Хмельницький національний університет
Факультет інформаційних технологій
Кафедра комп'ютерної інженерії та інформаційних систем

КВАЛІФІКАЦІЙНА РОБОТА

Галузь знань 12 – Інформаційні технології

Спеціальність 123 –Комп'ютерна інженерія

на тему «Метод та система ідентифікації номерних знаків автомобілів»

КвРКІП. 180118.22.01.26 ПЗ

Виконав: студент 2 курсу, група КІ2м-22-1 _____ Смоленюк Н.Р.
Підпис Ініціали, прізвище

Керівник канд. техн. наук, доцент _____ Грига В.М.
Науковий ступінь, вчене звання Підпис Ініціали, прізвище

До захисту допускаю:
Зав. кафедри КІІС, д.т.н., проф.
Т.О. Говорущенко
23 05 2024 р.

Хмельницький, 2024

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Інформаційних технологій

Кафедра Комп'ютерної інженерії та інформаційних систем

Освітній рівень магістр

Галузь знань 12 Інформаційні технології

Спеціальність 123 Комп'ютерна інженерія

Освітня програма освітньо-наукова програма «Комп'ютерна інженерія та програмування»

ЗАТВЕРДЖУЮ

Зав. кафедри Т.О. Говорущенко

" 01 " 09 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ МАГІСТРА

Смоленюку Назарію Руслановичу

Прізвище, ім'я, по батькові студента

1. Тема проекту (роботи) Метод та система ідентифікації номерних знаків автомобілів

Керівник проекту (роботи) Грига В.М., к.т.н., доцент

Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 01.01.2024 р. № 1

2. Строк подання студентом проекту (роботи) на кафедру 01.05.2024 р.

3. Вихідні дані до проекту (роботи) Завдання на дипломне проектування

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)

Теоретичний аналіз завдання розпізнавання автомобільних номерів

Метод та алгоритм розпізнавання номерних знаків

Розробка алгоритму розпізнавання номерних знаків

Проектування та реалізація системи розпізнавання номерних знаків

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень)

6. Консультанти розділів кваліфікаційної роботи магістра

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Лисенко С.М., професор кафедри КІІС		
Антиплагіат	Нічепорук А.О., доцент кафедри КІІС		

7. Дата видачі завдання « 01 » 09 2023р.

КАЛЕНДАРНИЙ ПЛАН

№з/п	Назва етапів (розділів) кваліфікаційної роботи магістра	Термін виконання етапів проекту (роботи)	Примітка
1	Вибір напряму дослідження та узгодження тематики КвРМ з керівником	01.09.2023	виконано
2	Ознайомлення з предметною областю; формулювання мети та задач дослідження; визначення об'єкта та предмета дослідження	01.10.2023	виконано
3	Робота над розділом 1 – аналіз відомих моделей, методів за темою; постановка задачі	01.11.2023	виконано
4	Робота над розділом 2 – розробка моделей для вирішення поставленої задачі	01.12.2023	виконано
5	Робота над науковою статтею	01.02.2024	виконано
6	Робота над розділом 3 – розробка методів для вирішення поставленої задачі	01.03.2024	виконано
7	Робота над розділом 4 – проектування та розробка ПЗ для вирішення поставленої задачі, експериментальна частина	01.04.2024	виконано
8	Оформлення пояснювальної записки згідно вимог	15.04.2024	виконано
9	Попередній захист ДРМ	26.04.2024	виконано
10	Захист ДРМ на засіданні ЕК	До 23.05.2024	

Студент _____ Підпис Смоленюк Н.Р. Ініціали, прізвище

Керівник _____ Підпис Грига В.М. Ініціали, прізвище

РЕФЕРАТ

Тема кваліфікаційної роботи: «Метод та система ідентифікації номерних знаків автомобілів»

Автор роботи: Смоленюк Назарій Русланович, студент групи КІ2м-22-1.

Керівник роботи: Грига Володимир Михайлович, канд. техн. наук, доцент.

Пояснювальна записка: 83 стр., 39 рис., 3 табл., 2 дод., 82 джер.

ПЕРЕЛІК КЛЮЧОВИХ СЛІВ: автомобільний номерний знак, система ідентифікації, система розпізнавання, алгоритм розпізнавання, програмне забезпечення, згорткові нейронні мережі.

