosind doar câteva click-uri.

Obiectivul principal al lucrării este cel menționat mai sus - ajutorul adus atât șoferilor de autovehicule, cât și a deținătorilor de parări, încercând automatizarea întregului proces de intrare și ieșire dintr-o parcare. Totuși, realizarea acestui proiect presupune următoarele obiective:

- Realizarea detecției plăcuței de înmatriculare a autovehiculului folosind procesarea de imagini și un model de inteligență artificială.
- Recunoașterea caracterelor folosind algoritmul de template matching.
- Găsirea locului de parcare pe care un autovehicul a parcat.
- Configurarea cu ușurință atât a camerelor video, cât și a locurilor de parcare din interiorul parării.
- Salvarea tuturor configurațiilor efectuate și a fiecărui număr de înmatriculare detectat, alături de locul în care a fost detectat (intrare/ieșire/loc de parcare)
- Realizarea unei interfețe cât mai prietenoase cu utilizatorul.

- Realizarea unei pagini web cu funcționalități cât mai simple, unde șoferii să poată vizualiza toate detaliile necesare legate de staționarea acestora în parcare.

1.2 Motivația alegerii temei

Motivul principal din spatele alegerii temei pentru dezvoltarea unui sistem de monitorizare și gestionare a unei parcuri auto, care integrează două aplicații distincte, este dorința de a crea o soluție completă și eficientă pentru administrarea și utilizarea facilităților de parcare. Prin combinarea unei aplicații în C++ ce se ocupă de detectarea și recunoașterea numerelor de înmatriculare, împreună cu salvarea acestora într-o bază de date, și a unei aplicații web în Java, care servește ca platformă pentru utilizatorii parcurii, se urmărește să se ofere o soluție cuprinzătoare și accesibilă.

Prin intermediul aplicației în C++, se pune accent pe automatizarea proceselor de monitorizare a intrărilor și ieșirilor dintr-o parcare, contribuind astfel la gestionarea eficientă a datelor legate de autovehiculele ce utilizează serviciile parcurii. Această aplicație este esențială pentru înregistrarea și stocarea precisă a informațiilor legate de autovehicule.

Pe de altă parte, aplicația web în Java oferă utilizatorilor parcurii o interfață intuitivă și accesibilă pentru gestionarea și utilizarea serviciilor oferite. Utilizatorii își pot crea cont folosind un cod special primit la intrarea în parcare și vor putea astfel să acceseze informații detaliate despre parcare lor, inclusiv istoricul parcurii și detaliile legate de locurile de parcare utilizate.

Prin combinarea acestor două aplicații se dorește să se ofere o soluție integrată care să simplifice procesele de administrare a parcurii și să îmbunătățească experiența utilizatorilor.

1.3 Structura lucrării

Lucrarea este împărțită în șapte capitole, care urmează să fie prezentate pe scurt, în continuare:

- **Introducere:** Acesta este capitolul introductiv. În acest capitol se vor prezenta scopul și obiectivele lucrării, alături de motivația alegerii temei.
- **Medii și concepte de programare:** prezintă limbajele de programare folosite în realizarea proiectului și concepte ce țin de partea de software, design și de procesul de compilare.
- **Detectia plăcuțelor de înmatriculare:** conține prezentarea în detaliu a fiecărei părți din cadrul procesului de detecție a plăcuțelor de înmatriculare. Astfel, sunt surprinse diferite informații referitoare la procesarea necesară a imaginilor capturate de la camerele video, având alături și rezultatul obținut după fiecare pas.
- **Recunoașterea caracterelor din plăcuța de înmatriculare:** conține prezentarea în detaliu a fiecărei părți din cadrul procesului de recunoaștere a caracterelor din plăcuța de înmatriculare, de la preprocesarea necesară, până la aplicarea algoritmului de recunoaștere pe fiecare caracter în parte. De asemenea, și în acest capitol vor exista imagini cu rezultatul obținut după fiecare pas.

- **Modelarea entităților:** descrie structura datelor și modelarea entităților prezente în sistem, oferind o viziune clară asupra organizării și gestionării datelor.
- **Prezentarea aplicațiilor:** prezintă în detaliu aplicațiile dezvoltate în cadrul acestui proiect, cuprinzând fiecare funcționalitate și fiecare meniu din cadrul acestora, precum și arhitecturile folosite în organizarea codului.
- **Concluzii și dezvoltări ulterioare:** este o recapitulare a realizărilor principale și o discuție despre potențialele direcții de dezvoltare și îmbunătățire a sistemului.

2 Medii si concepte de programare

2.1 Limbajul de programare C++

Inițial denumit "C cu clase", C++ este un limbaj de programare de nivel înalt, ce este considerat a fi o extensie și o îmbunătățire a limbajului C. Acesta păstrează toate caracteristicile limbajului C și aduce concepte specifice programării orientate pe obiecte, precum clase, obiecte, încapsulare, moștenire, polimorfism, gestionare de excepții și multe altele. Fiind un limbaj de programare orientat pe obiecte, accentul este pus mai mult pe "obiecte" și nu pe manipularea lor. Pe lângă caracteristica menționată anterior, C++ mai dispune și de alte caracteristici la care se numără:

- Viteza de compilare - deoarece C++ este un limbaj compilat, însemnând că codul sursă al acestuia este tradus direct în cod mașină, programul rezultat beneficiază de performanțe superioare în comparație cu alte limbaje de programare.
- Suport pentru pointeri - deși în ziua de astăzi acest lucru este indisponibil în multe alte limbaje de programare, C++ oferă o modalitate de a manipula și gestiona direct memoria cu ajutorul pointerilor.
- Portabilitate - fiind unul dintre cele mai utilizate limbaje din lume, codul C++ poate fi scris astfel încât să ruleze pe mai multe platforme hardware și sisteme de operare.

Având în vedere caracteristicile menționate [1], acestea contribuie la eficiența și adaptabilitatea sa, facilitând dezvoltarea aplicațiilor pe diverse platforme și în diverse contexte. Astfel, fiind unul dintre cele mai importante și puternice limbaje de programare, am decis ca implementarea codului sursă al aplicației pentru detectarea și recunoașterea numerelor de înmatriculare, dar și pentru gestionarea bazei de date a acesteia, să fie realizată în C++.

2.2 Limbajul de programare Java

Java, la fel ca și C++, este tot un limbaj de programare de nivel înalt, orientat pe obiecte, dar bazat pe clase, unde fiecare fișier este de fapt o clasă. A fost dezvoltat pentru a oferi portabilitate, performanță și securitate, fiind creat cu scopul de a fi independent de platformă, eficient și simplu. Atât Java, cât și C++, prezintă numeroase similarități din punct de vedere al sintaxei limbajului, însă limbajul Java este considerat a fi mai ușor de folosit [2]. Printre principalele caracteristici ale limbajului Java se numără:

- Programarea orientată pe obiecte - Java, la fel ca și C++, dispune de conceptele specifice programării orientate pe obiecte, îmbunătățind practicitatea limbajului de programare prin oferirea unei modalități mai eficiente de structurare a codului, promovând reutilizarea, flexibilitatea și modularitatea.
- Independent de platformă - programele create în Java pot fi rulate pe orice platformă hardware și sistem de operare datorită faptului că codul sursă este tradus într-un format intermediar numit "Java Bytecode" care permite rularea programului oriunde se dispune de o mașină virtuală Java instalată. JVM-ul (Java Virtual Machine) încarcă, verifică și execută Java

Bytecode-ul, asigurând totodată gestionarea memoriei, gestionarea excepțiilor, etc.

- Securitate - asupra securității s-a pus un accent deosebit, fiind asigurată prin diverse mecanisme, precum Sandboxing, care constă în izolarea resurselor periculoase ale sistemului de operare prin rularea aplicației Java într-un mediu controlat. Un alt mecanism este Garbage Collection, care previne scurgerile de memorie și depășirile de buffer.

În prezent, Java este utilizat în diverse aplicații și site-uri web, datorită numeroaselor avantaje pe care le oferă. Pentru site-ul web prezentat în această lucrare, implementarea acestuia va fi realizată cu ajutorul limbajului de programare Java și a framework-ului Spring.

Prin urmare, limbajul de programare Java se remarcă prin portabilitate, securitate și performanță. Datorită implementării în mod robust a conceptelor fundamentale ale programării orientate pe obiect, se facilitează crearea de cod bine structurat și ușor de întreținut.

2.3 Limbajul SQL

SQL (Structured Query Language) este un limbaj de programare dezvoltat de către IBM în anii 1970 și este folosit pentru gestionarea și manipularea datelor stocate într-o bază de date relațională [3]. În prezent, acesta este standard pentru majoritatea sistemelor de gestiune a bazelor de date.

Acest limbaj de programare este folosit într-o varietate largă de aplicații și domenii, de la aplicații web și mobile până la sisteme enterprise și analiză de date.

Principalele caracteristici ale limbajului SQL sunt:

- Structură - SQL este un limbaj care utilizează o sintaxă clară și structurată, împărțită în mai multe categorii, precum query-uri (interogări), actualizări, inserții, ștergeri și definiri de tabele și scheme.
- Interogări - Probabil cea mai frecventă utilizare a limbajului SQL este pentru interogarea bazelor de date. Cu ajutorul instrucțiunii SELECT, utilizatorii pot extrage date din tabele în funcție de anumite criterii.
- Actualizări, inserții și ștergeri: Limbajul SQL permite modificarea datelor existente în baza de date prin intermediul instrucțiunilor UPDATE, INSERT și DELETE. Acestea permit utilizatorilor să adauge, să șteargă sau să actualizeze date din tabelele existente.
- Crearea și administrarea tabelelor: SQL permite definirea structurii tabelelor și a altor obiecte din baza de date, precum constrângerile și view-urile, prin utilizarea instrucțiunilor CREATE, ALTER și DROP pentru a crea, a modifica sau a șterge aceste obiecte.

Astfel, cunoștințele de bază în limbajul SQL sunt un lucru esențial pentru dezvoltatorii de aplicații software, administratorii de baze de date și analiștii de date, pentru manipularea și gestionarea datelor în mod eficient. În implementarea aplicațiilor prezentate în această lucrare se folosește o bază de date comună iar datele din cadrul acesteia sunt accesate și manipulate cu ajutorul interogărilor SQL.

2.4 Rețele neurale artificiale

Rețelele neurale artificiale sunt modele matematice inspirate din structura și funcționarea creierului uman, folosite pentru a rezolva probleme complexe. Acestea sunt capabile să-și modifice structura internă în raport cu un obiectiv funcțional. În general, sunt potrivite pentru rezolvarea problemelor de tip neliniar reușind să reconstituie reguli neclare ce vor produce o soluție optimă pentru aceste probleme. Rețelele neurale artificiale sunt alcătuite din noduri, numite și elemente de procesare, conexiuni sau neuroni.

Astfel, o rețea neurală artificială este un ansamblu de neuroni interconectați și grupați în straturi care funcționează împreună pentru rezolvarea unei probleme.

Un neuron are ca scop reproducerea structurii și funcționării neuronului biologic, acesta fiind alcătuit dintr-un număr de intrări, fiecare dintre ele fiind caracterizate de o pondere proprie. Acesta își poate ajusta valorile ponderilor astfel încât pentru un set de perechi de intrare-ieșire să returneze un semnal de ieșire având o eroare minimă.

În timpul antrenării rețelei, aceasta își ajustează ponderile conexiunilor între neuroni pentru a minimiza o funcție de cost, cum ar fi eroarea dintre ieșirile prezise și valorile reale. Acest lucru se realizează folosind algoritmi de optimizare, cum ar fi gradient descent. Neuronii utilizează funcții de activare pentru a introduce non-liniaritate în rețea, permițându-i să învețe relații complexe între datele de intrare și ieșire.

La momentul actual, există o varietate largă de arhitecturi de rețele neurale artificiale, adaptate pentru diferite tipuri de probleme și de date. De exemplu, câteva arhitecturi populare sunt: rețelele cu straturi complet conectate (fully connected), rețelele convoluționale, rețelele recurente, rețelele generative adversariale (GANs) și altele. Aplicațiile acestor rețele sunt diverse, cum ar fi recunoașterea facială sau vocală, clasificarea și recunoașterea obiectelor într-o imagine, procesarea limbajului natural și multe altele.

În ultimii ani, au fost realizate progrese semnificative în acest domeniu, inclusiv dezvoltarea de noi arhitecturi sau algoritmi de antrenare mai eficienți și tehnici de regularizare pentru a preveni overfitting-ul. Rețelele neurale artificiale sunt astfel un instrument puternic cu aplicații largi și profunde în numeroase domenii și industrii.

Totuși, în aplicația principală ce a fost realizată pentru proiectul de diplomă, pentru detectarea plăcuței de înmatriculare a unui autovehicul s-a folosit o bibliotecă ce a permis antrenarea unei rețele neurale convoluționale folosind un set de date propriu și convertirea modelului astfel încât să se poată realiza inferența într-o aplicație C++.

2.5 Rețele neurale convoluționale

Rețelele neurale convoluționale (CNNs) reprezintă o ramură specializată a rețelelor neurale artificiale, adaptată pentru prelucrarea și analiza imaginilor. Ele sunt esențiale în domeniul viziunii artificiale și au fost dezvoltate pentru abordarea eficientă a provocărilor specifice ale analizei pe date spațiale, cum ar fi imagini și videoclipuri.

La fel ca și rețelele neurale artificiale, și cele convoluționale sunt formate din straturi de neuroni, însă structura acestora este adaptată astfel încât să se profite de proprietățile spațiale ale datelor unei imagini. Aceste rețele utilizează straturi speciale de convoluție pentru extragerea caracteristicilor relevante dintr-o imagine, reducând astfel complexitatea datelor și făcând posibilă identificarea de pattern-uri (modele) și obiecte în imagini [4].

Pe durata antrenării, rețelele convoluționale ajustează ponderile filtrelor convoluționale pentru a minimiza o funcție de cost, de obicei eroarea dintre predicțiile rețelei și etichetele reale ale imaginilor folosite pentru antrenare. La fel ca la rețelele neurale artificiale, algoritmi de optimizare precum gradient descent sunt folosiți și în acest caz.

Una dintre caracteristicile importante ale rețelelor neurale convoluționale este capacitatea lor de a captura caracteristici complexe din imagini, pornind de la caracteristici de bază precum marginile și texturile, până la caracteristici mai abstracte, precum formele obiectelor sau chiar obiecte întregi.

Aceste rețele sunt folosite într-o varietate largă de aplicații, incluzând până și detecția plăcuțelor de înmatriculare. În ultimii ani, au fost obținute progrese semnificative în domeniul arhitecturilor de rețele neurale convoluționale, precum ResNet, Inception și EfficientNet, care au îmbunătățit performanța și eficiența acestui tip de rețele.

2.6 Biblioteca OpenCV

OpenCV (Open Source Computer Vision Library) este o bibliotecă open-source specializată în vedere computerizată, procesarea imaginilor și învățare automată, deținând peste 2500 de algoritmi ce pot fi utilizați în diverse aplicații, precum recunoașterea facială, detectarea obiectelor, analiza medicală. Aceasta a fost scrisă în limbajul de programare C++, însă este compatibilă și cu alte limbaje de programare, inclusiv C++, Python, Java și Matlab. [5]

Pentru procesarea de imagini, OpenCV oferă o gamă variată de funcții pentru manipularea imaginilor, precum conversia de la un spațiu de culoare la altul, ajustarea luminii și a contrastului, transformări geometrice, filtrare și altele. De asemenea, în cadrul bibliotecii, există un modul numit OpenCV DNN (Deep Neural Networks), dedicat utilizării și implementării DNN-urilor în aplicații de computer vision. Oferă funcționalități puternice în implementarea inferenței modelelor DNN și suporta integrarea cu diferite framework-uri, precum TensorFlow, Caffe, Yolo, Darknet și altele.

În proiectul C++, pentru detecția și recunoașterea numerelor de înmatriculare ale mașinilor, s-a utilizat versiunea 4.8.0 de OpenCV. Biblioteca a fost utilizată de la începutul proiectului, în timpul detectării plăcuței de înmatriculare și a literelor de pe aceasta, unde s-au folosit algoritmi adecvați, până la final, la recunoașterea caracterelor, unde s-a folosit un set de date ce conține toate caracterele de pe plăcuțele din România și s-a realizat template matching.

2.7 Biblioteca YOLOv5

YOLOv5 (You Only Look Once, versiunea 5), dezvoltată de către echipa Ultralytics, este o bibliotecă de Deep Learning și un framework pentru detecția diferitelor obiecte în imagini și video-uri.

Ca structură, se bazează pe o abordare de tip "You Only Look Once", însemnând că în loc ca procesul de detecție să fie împărțit pe etape separate, cum ar fi detectarea, clasificarea și localizarea, acesta este tratat ca o singură rețea neurală, abordare ce permite furnizarea predicțiilor în timp real. Astfel, biblioteca YOLO este cunoscută pentru eficiența sa.

Biblioteca permite utilizatorilor să antreneze modele folosind seturi de date proprii, folosind una dintre arhitecturile variate pe care biblioteca le furnizează, cum ar fi YOLOv5s (Small), YOLOv5m (Medium), YOLOv5l (Large), și realizarea inferenței (detecție de obiecte) pe imagini noi [8].

În proiectul dezvoltat pentru aceasta lucrare de diplomă, detecția plăcuțelor de înmatriculare se poate realiza, la alegerea utilizatorului, atât prin folosirea algoritmilor de procesare de imagini din OpenCV, cât și prin folosirea unui model antrenat cu ajutorul acestei biblioteci.

2.8 Framework-ul Qt

Qt este un framework utilizat pentru crearea de aplicații cross-platform, precum serverele, care rulează pe diverse platforme software și hardware, cum ar fi Windows, Linux, OS X, Android, iOS, și altele. Preprocesorul MOC (Meta-Object Compiler) extinde limbajul C++ prin adăugarea de noi caracteristici necesare în dezvoltarea aplicațiilor cu interfață grafică, cum ar fi semnalele și sloturile. Există numeroase aplicații populare cu interfață grafică realizate cu ajutorul framework-ului Qt și folosite de milioane de utilizatori din întreaga lume, printre care se numără Skype, Google Earth și browser-ul Opera. [6]

Proiectarea și crearea de GUI (Graphical User Interface) se realizează cu ajutorul uneia grafice incluse în framework, Qt Designer. Elementele grafice sunt atașate codului prin folosirea mecanismului de sloturi și a semnalelor Qt. Qt Designer generează automat codul sursă asociat cu interfața, odată ce aceasta este proiectată. [7] Acest lucru elimină nevoia programatorilor de a scrie manual codul pentru fiecare element din interfață, economisind timp și evitând potențialele erori. Proiectarea interfețelor grafice ale aplicațiilor se realizează foarte ușor, având posibilitatea adăugării și configurării a diferitelor elemente GUI, cum ar fi butoanele, etichetele, textbox-uri și altele, utilizând pur și simplu drag-and-drop. Utilizarea Qt Designer implică patru etape de bază:

- Alegerea propriei interfețe și a obiectelor dorite;
- Așezarea obiectelor pe interfață;
- Conectarea semnalelor la sloturile corespunzătoare;
- Vizualizarea interfeței.

Pentru realizarea interfeței grafice a aplicației principale, responsabilă pentru detecția și recunoașterea automată a numerelor de înmatriculare ale mașinilor, s-a folosit framework-ul Qt 6.4.1. Meniul acesteia oferă utilizatorului o metodă ușoară și intuitivă de configurare a camerelor video și a locurilor de parcare, folosite pentru monitorizarea traficului din interiorul unei parări.

2.9 Framework-ul Spring

Spring este un framework open-source și cross-platform, ce oferă un model cuprinzător de programare și configurare a aplicațiilor moderne de întreprindere bazate pe limbajul de programare Java. Deoarece se axează pe suportul infrastructural al aplicațiilor, echipele de dezvoltare pot dedica mai mult timp logicii la nivel de aplicație, fără legături inutile cu medii de implementare specifice [9]. Printre principalele caracteristici ale framework-ului se numără:

- Inversiunea de control - containerul Spring preia responsabilitatea gestionării obiectelor și dependențelor, oferind astfel o structură modulară și ușor de gestionat.
- Spring Data - o abordare simplificată în ceea ce privește lucrul cu baze de date relaționale și non-relaționale.
- Spring Web - furnizează suport pentru dezvoltarea aplicațiilor web, oferind funcționalități precum gestionarea cererilor HTTP, manipularea parametrilor de solicitare, și facilitarea construirii de aplicații web robuste și scalabile.
- Spring Security - se ocupă de aspectele legate de securitatea aplicațiilor Java, furnizând suport pentru autentificare, autorizare și alte aspecte de securitate.

Pentru realizarea aplicației secundare, ce constă într-un site web, s-a folosit framework-ul Spring. Datorită componentei Spring Security, utilizatorii își pot crea un cont asociat numărului de înmatriculare corespunzător mașinii detectate la intrarea din parcare. Odată conectați, aceștia pot vizualiza informații, precum timpul petrecut în parcare, ora intrării, locurile de parcare unde a fost localizată mașina, și transmiterea unor întrebări în cadrul unui chat și a eventualelor plângeri.

2.10 Platforma CMake

CMake este un software open-source și cross-platform pentru automatizarea proceselor de build, testing, packaging și instalare a programelor prin utilizarea unei metode independente de compilator. CMake generează fișierele de compilare ale unui alt sistem, prin urmare acesta nu este un sistem de compilare [10].

Pentru generarea fișierelor de compilare, se vor utiliza fișiere numite "CMakeLists.txt" ce vor fi plasate în fiecare director sursă. Acest lucru funcționează prin producerea unui mediu de construire nativ unde se vor crea bibliotecile, se vor genera pachetele, se va compila codul sursă și ulterior va construi executabile.

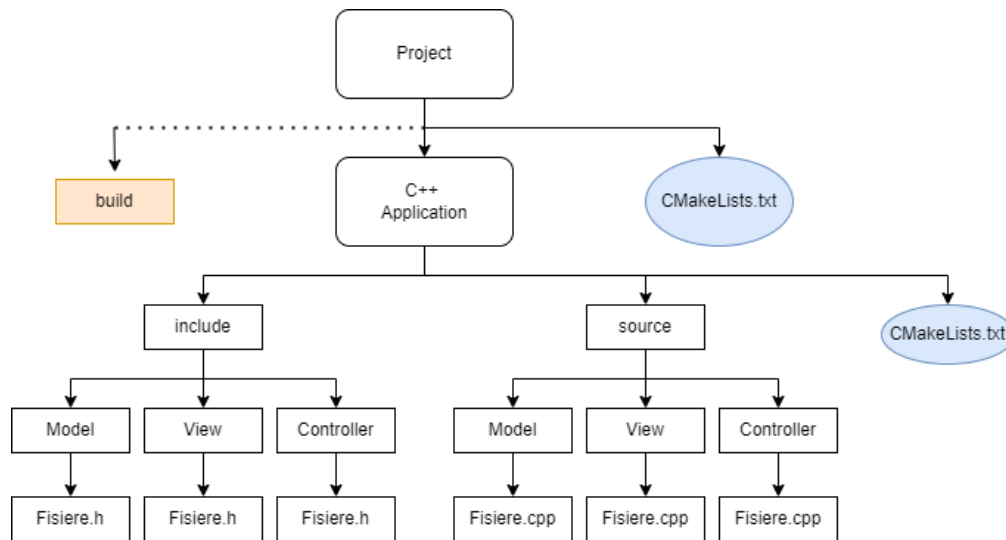


Figura 1: Structura directorului sursă al proiectului principal.

În figura 1 este evidențiată structura directorului sursă al proiectului principal, precum și utilizarea fișierului "CMakeLists.txt". După ce proiectul este generat și configurat, se creează directorul "build", unde vor fi prezente toate informațiile necesare rulării proiectului.

3 Detecția plăcuțelor de înmatriculare

Aplicația realizată a fost aleasă astfel încât să fie de ajutor persoanelor sau companiilor ce dețin o parcare, dar și persoanelor ce doresc să o folosească. Utilizarea acesteia poate duce la ușurarea monitorizării parcării, fără a mai avea nevoie în permanență de un supraveghetor, deoarece detecția și recunoașterea numărului de înmatriculare al unei mașini se va face automat la apăsarea unui buton de către șoferul acesteia. Odată detectat, numărul de înmatriculare este adăugat într-o bază de date, alături de ora intrării și un cod special, numit **SecretID**, ce va permite crearea unui cont de utilizator pe site-ul web destinat aplicației principale sau părăsirea parcării în cazul în care, din diverse motive, precum lumina nefavorabilă sau condiții meteo, numărul de înmatriculare nu a putut fi identificat.

3.1 Preprocesarea imaginilor

3.1.1 Redimensionarea imaginii

Rezoluția unei imagini se referă la detaliul sau claritatea imaginii și este măsurată în pixeli. Imaginile preluate de la o camera video pot avea o dimensiune variabilă în funcție de setările sau capacitățile tehnice ale acesteia. Astfel, indiferent de la camera video de la care provine imaginea, se va realiza o redimensionare a acesteia la o rezoluție fixă, de 640 de pixeli în lățime și 480 de pixeli în înălțime.

3.1.2 Conversia imaginii în grayscale

Atât imaginile preluate de la camerele video adăugate, cât și cele destinate testării, sunt imagini de tip BGR, mai precis, imagini care conțin trei canale de culori: canalul albastru (Blue), canalul verde (Green) și canalul roșu (Red). În contextul detectării plăcuțelor de înmatriculare, informația despre culoare nu este esențială deoarece asupra imaginii inițiale vor avea loc mai multe operații și identificări de contururi, reducând astfel complexitatea. Pentru realizarea conversiei unei imagini în grayscale s-au folosit 2 metode diferite, "Luminosity" și "Shades of Gray" [11].

Metoda "Luminosity" reprezintă o variantă mai avansată a metodei în care se face media valorilor de pe cele 3 canale. În esență, această metodă calculează o medie a valorilor, dar utilizează ponderi pentru a reflecta mai bine percepția umană asupra culorilor. Deoarece ochiul uman este mai sensibil la verde decât la alte culori, ponderea pentru verde este cea mai mare în calculul mediei ponderate.

Formula pentru realizarea conversiei unui pixel folosind metoda "Luminosity":

$$P_{\text{gray}} = 0.11 \cdot P_{\text{blue}} + 0.59 \cdot P_{\text{green}} + 0.30 \cdot P_{\text{red}} \quad (1)$$



Figura 2: Imaginea inițială (stânga); Imaginea convertită în grayscale folosind "Luminosity" (dreapta)

Metoda "Shades of Gray" este de asemenea o alta variantă a metodei în care se face media valorilor celor 3 canale, cu diferența că putem specifica numărul dorit de nuanțe de gri. Este acceptată orice valoare cuprinsă între 2 și 256, unde valoarea 2 va rezulta o imagine alb-negru, iar valoarea 256 va rezulta imaginea asemanatoare conversiei folosind metoda mediei clasice. Spre exemplu, în figura 3, s-au folosit 4 nuanțe pentru conversia imaginii din stânga, adică negru, gri închis, gri deschis și alb.

Formula pentru realizarea conversiei unui pixel folosind metoda "Shades of Gray":

$$P_{gray} = \left(\frac{P_{blue} + P_{green} + P_{red}}{3} \right) \cdot \frac{256}{ShadesNumber} + 0.5) \cdot \frac{256}{ShadesNumber} \quad (2)$$



Figura 3: Imaginea convertită în grayscale folosind metoda "Shades of Gray", în 4 nuanțe de gri (stânga); Imaginea convertită în grayscale folosind metoda "Shades of Gray", în 16 nuanțe de gri (dreapta)

În timp ce metoda "Shades of Gray" a oferit rezultate acceptabile în anumite condiții, s-a dovedit că alegerea unui număr de nuanțe prea scăzut rezultă în deteriorarea calității imaginii, pe când alegerea unui număr prea ridicat rezultă în imaginea obținută în urma folosirii metodei mediei clasice. Prin urmare, metoda "Luminosity" a reprezentat o opțiune mai fiabilă și mai eficientă, oferind în final cea mai mare acuratețe în detecția numerelor de înmatriculare.

3.1.3 Reducerea zgomotului

Zgomotul într-o imagine se constituie din perturbările nedorite sau aleatorii care sunt introduse în imagine în timpul captării, stocării, transmiterii sau prelucrării acesteia. Acesta poate avea diverse origini, precum fluctuațiile de lumină, interferențele electromagnetice sau erorile de captare a semnalului [12].

Algoritmii de eliminare a zgomotului diminuează sau elimină vizibilitatea zgomotului prin uniformizarea întregii imagini, păstrând totuși regiunile apropiate de limitele de contrast. Cu toate acestea, aceste metode pot masca detalii subtile ale contrastului scăzut. Pentru a extrage informații corecte dintr-o imagine, este crucial să minimizăm zgomotul la un nivel cât mai mic posibil.

Pentru reducerea zgomotului, am comparat patru algoritmi ce se găsesc în biblioteca OpenCV:

- Average
- Median
- Gaussian
- Bilateral

Filtrul "Average" calculează valoarea medie a intensității dintr-o vecinătate a fiecărui pixel din imagine și o atribuie pixelului corespunzător din imaginea de ieșire. Acest lucru se realizează prin parcurgerea imaginii cu ajutorul unui kernel (masca) peste imagine, iar fiecare pixel din imaginea de ieșire este calculat ca media pixelilor din fereastra corespunzătoare din imaginea de intrare. Un kernel de dimensiune 3x3 arată în felul următor:

$$Kernel = \frac{1}{9} \cdot \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad (3)$$



Figura 4: Imaginea inițială (stânga); Imaginea după aplicarea filtrului "Average" cu un kernel de dimensiune 5x5 (dreapta)

Filtrul median funcționează prin înlocuirea valorii fiecărui pixel cu valoarea mediană dintr-o regiune vecină a aceluși pixel. Prin sortarea valorilor pixelilor și alegerea celei din mijloc, acest filtru este eficient în reducerea zgomotului impulsiv, cum ar fi "zgomotul sării și piperului", ce constă într-o serie de pixeli foarte luminoși sau foarte întunecați, păstrând totodată detaliile marginilor în imagine. Cu toate acestea, poate fi mai lent decât alte filtre, deoarece necesită sortarea valorilor din cadrul kernel-ului, iar eficacitatea sa poate fi limitată în reducerea altor tipuri de zgomot, cum ar fi zgomotul Gaussian. Cu toate acestea, filtrul median rămâne o unealtă valoroasă și larg utilizată într-o varietate de aplicații de procesare a imaginilor. În ecuația de mai jos avem ca exemplu valori aleatorii a pixelilor din cadrul kernel-ului de dimensiune 3x3 ce parcurge o imagine și efectuează filtrul median:

$$Kernel = \begin{bmatrix} 5 & 4 & 3 \\ 6 & 9 & 2 \\ 7 & 8 & 1 \end{bmatrix} \Rightarrow [5 \ 4 \ 3 \ 6 \ 9 \ 2 \ 7 \ 8 \ 1] \Rightarrow [1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8 \ 9] \quad (4)$$



Figura 5: Imaginea inițială (stânga); Imaginea după aplicarea filtrului median cu un kernel de dimensiune 5x5 (dreapta)

Filtrul Gaussian funcționează prin aplicarea unui kernel de tip Gaussian peste imagine și convoluționarea acestuia peste imaginea originală. Acest filtru se caracterizează prin distribuția lui Gauss, care acordă o pondere ridicată pixelilor din centrul kernel-ului, în timp ce pixelii din apropierea marginilor primesc o pondere scăzută. Filtrul este dat de următoarea formulă:

$$G(x, y) = \frac{1}{2\pi\sigma^2} \cdot e^{-\frac{x^2+y^2}{2\sigma^2}} \quad (5)$$



Figura 6: Imaginea inițială (stânga); Imaginea după aplicarea filtrului Gaussian cu un kernel de dimensiune 5x5 (dreapta)

Filtrul bilateral folosește două filtre gaussiene separate, unul pentru spațiu, iar celălalt pentru intensitate. Prin aplicarea unui filtru gaussian în domeniul spațial se asigură că doar pixelii vecini sunt considerați în procesul de filtrare, păstrând astfel detaliile structurale ale imaginii. Al doilea filtru, aplicat în domeniul intensității, se asigură că doar pixelii cu intensități asemănătoare sunt

luați în considerare. Această combinație de filtre permite filtrului bilateral să mențină marginile clare ale obiectelor din imagine, deoarece pixelii de pe margini vor avea variații mari ale intensității.



Figura 7: Imaginea inițială (stânga); Imaginea după aplicarea filtrului bilateral cu un kernel de dimensiune 5x5 (dreapta)

În final, filtrul bilateral s-a dovedit a fi cel mai eficient în procesul de detecție a plăcuței de înmatriculare a unei mașini. Având în vedere capacitatea sa de a menține marginile clare, filtrul bilateral este ideal pentru a evidenția și izola plăcuța de înmatriculare în diverse condiții de lumină și în prezența a diferitor fundaluri. Acesta a obținut cea mai mare acuratețe în procesul de detecție a plăcuței de înmatriculare pe imaginile din setul de testare, set ce conține imagini din unghiuri nefavorabile, pe timp de zi și noapte, dar și pe condiții meteo mai puțin favorabile, precum zăpadă, astfel încât detecția plăcuțelor de înmatriculare să atingă o acuratețe mult mai mare în cazul plasării camerei video în vederea gestionării unei parcare. Scenariul real se referă la capturarea mai multor imagini din 4 fluxuri video, filmate cu un telefon mobil iPhone 12, în 1080p, în 4 zile diferite, simulând intrarea și părăsirea a 2 mașini dintr-o parcare.

	Average	Median	Gaussian	Bilateral
Set testare	22,22%	31.11%	71,11%	77,77%
Scenariu real	75%	55%	100%	100%

Figura 8: Acuratețea pentru cele 4 filtre încercate, atât pe setul de testare, cât și într-un scenariu real.

3.1.4 Operația morfologică de Opening și Thresholding

Operația morfologică de opening constă în aplicarea consecutivă a eroziunii și dilatării asupra unei imagini. Prin eroziune, se subțiază obiectele din imagine, în timp ce prin dilatare se umplu și se extind zonele rămase, ambele operații fiind realizate cu ajutorul unui kernel ce parcurge imaginea și modifica pixelii în funcție de vecinii lor [13]. Acest proces este eficient în eliminarea zgomotelor mici și a altor artefacte din imagine, păstrând astfel dimensiunea și forma generală a obiectelor de

interes. În aplicația realizată, operația morfologică de opening este folosită atât pe imagini binare, cât și pe imagini grayscale. Aplicarea acesteia pe o imagine grayscale este asemănătoare cu cea pe imagini binare, diferența fiind aplicarea pe intensități de gri. Prin urmare, eroziunea și dilatarea nu mai sunt efectuate pe valori binare (alb și negru), ci în funcție de intensitatea pixelilor. Așadar, pe imaginea rezultată în urma aplicării filtrului bilateral, se efectuează operația morfologică de opening pe imaginea grayscale.



Figura 9: Imaginea rezultată în urma aplicării filtrului bilateral (stânga); Imaginea rezultată după efectuarea operației morfologice de opening cu un kernel de dimensiune 13x13 (dreapta)

În continuare, se va efectua operația de scădere între imaginea obținută în urma aplicării filtrului bilateral, și imaginea obținută după realizarea operației de opening. Operația de scădere constă în scăderea valorilor de intensitate ale pixelilor corespunzători din două imagini. Rezultatul acestei operații este o imagine în care fiecare pixel din imaginea rezultată are o valoare care este diferența dintre valorile de intensitate ale aceluiași pixel din cele două imagini de intrare. Mai precis, pentru două imagini grayscale, A și B, și un pixel (x, y) din imaginea rezultată, rezultatul scăderii este:

$$R(x, y) = A(x, y) - B(x, y) \quad (6)$$

Pentru a converti o imagine color sau grayscale într-o imagine binară, în care pixelii pot avea doar două valori, negru (valoarea 0) sau alb (valoarea 1), se folosește o tehnică numită thresholding. Aceasta este una dintre cele mai simple tehnici de extragere a obiectelor dintr-o imagine din fundalul acesteia, și constă în alegerea unei valori T ca prag, astfel încât valorile pixelilor mai mari decât acest prag devin 1 (alb), iar cele sub acest prag devin 0 (negru).

$$\begin{cases} 1, & f(x, y) \geq T \\ 0, & \text{altfel} \end{cases} \quad (7)$$

Din diverse motive, cum ar fi culoarea mașinii sau fundalul imaginii, alegerea manuală a unui prag care să ofere rezultate optime în majoritatea situațiilor nu este posibilă. Astfel, este nevoie

de o soluție prin care pragul T să se determine automat, în funcție de imaginea de intrare. Determinarea automată a unui prag optim se poate realiza cu ajutorul metodelor Otsu și Triangle.

Metoda Triangle (triunghiului) este o tehnică de binarizare a imaginilor în care pragul optim este ales prin utilizarea unui triunghi dreptunghic. Inițial, se trasează o linie de la maximul histogramei până la minimul acesteia. Ulterior, pentru alegerea pragului optim se trasează o linie perpendiculară pe histogramă, astfel încât distanța față de histogramă să fie maximizată [14]. Această abordare funcționează foarte bine atunci când în histograma imaginii există un singur vârf dominant, pragul optim aflându-se la baza acestuia. Însă, în majoritatea imaginilor din setul de testare menționat, s-a observat faptul că histograma acestora conține cel puțin două vârfuri dominante iar rezultatele obținute în urma thresholding-ului nu erau favorabile, așadar, s-a renunțat la această metodă în favoarea metodei Otsu.

Metoda Otsu, la fel ca și metoda Triangle, este tot o tehnică de binarizare a imaginilor, utilizată pentru determinarea automată a unui prag optim. Această metodă se bazează pe analiza histogramei, propunându-și să găsească pragul optim prin adăugarea varianței pixelilor de fundal la varianța pixelilor de prim-plan, pentru toate pragurile posibile, urmând ca pragul optim să fie cel pentru care suma varianțelor este minimă [15].

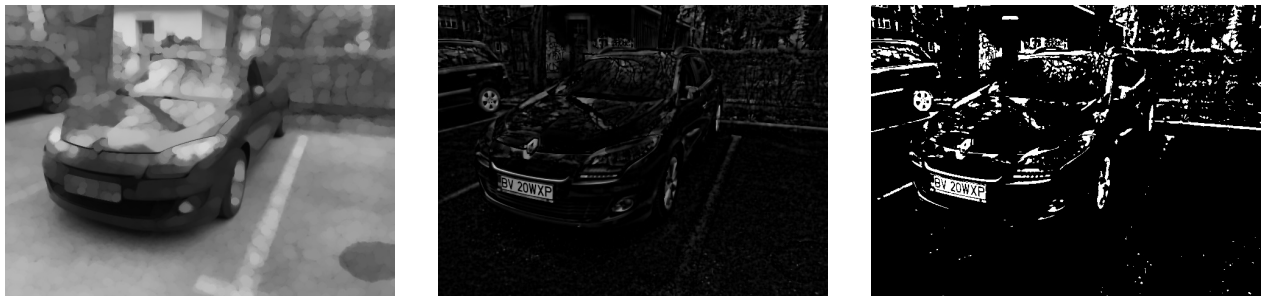


Figura 10: Imaginea rezultată în urma opening-ului (stânga); Imaginea rezultată în urma scăderii (mijloc); Imaginea rezultată în urma binarizării cu metoda Otsu (dreapta)

3.2 Extragerea contururilor

Contururile se descriu ca fiind linia care unește toate punctele continue, de-a lungul unei limite, care au aceeași culoare sau intensitate. Acestea sunt utile într-o varietate de aplicații ce necesită analiza și recunoașterea de obiecte într-o imagine, precum recunoașterea facială, detectarea și segmentarea obiectelor, etc.

Detectarea contururilor se poate realiza, de exemplu, prin identificarea tuturor componentelor conexe dintr-o imagine binară. Astfel, imaginea este scanată și pixelii sunt grupați în componente pe baza conectivității acestora, adică toți pixelii dintr-o componentă conexă care au, în cazul unei imagini binare, valoarea 1, pentru reprezentarea unui pixel alb, sau valori similare de intensitate a pixelilor, în cazul unei imagini grayscale. Deoarece imaginea candidat este binarizată, după determinarea grupurilor de pixeli, componentele conexe vor reprezenta posibila existență a unui obiect iar acestea vor fi etichetate cu o valoare sau o culoare. În continuare, după identificarea tuturor componentelor conexe, se va analiza granița dintre obiect și fundal, urmând ca contururile să fie

determinate de o serie de puncte continue ce delimitează regiunea obiectului [16]. Se observă că, în figura 11, numărul conturilor este foarte mare, așadar acestea vor trece printr-un proces de filtrare pentru a păstra doar conturile relevante ce ar putea delimita plăcuța de înmatriculare.

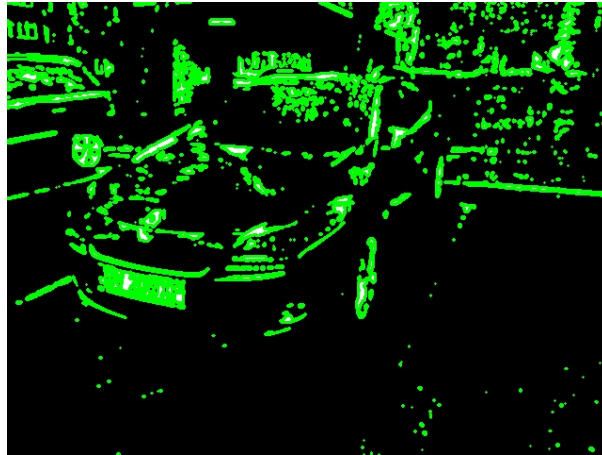


Figura 11: Toate conturile din imaginea candidat, colorate în verde.

Plăcuța de înmatriculare nu este mereu aliniată perfect cu axele orizontale și verticale ale imaginii, astfel că este nevoie de un obiect care să reprezinte un dreptunghi rotit, iar biblioteca OpenCV oferă acest lucru prin intermediul clasei `RotatedRect`. Spre deosebire de clasa `Rect`, care reprezintă un dreptunghi aliniat cu axele orizontale și verticale ale imaginii și este definit printr-un punct de origine, prezent în colțul din stânga sus, și o lățime și lungime, clasa `RotatedRect` este definită de o pereche de coordonate reprezentând centroidul dreptunghiului, lungimea și lățimea acestuia, dar și un unghi de rotație care specifică felul în care dreptunghiul este rotit în jurul centroidului său.

Pentru începutul procesului de filtrare a conturilor, se va găsi dreptunghiul rotit de cea mai mică suprafață care încadrează fiecare contur identificat în parte și se va calcula aria acestuia. Deoarece scopul proiectului este de a detecta plăcuțele de înmatriculare dintr-o parcare, am analizat toate imaginile din setul de testare și am observat că, în medie, aria plăcuței de înmatriculare este egală cu 2620 pixeli pătrați. De asemenea, într-un scenariu real, media ariei a fost de aproximativ 3400 pixeli pătrați, înregistrând o valoare maximă a ariei de 4500 pixeli pătrați. Având în vedere aceste statistici dar și faptul că imaginea candidat este redimensionată la 640 de pixeli pe lățime și 480 de pixeli pe lungime, am decis ca toate conturile a căror arie este mai mică de 1000 sau mai mare de 5000 vor fi excluse, renunțând astfel la contururi ce nu sunt relevante, precum alte obiecte din jurul mașinii sau chiar părți din mașină, ca de exemplu, farul, roțile sau chiar tabla acesteia, și păstrând astfel conturile valide în care s-ar putea afla plăcuța de înmatriculare.



Figura 12: Contururile valide din imaginea candidat, colorate în verde.

3.3 Extragerea plăcuței de înmatriculare

În continuare, după filtrarea contururilor, pentru fiecare contur valid se va calcula aria acestuia folosind funcția `contourArea` din biblioteca OpenCV sau utilizând Teorema lui Green pentru calculul ariei unei regiuni în planul xy delimitată de un contur C [17]:

$$A = \frac{1}{2} \oint_C x dy - y dx \quad (8)$$

Astfel, ariile contururilor valide vor fi stocate într-o listă iar în final, se va extrage conturul a cărui arie este maximă și va fi folosit pentru găsirea dreptunghiului de încadrare a conturului respectiv, urmând să fie folosit pentru decuparea plăcuței de înmatriculare de pe poza originală. În figura 12 se poate observa faptul că conturul din jurul plăcuței de înmatriculare este conturul a cărui arie este maximă.



Figura 13: Plăcuța de înmatriculare extrasă folosind metoda `IMAGE_PROCESSING`.