Об’єктом дослідження є процес розпізнавання автомобільних номерних знаків на основі алгоритму який повинен бути реалізований у програмному забезпеченні. Основна увага зосереджена на розумінні складної роботи програмного забезпечення для розпізнавання номерних знаків. Це передбачає вивчення алгоритмів, методологій і технологій, які використовуються для ефективної ідентифікації та розшифровки інформації про номерні знаки.

Предметом дослідження є метод та система ідентифікації номерних знаків автомобілів, що включає оптичне розпізнавання символів, розпізнавання образів та підходи глибокого навчання.

Метою роботи є розробка алгоритму розпізнавання номерних знаків в будь-яких умовах (наприклад за умов слабкої видимості або у випадках, коли номер знаходиться не під прямим кутом, коли складно виділити межі самого номера, а також якщо масштаб номера досить маленький) автотранспортних засобів та розробка системи з програмним забезпеченням, що використовує даний алгоритм.

Наукова новизна отриманих результатів:

1) У результаті дослідження поставленої задачі набув подальшого розвитку метод розпізнавання номерних знаків на основі згорткових нейронних мереж з використанням бібліотеки Caffe, який відрізняється від інших своєю високою ефективністю та універсальністю, що дозволяє успішно розпізнавати автомобільні номерні знаки з точністю приблизно 85%;

2) Було удосконалено мікроархітектуру програмного забезпечення системи ідентифікації номерних знаків, яка на відміну від інших, дозволяє успішно розпізнавати автомобільні номерні знаки за короткий час до 0,3 секунди та із більш високою точністю, яка становить приблизно 90%.

Практична значущість отриманих результатів:

Метод та система мають безпосереднє практичне застосування в різних областях, включаючи правоохоронні органи, управління дорожнім рухом, збір плати та управління паркуванням.

Правоохоронні органи можуть використовувати цю систему для автоматичного виявлення транспортних засобів, причетних до злочинних дій або порушень правил дорожнього руху, для підвищення громадської безпеки.

В управлінні дорожнім рухом система полегшує моніторинг руху транспортних засобів у режимі реального часу, дозволяючи органам влади оптимізувати дорожній потік, виявляти затори та ефективно контролювати дотримання правил дорожнього руху.

Системи збору плати за проїзд виграють від здатності системи швидко ідентифікувати транспортні засоби, забезпечуючи безперебійні та ефективні процеси оплати.

Системи керування паркуванням можуть використовувати систему для моніторингу паркувальних місць, виявлення несанкціонованих транспортних засобів і оптимізації паркувальних операцій у міських районах. Загалом практичне значення отриманих результатів виходить за межі академічних досліджень, впливаючи на різні сектори та сприяючи розвитку суспільства.

Рекомендації з використання результатів роботи:

Результати, досягнуті в даній роботі включають нові методи та технології, що можна впроваджувати в існуючі системи розпізнавання номерних знаків для підвищення точності та ефективності, наприклад: впровадження в інфраструктуру безпеки, інтеграція в систему «розумного міста», подальші дослідження роботи прорамного забезпечення, оцінка та ітерація розгорнутих систем для їх удосконалення.

Важливість роботи і висновки:

Підсумовуючи, у цій роботі висвітлюється наукова новизна, досягнута в розпізнаванні номерних знаків завдяки впровадженню інноваційних технологій. Гібридний підхід до глибокого навчання та вдосконалена архітектура, представлені в даній роботі представляють значні досягнення в галузі, пропонуючи підвищену точність, ефективність та адаптивність порівняно з існуючими методами. Такі інновації відкривають шлях до розробки надійніших систем розпізнавання номерних знаків з широким застосуванням у багатьох галузях та сферах.

Публікації. За темою кваліфікаційної роботи опубліковано одну публікацію.

Структура та об’єм кваліфікаційної роботи. Кваліфікаційна робота складається з вступу, чотирьох розділів, висновку та додатків, її повний зміст сторінок, основний зміст викладено на 83 сторінках, 2-х додатках, містить 39 рисунків, 3 таблиці, включає 82 найменування вітчизняної та зарубіжної літератури.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

РЗТЗ – реєстраційний знак транспортних засобів

OCR – Optical Character Recognition

XML – eXtensible Markup Language

ReLU – Rectified Linear Unit

CPU – Central Processing Unit

GPU – Graphics Processing Unit

CNN – Convolutional Neural Network

LPR – Licence Plate Recognition

OpenCV – Open Source Computer Vision Library

Caffe – Convolution Architecture For Feature Extraction

ANPR – Automatic Number Plate Recognition

АРНЗ – автоматичне розпізнавання номерних знаків

ВСТУП

Завдання розпізнавання автомобільних номерних знаків користується попитом у програмному забезпеченні для контролю в'їзду та виїзду транспортних засобів з території підприємств, паркувань, контролю потоку автотранспорту. Дане програмне забезпечення може бути розміщене в автосервісах, контрольно-пропускних пунктах, пунктах контролю швидкості [1].