3.4 Metodă alternativă folosind rețele neurale convoluționale

Pe durata staționării a unei mașini în parcare, camerele destinate locurilor de parcare vor încerca, la un anumit interval de timp, localizarea mașinii într-unul dintre locurile de parcare configurate. Imaginile capturate de la camerele destinate locurilor de parcare pot conține mai multe mașini. Prin urmare, fiecare loc de parcare configurat este decupat într-o imagine nouă, rezultând astfel mai multe imagini, în funcție de numărul de locuri de parcare acoperite de camera video respectivă. Având în vedere distanța dintre camera video și plăcuța de înmatriculare, calitatea imaginii

în urma decupării și luând în considerare faptul că viteza detecției nu este la fel de importantă ca în cazul camerelor destinate intrării și ieșirii mașinilor, detecția facându-se doar la un anumit interval de timp mai ridicat (cel puțin 1 minut), am decis ca metoda folosită să fie diferită față de cea destinată celorlalte tipuri de camere video din cadrul aplicației în vederea obținerii unor rezultate mai optime.

Metoda propusă constă în folosirea rețelelor neurale convoluționale, mai precis, antrenarea unui model folosind biblioteca YOLOv5 pe un set de imagini ce are plăcuțele de înmatriculare etichetate, și efectuând predicții în C++ folosind modulul DNN (Deep Neural Networks) din biblioteca OpenCV.

3.4.1 Setul de date utilizat

În antrenarea unui model de rețea neurală convoluțională (CNN), alegerea unui set de date corespunzător este esențial în vederea obținerii unui model precis. Pentru a permite modelului să învețe caracteristicile relevante pentru problema pe care dorim să o rezolvăm, setul de date trebuie să cuprindă o varietate suficientă de exemple. Setul de date folosit în rezolvarea problemei pentru detecția plăcuței de înmatriculare este o combinație a două seturi de date diferite obținute de pe platforma Roboflow Universe [18], dintre care pe unul dintre acestea s-a efectuat augmentarea pe întreg setul cu ajutorul platformei Roboflow.

Roboflow este o platformă ce oferă instrumente și servicii pentru a gestiona și pregăti seturi de date pentru diverse aplicații de inteligență artificială, precum detectarea obiectelor din imagini, segmentarea semantică a obiectelor din imagini, etc. Platforma deține funcționalități ce au ca scop accelerarea și simplificarea procesului de dezvoltare a aplicațiilor de inteligență artificială, având capacitatea de a importa, eticheta, procesa și exporta seturi de date, precum și de a antrena și evalua modelele de inteligență artificială. Folosind aceasta platformă, setul de date a fost augmentat folosind următoarele tehnici: schimbarea luminozității și a contrastului, adăugare de zgomot, rotire și flip orizontal/vertical.

Roboflow Universe este un ecosistem vast destinat comunității de inteligență artificială ce oferă o gamă largă de resurse. Prin intermediul Roboflow Universe, utilizatorii au acces la seturi de date deja etichetate și pregătite pentru diverse sarcini, precum antrenarea unui model de inteligență artificială.

În final, setul de date conține 7593 de imagini dedicate procesului de antrenare, și 2077 de imagini dedicate procesului de validare, având atât imagini capturate din fața mașinilor, cât și din spațiile lor, dar și imagini ce conțin doar plăcuța de înmatriculare, fără a se observa plasarea acesteia pe o mașină. Toate imaginile au o dimensiune de 640 de pixeli atât în lungime, cât și în lățime. În figura 14 sunt surprinse 6 imagini aleatorii din setul de date.

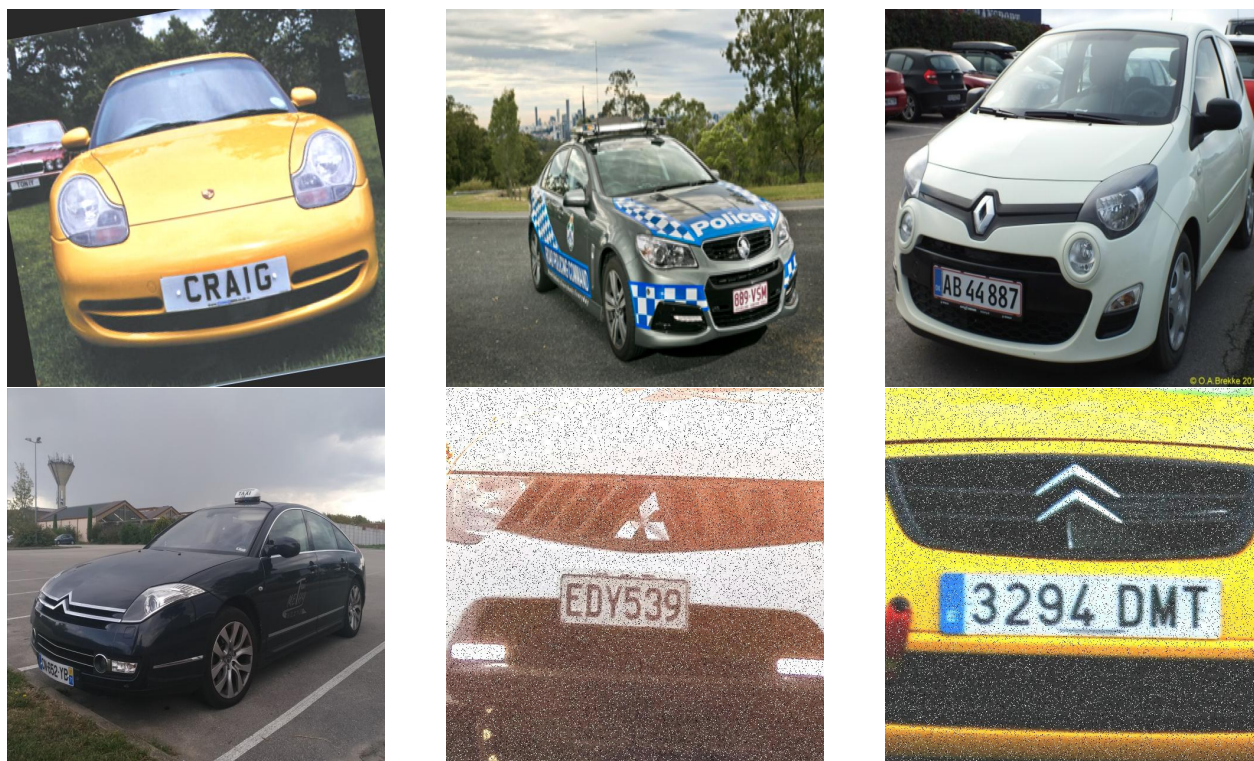


Figura 14: Imagini din setul de date.

3.4.2 Antrenarea modelului

Arhitectura YOLO este alcătuită din 3 componente, backbone (schelet), neck (gât) și head (cap), toate 3 fiind esențiale pentru structura și funcționarea rețelei. Backbone-ul reprezintă prima componentă a rețelei iar aceasta este responsabilă de extragerea caracteristicilor din imaginea de intrare. Următoarea componentă, neck-ul, este o parte intermediară care se ocupă de rafinarea caracteristicilor extrase anterior. În final, head-ul reprezintă partea finală care este responsabilă pentru predicția finală, cum ar fi clasele obiectelor și coordonatele bounding box-urilor.

În acest stadiu, se definesc și se ajustează parametrii esențiali ai procesului de antrenare, precum arhitectura modelului, care în acest caz este YOLOv5n, dimensiunea batch-ului, setată la 16, learning rate-ul, setat la 0.01 și numărul de epoci, care este 10. Antrenarea modelului poate fi un proces care poate dura ore sau chiar zile, în funcție de mărimea setului de date și complexitate modelului. În cazul de față, setul de date este relativ mic, conținând doar o clasă (plăcuța de înmatriculare) pentru care se va face predicții. La finalul celei de a 10-a epocă, acuratețea a fost de 92,9% pe setul de validare și 100% pe setul de testare prezentat la finalul subcapitolului de reducere a zgomotului (3.1.3).

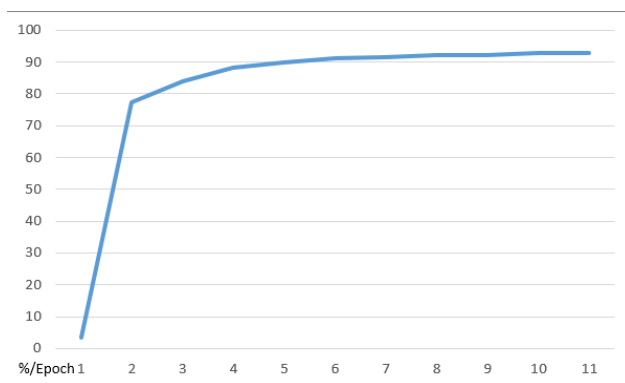


Figura 15: Acuratețea la începutul fiecărei epoci.

3.4.3 Inferența în C++

Odată ce modelul a fost antrenat și evaluat cu succes, acesta poate fi utilizat în aplicații pentru a face predicții în timp real. Astfel, inferența reprezintă procesul prin care un model antrenat va face predicții pe baza unor date noi de intrare, diferite față de cele din momentul antrenării. Pentru realizarea inferenței se va folosi modulul DNN din OpenCV, fiind astfel necesară convertirea modelului PyTorch rezultat în urma antrenării, în format ONNX, deoarece la momentul actual modelele antrenate folosind biblioteca YOLOv5 necesită convertirea la formatul ONNX.

Pentru realizarea pașilor din cadrul procesului de inferență se va folosi un obiect de tip Object-Detector ce oferă toate funcționalitățile necesare, de la încărcarea oricărui model în format ONNX, până la afișarea predicțiilor pe o imagine de intrare, oferind posibilitatea utilizării unui alt model decât cel antrenat anterior dar și a folosirii altor modele ce pot detecta orice număr de clase pentru eventuale dezvoltări ulterioare [19].

Primul pas constă în încărcarea arhitecturii și a ponderilor modelului dorit a fi folosit pentru detecția obiectelor. Acesta se poate realiza prin apelarea metodei `ChangeDetectionModel` din cadrul clasei `ObjectDetector`, la care se specifică detaliile necesare legate de modelul încărcat, ca de exemplu numele fișierului ONNX, dimensiunile imaginii de intrare, fișierul ce conține lista claselor și un confidence threshold, adică acuratețea minimă necesară pentru ca detecția să fie considerată validă.

După ce modelul a fost încărcat, următorul pas este prelucrarea imaginii de intrare ce se realizează prin intermediul metodei `PreProcess`, unde are loc scalarea și normalizarea imaginii în vederea pregătirii imaginii pentru a fi introdusă prin rețeaua neurală încărcată.

În final, au loc ultimii 2 pași, și anume: aplicarea inferenței și desenarea rezultatelor. Se va aplica inferența modelului asupra imaginii de intrare prelucrată anterior folosind metoda `PostProcess`, unde se va inițializa un vector ce va stoca informațiile legate de predicțiile obiectelor și se vor calcula factorii de scalare pentru redimensionarea predicțiilor la dimensiunile imaginii de intrare, urmând extragerea rezultatelor din matricea de ieșire a modelului. Matricea de ieșire conține informații precum coordonatele bounding box-urilor, clasele obiectelor ce au fost detectate și acuratețea predicțiilor rezultate. Predicțiile a căror acuratețe este mai mare sau egală cu valoarea

minimă oferită la încărcarea modelului (confidence threshold) sunt stocate în vectorul inițializat la început. Pentru selectarea celor mai bune predicții și eliminarea celor redundante se va aplica algoritmul Non-Maximum Suppresion. În cele din urmă, pentru fiecare predicție rezultată se poate apela metoda DrawLabel care se va folosi de coordonatele bounding box-ului pentru a desena un dreptunghi în jurul obiectului detectat, alături de clasa prezisă. De asemenea, pentru ușurință, în situația detectării plăcuțelor de înmatriculare, am decis ca plăcuța detectată să fie decupată și nu desenată. Dacă nu a putut fi detectată nicio plăcuță de înmatriculare, atunci se va încerca automat detecția acestora folosind metoda prezentată la început, denumită **IMAGE_PROCESSING** în cadrul proiectului.

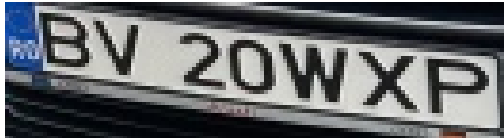


Figura 16: Plăcuța de înmatriculare extrasă folosind metoda IMAGE_PROCESSING (stânga); Plăcuța de înmatriculare extrasă folosind metoda DNN (dreapta).

3.5 Concluzie

Comparând cele două metode de detectare a plăcuței de înmatriculare, metoda DNN, bazată pe tehnologii de inteligență artificială, oferă o adaptabilitate mai mare la diferite condiții de iluminare și un grad mai mare de generalizare, fiind mai potrivită pentru aplicații ce necesită o acuratețe cât mai mare, pe când metoda IMAGE_PROCESSING este mai simplă și mai eficientă, fiind o opțiune bună pentru situațiile în care timpul de procesare este foarte important.

Având în vedere că camerele video destinate intrării și ieșirii sunt deseori plasate la același nivel cu plăcuțele de înmatriculare sau la nivelul tavanului, și la o distanță scurtă față de locul în care șoferii mașinilor opresc, de exemplu, pentru așteptarea ridicării unei bariere, acuratețea este la fel de bună pentru ambele metode. Dacă numărul de înmatriculare nu ar putea fi detectat, atunci, în cazul intrării, bariera se va deschide oricum și șoferul va primi un cod special, numit **SecretID**, pe care îl va folosi pentru a părăsi parcare. De asemenea, pentru aceste tipuri de camere video, timpul de procesare nu ar trebui să fie un factor de decizie deoarece diferența de timp dintre cele două metode este de 0.24 secunde și în cazul unei cozi la intrare/ieșire, durata înaintării mașinii spre camera video va dura cu siguranță cel puțin două, trei secunde. Totuși, la folosirea unei camere video de calitate scăzută sau la plasarea acesteia la o distanță mai mare față de mașină, metoda aleasă ar putea conta în procesul de detectare a plăcuței și de aceea tipul metodei folosite poate fi schimbat oricând din interiorul aplicației C++.

În concluzie, alegerea uneia dintre cele două metode depinde atât de condițiile și localizarea camerelor video, cât și de preferințele personale. Implicit, camerele video destinate intrării și ieșirii vor folosi metoda IMAGE_PROCESSING, fiind o metodă mai simplă și mai rapidă, iar camerele video destinate locurilor de parcare vor folosi metoda DNN, deoarece acestea sunt plasate la o distanță considerabilă, astfel încât să cuprindă mai multe locuri de parcare, și fiind verificate doar la un anumit interval de timp, pe rand. Timpul de procesare puțin crescut nu va fi oricum sesizat, operația fiind efectuată pe alt fir de execuție.

4 Recunoașterea caracterelor din plăcuța de înmatriculare

Recunoașterea caracterelor, sau OCR (Optical Character Recognition), reprezintă procesul de identificare și interpretare a textului din imagini sau alte forme de date vizuale. Acesta este utilizat într-o varietate de domenii, precum digitalizarea documentelor, scanarea codurilor de bare, recunoașterea plăcuțelor de înmatriculare și multe altele. Totuși, odată cu avansarea tehnologiei, algoritmi și tehnicile de recunoaștere a caracterelor sunt îmbunătățite pentru a fi din ce în ce mai precise și mai robuste, ajungând astfel să fie utilizate chiar și în aplicarea legii, ca de exemplu, aplicarea unei amenzi pentru depășirea limitei maxime de viteză admise.

Inițial, procesul începe prin prelucrarea imaginii care conține text în vederea îmbunătățirii calității și clarității acesteia. Aceasta poate include diverse operații precum, conversia la o imagine grayscale, eliminarea zgomotului, corectarea distorsiunii, etc. În continuare este necesar ca fiecare caracter din imagine să fie segmentat, aceasă etapă fiind crucială pentru izolarea fiecărui caracter și pregătirea acestuia pentru recunoaștere. După aceea, caracterele sunt extrase și pregătite pentru a fi recunoscute, identificându-se contururile și marginile acestora. În final, caracterele extrase sunt recunoscute și convertite într-un tip de date text (string) folosind diverși algoritmi de recunoaștere a caracterelor ce pot varia în funcție de complexitatea și domeniul aplicației, de la rețele neurale, până la template matching.

4.1 Preprocesarea imaginii

Înainte de extragerea caracterelor din imaginea de intrare, care este imaginea plăcuței de înmatriculare detectate anterior, sunt necesare diferite operații pentru îmbunătățirea acesteia. Operațiile sunt efectuate în ordinea următoare: conversia imaginii în grayscale, reducerea zgomotului folosind filtrul gaussian cu un kernel de dimensiune 7x7, egalizarea adaptivă a histogramei folosind algoritmul CLAHE [20], redimensionarea imaginii la 400 de pixeli în lățime și 125 pixeli în înălțime, aplicarea algoritmului de thresholding Otsu și în final, unul dintre cei doi algoritmi de corectare a distorsiunii ce vor fi prezentați.

În continuare se va prezenta egalizarea adaptivă a histogramei urmată de cei doi algoritmi de corectare a distorsiunii, denumiți Undistortioning și Skew Correction, celelalte operații fiind deja prezentate în capitolul 3.



Figura 17: Plăcuța de înmatriculare detectată (stânga); Plăcuța de înmatriculare după preprocesare, fără corectarea distorsiunii (dreapta).

4.1.1 Egalizarea adaptivă a histogramei

Histograma unei imagini este o reprezentare grafică a numărului de pixeli din imagine în funcție de intensitatea lor. Acestea sunt alcătuite din "bini", fiecare reprezentând un anumit interval de valori de intensitate. Calcularea histogramei se realizează prin parcurgerea tuturor pixelilor din imagine și atribuirea fiecăruia unui bin în funcție de intensitatea acestuia [?]. Astfel, valoarea unui bin reprezintă numărul de pixeli atribuiți acestuia. De exemplu, dacă primul bin (bin-ul 0) are o valoare de 2500, înseamnă că în imagine sunt 2500 de pixeli ce au valoarea de intensitate 0 [21].

Egalizarea constă în maparea unei distribuții, reprezentând histograma, pe o altă distribuție mai largă și mai uniformă a valorilor de intensitate, astfel încât histograma rezultată să fie cât mai uniformă posibil. Fie $P(k)$ probabilitatea de apariție a intensității k în imaginea inițială, iar y noua intensitate atribuită lui k după egalizarea histogramei. Pentru o imagine a cărei dimensiune totală este de n pixeli, funcția densității de probabilitate a intensității de lumină este:

$$P(k) = \frac{n_k}{n} \quad (9)$$

Pentru realizarea efectului de egalizare se va găsi o noua intensitate y pentru intensitatea inițială k folosind funcția de distribuție cumulativă definită astfel:

$$y = T(k) = \sum_{i=0}^k P(i) \quad (10)$$

În final, noua intensitate pentru fiecare pixel (x,y) din imaginea egalizată I' este dată de următoarea formulă:

$$I'(x, y) = T(I(x, y)) \quad (11)$$

Egalizarea adaptivă a histogramei reprezintă tehnica utilizată pentru a îmbunătăți contrastul în zone cu iluminare inegală. Deci, imaginea este împărțită în regiuni mai mici pe care se va aplica egalizarea histogramei în mod individual. Această tehnică este folosită în situațiile în care iluminarea unei imagini este inegală sau variază semnificativ, lucru ce poate fi întâlnit în scenarii precum imagistică medicală sau camerele de supraveghere, cum este cazul în aplicația responsabilă detectării și recunoașterii plăcuței de înmatriculare, prezentată în această lucrare.

Pentru realizarea acesteia am folosit algoritmul CLAHE (Contrast Limited Adaptive Histogram Equalization) din OpenCV. Imaginea este împărțită în zone mici numite "tiles", având un tileSize de 8x8, care este de asemenea și valoarea implicită în OpenCV. Ulterior, pentru fiecare zonă histograma intensităților pixelilor este calculată și apoi egalizată. Totuși, pentru a evita amplificarea zgomotului, se aplică o tehnică de limitare a contrastului, însemnând că dacă un bin din histograma unei zone depășește o anumită limită de contrast, pixelii corespunzători sunt redistribuiți uniform

în alte bin-uri înainte de aplicarea egalizării pe histogramă. După aceea, se aplică interpolarea biliniară în jurul marginilor zonelor pentru a asigura o tranziție mai lină între ele. În figura 18 se poate observa egalizarea imaginii unei plăcuțe de înmatriculare rezultate în urma detecției sale dintr-o imagine.



Figura 18: Plăcuța de înmatriculare detectată (stânga); Plăcuța de înmatriculare după aplicarea algoritmului CLAHE (dreapta).

4.1.2 Undistortion

Distorsiunile geometrice sunt deformări nedorite a unei imagini care apar ca rezultat al proiecției tridimensionale a obiectelor pe un plan bidimensional, fiind cauzate de faptul că obiectelor pot părea diferite în imaginea bidimensională față de cum arată în realitate, din cauza unghiurilor de vizualizare și a perspectivei [22]. Există mai multe tipuri de distorsiuni geometrice, dintre care cele mai comune sunt:

- Distorsiuni de înclinare - reprezintă distorsiunile cauzate de obiectele ce sunt fotografiate sub un anumit unghi, ducând astfel la apariția unei distorsiuni în formă de înclinare a acestora. În cazul plăcuțelor de înmatriculare, acesta este tipul de distorsiune geometrică care va fi corectat, deoarece camera video se poate afla în partea stângă sau dreaptă a mașinii, la un nivel ridicat față de plăcuță, astfel cauzând acest tip de distorsiune.
- Distorsiuni de scalare - reprezintă distorsiunile ce apar atunci când obiectele dintr-o imagine sunt reduse sau mărite într-un anumit raport, ceea ce poate distorsiona proporțiile lor. De exemplu, o imagine a unei mașini care este redimensionată în lățime poate duce la alungirea sau comprimarea acesteia.
- Distorsiuni de înălțime și lățime - reprezintă distorsiunile ce apar atunci când obiectele sunt văzute din unghiuri diferite. De exemplu, dacă un obiect este fotografiat sub un unghi lateral poate părea mai lung sau mai lat în direcția în care este fotografiat, în funcție de unghiul de observare.