Ця магістерська робота присвячена розробці та реалізації алгоритму розпізнавання автомобільних номерів. Завдання розпізнавання автомобільних знаків можна розділити на два підзавдання: пошук номерного знаку та розпізнавання символів номерного знаку. В основному розпізнавання відбувається у три етапи: передобробка зображення, сегментація та безпосередньо розпізнавання символів.

Передобробка зображення включає виділення номерної пластини і обробку спеціальними фільтрами, щоб покращити якість. За допомогою етапу сегментації виділяються символи, після чого відбувається розпізнавання будь-яким методом.

Об’єктом дослідження є процес розпізнавання автомобільних номерних знаків на основі алгоритму який повинен бути реалізований у програмному забезпеченні. Основна увага зосереджена на розумінні складної роботи програмного забезпечення для розпізнавання номерних знаків. Це передбачає вивчення алгоритмів, методологій і технологій, які використовуються для ефективної ідентифікації та розшифровки інформації про номерні знаки.

Предметом дослідження є метод та система ідентифікації номерних знаків автомобілів, що включає оптичне розпізнавання символів, розпізнавання образів та підходи глибокого навчання.

Центральним у цьому дослідженні є спроба вдосконалити технологію, що лежить в основі розпізнавання автомобільних номерних знаків, спираючись на обробку зображень і алгоритми розпізнавання символів.

Прийнята методологія передбачає формулювання алгоритму, розробленого для вирішення проблем, пов’язаних із умовами поганої видимості, такими як похиле розміщення номерів або затемнені межі табличок.

Розробка цього алгоритму доповнюється створенням програмного забезпечення, готового використовувати його можливості.

Метою роботи є розробка алгоритму розпізнавання номерних знаків в будь-яких умовах (наприклад за умов слабкої видимості або у випадках, коли номер знаходиться не під прямим кутом, коли складно виділити межі самого номера, а також якщо масштаб номера досить маленький) автотранспортних засобів та розробка системи з програмним забезпеченням, що використовує даний алгоритм.

Дана дипломна робота представляє комплексне дослідження сфери технології розпізнавання автомобільних номерних знаків. Поєднуючи досягнення в обробці зображень і розпізнаванні символів разом із інтеграцією штучного інтелекту та нейронних мереж, дослідження спрямоване на створення надійних алгоритмів, здатних точно ідентифікувати номерні знаки в різноманітних і складних умовах. Такі досягнення обіцяють підвищення безпеки, ефективності та контролю в безлічі сценаріїв управління транспортними засобами.

Для досягнення поставленої мети необхідно:

- ознайомитись з методами розпізнавання тексту на зображеннях;
- розробити алгоритм для розпізнавання номерних знаків автотранспортних засобів за умов слабкої видимості;
- розробити програмне забезпечення, що використовує цей алгоритм.

За темою дипломної роботи опубліковані тези доповіді (додаток А), також взято участь у IV Міжнародній науково-практичній конференції «Прикладні науково-технічні дослідження» (14-16 травня, м. Івано-Франківськ, Україна:

1.) Смоленюк Н.Р. Метод та система ідентифікації номерних знаків автомобіля. VI міжнародна науково-практична конференція “Прикладні науково-технічні дослідження” – Івано-Франківськ, Україна, 2024, с. 236-238.

1 ТЕОРЕТИЧНИЙ АНАЛІЗ ЗАВДАННЯ РОЗПІЗНАВАННЯ АВТОМОБІЛЬНИХ НОМЕРІВ

Існує велика кількість промислового програмного забезпечення для розпізнавання автомобільних номерів. Цей розділ пропонує детальний аналітичний огляд деяких програмних продуктів, систем розпізнавання номерних знаків а також методів обробки зображень. Опис кожного продукту містить інформацію про назву, область застосування та технології розпізнавання, що в ній використовуються.