În vederea recunoașterii caracterelor dintr-o plăcuță de înmatriculare, corectarea distorsiunilor geometrice este foarte importantă deoarece asigură faptul că caracterele din plăcuță sunt reprezentate în mod uniform. În mare parte, algoritmul ce urmează a fi prezentat se ocupă de corectarea înclinației și a rotației, eliminând astfel distorsiunile ce pot apărea din cauza unghiurilor de filmare neregulate, și este denumit în aplicație: **Undistortion**.

Corectarea acestei distorsiuni se va realiza folosind un proces matematic utilizat pentru a modifica poziția și orientarea unei imagini în funcție de un punct de vedere specific sau de o perspectivă observată, numit transformare de perspectivă. Mai precis, acesta implică modificarea geometrică a imaginii astfel încât să reflecte corect apariția obiectelor atunci când acestea sunt privite dintr-un

unghi sau o poziție specifică. Această transformare implică calculul unei matrice de transformare H , care să reprezinte relația între poziția originală a pixelilor din imagine și noua poziție dorită în imaginea rezultată. Elementele matricei H pot fi calculate cu ajutorul algoritmului de transformare liniară directă (DLT). Totuși, pentru a determina elementele matricei folosind algoritmul DLT, sunt necesare cel puțin cinci perechi de puncte corespunzătoare, pe când în aceasta lucrare sunt folosite patru puncte [23]. Astfel, matricea de transformare H se calculează cu ajutorul unor puncte de origine, care sunt cele patru colțuri ale plăcii de înmatriculare, și cu puncte de destinație, care sunt pozițiile dorite ale acestor colțuri în imaginea transformată. O transformare liniară este exprimată prin ecuația:

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = H \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \quad (12)$$

În primul rând, pentru găsirea punctelor de origine și de destinație, rezoluția imaginii plăcuței de înmatriculare este redimensionată la o dimensiune fixă, setată la 400 pixeli în lățime și 125 pixeli în înălțime pe care se va aplica operația de thresholding folosind metoda Otsu, urmând a se obține imaginea pregătită pentru corectarea distorsiunii, ca în figura 17.

Pentru găsirea punctelor de origine, adică cele patru colțuri ale plăcuței de înmatriculare se va clona imaginea preprocesată până în acest punct iar pe aceasta se va căuta cel mai mare contur, care va fi conturul plăcuței de înmatriculare. Odată găsit conturul, se va colora cu alb interiorul acestuia, eliminând astfel caracterele din interiorul plăcii deoarece sunt irelevante în găsirea colțurilor plăcuței. De asemenea, se va efectua și o operație morfologică de opening în care erodarea va folosi un kernel de dimensiune 7x7 iar dilatarea un kernel de dimensiune 5x5.

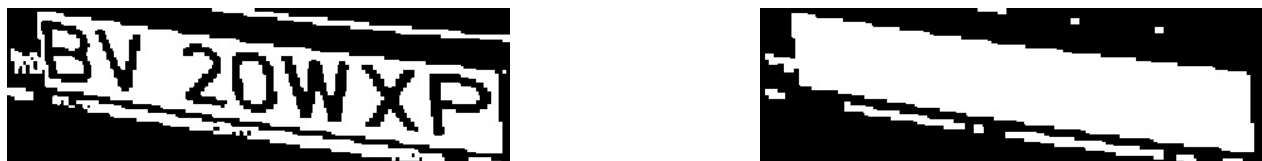


Figura 19: Plăcuța de înmatriculare preprocesată (stânga); Plăcuța de înmatriculare după eliminarea caracterelor și opening (dreapta)

În continuare, se vor identifica toate colțurile plăcuței de înmatriculare prin obținerea unui poligon simplificat care aproximează forma acesteia. Acest lucru se realizează cu ajutorul funcției `approxPolyDP` din biblioteca OpenCV, iar valorile poligonului rezultat reprezintă colțurile plăcuței. Toate aceste coordonate ale colțurilor sunt ulterior sortate descrescător în funcție de coordonata lor pe axa X. După sortare, primele două colțuri din lista vor fi cele din partea dreaptă a plăcuței. Deși ultimele două colțuri din lista ar reprezenta colțurile din partea stângă, din cauza literelor prezente pentru identificarea țării de origine a plăcuței, este posibil ca după operația de opening să existe în continuare un gol în această zonă, asemănător ca în figura 19, ducând astfel identificarea unor colțuri suplimentare în interiorul acesteia. Pentru rezolvarea acestei probleme, se vor parcurge toate colțurile identificate din partea stângă și se vor găsi colțurile a cărei coordonată pe axă

Y este maximă, respectiv minimă. Pentru a decide care dintre colțuri se află în partea de sus și care se află în partea de jos se va face o verificare pe baza coordonatelor de pe axa Y. De exemplu, dacă coordonata Y a unui colț c1 este mai mică decât coordonata Y a altui colț c2, ambele aflându-se pe aceeași parte, atunci colțul c1 va fi poziționat stânga sus iar c2 va fi poziționat stânga jos. Atât punctele de origine finale, cât și cele de destinație, vor fi adăugate într-o listă în următoarea ordine: stânga sus, dreapta sus, stânga jos, dreapta jos. În figura 20 se pot observa toate colțurile identificate, colțurile selectate pentru partea stângă sunt colorate cu verde, pentru partea dreaptă sunt colorate cu roșu, și cu albastru sunt colorate restul colțurilor ce au fost identificate, dar nu și selectate.



Figura 20: Colțurile identificate în plăcuța de înmatriculare

Este important să obținem aceste colțuri cu precizie, deoarece orice eroare în identificarea lor va afecta negativ transformarea și ar putea duce la distorsiuni în imaginea rezultată. Pe baza conținutului plăcuței se va găsi și dreptunghiul rotit de cea mai mică suprafață care îl încadrează iar cu ajutorul acestuia se va decide, pe baza unghiului de rotație, dacă este necesară transformarea sau nu.

Punctele de destinație sunt stabilite astfel încât imaginea plăcuței de înmatriculare rezultată din corectarea distorsiunii să corespundă unei forme geometrice dreptunghiulare orientate frontal. Astfel, coordonatele fiecărui colț al dreptunghiului este determinat în funcție de dimensiunea imaginii $W \times H$, unde W reprezintă lățimea și H reprezintă înălțimea și care implicit are dimensiunea 400×125 , însă constanta poate fi modificată din interiorul codului, din clasa `Utils`. Formula pentru determinarea coordonatelor punctelor de destinație este:

$$\begin{pmatrix} (x, y)_{\text{TopLeft}} \\ (x, y)_{\text{TopRight}} \\ (x, y)_{\text{BottomLeft}} \\ (x, y)_{\text{BottomRight}} \end{pmatrix} = \begin{pmatrix} (\frac{W}{8}, \frac{H}{3}) \\ (W - \frac{H}{3}, \frac{H}{3}) \\ (\frac{W}{8}, \frac{2 \cdot H}{3}) \\ (W - \frac{H}{3}, \frac{2 \cdot H}{3}) \end{pmatrix} \quad (13)$$

Deoarece au fost aflate toate punctele necesare pentru calculul matricei de transformare, se dorește păstrarea doar a caracterelor din interiorul plăcuței și realizarea transformării pe o imagine ce le include doar pe acestea. Având deja atât punctele de origine cât și cele de destinație, faptul că se exclud restul detaliilor din imagine este neimportant, punctele având oricum aceleași coordonate. Astfel, pentru a păstra doar caracterele din imagine se va realiza o operație în care se compară valoarea pixelilor din imaginea plăcuței de înmatriculare ce nu include caracterele, notată cu A , cu imaginea plăcuței de înmatriculare originală, rezultată în urma preprocesării, notată cu B . Scopul este să avem pixelii corespunzători caracterelor colorați cu negru, adică valoarea pixelilor să

fie 0, și ceilalți pixeli să fie colorați cu alb, adică valoarea pixelilor să fie 255. Pixelii ce au deja valoarea 255 vor rămâne la fel, cei a căror valoare este egală în ambele imagini vor lua valoarea 255, iar pixelii a căror valoare este 255 (alb) în imaginea A dar este 0 (negru) în imaginea B, vor lua valoarea 0 (negru), obținând astfel o imaginea ce va conține doar caracterele, cu eventualitatea aparițiilor a unor pixeli de valoare 0 ce nu corespund unui caracter. Aceștia vor fi eliminați după corectarea distorsiunii, în capitolul 4.2. Mai precis, pixelii vor fi modificați utilizând următoarea formulă:

$$\begin{cases} 0, & A(x, y) = 255 \text{ și } B(x, y) = 0 \\ 1, & \text{altfel} \end{cases} \quad (14)$$



Figura 21: Plăcuța de înmatriculare după eliminarea caracterelor și opening (stânga); Plăcuța de înmatriculare originală, preprocesată (mijloc); Caracterele din interiorul plăcuței (dreapta)

Transformarea de perspectivă se va realiza pe imaginea obținută, ce conține doar caracterele imaginii și utilizând matricea de transformare H calculată cu ajutorul punctelor de origine și destinație găsite. În figura 22 se poate observa corectarea distorsiunii geometrice, caracterele nefiind înclinate.



Figura 22: Caracterele din interiorul plăcuței (stânga); Caracterele din interiorul plăcuței după corectarea distorsiunii folosind Undistortion (dreapta)

4.1.3 Skew Correction

Asemănător cu algoritmul prezentat anterior, și **SkewCorrection** (corectarea înclinării) este tot un proces în prelucrarea imaginilor ce se ocupa cu corectarea oricărei înclinări sau distorsiuni dintr-o imagine.

Acest proces constă în detecția unghiului de rotație cu ajutorul dreptunghiului rotit de cea mai mică suprafață care încadrează conturul maxim găsit în imaginea plăcuței de înmatriculare. După obținerea unghiului, se verifică dacă imaginea are nevoie de aplicarea algoritmului Skew Correction cu ajutorul valorii unghiului. Dacă unghiul este mai mare de 85 de grade sau mai mic de 5 grade, atunci imaginea plăcuței nu necesită aplicarea acestui pas, imaginea fiind aproape, sau chiar deloc înclinată.

În continuare, se va calcula matricea de rotație specificând centrul rotației, unghiul de rotație, astfel încât să fie între -45 și 45 de grade, și un factor de scalare care este 1.0 în acest caz. Matricea de rotație este un tip specific de matrice de transformare ce descrie o rotație aplicată unei imagini sau a unui obiect geometric din jurul unui anumit punct sau axă. Într-un spațiu bidimensional, matricea de rotație este de dimensiune 2x2 și este definită în funcție de unghiul de rotație. O matrice de rotație pentru unghiul specific de rotație θ este definită astfel:

$$M = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \quad (15)$$

În final, se aplică transformarea afină folosind funcția `warpAffine` din biblioteca OpenCV cu matricea de rotație calculată, pentru a roti imaginea ce conține caracterele din interiorul plăcuței de înmatriculare. Pentru obținerea imaginii ce conține doar caracterele plăcuței, imaginea originală a acesteia este clonată și colorată cu alb pe suprafața celui mai mare contur găsit. După aceea, se va modifica fiecare pixel dintre imaginea originală și imaginea obținută anterior folosind formula prezentată în ecuația 14. După rotire se pot crea spații goale în jurul imaginii iar pentru acest lucru se folosește interpolarea cubică. Aceasta este folosită pentru estimarea valorilor pixelilor din spațiile goale. În figura 23 se poate observa corectarea distorsiunii folosind tehnica prezentată.



Figura 23: Caracterele din interiorul plăcuței (stânga); Caracterele din interiorul plăcuței după corectarea distorsiunii folosind `SkewCorrection` (dreapta)

4.1.4 Rezultatele celor două metode de corectare a distorsiunii

Încă de la prima vedere se poate observa faptul că algoritmul `SkewCorrection` a obținut o înclinare dreaptă a textului din interiorul plăcuței de înmatriculare, dar nu și a caracterelor, acestea fiind puțin înclinate spre stânga, așa cum se poate observa în figura 23. Acest lucru se poate întâmpla și în cazul metodei `Undistortion` atunci când cele 4 colțuri ale plăcuței nu sunt alese cu precizie, însă alegerea acestora poate fi îmbunătățită, pe când găsirea unui contur mai precis în cazul metodei `SkewCorrection` va duce la același unghi de rotație și prin urmare, la aceeași imagine rezultată. De asemenea, în urma aplicării ambelor metode pe fiecare imagine din setul de testare, s-a constatat că metoda `SkewCorrection` corectează înclinarea textului pe aproape fiecare imagine însă din cauza înclinării caracterelor, acestea nu pot fi recunoscute cu succes în pasul următor, pe când metoda `Undistortion`, deși nu corectează distorsiunea geometrică cu o acuratețe la fel de mare, atunci când vine vorba de recunoașterea caracterelor, această metodă câștigă detașat, având o acuratețe de peste două ori mai mare în recunoașterea întregului text de pe plăcuțele de înmatriculare.

Metoda `Undistortion` este adesea considerată o abordare superioară în comparație cu `SkewCorrection` în contextul corectării distorsiunilor dintr-o imagine. Metoda `Undistortion` aplică trans-

formări complexe pentru corectarea distorsiunilor geometrice, astfel încât obiectele să apară cu forme și proporții corecte în imagine, acest lucru fiind esențial în contextul aplicațiilor ce necesită o precizie ridicată, precum recunoașterea caracterelor. În schimb, metoda SkewCorrection se concentrează pe ajustarea orientării obiectelor din imagine, lucru ce poate fi suficient în unele aplicații, dar nu corectează complet distorsiunile geometrice. Prin urmare, metoda folosită în continuare pentru recunoașterea caracterelor este **Undistortion**. În figura 24 se observă rezultatele ambelor metode pe diferite imagini a unor plăcuțe de înmatriculare.

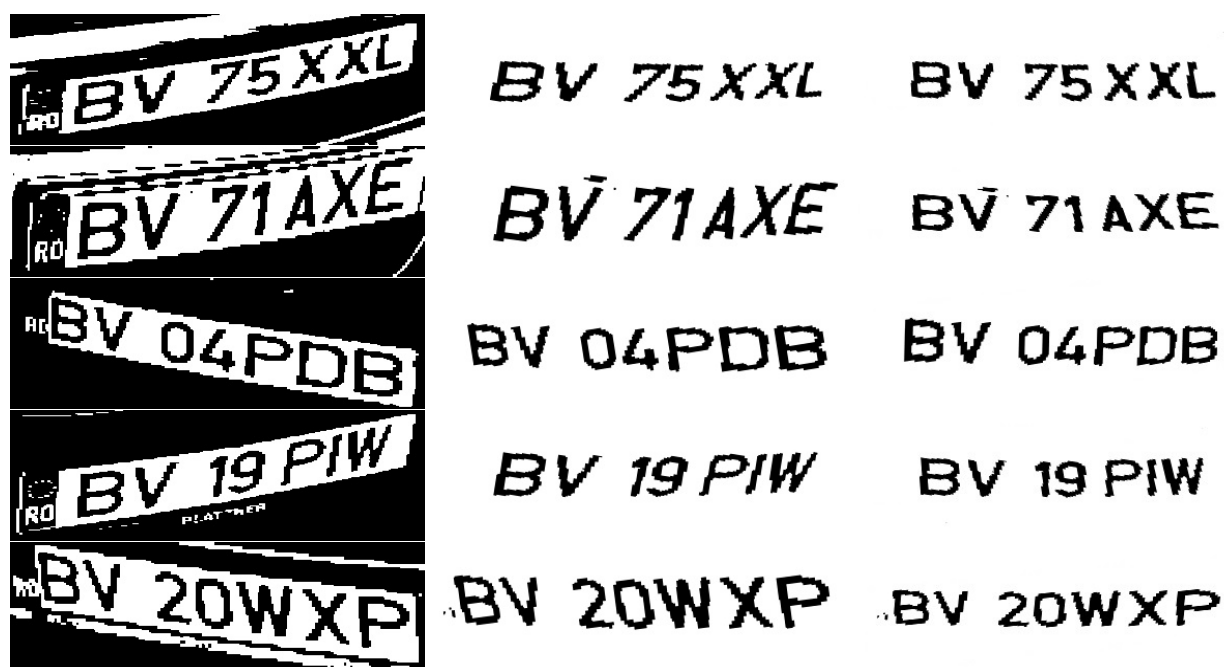


Figura 24: Plăcuța de înmatriculare (stânga); Corectarea distorsiunii folosind SkewCorrection (mijloc); Corectarea distorsiunii folosind Undistortion (dreapta)

4.2 Detectarea caracterelor

Recunoașterea caracterelor din interiorul plăcuței nu se va face utilizând întreg textul, ci pe fiecare caracter în parte. Așadar, este necesar ca fiecare caracter să fie detectat și salvat pentru aplicarea algoritmului de recunoaștere a acestuia. Având deja o imagine cu fundal alb ce conține tot interiorul plăcuței, detectarea caracterelor se face foarte simplu, găsind toate contururile din această imagine și selectarea celor relevante. Tehnica pentru extragerea contururilor este aceeași ca cea prezentată în capitolul 3.2, unde s-a realizat detectarea plăcuței de înmatriculare.

Totuși, este important de evidențiat că nu toate contururile găsite reprezintă un caracter valid. Unele dintre aceste contururi pot fi rezultatul zgomotului sau a artefactelor din imagine, iar altele ar putea fi segmente din caracterele deja existente, de exemplu, pentru litera "O" ar putea exista 2 contururi diferite, unul fiind litera în sine, iar celălalt fiind interiorul acesteia, rezultând astfel în două litere "o" recunoscute. De asemenea, în România, în urma finalizării cu succes a inspecției tehnice periodice (ITP) la mașină, unii șoferi primesc un abțibild de dimensiune mică și rotund pe care ajung să îl lipească pe interiorul plăcuței de înmatriculare. În consecință, este necesară

aplicarea unui proces de filtrare și validare a conturilor găsite pentru a asigura că doar caracterele reale sunt recunoscute și interpretate corect.

Procesul de filtrare și validare implică plasarea unui dreptunghi în jurul fiecărui contur, delimitând astfel o regiune de interes care ar putea include un caracter. Acest dreptunghi care înconjoară conturul este folosit pentru izolarea și extragerea caracterului corespunzător din imaginea originală, adică imaginea rezultată în urma corectării distorsiunilor. Pentru a decide dacă un contur este valid sau nu, acesta trebuie să îndeplinească următoarele condiții: înălțimea dreptunghiului este mai mare decât 20% din înălțimea imaginii originale și lățimea dreptunghiului este mai mică decât jumătate din lățimea imaginii originale, iar aria conturului să fie cuprinsă între 75 și 4000. În continuare, pentru fiecare contur valid se va salva imaginea acestuia, rezultată în urma decupării din imaginea originală folosind dreptunghiul acestuia iar acestea vor fi plasate într-un vector ce conține perechi de tipul `cv::Mat, cv::Rect`, unde `cv::Mat` reprezintă imaginea caracterului și `cv::Rect` reprezintă dreptunghiul ce îl înconjoară. Salvarea dreptunghiului este necesară pentru sortarea conturilor după coordonata axei X, astfel încât să se obțină ordinea corectă a caracterelor din textul plăcuței. După sortare, fiecare contur valid este parcurs și analizat pentru a se asigura faptul că acesta nu face parte din interiorul altui contur, așa cum este în exemplul prezentat mai sus în cazul literei "O".

În final, după ce conturile sunt filtrate și validate, se consideră că aplicarea algoritmului pentru recunoașterea caracterelor îndeplinește toate condițiile necesare, asigurând că doar datele relevante vor fi transmise. Atunci când algoritmul va parcurge conturile valide, acesta va porni de la primul caracter, adică cel mai din stânga, până la ultimul caracter, adică cel mai din dreapta, deoarece caracterele au fost sortate după localizarea acestora. De asemenea, este încă posibil ca unele conturi să nu reprezinte un caracter valid, însă algoritmul pentru recunoașterea acestora va elimina la rândul său alte date irelevante.

4.3 Template Matching

Template matching reprezintă o metodă de căutare și găsim a locației unei imagini șablon într-o imagine mai mare. Aceasta glisează imaginea șablon peste imaginea de intrare, asemănător ca în convoluția 2D, și compară șablonul și fragmentul de imagine de intrare sub imaginea șablon. În final, rezultatul este o imagine grayscale în care fiecare pixel indică cât de mult se aseamănă vecinătatea aceluia pixel cu șablonul. Dacă imaginea de intrare are dimensiunea $W \times H$ și imaginea șablon are dimensiunea $w \times h$, imaginea de ieșire va avea dimensiunea $W-w+1, H-h+1$ [24]. După obținerea rezultatului, se va afla valoarea maximă care va determina care dintre caractere seamănă cel mai mult cu imaginea de intrare, care în cazul recunoașterii plăcuțelor de înmatriculare, va fi imaginea unui caracter din textul acesteia.

Pentru început, se va alege ca imagine de intrare, o imagine care conține litera sau cifra care urmează să fie recunoscută. În recunoașterea caracterelor de pe plăcuța de înmatriculare, imaginea reprezintă o anumită literă sau cifră din interiorul acesteia. Având deja o listă sortată cu toate caracterele plăcuței, se va considera, pe rând, fiecare element ca fiind imaginea de intrare, urmând a se executa algoritmul pentru fiecare dintre caracterele acesteia. Aceste imagini de intrare sunt comparate mai departe cu o altă listă de imagini, predefinite, numite imagini template, care

conțin toate caracterele și cifrele posibile pe plăcuțele din România [25]. De asemenea, acest set de imagini poate fi modificat astfel încât să conțină și alte caractere dintr-un alt alfabet prin simpla adăugare/ștergere a unor imagini în folderul specific acestuia (CharTemplates) și în fișierul text numit "charTemplates.txt" din interiorul folderului Resources, unde se află lista imaginilor dorite. Denumirea imaginilor trebuie să înceapă neapărat cu litera care se află în imaginea template.



Figura 25: Toate caracterele posibile pe o plăcuță din România.

Astfel, algoritmul de template matching calculează un grad de asemănare între diferite regiuni ale fiecărei imagini de intrare, adică imaginile caracterelor, și imaginile template. Pentru măsurarea similarității se pot folosi mai multe metode, cum ar fi:

- Diferența absolută medie (MAD): Calculează media diferenței absolute dintre pixelii din șablon și cei din regiunea corespunzătoare a imaginii de intrare.
- Diferența pătratică medie (MSE): Calculează media pătratelor diferențelor dintre pixelii din șablon și cei din regiunea corespunzătoare a imaginii de intrare.
- Coeficientul de corelație: Măsoară gradul de corelație între intensitățile pixelilor din șablon și cele din regiunea corespunzătoare a imaginii de intrare.

Pentru măsurarea similarității în cazul recunoașterii de caractere, se va folosi metrica coeficientul de corelație normalizat, având următoarea formulă, unde A și B sunt matricile de pixeli corespunzătoare imaginii template și a regiunii imaginii de intrare, $A'(x,y)$ și $B'(x,y)$ sunt valorile normalizate ale intensităților pixelilor din matricele A și B , \bar{A} și \bar{B} reprezintă mediile pixelilor din matricile A și B [24].

$$R(A, B) = \frac{\sum_{x,y} (A'(x, y) - \bar{A})(B'(x, y) - \bar{B})}{\sqrt{\sum_{x,y} (A'(x, y) - \bar{A})^2 \sum_{x,y} (B'(x, y) - \bar{B})^2}} \quad (16)$$

Pentru identificarea unui caracter specific dintr-o imagine, se va aplica algoritmul de template matching pentru imaginea caracterului respectiv pe toate imaginile template ale tuturor caracterelor posibile ce pot fi pe o plăcuță de înmatriculare din Romania. După aceea, caracterul ce prezintă cea mai mare asemănare cu imaginea de intrare va fi considerat, cu excepția cazului în care valoarea maximă a asemănării nu depășește 0.40 (40%), caz în care imaginea de intrare va fi ignorată și niciun caracter nu va fi returnat.

BV 71 AXE

(a) Text: BV 71 AXE

BV 24 NSC

(b) Text: 8V 24 NSC

BV 75 XXL

(c) Text: BV 75 XXL

Figura 26: Rezultatele recunoașterii pe diferite plăcuțe.

În final, algoritmul prezentat s-a dovedit a fi eficient și precis în recunoașterea caracterelor de pe plăcuțele de înmatriculare ale mașinilor, obținând o acuratețe finală de 89.30% pe caracterele recunoscute, și de 74.46% pe numerele de înmatriculare recunoscute în totalitate, majoritatea greșelilor fiind pentru caracterul "B", care este confundat uneori cu caracterul "8", așa cum se poate observa și în figura 26 (b), caracterul "1", care este confundat cu caracterul "l", și caracterul "O", care este confundat cu caracterul "o". Totuși, în aproape toate cazurile în care au avut loc aceste greșeli, s-a observat că distorsiunile geometrice nu au fost corectate în totalitate iar unele caractere sunt în continuare mai mult sau mai puțin înclinate, asemănător figurii 26 (b).

5 Modelarea entităților

Modelarea entităților reprezintă o etapă fundamentală în dezvoltarea unei aplicații software ce implică definirea și organizarea structurilor de date necesare pentru funcționarea corectă a aplicației. Primul pas în acest proces de modelare a entităților este identificarea entităților principale ale sistemului, fie că acestea sunt obiecte, concepte sau componente semnificative. De exemplu, în aplicațiile ce compun sistemul de monitorizare și gestionare a unei parări, entitățile importante sunt "Camera", "Session", "ParkingSession" și "Users"

Odată ce entitățile au fost identificate, următorul pas constă în definirea atributelor asociate fiecărei entități. Atributele reprezintă informațiile specifice pe care entitatea le are în posesie. Pentru entitatea "Users", de exemplu, atributele pot include un număr de identificare (ID), numărul de înmatriculare (username) și o parolă.

După definirea atributelor, urmează definirea relațiilor dintre entități. Acestea reprezintă modul în care o entitate este conectată și interacționează cu o altă entitate. În cazul sistemului de monitorizare și gestiune a unei parări, o relație importantă este între "Session" și "ParkingSession", însemnând că într-o sesiune principală pot exista mai multe "sesiuni de parcare", mai exact, un șofer poate parca pe mai multe locuri de parcare (ParkingSession) în cadrul unei singure sesiuni (Session).

Procesul de modelare a entităților este adesea iterativ, iar acesta poate suferi modificări pe parcurs, în funcție de evoluția cerințelor și a specificațiilor. Prin urmare, este necesar ca acesta să fie deschis la ajustări.

În continuare vor fi prezentate toate entitățile și atributele sale, alături de o descriere detaliată a acestora. Lista entităților este următoarea: "Camera", "CameraType", "CameraKey", "Session", "ParkingSession", "ParkingSpace", "Users" și "Reports"

5.1 Entitatea "Camera"

Entitatea "Camera" reprezintă tabela în care se stochează toate informațiile legate despre o cameră video a parării. Aceste camere video sunt amplasate strategic în diverse locații din parcare auto, precum intrarea, ieșirea și alte zone din parcare de unde se poate observa mișcarea autovehiculelor ce parchează pe anumite locuri de parcare, pentru a captura imagini ale acestora în vederea detectării și recunoașterii numerelor de înmatriculare ale autovehiculelor.

Nume câmp	Tip câmp	Descriere
id	int	Numărul unic de identificare a camerei video în baza de date.
camera_type_id	int	Numărul unic de identificare a tipului dorit pentru camera video.
location	String	Path-ul către camera video sau index-ul acesteia.
name	String	Numele atribuit camerei video.

Tabela 1: Entitatea "Camera"

Pentru adăugarea unei înregistrări tabeli "Camera", vor fi necesare doar ultimele trei atribute,

"camera_type_id", "location" și "name". Primul atribut, ID, se va autogenera. Cel de-al doilea atribut, "camera_type_id", va fi un număr între 1 și 3 ce va reprezenta tipul camerei dorit, mai exact, 1 pentru tipul ENTRANCE, 2 pentru tipul EXIT și 3 pentru tipul PARKING. Atributul "location" poate fi atât path-ul către device-ul sau video-ul ce reprezintă o cameră video (de exemplu, "/dev/video0"), sau index-ul acesteia (de exemplu, "0"). Atunci când vor fi citite datele camerelor video în aplicație, se va verifica dacă "location" este un path sau un index. În final, atributul "name" reprezintă numele atribuit camerei video și deși nu este obligatoriu să fie unic, se recomandă totuși evitarea numelor duplicate pentru evitarea greșelilor în cazul modificării acestora. Dacă se merge pentru varianta numelor duplicate, atunci modificarea camerelor video va necesita o atenție sporită, utilizatorul asigurându-se că modifică camera corectă pe baza ID-ului acesteia.

5.2 Entitatea "CameraType"

Entitatea "CameraType" reprezintă o tabelă ce stochează toate tipurile existente ce pot fi atribuite unei camere video. Aceasta tabelă este modificată, validată și corectată automat la rularea aplicației principale, implementată în C++.

Nume câmp	Tip câmp	Descriere
id	int	Numărul unic de identificare a unui tip de cameră în baza de date.
type	String	Numele tipului de cameră video.

Tabela 2: Entitatea "CameraType"

Astfel, la rularea aplicației se vor verifica toate înregistrările din această tabelă iar în cazul în care acestea corespund celor implicite, ele vor fi șterse și adăugate din nou.

Dacă dintr-un motivul sau altul se dorește adăugarea unui nou tip de cameră video, atunci va fi necesară modificarea acestei verificări la rularea aplicației și a adăugării unei noi valori în obiectul de tip enum ce reprezintă tipul camerei video. Acest enum se află în fișierul header destinat acestei entități ("CameraType.h") aflat în cadrul componentei Model al arhitecturii MVC, urmată de layer-ul "Entities" (Model) al arhitecturii pe trei nivele pentru baza de date. Mai departe, se va adăuga funcționalitatea dorită la declanșarea acțiunii pentru acest tip de cameră video în clasa ActionManagement. Acțiunea reprezintă felul în care imaginea capturată de la camera video și numărul de înmatriculare recunoscut sunt manipulate în funcție de tipul camerei video.

id	type
1	ENTRANCE
2	EXIT
3	PARKING

Tabela 3: Înregistrările implicite ale entității "CameraType"

5.3 Entitatea "CameraKey"

Entitatea "CameraKey" reprezintă tabela în care sunt stocate tastele folosite pentru declanșarea acțiunii unei camere video, ca de exemplu, la intrarea în parcare unde camera video va prelua

un frame pe care îl va folosi pentru detectarea și recunoașterea numărului de înmatriculare al autovehiculului prezent. O tasta (key) poate fi orice literă de la A la Z, reprezentată ca valoare numerică în cod ASCII, și este unică pentru fiecare cameră video în parte.

Nume câmp	Tip câmp	Descriere
camera_id	int	Numărul unic de identificare a unei camerei video.
key	int	Valoarea numerică asociată tastei alese în codul ASCII.

Tabela 4: Entitatea "CameraKey"

5.4 Entitatea "Session"

Entitatea "Session" reprezintă tabela în care sunt stocate toate sesiunile șoferilor. Sesiunea în acest context se referă la intervalul de timp în care un șofer utilizează parcare. Această sesiune începe în momentul în care șoferul intră în parcare și se încheie atunci când acesta o părăsește.

O înregistrare a unei sesiuni se crează atunci când un șofer va acționa un buton sau o tastă atribuită camerei video a intrării respective. Dacă numărul de înmatriculare al autovehiculului este recunoscut cu succes, atunci acesta va fi inclus în atributul "license_plate" al sesiunii. Pe lângă numărul de înmatriculare și numărul unic de identificare al sesiunii ("id"), va exista un alt număr unic de identificare al sesiunii, mai complex, numit SecretID, ce este compus dintr-un șir de numere, urmat de text-ul plăcuței de înmatriculare ce a fost recunoscut. Acest "SecretID" va fi folosit pentru crearea unui cont de utilizator pe aplicația web și pentru ieșirea forțată din parcare pentru cazul în care numărul de înmatriculare nu a putut fi detectat la intrare sau a fost detectat la intrare, dar nu mai poate fi detectat la ieșire.

Nume câmp	Tip câmp	Descriere
id	int	Numărul unic de identificare al sesiunii.
license_plate	String	Numărul de înmatriculare detectat.
entrance_time	timestamp	Data și ora începerii sesiunii.
exit_time	timestamp	Data și ora încheierii sesiunii.
secret_id	String	Numărul special de identificare al sesiunii, oferit șoferilor la intrare.

Tabela 5: Entitatea "Session"

5.5 Entitatea "ParkingSpace"

Entitatea "ParkingSpace" este responsabilă pentru stocarea informațiilor pentru fiecare loc de parcare definit, pe care să se realizeze detecția numărului de înmatriculare. Se consideră că o cameră video de tip PARKING este statică și astfel se pot crea zone delimitatoare pentru fiecare loc de parcare disponibil din câmpul vizual al camerei video respective. Având această zonă delimitatoare pentru un loc de parcare dar și un nume atribuit acestuia (cum ar fi, M23, unde M reprezintă rândul și 23 locul de parcare de pe rândul respectiv), se poate realiza detecția pe această zonă iar ca urmare, șoferii vor putea vedea în cadrul aplicației web, locul de parcare pe care și-au parcat autovehiculul. De asemenea, cu ajutorul acestei entități se pot realiza și diferite statistici precum gradul de ocupare al parării, locurile de parcare cele mai des ocupate și altele.

Nume câmp	Tip câmp	Descriere
id	int	Numărul unic de identificare al locului de parcare.
camera_id	int	Numărul unic de identificare a unei camere video.
name	String	Numele atribuit locului de parcare.
x1	int	Coordonata X a primului punct ce reprezintă zona delimitatoare.
y1	int	Coordonata Y a primului punct ce reprezintă zona delimitatoare.
x2	int	Coordonata X a celui de-al doilea punct ce reprezintă zona delimitatoare.
y2	int	Coordonata Y a celui de-al doilea punct ce reprezintă zona delimitatoare.

Tabela 6: Entitatea "ParkingSpace"

5.6 Entitatea "ParkingSession"

Entitatea "**ParkingSession**" este asemănătoare entității "Session", ambele referindu-se la un interval de timp în care șoferul utilizează un lucru anume. În cazul entității "Session", aceasta se referea la intervalul de timp în care conducătorul autovehiculului utilizează parcare, pe când în cazul entității "**ParkingSession**", aceasta se referă la intervalul de timp în care un autovehicul a cărui număr a fost detectat pe unul din locurile de parcare definite se află parcat pe acesta.

Nume câmp	Tip câmp	Descriere
id	int	Numărul unic de identificare a sesiunii de parcare.
session_id	int	Numărul unic de identificare a unei sesiuni.
parking_space_id	int	Numărul unic de identificare a unui loc de parcare.
start_time	timestamp	Data și ora la care a fost detectat autovehiculul pe locul de parcare.
end_time	timestamp	Data și ora la care autovehiculul a părăsit locul de parcare.

Tabela 7: Entitatea "ParkingSpace"

Atunci când un număr de înmatriculare este detectat pe unul din locurile de parcare, se verifică să nu existe deja o sesiune de parcare (ParkingSession) pentru acest loc, și dacă nu există, se consideră ca autovehiculul abia a parcat și se creează o sesiune de parcare nouă. De asemenea, dacă pentru numărul de înmatriculare respectiv a existat o sesiune de parcare dar pentru alt loc, atunci sesiunea trecută va fi considerată încheiată. Sesiunea de parcare nouă va include informații precum sesiunea curentă a numărului de înmatriculare, data și ora începerii și încetării staționării și locul de parcare pe care a fost detectat și va fi creată doar dacă se găsește o sesiune principală (Session) pentru numărul de înmatriculare detectat.

5.7 Entitatea "Users"

Entitatea "Users" este destinată doar aplicației web și reprezintă un cont al unui utilizator. Așa cum a fost menționat anterior, utilizatorii parcării au posibilitatea creării unui cont de utilizator pe aplicația web unde își vor putea vizualiza toate sesiunile, locurile pe care au parcat în cadrul acestora, vor putea adresa întrebări și vor putea crea rapoarte în cazul întâmpinării unei probleme. Această entitate va stoca informațiile unui utilizator, mai precis username-ul acestuia și parola sa. Parolele stocate în baza de date sunt criptate folosind BCrypt.

Nume câmp	Tip câmp	Descriere
id	int	Numărul unic de identificare al contului de utilizator.
license_plate	String	Username-ul utilizatorului (numărul de înmatriculare).
password	String	Parola utilizatorului, criptată.

Tabela 8: Entitatea "Users"

5.8 Entitatea "Reports"

Entitatea "Reports" este destinată aplicației web și este utilizată pentru înregistrarea și gestionarea rapoartelor referitoare la problemele întâmpinate în cadrul parcării. Aceasta conține mai multe câmpuri care descriu detaliat fiecare raport în parte. Fiecare raport este asociat unei sesiuni, identificată prin câmpul "session_id". Utilizatorii pot crea un raport direct de pe pagina principală a aplicației web doar dacă numărul lor de înmatriculare a fost detectat pentru sesiune respectivă. Lista rapoartelor poate fi vizualizată numai din contul special de administrator.

Nume câmp	Tip câmp	Descriere
id	int	Numărul unic de identificare al raportului.
session_id	int	Numărul unic de identificare a sesiunii.
reporter	String	Numărul de înmatriculare a autovehiculului care a raportat.
suspect	String	Numărul de înmatriculare a autovehiculului raportat.
time_period	String	Perioada întâmpinării problemei, oferită de utilizator.
contact	String	Informațiile de contact ale persoanei care a raportat.
details	String	Detaliile problemei.
opened	boolean	Statusul raportului.

Tabela 9: Entitatea "Reports"

6 Prezentarea aplicațiilor

În cadrul acestei lucrări, vor fi prezentate două aplicații legate între ele ce se ocupă de monitorizarea și gestionarea parcării: una dezvoltată în limbajul de programare C++ și cealaltă în Java.

Prima aplicație este o implementare în C++ a unui sistem de monitorizare și gestionare a unei parări ce se ocupă de configurarea camerelor video din interiorul parării și a locurilor de parcare, de detecția și recunoașterea numerelor de pe plăcuțele de înmatriculare ale mașinilor și de salvarea datelor legate de fiecare mașină și camera video.

În al doilea rând, se va adresa o altă perspectivă prin prezentarea unei aplicații web implementate în Java, care permite utilizatorilor să vizualizeze diferite detalii pentru numărul de înmatriculare cu care și-au creat contul pe aplicație, precum data și ora în care a intrat în parcare, timpul rămas, locul de parcare pe care a parcat, etc. Această aplicație web oferă o interfață prietenoasă și accesibilă, permițând utilizatorilor să vizualizeze toate detaliile legate de sesiunea de parcare curentă, fara a fi nevoie de instalarea și rularea unui software suplimentar pe dispozitivele acestora.

6.1 Aplicația C++

6.1.1 Arhitectura MVC

În implementare aplicației C++ s-a folosit arhitectura Model-View-Controller (MVC), care este un model de proiectare arhitectural a aplicațiilor software care separă componentele sale în trei părți distincte: Model, View și Controller. Această separare permite dezvoltatorilor să își organizeze mai eficient codul și să își mențină aplicațiile mai ușor de întreținut și de extins.

Modelul reprezintă componenta care gestionează datele și logica aplicației. Aici sunt definite clasele și funcțiile care permit manipularea acestora. Modelul nu este conștient de interfața utilizatorului sau de cum datele sunt prezentate pe aceasta, ci se concentrează exclusiv pe gestionarea informațiilor și a logicii asociate.

View-ul este responsabil cu prezentarea datelor utilizatorului. Aceasta reprezintă interfața utilizatorului, care poate fi o interfață grafică, o pagină web sau orice altă formă de prezentare a informațiilor. Acesta primește datele de la model și le afișează într-o formă ușor de înțeles pentru utilizatori.

Controller-ul este intermediar între Model și View. El primește input-ul de la utilizator prin intermediul interfeței utilizatorului și decide cum să proceseze aceste input-uri. Mai departe acesta interacționează cu Model-ul pentru a obține datele necesare și le transmite apoi către componenta View pentru afișare.

Separarea clară a problemelor între diferitele componente este unul din avantajele principale ale folosirii arhitecturii MVC. Această separare permite lucrarea în mod independent la fiecare componentă în parte, ceea ce facilitează extinderea, testarea și întreținerea aplicației. De exemplu, schimbarea modului în care datele sunt prezentate utilizatorului în interfața grafică nu afectează deloc logica din spate, deoarece aceasta este gestionată de componenta Model [26].

Astfel, având clase dedicate bazei de date ce se ocupă cu manipularea informațiilor din aceasta, clase ce se ocupă cu prelucrarea imaginilor preluate din camerele video definite și ulterior cu detectarea și recunoașterea caracterelor din numărul de înmatriculare al mașinii, și clase dedicate interfeței grafice, arhitectura MVC a fost cea mai potrivită alegere pentru dezvoltarea acestei aplicații, facilitând organizarea și gestionarea codului.

6.1.2 Arhitectura pe trei nivele pentru baza de date

În cadrul componentei Model din arhitectura MVC se află și implementarea claselor necesare pentru manipularea datelor din baza de date, iar aceasta este, la rândul ei, implementată de asemenea pe mai multe nivele. Astfel, arhitectura pe trei nivele este o modalitate comună pentru organizarea codului și pentru separarea clară dintre manipularea datelor, logică și interfața utilizatorului. Cele trei nivele din cadrul arhitecturii sunt Data Access, Business Logic și Entity [27].

În primul rând, nivelul Data Access cuprinde clasele și funcțiile ce se ocupă direct de interacțiunea cu baza de date, fiind cel mai apropiat nivel față de aceasta. Clasele și funcțiile includ operații CRUD, adică creare, citire, actualizare și ștergere a datelor. Pentru conectarea și interacțiunea cu baza de date s-au folosit clasele specifice din biblioteca Qt, de exemplu, QSql, QSqlDatabase, QSqlQuery.

În al doilea rând, nivelul Business Logic include logica aplicației, mai precis modul în care datele sunt procesate și utilizate în cadrul aplicației. Acest nivel cuprinde clasele și funcțiile care definesc operațiile și anumite reguli, precum validări, calcule sau transformări, folosind datele primite de la nivelul DataAccess.

În final, nivelul Entity cuprinde clasele ce definesc structura datelor utilizate în aplicație și sunt utilizate de către nivelul Business Logic pentru manipularea datelor în conformitate cu regulile și operațiile definite și de către nivelul Data Access pentru interacțiunea cu baza de date.

Prin utilizarea acestei arhitecturi pe trei nivele, se realizează o separare clară a responsabilităților din cadrul aplicației, favorizând astfel un cod mai ușor de înțeles, de testat și de întreținut.

6.1.3 Realizarea interfeței grafice

Rolul interfeței grafice este acela de a oferi utilizatorului un mijloc eficient și intuitiv pentru interacțiunea cu funcționalitățile oferite. Toate informațiile și acțiunile disponibile în interfață sunt prezentate folosind anumite texte sugestive pentru realizarea fiecărei componente.

Pentru realizarea interfeței grafice a proiectului s-a folosit mediul de dezvoltare Qt. Crearea și utilizarea unui GUI se poate realiza cu ușurință folosind cele două instrumente oferite de Qt, și anume, Qt Designer și Qt Creator. Diferența dintre cele două instrumente este că Qt Creator este un mediu de dezvoltare integrat pentru dezvoltarea aplicațiilor Qt, pe când Qt Designer se ocupă, așa cum reiese și din denumirea sa, doar cu proiectarea interfeței grafice.

Interfața acestei aplicații a fost creată utilizând instrumentul Qt Designer. Astfel, în urma creării unui GUI, se obține un fișier cu extensia "*.ui", ce poate fi integrat cu ușurință în orice aplicație.

În continuare, se vor prezenta câteva componente de bază, numite widgets, ce sunt prezente pe interfața grafică a aplicației.

Fereastra principală care oferă un cadru pentru adăugarea tuturor elementelor necesare proiectării unui GUI se numește **QMainWindow**.

Obiectele din interfața grafică sunt moștenite de clasa **QWidget**. Este posibil ca prin intermediul acesteia să se poată primi anumite evenimente de la tastatură, precum apăsarea unei taste, și mouse, precum un clic, cărora să le răspundă prin îndeplinirea unor acțiuni.

Sistemul de layout-uri al Qt-ului se ocupă de aranjarea automată a tuturor widget-urilor copil în zona destinată unui widget părinte, astfel încât tot spațiul disponibil să fie umplut. Rolul layout-urilor este de a redimensiona și poziționa elementele din interfața grafică în momentul în care dimensiunea a schimbată, asigurând astfel rearanjarea constantă a widget-urilor. **QGridLayout** este o altă clasă din Qt ce se ocupă de aranjarea elementelor din interfață într-un tablou bidimensional, acestea având posibilitatea de a ocupa mai multe celule. De regulă, la adăugarea unui element în **QGridLayout**, se va atribui o anumită dimensiune pentru ocuparea spațiului disponibil în funcție de restul componentelor ce urmează a fi adăugate [28].