1.1 Дослідження предметної області

Реєстраційні знаки транспортних засобів України – це розпізнавальні знаки, які кріпляться у спеціально встановлених місцях на транспортних засобах а також причепах. Їхнє призначення - реєстрація транспортних засобів в уповноваженому органі Міністерства внутрішніх справ [2].

На сьогодні в Україні існує 15 видів номерних знаків (понад двадцять підвидів). В основному номерні знаки мають українську літературу, яка має графічні відповідники латинським алфавітом (всього 12: A, B, C, E, H, I, K, M, O, P, T, X) а інші типи знаків використовують такі літери: A, B, Г, E, I, K, M, H, O, P, C, T, Ф, X, Ч, Ю, Я [3].

Перелік основних і додаткових типів номерних знаків:

1) знаки для всіх типів автомобілів, причепів, автобусів, розмір 520×112 мм, білий фон, чорні символи, комбінація букв зліва означає регіон, праворуч - серію номерного знаку;

2) номерні знаки для електромобілів, зелені символи;

3) номерні знаки для маршрутних автобусів, мікроавтобусів і таксі, розмір 520x112 мм, жовтий фон, чорні символи, комбінація букв зліва позначає регіон, праворуч - серію номерного знаку;

4) номерні знаки тимчасового користування на легкові та вантажні автомобілі, автобуси та мікроавтобуси, причепи, напівпричепи та напівпричепи, розмір 520х112 мм, червоний фон, білі символи;

5) номерні знаки, видані комерційним підприємством у тимчасове користування на легкові та вантажні автомобілі, автобуси та мікроавтобуси, причепи, напівпричепи та напівпричепи, розмір 520х112 мм, червоний фон, білі символи;

6) номерні знаки транспортних засобів з об’ємом двигуна до 50 см³, розмір 140×114 мм, білий фон, чорні символи;

7) номерні знаки для тимчасового використання на транспортних засобах з об'ємом двигуна до 50 см³, розмір 140х114 мм, червоний фон, білі символи;

8) номерні знаки, видані комерційним підприємством для тимчасового використання на транспортних засобах з об’ємом двигуна не більше 50 см³, розмір 140х114 мм, червоний фон, білі символи;

9) номерні знаки на автотранспортні засоби дипломатичних місій, консульств, представництв міжнародних організацій та членів сімей їх співробітників, розмір 520×112 мм, білий фон, чорні символи;

10) номерні знаки для мотоциклів, скутерів і скутерів, розмір 220х174 мм, білий фон, чорні символи;

11) номерні знаки для тимчасового використання на мотоциклах, мопедах і скутерах, розмір 220х174 мм, червоний фон, білі символи;

12) номерні знаки, видані підприємством для тимчасового користування на мотоциклах, скутерах і моторолерах, розмір 220х174 мм, червоний фон, білі символи;

13) номерні знаки, виготовлені за індивідуальним замовленням для автотранспорту. В Україні ситуація із зображеннями на індивідуальних номерних знаках досить унікальна: власник транспортного засобу обирає зображення на власний розсуд, в той час як, наприклад, у США існує процедура ініціативної реєстрації групи для кожного окремого тиражу, яка стає доступною для всіх бажаючих;

14) номерні знаки для мотоциклів на замовлення, розмір 220х174 мм, білий фон, чорні символи;

15) номерні знаки спеціальної форми для автомобілів американського або японського виробництва, розмір (від 220 до 400 мм) або (від 110 до 320 мм), білий фон, чорні символи;

16) номерні знаки спеціальної форми, виготовлені за індивідуальним замовленням для автомобілів зарубіжного виробництва, розміри (від 220 до 400 мм) або (від 110 до 320 мм), білий фон, чорні символи;

17) номерні знаки тракторів та причепів до них, що використовуються в сільському господарстві, розмір 288×226 мм, нижній край 140 мм, білий фон, чорні символи.

18) знаки для всіх типів автомобілів, автопричепів, автобусів, самохідних машин і механізмів Збройних Сил України, Держспецзв'язку, Державної служби спеціального зв'язку та захисту інформації України;

19) знаки для мотоциклів, скутерів, моторолерів, квадроциклів, трициклів та інших прирівняних до них транспортних засобів, тягачів і тягачів Збройних Сил України, Держспецзв'язку, Державної служби спеціального зв'язку та захисту інформації України.

Сполучення літер на номерних знаках. Знаки типу 1 мають містять:

- дві літери зліва це інформація про адміністративно-територіальну належність;

- чотири цифри по середині це порядковий номер;

- дві літери праворуч це серія.