QLabel este un element ce permite afișarea de texte, imagini sau video-uri în interfața grafică. Aceștia i se pot adăuga diferite conținuturi astfel: în cazul text-ului, se va utiliza funcția **setText()**, care necesită un parametru de tip **QString**, iar în cazul afișării sau manipulării unei imagini se va utiliza **setPixmap()**, ce necesită un parametru de tip **QPixmap**. În această aplicație se vor utiliza aceste două elemente chiar din prima pagină a acesteia, adică pagina principală, unde vor exista două chenare pe care se vor afișa imagini preluate de la camerele video, iar deasupra lor, vor exista două **QLabel**-uri ce conțin texte. Avantajul elementului **QPixmap** este că acesta are proprietatea de a nu umple memoria atunci când imaginea este schimbată. Astfel, programul este optim și se pot realiza oricâte încărcări de imagini în cadrul unui **QPixmap**.

QLineEdit este elementul utilizat pentru crearea unui câmp de text simplu unde utilizatorul poate introduce sau edita text.

QComboBox reprezintă un control de listă derulantă ce oferă utilizatorului posibilitatea de a selecta una din mai multe opțiuni predefinite dintr-o listă.

QListWidget este un element ce oferă o listă afișată în interfața grafică, unde fiecare element din listă poate fi selectat. Poate fi utilizată pentru afișarea unor liste simple de elemente, cu sau fără posibilitatea de a le edita. În cadrul aplicației, acest element este folosit pentru a afișa o listă de elemente, mai precis camere video, pe care utilizatorul le poate selecta pentru a putea vizualiza/completa automat toate detaliile despre aceasta, ușurându-i astfel munca.

QMenuBar este elemente care conține o bară de meniuri. În aplicația prezentată se va folosi un astfel de meniu pentru a oferi mai mult loc celor două chenare în care se vor afișa fluxuri video.

QMenu creează un meniu ce poate fi utilizat în interiorul bării de meniu, mai precis într-un **QMenuBar**. Acest meniu poate fi de două tipuri: de sine stătător sau derulant (afișat în jos). În

cadrul aplicației vor exista trei meniuri de tip pull-down.

În cadrul aplicațiilor, multe dintre comenzi sunt apelate prin intermediul meniurilor, butoanelor sau a comenzilor rapide de la tastatură. **QAction** poate conține atât o comandă rapidă, cât și un text sau o pictogramă. Odată cu adăugarea unui QMenu, se va adăuga implicit și un QAction iar această acțiune trebuie conectată neapărat la un slot care o va efectua.

Qt oferă și o metodă de comunicare bidirecțională prin utilizarea semnalelor și a sloturilor, cunoscută sub numele de tehnica de apelare inversă. **Semnalul** (signal) este emis mereu când are loc un eveniment iar widget-urile din Qt au mai multe semnale predefinite, precum semnalul declanșat (triggered) care este și corespunzător pentru QAction și este activat la fiecare conexiune a unei acțiuni cu un slot corespunzător. Qt oferă de asemenea posibilitatea adăugării în orice moment a semnalelor proprii. De exemplu, în evenimentul creat în proiect, atunci când se face click cu mouse-ul pe imaginea emisă de camera video în momentul definirii locurilor de parcare, se va emite un semnal ce este recepționat atunci când se conectează cu funcția corespunzătoare îndeplinirii acțiunii. Un **Slot** este o funcție ce reprezintă răspunsul unui semnal. În Qt exista slot-uri predefinite, ca de exemplu slot-ul close() ce este folosit în cadrul proiectului și implică închiderea interfeței grafice.

Având toate elementele ce compun interfața grafică din proiect prezentate, se poate trece la evidențierea funcționalității acestora.

6.1.4 Camerele Video

Pentru ca sistemul de monitorizare și gestionare a parcării să funcționeze, sunt necesare cel puțin două camere video, una destinată intrării în parcare, și cealaltă destinată ieșirii. Acest lucru este necesar pentru a putea detecta și recunoaște numărul de înmatriculare în vederea înregistrării autovehiculelor ce intră și părăsesc parcare. La alegerea administratorului parcării, se pot introduce și camere video destinate locurilor de parcare ce vor putea înregistra mașinile ce au parcat pe un loc anume.

În aplicația ce urmează a fi prezentată vor exista trei tipuri de camere video numite **ENTRANCE**, **EXIT** și **PARKING**, ce vor putea fi atribuite camerelor video configurate. Fiecare tip va avea ca urmare o "Acțiune" specifică. Acțiunea reprezintă felul în care imaginea capturată de la camera video și numărul de înmatriculare recunoscut sunt manipulate în funcție de tipul camerei video.

Tipul **ENTRANCE** se referă la camerele video destinate intrării în parcare și pot fi oricâte la număr. Spre deosebire de tipul **PARKING**, aceasta necesită declanșarea acțiunii manual prin apăsarea unui buton de către șofer, sau în cazul testării, prin apăsarea unei taste alese de administratorul parcării la configurarea camerei video, asemănător parcarilor din mall-uri, aeroporturi și așa mai departe. Acțiunea specifică acestui tip constă în încercarea de a detecta și recunoaște numărul de înmatriculare a mașinii prezente în fluxul video și de a înregistra o nouă sesiune în baza de date. Această nouă sesiune va include data și ora la care s-a realizat intrarea în parcare, numărul de înmatriculare al mașinii dacă a fost recunoscut cu succes, iar dacă nu, atunci nu va exista niciun număr, și un cod special, numit **SecretID**, care va putea fi folosit pentru crearea unui cont de utilizator pe pagina web a parcării, dar și pentru a putea părăsi parcare în cazul eșecului la recunoașterea

numărului de înmatriculare. Se va considera că "Acțiunea" s-a terminat cu succes dacă s-a putut crea această nouă sesiune. Motivul pentru care nu s-ar putea crea o nouă sesiune este pentru că numărul de înmatriculare detectat se află, teoretic, în interiorul parării, caz în care va fi nevoie de intervenția administratorului pentru a încheia vechea sesiune, lucrul realizat din meniul "Manage Entries" al aplicației. Utilizatorii vor putea vedea toate detaliile legate de sesiunile sale pe pagina web a parării.

Tipul **PARKING** reprezintă camerele video destinate locurilor de parcare și vor cuprinde, în general, trei sau patru locuri de parcare, în funcție de calitatea acestora. La alegerea administratorului, acesta poate opta pentru renunțarea la acest tip de camere. La adăugarea unei camere video de acest tip va fi necesară o configurare mai detaliată ce va cuprinde și datele locurilor de parcare pe care aceasta le are în vedere, însă aceasta va fi prezentată mai în detaliu într-un capitol viitor unde se va prezenta pagina specială pentru configurarea acesteia. Declanșarea acțiunii se face atât automat, la un anumit interval de timp ce poate fi modificat, dar și manual prin apăsarea unei taste de către administrator. Acțiunea constă în detectarea numărului de înmatriculare de pe fiecare loc de parcare configurat, fiecărei camere video de acest tip. Dacă un număr de înmatriculare este detectat pe un loc de parcare A, se va verifica dacă există o sesiune activă pe acesta iar în caz afirmativ, se va crea o sesiune de parcare nouă pentru numărul de înmatriculare detectat pentru locul de parcare A și cu timpul curent, care va reprezenta timpul de început al acesteia, punând capăt sesiunilor de parcare anterioare, dacă este cazul. Încheierea unei sesiuni de parcare se referă la adăugarea unui timp final acesteia. Dacă niciun număr de înmatriculare nu a putut fi detectat sau dacă acesta a fost detectat în parcare, dar nu și la intrare, adică sesiunii curente nu îi este atribuit niciun număr de înmatriculare, atunci se va ignora locul curent de parcare și se va trece la verificarea următorului. Utilizatorul va putea vedea pe pagina web a parării toate detaliile legate atât de sesiunea curentă, cât și de sesiunile trecute, precum data și ora la care a intrat/ieșit din parcare, dar și locurile de parcare pe care acesta a parcat, eliminând nevoia șoferilor de a ține minte locul pe care au parcat.

Tipul **EXIT** se referă la camerele video destinate ieșirii din parcare și pot fi, de asemenea, oricâte la număr. Declanșarea acțiunii se face tot prin apăsarea unui buton, însă aceasta diferă față de tipul **ENTRANCE**. Spre deosebire de tipul **ENTRANCE**, care creează o nouă sesiune, tipul **EXIT** actualizează sesiunea curentă prin adăugarea timpului (data și oră) la care s-a realizat ieșirea din parcare. Dacă la intrare nu a putut fi detectat numărul de înmatriculare sau nu a fost găsită nicio sesiune curentă specifică numărului detectat, atunci părăsirea parării se va realiza prin introducerea **SecretID-ului** aferent. În cazul întâmpinării unei probleme, șoferii vor putea crea o sesizare pe pagina web.

Tastele pentru declanșarea acțiunilor camerelor video se pot atribui atât la configurarea camerei video, cât și după, cu condiția ca acestea să fie unice și să fie orice literă din alfabet, de la A la Z. Camerele video de tip **PARKING** nu necesită neapărat o tastă definită, acțiunea pentru acestea fiind declanșată automat. De asemenea, aceste taste pot fi schimbate în orice moment din meniul "Manage Cameras".

Pentru ca aplicația să funcționeze, este necesar ca toate camerele video să fie configurate corect, în caz contrar, aplicația va intra în "repair mode" până când problemele sunt rezolvate. De exemplu, dacă o cameră video este scoasă din funcțiune, dar nu și ștearsă din meniul destinat ad-

ministrării camerelor video, atunci aplicația nu va mai funcționa decât după ștergerea acestora din lista camerelor.

6.1.5 Pagina principală

La deschiderea aplicației se poate observa pagina principală a acesteia, ce conține două chenare în care se poate vizualiza fluxul video a unor camere video, la alegere, din interiorul parcării, un buton "Refresh" și o bară de meniuri în stânga sus având la rândul ei, meniurile **"Settings"**, **"Manage Cameras"** și **"Manage Entries"**.

Butonul din josul paginii, **"Refresh"**, este folosit pentru a actualiza camerele video din întreaga aplicație. Odată acționat, toate camerele video existente vor fi actualizate alături de cele două chenare. Acest lucru este necesar de fiecare dată când o cameră video este adăugată, ștearsă, sau actualizată. De exemplu, în cazul în care se dorește vizualizarea altor camere video în cele două chenare, este necesară alegerea unui camera slot pentru o cameră video specifică din meniul **"Manage Cameras"**, urmată de acționarea butonului **"Refresh"**. De asemenea, în cazul testării, unde se folosesc video-uri predefinite în loc de o cameră video, la finalizarea acestora este necesară apăsarea butonului pentru ca video-urile să înceapă din nou.

Pentru realizarea unei acțiuni pentru o cameră video se va apăsa tasta atribuită acesteia, indiferent dacă camera video respectivă se află sau nu pe unul dintre cele două chenare. Se pot realiza oricâte acțiuni pentru orice tip de cameră video, cu condiția ca aceasta să fie funcțională la acel moment.

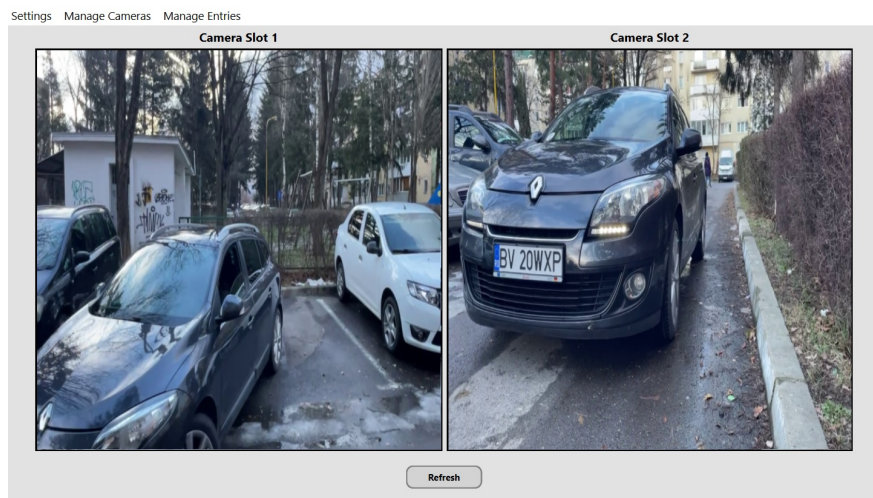


Figura 27: Pagina principală a aplicației C++.

6.1.6 Meniul "Manage Cameras"

Meniul **"Manage Cameras"** se găsește pe pagina principală a aplicației, mai exact în bara de meniuri aflată în susul paginii. Acționarea acestuia va deschide o listă de opțiuni ce țin de administrarea camerelor video din interiorul aplicației:

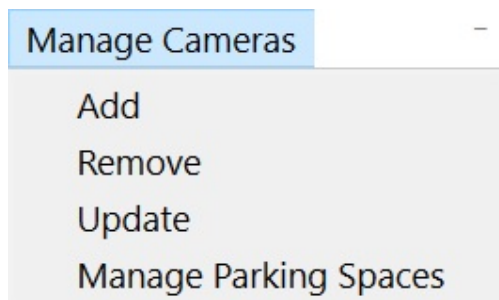


Figura 28: Meniul "Manage Cameras".

Administrarea camerelor video este cel mai important aspect în configurarea sistemului de monitorizare și gestionare a parcării, acestea fiind responsabile pentru capturarea frame-urilor în vederea detectării și recunoașterii numerelor de înmatriculare ale autovehiculelor, iar acest meniul oferă toate uneltele necesare pentru manipularea camerelor video.

- Add - deschide o fereastră unde utilizatorul va putea adăuga o nouă cameră video.
- Remove - deschide o fereastră unde utilizatorul va putea șterge o cameră video.
- Update - deschide o fereastră unde utilizatorul va putea modifica detaliile camerelor video.
- Manage Parking Spaces - deschide o fereastră unde utilizatorul va putea configura locurile de parcare pentru camerele video de tip **PARKING**.

Opțiunile "Add", "Remove" și "Update" vor deschide o nouă fereastră numită "Camera Management", destinată adăugării, ștergerii și modificării camerelor video, care, în funcție de alegerea utilizatorului, va avea diferite modificări. Această fereastră este explicată în detaliu în capitolul următor.

Pe de altă parte, opțiunea "Manage Parking Spaces" va deschide o altă fereastră, total diferită față de "Camera Management", ce este destinată configurării locurilor de parcare pentru camerele video de tip **PARKING**.

6.1.7 Pagina "Camera Management"

Pagina "Camera Management" este destinată configurării camerelor video, oferind posibilitatea utilizatorului de a adăuga, șterge sau modifica camerele video cu ușurință. Aceasta este alcătuită din patru butoane, trei textbox-uri, două combobox-uri și o listă de widget-uri. Se poate observa că această pagină este totuși împărțită în două părți, una pentru detaliile camerei video, iar cealaltă pentru lista camerelor video.

Deschiderea acestei pagini (ferestre) se realizează prin acționarea unuia din cele trei butoane ale meniului "Manage Cameras", denumite "Add", "Remove" și "Update". În funcție de butonul acționat, pagina va fi deschisă în modul de utilizare dorit, însemnând că anumite widget-uri vor dezactivate iar butonul de confirmare, care va purta numele modului ales, va efectua operația

necesară, salvând modificările în baza de date.

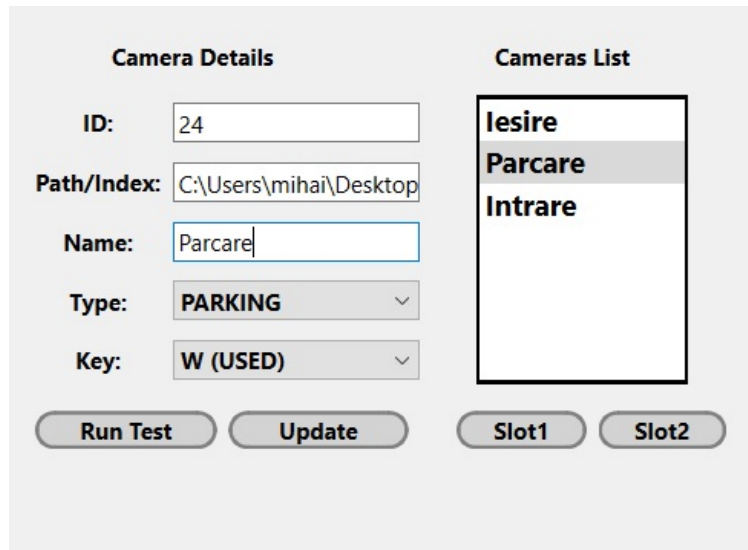


Figura 29: Pagina "Camera Management".

Prima parte a ferestrei, destinată introducerii și afisării detaliilor camerelor video și poziționată în partea stângă, conține următoarele elemente:

- ID (QLineEdit) - numărul unic de identificare a unei camere video, nemodificabil indiferent de modul de utilizare în care fereastra a fost deschisă.
- Path/Index (QLineEdit) - path-ul sau index-ul camerei video, necesar pentru deschiderea acesteia.
- Name (QLineEdit) - numele atribuit camerei video.
- Type (QComboBox) - tipul camerei video.
- Key (QComboBox) - tasta atribuită camerei video pentru declanșarea acțiunii.
- Update (QPushButton) - buton pentru confirmarea operației, numele acestuia fiind identic cu operația dorită.
- Run Test (QPushButton) - buton pentru testarea camerei video folosind detaliile din textbox-uri și combobox-uri.

În tabelul de mai jos sunt toate elementele din prima parte a ferestrei și starea lor (activ, dacă acesta poate fi modificat, și inactiv dacă acesta nu poate fi modificat) în funcție de modul de utilizare ales la deschiderea ferestre.

Deși unele elemente sunt inactive în anumite situații, acestea sunt vizibile pe pagină și actualizate automat doar de către aplicație, nefiind modificabile de către utilizator.

	Add	Remove	Update
ID	Inactiv	Inactiv	Inactiv
Path/Index	Activ	Inactiv	Activ
Name	Activ	Inactiv	Activ
Type	Activ	Inactiv	Activ
Key	Inactiv	Inactiv	Activ
Run Test	Activ	Activ	Activ

Tabela 10: Elementele ferestrei și starea lor de funcționare pentru fiecare mod de utilizare

A doua parte a ferestrei, destinată vizualizării tuturor camerelor video existente și a schimbării chenarului implicit în care aceasta se poate vizualiza în direct, este poziționată în partea dreaptă și conține următoarele elemente:

- Cameras List (QListWidget) - lista de widget-uri ce conține toate camerele existente în aplicație.
- Slot1 (QPushButton) - buton pentru schimbarea chenarului implicit în chenarul 1 pentru camera selectată din lista de widget-uri.
- Slot2 (QPushButton) - buton pentru schimbarea chenarului implicit în chenarul 2 pentru camera selectată din lista de widget-uri.

Indiferent de modul de utilizare ales, toate elementele acestei părți sunt complet funcționale.

Lista de widget-uri "**Cameras List**" conține de fapt mai multe widget-uri ce mapează către un obiect de tip Camera (cameră video). Astfel că selectarea unei camere video din listă va duce automat la completarea detaliilor despre aceasta în elementele din stânga paginii și reținerea acesteia în cazul schimbării chenarului implicit.

Adăugarea (Add) unei camere video constă într-un proces simplu în care utilizatorul este nevoit să introducă path-ul sau index-ul camerei video, numele dorit pentru aceasta și tipul ei (ENTRANCE, EXIT, PARKING). Chiar dacă ID-ul este gol sau a fost completat automat în urma selectării unei camere video din listă, acesta nu va fi luat în vedere iar generarea acestuia va fi efectuată automat de către baza de date la salvarea camerei video. Camera video adăugată va fi pornită doar după acționarea butonului "Refresh" din pagina principală sau după repornirea aplicației.

Procesul de **ștergere** (Remove) a unei camere video este probabil cel mai simplu, utilizatorul fiind nevoit să selecteze o cameră video din listă, să se asigure prin verificarea detaliilor camerei video selectate că aceasta este cea corectă și după aceea să acționeze butonul "Remove". Totuși, aceasta va continua să funcționeze până când se va acționa butonul "Refresh" din pagina principală sau se va reporni aplicația.

Actualizarea (Update) se realizează prin selectarea din listă a camerei video dorite, urmând a se modifica detaliile dorite în partea stângă a paginii. Actualizarea este absolut necesară dacă se dorește adăugarea unei taste în vederea declanșării unei acțiuni la apăsarea acesteia, acest lucru

nefiind posibil direct la adăugare. Tasta adăugată poate fi orice literă de la A la Z și trebuie să fie neapărat unică. Pentru ca modificările să intre în efect, este necesară acționarea butonului "Refresh" din pagina principală sau repornirea aplicației.

Schimbarea chenarului implicit se efectuează prin simpla selectare a unei camere video din listă și a chenarului dorit (Slot1 sau Slot2). Chenarele vor fi actualizate cu noile camere video după repornirea aplicației sau a acționării butonului "Refresh" din pagina principală. Spre deosebire de camerele video în sine, ce sunt salvate într-o bază de date, camerele video implicite din chenare sunt salvate într-un fișier JSON numit "cameras.json", unde pentru fiecare din cele două chenare se va specifica ID-ul camerei video. **Modificarea camerei video dintr-un chenar prin intermediul meniului "Settings" este temporară!**

Este important ca înainte de acționarea butonului "Add" sau "Update" să se verifice mai întâi funcționalitatea camerei video introduse prin acționarea butonului "Run Test", care va încerca să deschidă camera video și să preia un frame din aceasta. Existența unei camere video nefuncționale va rezulta în activarea modului de reparare (repair mode) până când toate camerele existente vor fi funcționale. Aplicația va intra în repair mode doar la reîncărcarea camerelor video, mai precis la deschiderea aplicației sau la acționarea butonului "refresh" de pe pagina principală, iar în cazul apariției unui defect la o cameră video în timpul rulării, atunci nu se vor mai putea declanșa acțiuni de la aceasta, nefiind posibilă preluarea unui frame, însă aplicația va rula în continuare fără probleme.

6.1.8 Pagina "Parking Spaces Management"

Pagina "Parking Spaces Management" oferă posibilitatea utilizatorului de a adăuga, șterge sau modifica locurile de parcare ce aparțin de o cameră video de tip PARKING. Aceasta este alcătuită din patru butoane, două liste de widget-uri și un chenar folosit pentru afișarea unei imagini. Deschiderea aceste pagini se realizează prin acționarea butonului "Manage Parking Spaces" din cadrul meniului "Manage Cameras".