Коди адміністративних регіонів [4].

З 2004 року було розподілено коди регіонів з використанням літер, що збігаються в латинській та українській абетках, починаючи з Києва: АА, АВ, АС тощо, в інтервалах АА–АХ, ВА–ВО, ВТ–ВХ, СА–СВ, СЕ–СН.

Окремим кодом, що не входив до жодного інтервалу, стала серія ІІ. З 2006 року серія ІІ була виключена і підлягала вилученню.

З 2013 року змінено якими кодами регіонів список доповнюється з серіями в інтервалах НА–НХ, ІА–ІН, КА–КХ, за виключенням серій НР та ІС, які не мали аналогів у кодуванні з 2004 року.

З 2021 року кожному регіону України було додано ще по два коди регіону.

На рисунку 1.1 показано типи номерних знаків.

Рисунок 1.1 – Типи номерних знаків [5]

На рисунку 1.2 показано територіальний розподіл кодів регіонів реєстрації транспортних засобів.

Рисунок 1.2 – "Географія" кодів номерних знаків [6]

Права комбінація літер (типи 1 та 2) має видаватися в порядку латинської абетки (з використанням лише літер, що збігаються в латинській та кириличній абетках): з початку абетки – для автомобілів (AA, AB, AC і т. д.), з кінця – для причепів та напівпричепів (XX, XT, XP, тощо.) [7].

1.2 Огляд існуючих систем розпізнавання автомобільних номерів

До найпопулярнішого в Україні програмного забезпечення для розпізнавання автомобільних номерів належать:

- Програмне забезпечення «НОМЕРОК MMCR 6»;
- Програмне забезпечення «НОМЕРОК SMB 9»;
- Програмне забезпечення «НОМЕРОК Lite 12»;
- Програмне забезпечення «Vector»;
- Програмне забезпечення «SecurOS Auto»;
- Програмне забезпечення «Nomeroff Net»;
- ПЗ та Модуль керування RM-44 (БАРБОС);
- ПЗ та Контролер ATIS RME-22.

1.2.1 Система розпізнавання «НОМЕРОК Lite 12»

«НОМЕРОК Lite 12» [8] це програма для розпізнавання автомобільних номерних знаків з функцією відправлення результатів детекції до зовнішніх додатків.

Принцип роботи «НОМЕРОК Lite» наступний: при фіксації автомобільного номера у відеопотоці, програма НОМЕРОК зберігає скріншот транспортного засобу та робить запис до бази даних про подію (дата/час) з результатами розпізнавання.

Кожна версія програмного забезпечення Lite включає модуль розпізнавання автомобільних номерів та передбачає можливість подальшої інтеграції із системами контролю доступу, 1С, ваговими та/або іншими додатками.

«НОМЕРОК Lite» легко інтегрується з іншими додатками, використовуючи базу даних безкоштовного формату (FireBird). Також можна отримувати дані з сервера цього програмного забезпечення в режимі реального часу.

Країни, номерні знаки яких розпізнаються програмою «НОМЕРОК Lite»: Україна, Білорусь, Молдова, Євросоюз, Туреччина, Ізраїль.

До основних відомостей роботи ПЗ «НОМЕРОК Lite» можна віднести:

- сумісна з будь-якою версією ОС Windows;
- висока якість розпізнавання – до 95%;
- робота з будь-якими IP-камерами та аналоговими системами відеоспостереження, що підтримують передачу RTSP-потоку;
- розпізнавання номера при швидкості автомобіля до 250 км/год;
- фіксує розпізнаний номер, зберігає скріншот в базу.

1.2.2 Система розпізнавання «Vector»

Система розпізнавання автомобільних номерів Vector [9] це апаратно-програмний комплекс, що складається з камер відеоспостереження, центрального комп`ютера та спеціального програмного забезпечення (ядро розпізнавання) для аналізу відеопотоку.

Дана система розпізнавання номерних знаків застосовується в різних сферах, де, крім відеоспостереження, важливою є здатність контролювати потік руху автомобілів:

- автомобільні паркування;
- платні автомобільні дороги;
- організація роботи автомийок;
- КПП на комерційних підприємствах;
- житлові комплекси та заміські будинки.

За технологією програмного забезпечення системи «Vector» можна:

- визначати час перебування автомобіля на платному паркінгу;
- підраховувати час та вартість простою автомобільного транспорту;

- реєструвати автомобільний потік на мийці, тим самим виключити платежі повз касу;
- контролювати потік автомобілів через прохідні великих підприємств.