Figura 30: Pagina "Parking Spaces Management".

În figura 30 se observă existența a două liste de widget-uri poziționate pe partea dreaptă, numite "Cameras" și "Spaces", și a unei imagini aflate în partea stângă a paginii. Aceste trei elemente sunt strâns legate între ele, imaginea din chenar schimbându-se în funcție de camera video și locul de parcare ales.

Lista de widget-uri numită "**Cameras**" conține toate camerele video de tip PARKING existente în aplicație sub formă de widget-uri care mapează la un obiect de tip Camera (cameră video). Astfel, selectarea uneia dintre camere va rezulta în preluarea frame-ului curent de la camera video și afișarea acestuia în chenar, dar și actualizarea listei de widget-uri "Spaces" cu locurile de parcare ale camerei respective.

După selectarea unei camere din lista "Cameras", lista de widget-uri "**Spaces**" va conține toate locurile de parcare existente pentru camera respectivă sub formă de widget-uri care mapează la un obiect de tip ParkingSpace (loc de parcare). Selectarea unui loc de parcare va duce la desenarea pe frame-ul preluat de la camera video a unei cutii delimitatoare (bounding box) în jurul acestuia. De asemenea, acesta va fi reținut în cazul în care se dorește ștergerea sau modificarea acestuia.

Desenarea cutiei delimitatoare se realizează prin setarea a două puncte pe imaginea din chenar printr-un simplu click de la mouse. Primul punct poate reprezenta oricare dintre colțurile locului de parcare iar al doilea punct trebuie să reprezinte colțul de pe diagonala celui ales anterior. Imediat după selectarea celui de-al doilea punct se va desena cutia delimitatoare (bounding box) pe imagine.

Pentru **adăugarea** unui loc de parcare nou este necesară selectarea camerei video dorite și desenarea unei cutii delimitatoare pentru locul de parcare. Dacă zona delimitatoare a locului de parcare este corectă se va acționa butonul "**Confirm (Add)**", iar în caz contrar, butonul "**Reset Box**" care va șterge desenul de pe imagine și va reseta punctele alese. După adăugarea locului de parcare,

detectarea numerelor de înmatriculare de pe acesta va începe de îndată, fără a mai fi nevoie de acționarea butonului "Refresh" de pe pagina principală.

Ștergerea unui loc de parcare se realizează prin selectarea camerei video din lista "Cameras" în care se află locul de parcare dorit, urmat de selectarea acestuia din lista "Spaces" și acționarea butonului "**Remove**". Detectarea numerelor de înmatriculare de pe acesta va lua sfârșit imediat, nefiind nevoie de acționarea butonului "Refresh" din pagina principală.

Actualizarea (Update) sau modificarea unui loc de parcare se obține prin selectarea locului de parcare dorit din lista "Spaces", urmat de schimbarea cutiei delimitatoare dacă se dorește schimbarea delimitării zonei, lucru realizat prin acționarea butonului "Reset Box" urmat de desenarea zonei noi, urmând în final acționarea butonului "**Update**" și introducerea unui nume nou pentru locul de parcare respectiv, dacă este cazul. Noile schimbări vor intra în efect imediat iar șoferii vor putea vizualiza noul nume al locului de parcare pe pagina web și detecția numerelor de înmatriculare se va efectua pe noua zonă.

6.1.9 Meniul "Manage Entries"

Meniul "Manage Entries" este situat în bara de meniuri aflată în susul paginii pe pagina principală. Acționarea acestuia va deschide o listă de opțiuni ce țin de administrarea înregistrărilor în baza de date. Opțiunile disponibile în acest meniu sunt "**Force exit for an entry**" și "**Force action by photo**".

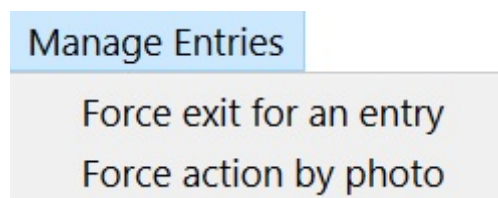


Figura 31: Meniul "Manage Entries".

Opțiunea **Force exit for an entry** este folosită pentru a putea înregistra manual încheierea unei sesiuni în situația în care unui șofer nu i-a putut fi detectat și recunoscut numărul de înmatriculare și acesta a pierdut SecretID-ul sau nu reușește să-l folosească. În figura 32 se observă existența a două QComboBox-uri, unul pentru selectarea sesiunii dorite a fi încheiate unde este scris ID-ul sesiunii și numărul de înmatriculare atribuit, dacă este cazul, și celălalt pentru selectarea camerei video de ieșire, unde este scris numele camerei alături de ID-ul acesteia între paranteze. Camera video de ieșire selectată va corespunde cu camera video la care se situează autovehiculul în momentul utilizării acestei opțiuni.

Figura 32: Meniul opțiunii "Force exit for an entry".

Opțiunea **Force action by photo** este utilizată pentru a putea forța declanșarea unei acțiuni pentru o anumită cameră video folosind o imagine. Folosirea acestei opțiuni este recomandată în momentul testării. În figura 33 se poate observa existența unui QLineEdit, utilizat pentru introducerea path-ului către imaginea dorită, și un QComboBox utilizat pentru selectarea camerei video dorite pentru care se va declanșa o acțiune.

Figura 33: Meniul opțiunii "Force action by photo".

6.1.10 Meniul "Settings"

Meniul "Settings" se află în bara de meniuri aflată în susul paginii pe pagina principală. Acționarea acestuia va deschide o listă de opțiuni destinate schimbării anumitor setări. Opțiunile disponibile în acest meniu sunt **View Specific Camera** și **Change Detection Type**.

Figura 34: Meniul "Settings".

Opțiunea **View Specific Camera** este folosită pentru a schimba temporar camera video ce este vizualizată în cadrul unuia dintre cele două chenare de pe pagina principală. După repornirea aplicației sau acționarea butonului "Refresh", chenarele vor avea înapoi camerele video implicate.

Schimbarea permanentă a camerei video dintr-un chenar se realizează prin accesarea meniului "Manage Cameras", opțiunea "Add", "Remove" sau "Update" și acționarea butonului "Slot1" sau "Slot2", în funcție de chenarul dorit. În cazul de față, se va selecta slotul dorit din primul QComboBox, urmat de camera video dorită a fi vizualizată, de pe al doilea QComboBox. Schimbarea va fi imediată iar acționarea butonului "Refresh" pentru actualizarea camerelor video nu va fi necesară!

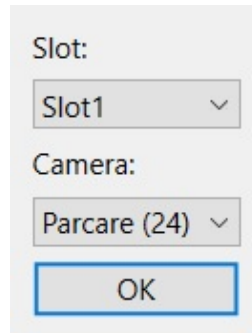


Figura 35: Meniul opțiunii "View Specific Camera".

Opțiunea **Change Detection Type** este utilizată pentru schimbarea modului în care se realizează detecția plăcuței de înmatriculare pentru un anumit tip de cameră. Implicit, detecția se realizează folosind modul **IMAGE_PROCESSING** pentru tipurile **ENTRANCE** și **EXIT**, și **DNN** pentru tipul **PARKING**. Aceste schimbări vor fi salvate într-un fișier JSON numit "detections.json" și vor intra în efect de îndată, nefiind necesară actualizarea camerelor video folosind butonul "Refresh". În figura 36 se poate observa existența a două QComboBox-uri, unul pentru selectarea tipului de cameră (ENTRANCE, EXIT sau PARKING) ce va folosi noul mod de detecție, și celălalt pentru selectarea noului mod de detecție (IMAGE_PROCESSING sau DNN).

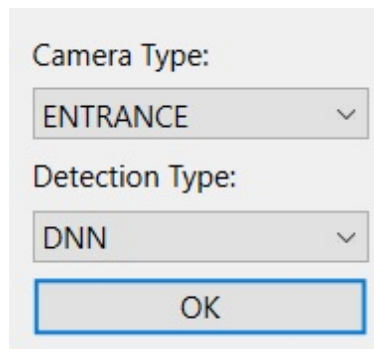


Figura 36: Meniul opțiunii "Change Detection Type".

6.2 Aplicația Java

6.2.1 Arhitectura

Asemănător aplicației C++, și în implementarea aplicației web folosind Java s-a folosit arhitectura Model-View-Controller (MVC), însă cu unele diferențe. Dacă în aplicația C++ se folosea în

principal arhitectura MVC iar în cadrul componentei Model se folosea arhitectura pe trei nivele pentru manipularea înregistrărilor în baza de date, în aplicația web implementată în Java se folosesc toate cele trei componente, Model, View și Controller, împreună cu încă două componente, Service și Repository. Acestea din urmă sunt responsabile cu manipularea înregistrărilor bazei de date, în timp ce celelalte trei componente se ocupă de logica aplicației și prezentarea datelor către utilizatori.

- Model - este reprezentarea datelor aplicației și include clasele Java care definesc structura datelor și logica asociată acestora.
- Controller - este responsabil pentru gestionarea cererilor HTTP primite de la client și pentru a comunica cu celelalte componente ale aplicației. Acestea sunt punctele de intrare în aplicație și sunt responsabile pentru direcționarea cererilor către serviciile adecvate și pentru returnarea răspunsurilor către client.
- Fișierele HTML (View) - sunt fișierele care definesc interfața utilizatorului. Ele pot folosi șabloane sau framework-uri de frontend, cum ar fi Thymeleaf sau AngularJS, pentru a obține datele din backend și pentru a le putea afișa în mod corespunzător.
- Repository - este componenta responsabilă de interacțiunea cu baza de date și include operații CRUD (Create, Read, Update, Delete) pentru a accesa și manipula datele.
- Service - este componenta care conține logica de afaceri (engl. BusinessLogic) a aplicației și este responsabilă pentru procesarea datelor și implementarea logicii aplicației. De obicei, componenta Controller va comunica cu componenta Service pentru executarea operațiilor necesare pe date.

6.2.2 Realizarea interfeței grafice

În cadrul aplicației Java a fost integrat motorul de șabloane numit Thymeleaf în vederea gestionării fișierelor HTML și realizării interfeței utilizatorului. Thymeleaf a fost alegerea preferată datorită flexibilității și puterii sale în crearea paginilor web dinamice. Prin intermediul Thymeleaf, au fost create și gestionate pagini web interactive, care comunică eficient cu logica de backend a aplicației. Thymeleaf oferă o modalitate elegantă de a combina codul HTML static cu expresii dinamice și logica de afișare a datelor din backend.

Pentru început, se creează fișierele HTML care definesc structura și aspectul paginii web. Thymeleaf permite utilizarea unor construcții speciale în cadrul acestor fișiere pentru a face referire la obiectele și datele din backend. Aceste construcții speciale sunt adesea integrate direct în codul HTML folosind o sintaxă specifică sau diferite atribute, facilitând astfel manipularea și afișarea dinamică a datelor.

Pe durata dezvoltării interfeței se folosesc funcționalitățile oferite de motorul de șabloane pentru a se utiliza, de exemplu, bucle care iterează printr-o listă de obiecte și afișarea fiecărui element într-un mod structurat pe pagină, sau condiții pentru afișarea sau ascunderea anumitor elemente pe baza unor condiții din backend.

De asemenea, o altă caracteristică puternică este capacitatea sa în procesarea formularelor HTML și gestionarea trimerii datelor înapoi către server. De exemplu, se pot realiza diferite validări sau prelucrări a datelor introduse de către utilizatori.

Pentru adăugarea unui stil și a unui aspect plăcut paginilor web s-a folosit și CSS (Cascading Style Sheets). Acesta permite controlarea modului în care elementele HTML sunt afișate în browser, de la fonturi și culori, la layout-uri și dimensiuni.

Un exemplu de utilizare în cadrul aplicației este definirea de stiluri CSS pentru elementele de tip text, precum dimensiunea fontului, culoarea și tipul de font utilizat. Au fost create clase CSS pentru a standardiza aspectul textului în întreaga aplicație, asigurând faptul că mesajele și informațiile sunt prezentate într-un mod clar și ușor de citit pentru utilizatori. Un alt exemplu în folosirea CSS-ului este adăugarea de culori și tabele pentru îmbunătățirea aspectului general al paginilor web din cadrul aplicației, creându-se astfel un mediu vizual atrăgător pentru utilizatori.

În plus, CSS-ul a fost esențial pentru definirea comportamentului elementelor de interfață, cum ar fi butoanele, meniurile și alte elemente interactive. Au fost utilizate stiluri CSS pentru evidențierea elementelor de acest tip, pentru a le putea oferi efecte de hover sau pentru a le adapta aspectul în funcție de acțiunile utilizatorului.

Având modul în care tehnologiile prezentate compun interfața grafică a aplicației, se poate trece la evidențierea funcționalității și prezentării acesteia.

6.2.3 Autentificarea (/login, /register, /passwordReset)

Autentificarea este un proces esențial pentru securizarea aplicațiilor web bazate pe framework-ul Spring ce se realizează cu ajutorul componentei Spring Security. Aceasta oferă un set puternic de funcționalități pentru gestionarea autentificării utilizatorilor și autorizarea accesului la resursele aplicației.

Pentru implementarea autentificării în Spring Security, primul pas constă în configurarea unui sistem de autentificare. Acest lucru constă în definirea modului de autentificare, cum ar fi autentificarea pe baza unui utilizator și a unei parole stocate într-o bază de date sau autentificarea pe baza unor mecanisme externe, precum LDAP sau OAuth. În cazul aplicației web prezentate, s-a folosit modul de autentificare bazat pe un utilizator și o parolă stocată într-o bază de date.

Odată ce sistemul de autentificare este configurat, Spring Security gestionează întregul proces de autentificare, inclusiv verificarea datelor (credențialelor) utilizatorului și gestionarea sesiunilor. De asemenea, Spring Security oferă funcționalități avansate de securitate, precum protecția împotriva atacurilor de tip "brute force" și gestionarea centralizată a politicilor de securitate.

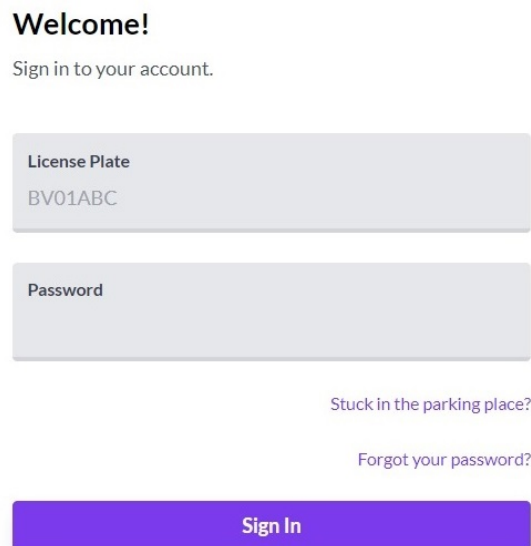
În plus, Spring Security facilitează implementarea autorizării, permițând definirea unor reguli detaliate pentru controlul accesului la resursele aplicației în funcție de rolurile și permisiunile utilizatorilor.

La accesarea aplicației web, utilizatorul este redirectionat automat către pagina de autentificare

(/login). Aici, utilizatorul este nevoit să își introducă datele de conectare pentru a putea accesa funcționalitățile aplicației.

Autentificarea se realizează folosind un număr de înmatriculare și o parolă. În 37 se poate observa meniul de login folosit pentru autentificare. Acesta include două textbox-uri, unul pentru introducerea numărului de înmatriculare (License Plate) și unul pentru introducerea parolei, două texte ce redirecționează utilizatorul către o pagină destinată ieșirii forțate din parcare folosind SecretID-ul ("**Stuck in the parking place?**"), nefiind nevoie de autentificare, și către o altă pagină destinată schimbării parolei ("**Forgot your password?**"), și de asemenea un buton pentru finalizarea autentificării. Pentru o vizualizare mai adecvată a acestui meniu, fundalul acestuia a fost șters astfel încât să se poată observa în figura 37 toate detaliile necesare.

După acționarea butonului "**Sign in**", dacă datele introduse sunt corecte, utilizatorul este logat și redirecționat către pagina principală (**/home**), iar în caz contrar, acesta va fi anunțat de faptul că numărul de înmatriculare sau parola nu sunt corecte.



The image shows a login interface with the following elements:

- Welcome!** header.
- Text: "Sign in to your account."
- License Plate** input field containing the text "BV01ABC".
- Password** input field.
- Link: "Stuck in the parking place?"
- Link: "Forgot your password?"
- Sign In** button.

Figura 37: Meniul de login din pagina de autentificare.

Înainte ca un utilizator să se logheze, acesta trebuie să își creeze mai întâi un cont de utilizator folosind SecretID-ul primit la intrarea în parcare. Acest cod special (SecretID) este unic pentru fiecare intrare înregistrată în parcare și poate fi folosit pentru crearea unui cont doar dacă aparține unei sesiuni de parcare în curs, nu există deja un cont și s-a putut detecta cu succes numărul de înmatriculare. Acest SecretID va conține o secvență de numere, urmată de numărul de înmatriculare detectat pentru sesiunea respectivă, de exemplu, "1705533880505549**BVo1ABC**". De asemenea, utilizatorului va trebui să introducă o parolă ce conține minim 8 caractere, incluzând cel puțin o literă mare, un număr și un caracter special. Pentru a începe procesul de înregistrare a unui cont, utilizatorul va apăsa pe textul de sub meniul de login "**Don't have an account? Sign up!**" și va fi

redirecționat către pagina de creare a unui cont de utilizator (**/register**).

Hello!

To create an account, you must get your secret ID from your entrance ticket.

Secret ID

17055333880505549BV01ABC

Password

Repeat Password

Sign Up

Figura 38: Meniul de register din pagina de creare a unui cont de utilizator.

Așa cum se poate observa în meniul din figura 38, utilizatorul este nevoit să introducă un SecretID și parola dorită, urmată de repetarea acesteia. După acționarea butonului **"Sign Up"**, dacă datele introduse sunt corecte, acesta va fi redirecționat către pagina de autentificare și se poate autentifica folosind numărul său de înmatriculare și parola introdusă.

În cazul uitării parolei, utilizatorii pot accesa pagina destinată resetării parolei **"passwordReset"** din pagina de autentificare printr-un click pe text-ul **("Forgot your password?"**). Pentru a realiza acest lucru cu succes, utilizatorul va avea nevoie de SecretID-ul sesiunii aflate în desfășurare, însemnând că șoferul nu a părăsit încă parcare, cu condiția ca SecretID-ul să conțină numărul de înmatriculare.

Toate parolele conturilor vor fi criptate și nu vor putea fi vizualizate în baza de date.

6.2.4 Pagina principală (**/home**)

Odată autentificat, utilizatorul este redirecționat către pagina principală (**/home**). Aceasta este compusă dintr-un meniu aflat în partea de sus a paginii, ce conține trei butoane ce vor redirecționa utilizatorul către o altă pagină, de exemplu "Home", către pagina principală, "Chat", către pagina de întrebări și răspunsuri, și "Log out" pentru delogarea din contul curent, și un tabel ce va include toate sesiunile în care numărul de înmatriculare cu care s-a efectuat autentificarea a fost detectat.

Parking Management System						Home	Chat	Log out
ID	Entrance Date	Exit Date	Time Spent (min)	Parking History	Status	SecretID	Report	
36	2024-04-10 20:21	Still in parking!	33274	View	NOT EXITED	View	Create Report	
35	2024-04-09 21:38	2024-04-09 21:40	1	View	EXITED	View	Create Report	
33	2024-04-02 17:34	2024-04-02 17:34	0	View	EXITED	View	Create Report	

Figura 39: Pagina principală (/home).

Așa cum se poate observa în figura 39, pagina constă în mare parte într-un tabel alcătuit din opt coloane, unde fiecare înregistrare a acestuia reprezintă o sesiune de parcare. Mai jos vor fi prezentate pe rând toate coloanele tabelului:

- ID - Număr unic de identificare a unei sesiuni.
- Entrance Date - Data și ora la care a fost începută sesiunea, mai precis când a intrat în parcare).
- Exit Date - Data și ora la care a fost încheiată sesiunea, mai precis când a ieșit din parcare).
- Time Spent (min) - Timpul petrecut în minute în interiorul parcării.
- Parking History - Buton pentru vizualizarea locurilor de parcare unde numărul de înmatriculare al utilizatorului a fost detectat. Mai exact, locurile de parcare pe care șoferul a parcat.
- Status - Starea sesiunii.
- SecretID - Buton pentru vizualizarea SecretID-ului asociat sesiunii.
- Report - Buton pentru raportarea unei probleme pe durata sesiunii.

La acționarea butonului **"View"** de pe coloana **"Parking History"**, utilizatorul este redirecționat către o pagină (**/home/spaces/{ID sesiune}**) unde acesta va putea vizualiza, sub formă de tabel, toate locurile de parcare pe care acesta a parcat în cadrul sesiunii respective. În cadrul tabelului va apărea numele acestuia, care se recomandă a fi cât mai sugestiv astfel încât, în cazul uitării acestuia, șoferul să își poată localiza mașina cu ușurință. Tabelul mai conține data și ora la care numărul de înmatriculare a fost localizat în zona delimitatoare a locului de parcare și la care l-a părăsit.

Parking Management System			Home	Chat	Log out
Name	Start Time	End Time			
ALEE_23	2024-02-08 23:00	2024-02-08 23:36			

Figura 40: Pagina pentru vizualizarea locurilor de parcare în care numărul de înmatriculare pentru sesiunea respectivă a fost detectat. (**/home/spaces/{ID sesiune}**).

Așadar, utilizatorul este redirecționat către o pagină în funcție de ID-ul sesiunii respective. În

cazul sesiunii cu ID 2, utilizatorul este redirecționat către `/home/spaces/2`. Același lucru se va întâmpla și la acționarea butonului **"View"** de pe coloana **"SecretID"**, utilizatorul fiind redirecționat către pagina `/home/viewSecretID/{ID sesiune}` unde își va putea vizualiza SecretID-ul pentru sesiunea dorită.

Acționarea butonului **"Create Report"** va redirecționa utilizatorul către o pagină destinată inițializării unui raport, pagină ce va fi prezentată în capitolul 6.2.6. Link-ul acestei pagini va fi tot de tipul celor prezentate anterior, acesta fiind `/report/create/{ID sesiune}`.

Butonul **"Chat"** din partea de sus a paginii va redirecționa utilizatorul către pagina de întrebări și răspunsuri (`/chat`), aceasta fiind disponibilă tuturor utilizatorilor. Butonul **"Home"** va redirecționa utilizatorul către pagina principală (`/home`). Cele două butoane, alături de **"Log out"**, vor fi disponibile în orice moment în cadrul meniului din partea de sus a unei pagini.

Este evident faptul că un utilizator va putea intra pur și simplu pe o pagină, ca de exemplu `/home/spaces/1000`, a cărui ID de sesiune nu îi aparține. În acest caz, utilizatorul va fi putea accesa pagina însă, în loc să fie întâmpinat de tabelul cu locurile de parcare, acesta va vedea un mesaj care îl va anunța că nu îi este permis să vizualizeze această pagină, exact ca în figura 41.

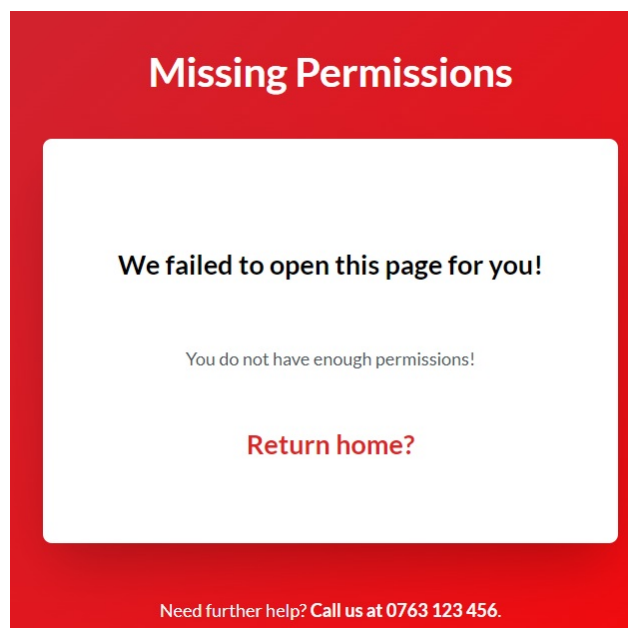


Figura 41: Accesarea unei pagini la care utilizatorul nu are acces.

6.2.5 Pagina și contul de administrator (`/admin`)

Din oficiu, va exista un cont special de administrator ce va avea numărul de înmatriculare (username-ul) **"ADMIN"** și parola **"pass"**. Acesta va putea accesa pagini suplimentare, însă nu va putea vizualiza date legate de sesiunile utilizatorilor.

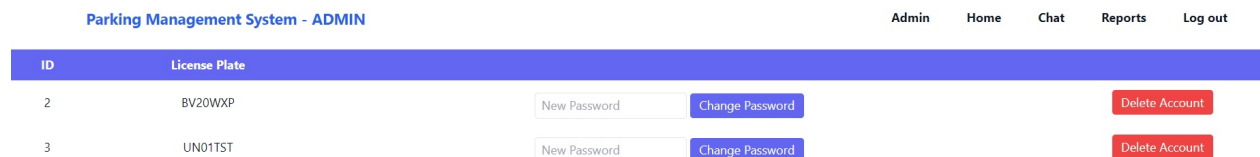
Spre deosebire de contul de utilizator ce este redirecționat către pagina principală după autentificare, contul de administrator este redirecționat către o pagină specială numită pagina de

administrator (**/admin**), ce este accesibilă doar acestui cont.

Această pagină este compusă dintr-un meniu de butoane aflat în partea de sus a paginii și un tabel ce se află sub acesta.

Meniul de butoane seamănă cu cel al unui cont de utilizator, având cele trei butoane clasice, "Home", "Chat", și "Log out", plus încă două butoane destinate contului de administrator, "**Admin**" și "**Reports**". Butonul "Admin" va redirecționa administratorul către pagina de administrator (**/admin**), iar "Reports" către pagina de raporturi (**/report/list**).

Tabelul din această pagină este compus din patru coloane în care fiecare înregistrare a acestuia va reprezenta un cont de utilizator. Administratorul va avea posibilitatea vizualizării tuturor conturilor de utilizator, alături de ID-ul și numărul de înmatriculare al acestora, și a schimbării parolei sau chiar a ștergerii contului, dacă este cazul. În figura 42 se poate observa pagina de administrator, alături de tabelul și meniul de butoane specific al acesteia.



ID	License Plate	New Password	Change Password	Delete Account
2	BV20WXP			
3	UN01TST			

Figura 42: Pagina de administrator (**/admin**).

6.2.6 Pagina de raporturi (**/report/list**)

Este inevitabil ca șoferii să nu întâmpine anumite dificultăți sau probleme pe durata șederii în parcare, astfel că, posibilitatea raportării acestora este necesară într-un mod sau altul.

Utilizatorii vor putea iniția un raport prin acționarea butonului "**Create Report**" de pe linia sesiunii în care a avut loc problema, aceștia fiind redirecționați către pagina dedicată creării unui raport (**/report/create/ID sesiune**). Aceștia vor putea crea oricâte rapoarte doresc pentru oricare dintre sesiuni, fie că ele sunt încă active sau nu. În cadrul unui raport, utilizatorul va introduce numărul de înmatriculare al suspectului, intervalul orar aproximativ la care s-a întâmplat, detaliile problemei și metoda prin care să fie contactat. Spre exemplu, în cazul în care un șofer cu numărul de înmatriculare A observă o tamponare, în care un șofer cu numărul B atinge o mașină parcată cu numărul C, acesta va crea un raport specificând numărul de înmatriculare al autovehiculului suspect, adică B, și al victimei în detalii, adică C.

Raporturile create de către utilizatori vor putea fi vizualizate doar de către un cont de administrator pe pagina de raporturi "**/report/list**". Acesta va putea accesa această pagină fie prin acționarea butonului "**Reports**" de pe pagina de administrator, fie prin accesarea link-ului **/report/list** din aplicația web. Lista de raporturi din cadrul acestei pagini este tot sub formă de tabel, fiind compusă din 6 coloane:

- ID - Număr unic de identificare a raportului.

- Session ID - Numărul unic de identificare a sesiunii pentru care a fost creat raportul.
- Reporter - Numărul de înmatriculare al persoanei ce a creat raportul sau a victimei.
- Suspect - Numărul de înmatriculare al persoanei raportate sau a suspectului.
- Status - Starea raportului, OPEN, dacă nu a fost soluționat, și CLOSED, dacă a fost soluționat.
- View - buton pentru vizualizarea detaliilor legate de raport și pentru setarea stării acestuia.

Parking Management System - ADMIN

Admin Home Chat Reports Log out

ID	Session ID	Reporter	Suspect	Status	
1	3	UN00TST	BV20WXP	CLOSED	View
2	2	BV20WXP	UN01TST	OPEN	View

Figura 43: Pagina de rapoarturi (/report/list).

Odată acționat, butonul **"View"** va rezulta în redirecționarea administratorului către pagina raportului (**"/report/view/ID Raport**). Acolo, administratorul va putea vizualiza orice detaliu legat de raport ce a fost oferit de către persoana ce a întocmit raportul, detaliu precum ora la care s-a întâmplat, persoanele implicate și altele. După eventuala soluționare a acestuia, administratorul va putea folosi butonul **"Close Report"** de pe această pagină pentru a marca starea raportului ca fiind **CLOSED**. Totuși, dacă dintr-un motiv sau altul raportul are nevoie de continuarea investigației, administratorul va acționa butonul **"Open Report"** pentru a marca starea raportului ca fiind **OPEN**.

6.2.7 Pagina de întrebări și răspunsuri (/chat)

Pagina de întrebări și răspunsuri poate fi accesată folosind atât un cont de utilizator, cât și un cont de administrator, prin intermediul butonului **"Chat"** aflat în partea de sus a paginii. Această pagină este destinată adresării de întrebări de către utilizatori unde vor putea fi răspunsuri de alți utilizatori sau chiar de administrator. O altă folosință a acesteia ar mai putea fi și comunicarea de anunțuri importante ce țin de parcare, precum închiderea acesteia de sărbători, evitarea unei ieșiri închise, renovarea acesteia, existența unor lucrări pe anumite zone și altele.

În vederea trimiterii unui mesaj în "chat", utilizatorii vor apăsa pe textul **"Have any questions? Post them here!"** aflat sub meniul din partea de sus a paginii de întrebări și răspunsuri după care vor introduce mesajul dorit.

Pentru ca mesajele trimise de către un cont de administrator să iasă în evidență, acestea vor avea text de culoare albă pe un fundal de culoare roșie, pe când cele trimise de către un cont de utilizator vor avea text de culoare neagră pe fundal de culoare gri.

Have any questions? [Post them here!](#)

ADMIN 2024-02-02 13:35

Cu placere!

BV20WXP 2024-02-01 20:37

Am reusit, va multumesc!

ADMIN 2024-02-01 20:36

Introduceti Secret ID-ul in aparatul de la iesire sau accesati pagina /stuck fie din bara de cautare, fie prin click pe textul: Stuck in the parking lot? de pe pagina de autentificare. Daca inca aveti probleme, creati un report la sesiunea respectiva sau contactati-ne la numarul de telefon 0763 123 456

BV20WXP 2024-02-01 20:33

Ce fac daca nu mi-a fost detectat numarul de inmatriculare iar la iesire nu mi se deschide bariera?

BV20WXP 2024-02-01 18:20

Super!

ADMIN 2024-02-01 18:20

In perioada 12-15 februarie, parcare de la etajul 1 este inchisa pentru renovare!

Figura 44: Pagina de întrebări și răspunsuri (/chat).

Mesajele trimise în acest chat nu vor fi salvate în baza de date, ci într-un fișier de tip JSON numit **"globalMessages.json"**, unde fiecare mesaj va conține un **sender**, reprezentând persoana ce a trimis mesajul, un **timestamp**, reprezentând data și ora la care a fost trimis și un **text**, fiind text-ul mesajului în sine.

6.2.8 Părăsirea parcării folosind SecretID (/stuck)

Se poate întâmpla ca la intrare, numărul de înmatriculare să nu poată fi detectat sau să fi fost incorect detectat la intrare dar corect detectat la ieșire, caz în care ieșirea nu va putea fi efectuată cu succes.

Ca alternativă la apelarea la o persoană autorizată pentru ajutor, s-a decis crearea unei pagini speciale, ce poate fi accesată fără a avea cont de utilizator, prin care un șofer să poată forța declanșarea unei acțiuni pentru o cameră video de tip EXIT. Mai exact, se va marca sfârșitul sesiunii iar bariera parcării s-ar deschide, dacă este cazul, la ieșirea dată.

Astfel, părăsirea parcării folosind SecretID se realizează prin completarea datelor necesare de pe pagina specifică ce poate fi accesată printr-un clic pe textul **"Stuck in the parking lot?"** aflat pe pagina de autentificare. Datele necesare pentru efectuarea cu succes a părăsirii forțate a parcării sunt SecretID-ul sesiunii și "Leaving Gate-ul", adică ID-ul porții de ieșire la care se află șoferul în momentul completării acestora.

7 Concluzii și dezvoltări ulterioare

7.1 Concluzii generale

În concluzie, prin această lucrare, intitulată "Sistem integrat de monitorizare și gestionare a unei parări auto cu platformă web interactivă", s-au realizat două aplicații, una care să se ocupe de detectarea și recunoașterea automată a numerelor de înmatriculare ale autovehiculelor și de gestionarea lor, și cealaltă care să cuprindă cât mai profund nevoile șoferilor de autovehicule atunci când staționează într-o parcare, precum vizualizarea timpului petrecut în parcare, locurile pe care aceștia au parcat, adresarea cu ușurință a eventualelor întrebări sau chiar sesizarea unor probleme direct de pe aplicația web. Astfel, principalele caracteristici ale celor două aplicații dezvoltate sunt: ușurința în utilizare, robustețea, extensibilitatea și folosirea de tehnologii moderne.

Pentru ca produsul final să fie unul pe măsura așteptărilor, au fost urmați anumiți pași ce au oferit rezultate bune. Deși au fost încercate mai multe variante pentru implementarea anumitor algoritmi în procesul de detectare și recunoaștere a numerelor de înmatriculare, proiectul conține doar acele metode care au oferit cele mai bune rezultate.

În acest sens, primul pas urmărit a fost cel de detectare și recunoaștere a numerelor de înmatriculare ale unui autovehicul. În primul rând, pentru detectarea plăcuței unde se află numărul de înmatriculare, s-a făcut o preprocesare, care a servit scopul de a prelucra imaginea astfel încât să fie mai ușor de identificat și segmentat plăcuța de înmatriculare. Apoi, folosindu-ne de contururi, s-a realizat o filtrare a acestora, urmând a fi ales conturul ce delimitează plăcuța de înmatriculare. Detectarea plăcuței folosind metoda prezentată a fost un succes, reușind identificarea plăcuței în aproape toate cazurile reale testate. Având în vedere evoluția și popularitatea inteligenței artificiale, am decis să adaug și o metodă prin care detectarea plăcuței să se realizeze folosind o rețea neurală convoluțională, lăsând la alegerea utilizatorului metoda dorită a fi folosită în cadrul sistemului de monitorizare și gestionare al parării auto. Mai departe, pe imaginea decupată ce conține doar plăcuța de înmatriculare, se efectuează din nou o preprocesare ce constă în egalizarea adaptivă a histogramelor și corectarea distorsiunilor geometrice, dacă este cazul, urmând aplicarea algoritmului de template matching pe fiecare caracter în parte pentru recunoașterea finală a textului de pe plăcuță.

În continuare, următorul pas a constat în realizarea interfeței grafice din cadrul aplicației, gestionarea camerelor video ce se ocupă de detectarea și recunoașterea numerelor de înmatriculare și conectarea la baza de date și manipularea și accesarea înregistrărilor din cadrul acesteia. Interfața a fost proiectată astfel încât să fie ușor de utilizat, având denumiri clare ale acțiunilor ce urmează a fi efectuate. Simplitatea interfeței grafice, aceasta având un număr mic de butoane și meniuri, este utilă, sistemul de monitorizare și gestionare a parării fiind configurat cu ușurință indiferent dacă utilizatorul acestuia a citit sau nu instrucțiunile de utilizare. De asemenea, configurarea acestuia constă în simpla adăugare a unor camere video ce vor fi de un anumit tip, intrare, ieșire sau parcare, iar cu ajutorul acestora, toate evenimentele (întrări, ieșiri, parări) din parcare sunt înregistrate într-o bază de date și sunt folosite în cea de a doua aplicație, ce constă într-un site web, unde utilizatorii vor putea vedea toate detaliile necesare legate de parcare.

În final, ultimul pas a fost dezvoltarea aplicației web unde utilizatorii să-și poată vizualiza toate detaliile legate de sesiunile lor din parcare. Aceștia pot vizualiza detalii precum, data și ora începerii sau încheierii unei sesiuni, locurile pe care a parcat în cadrul unei sesiuni dorite și timpul petrecut. Pe lângă asta, în cazul întâmpinării unor dificultăți, aceștia vor putea adresa întrebări pe un mini-chat la care vor putea primi răspuns fie de la un administrator, fie de la un alt utilizator (șofer), sau chiar întocmirea unei sesizări în cazul unei dificultăți mai grave.

7.2 Dezvoltări ulterioare

Există mai multe direcții ce prezintă potențial atât în îmbunătățirea rezultatelor detecției și recunoașterii numerelor de înmatriculare, cât și în îmbunătățirea aplicației web, oferind mult mai multe funcționalități ce ar putea veni în sprijinul șoferilor.

- Găsirea unei metode alternative pentru detectarea locurilor de parcare - În cadrul aplicației principale, să se găsească o metodă mai eficientă pentru detectarea locurilor de parcare pe care un șofer a parcat.
- Crearea unui sistem de plăți - În cadrul aplicației web s-ar putea adăuga un sistem de plăți prin care un șofer va trebui să plătească o sumă în funcție de timpul petrecut în parcare și doar după efectuarea plății să poată părăsi parcare.
- Rețele neurale pentru recunoașterea caracterelor - Antrenarea și folosirea unei rețele neurale pentru recunoașterea caracterelor din plăcuțele de înmatriculare.
- Extinderea setului de date pentru template matching - Un set de date ce conține imagini a tuturor caracterelor în majoritatea unghiurilor posibile ar putea duce la o acuratețe mai mare în recunoașterea caracterelor.
- Posibilitatea recomandării unui loc de parcare - La intrarea în parcare, șoferul să poată primi ca recomandare un loc de parcare avantajos, lângă care să nu existe alte mașini, dacă este posibil.
- Posibilitatea achiziționării de abonamente - Șoferii ar putea avea posibilitatea achiziționării unui abonament. De exemplu, dacă ora în parcare ar costa 2 RON și șoferii vor fi taxați non-stop, s-ar putea adăuga un abonament de o zi care ar putea costa 16 RON și ar fi disponibil pentru 24 ore de la data cumpărării abonamentului. După expirarea celor 24 ore, șoferul va fi taxat din nou cu 2 RON pe oră.
- Dezvoltarea unei aplicații mobile - Pe lângă aplicația web, s-ar putea dezvolta o aplicație mobile cu aceleași funcționalități ca în prezent, plus trimiterea de notificări privind timpul petrecut/rămas în parcare, expirarea abonamentelor, eventuale anunțuri și chiar realizarea plăților prin intermediul acesteia.

Bibliografie

- [1] Albatross, cplusplus.com, n.d.. <https://cplusplus.com/info/description/> [Accessed: 11.02.2024]
- [2] Desiree D. Martinez, Axl Heart P. Remegio, Darllaine Lincopinis. "A Review on Java Programming Language." May 2023. https://www.researchgate.net/publication/371166744_A_Review_on_Java_Programming_Language. [Accessed: 11.02.2024].
- [3] "SQL", Wikipedia, 2024, <https://en.wikipedia.org/wiki/SQL>. [Accessed: 20.02.2024]
- [4] Asifullah Khan, Anabia Sohail, Umme Zahoora, and Aqsa Saeed. "A survey of the recent architectures of deep convolutional neural networks.", December 2020.
- [5] OpenCV, n.d.. <https://opencv.org/> [Accessed: 26.03.2024]
- [6] "About Qt," Qt, https://wiki.qt.io/About_Qt. [Accessed 20.03.2024].
- [7] "Qt Designer Manual," Qt, n.d.. <https://doc.qt.io/qt-5/qtdesigner-manual.html>. [Accessed 20.03.2024].
- [8] "Comprehensive Guide to Ultralytics YOLOv5", Ultralytics, n.d.. <https://docs.ultralytics.com/yolov5/> [Accessed: 14.03.2024].
- [9] "Spring Framework", Spring, n.d.. <https://spring.io/projects/spring-framework> [Accessed: 30.04.2024].
- [10] "About CMake," CMake, n.d.. <https://cmake.org/overview/>. [Accessed 20.03.2024].
- [11] T. Helland, "Seven grayscale conversion algorithms," tannerhelland.com, 01.10.2011. <https://tannerhelland.com/2011/10/01/grayscale-image-algorithm-vb6.html>. [Accessed 17.02.2024].
- [12] Owotogbe Segun Joshua, Sunday Ibiyemi and B. A. Adu, "A Comprehensive Review On Various Types of Noise in Image Processing", November 2019. https://www.researchgate.net/publication/338112901_A_Comprehensive_Review_On_Various-Types_of_Noise_in_Image_Processing. [Accessed: 20.02.2024].
- [13] Kaehler, A., Bradski, G., Learning OpenCV 3: computer vision in C++ with the OpenCV library, O'Reilly Media, Editura Inc., 2016.
- [14] "Otsu's method", Wikipedia, 2024. https://en.wikipedia.org/wiki/Otsu%27s_method. [Accessed: 04.04.2024].
- [15] E. Hodneland, "Segmentation of Digital Images," pp. 17-18, 22th of July 2003.
- [16] Connected Components Labeling, n.d.. <https://homepages.inf.ed.ac.uk/rbf/HIPR2/>

[label.htm](#) [Accessed: 04.04.2024].

- [17] "Green's theorem", Wikipedia, 2024, https://en.wikipedia.org/wiki/Green%27s_theorem. [Accessed: 20.05.2024]
- [18] Roboflow Universe, n.d.. <https://universe.roboflow.com/> [Accessed: 10.04.2024]
- [19] "Object Detection using YOLOv5 OpenCV DNN in C++ and Python", LearnOpenCV, April 2022, <https://learnopencv.com/object-detection-using-yolov5-and-opencv-dnn-in-c-and-python/> [Accessed: 15.04.2024]
- [20] "Image Contrast Enhancement Using CLAHE", Analytics Vidhya, July 2024, <https://www.analyticsvidhya.com/blog/2022/08/image-contrast-enhancement-using-clahe/> [Accessed: 25.04.2024].
- [21] "Image histogram", Scientific Volume Imaging, n.d.. <https://svi.nl/ImageHistogram> [Accessed: 20.04.2024].
- [22] Qiang Wu, Fatima A. Merchant, and Kenneth R. Castleman, "Microscope Image Processing", pp. 47-54, 2022.
- [23] Kim, T.-G., Yun, B.-J., Kim T.-H., Lee, J.-Y., Park, K.-H., Jeong, Y., Kim, H.D., "Recognition of Vehicle License Plates Based on Image Processing", 7th of July 2021, <https://www.mdpi.com/2076-3417/11/14/6292> [Accessed: 27.05.2024].
- [24] "Template Matching", OpenCV, n.d.. https://docs.opencv.org/4.x/d4/dc6/tutorial_py_template_matching.html [Accessed: 24.05.2024].
- [25] Nadira Muda, Nik Kamariah Nik Ismail, Siti Azami Abu Bakar, Jasni Mohamad Zain, "Optical Character Recognition By Using Template Matching (Alphabet)" <https://core.ac.uk/download/pdf/159177553.pdf> [Accessed: 24.05.2024]
- [26] "Model-View-Controller Design Pattern", GeeksForGeeks, 19th of February 2024. <https://www.geeksforgeeks.org/mvc-design-pattern/> [Accessed: 20.05.2024].
- [27] Alain Sondrae, "Three-Layered Architecture (With Example)", 22nd of October 2022. <https://medium.com/@asoldan1459/three-layered-architecture-with-example-b597a2161538> [Accessed: 20.03.2024].
- [28] "Widgets and Layouts," Qt, n.d.. https://wiki.qt.io/Widgets_and_Layouts/ro. [Accessed: 20.03.2024].