Таким чином, встановлення системи розпізнавання номерних знаків актуальне як для приватних рішень, так і для організації повноцінної СКУД з подальшою інтеграцією з тривожними та виконавчими засобами.

1.2.3 Система розпізнавання «SecurOS Auto»

Система інтелектуального відеоаналізу SecurOS Auto [10] це система, що забезпечує розпізнавання державних номерів транспортних засобів.

SecurOS Auto функціонує на базі інтеграційної платформи відеоменеджменту SecurOS, що дозволяє створити комплекс безпеки з необхідним замовнику функціоналом.

Область застосування SecurOS Auto охоплює значне коло завдань від забезпечення безпеки на парковках до контролю потоків машин у масштабах міста.

Можливості системи SecurOS Auto:

- розпізнавати номерні знаки понад 50 країн світу;
- розпізнавати номери за окремими кадрами без використання відео;
- швидко налаштувати оновлення для розпізнавання нових стандартів;
- формувати базу даних розпізнаних номерів зі збереженням супутньої інформації про дату, час і місце виявлення автомобіля, швидкість і напрямок його руху, а також посилання на відеофрагмент;
- організувати пошук розпізнаних раніше номерів за заданими параметрами;
- підтримувати одночасно зовнішні та внутрішні списки номерів (інформаційні, білі, чорні);
- автоматично перевіряти розпізнаний реєстраційний знак транспортних засобів за внутрішніми та зовнішніми списками (базами даних);

- налаштувати необхідні реакції системи на розпізнавання номера, результати пошуку;
- оперативно інформувати оператора (підтримується, в тому числі, і голосове сповіщення) та/або надіслати повідомлення (email, SMS та ін.) зовнішнім службам про результати розпізнавання номера та/або про результати зіставлення даних про автомобіль зі списками номерів;
- автоматично генерувати звіти різних видів на основі результатів розпізнавання, пошуку баз даних.

Модуль локалізації формує гіпотези про можливу присутність номерної пластини на зображенні (у кадрі) та коректно обробляє ситуації знаходження у кадрі кількох номерних пластин. Аналіз заснований на пошуку ділянок кадру, що мають специфічну для тексту структуру перепадів яскравості. У налаштуваннях режиму розпізнавання можна вказати, чи потрібно шукати номерні знаки у нерухомих зонах кадру.

Положення номерної пластини в кадрі передбачається не тільки на основі аналізу поточного зображення, але й шляхом екстраполяції результатів розпізнавання на попередніх кадрах.

Масштабування та бінаризація зображення. Розпізнавання реєстраційних знаків транспортних засобів здійснюється у кілька етапів аналізів двоградаційного (бінаризованого) та напівтонового зображень.

Область кадру, вказана локалізатором, масштабується і потім бінаризується за допомогою власного алгоритму. В результаті виникає зручне для швидкого аналізу зображення, що складається лише з білих та чорних пікселів.

На етапі аналізу бінарного зображення відбувається виявлення та розпізнавання символів номера, їх вибудовування в послідовність знаків, а також пошук символів, що відсутні, виходячи з номерних шаблонів конкретної країни.

Порівняння зображення з набором шаблонів дозволяє уникнути помилкового розпізнавання «артефактів», визначення неправильної кількості символів у номері, неправильного розпізнавання символів, що близько розташовані один до одного, тощо.

У SecurOS Auto використовуються шаблони, що враховують геометрію розташування символів на номерних знаках. Як правило, розпізнавання трьох символів у номері достатньо, щоб коректно накласти на зображення шаблон. Це дозволяє визначити місця розташування інших символів та розпізнати їх із високим ступенем ймовірності. Шаблони містять інформацію про допустимі послідовності букв і цифр, їх кількість, про фізичні розміри номерів - співвідношення довжини та ширини рамок, наявність зображення прапора держави або інших графічних елементів.

SecurOS Auto дозволяє оперувати одночасно майже необмеженою кількістю наборів шаблонів ГРЗ різних держав, проте для впевненого розпізнавання в темпі надходження відео рекомендується використовувати шаблони не більше ніж п'яти-шести країн.

Реалізований аналіз напівтонового зображення дозволяє суттєво підвищити точність розпізнавання. Це особливо важливо, якщо необхідно зробити вибір між декількома «конкуруючими» варіантами, що мають подібне написання, ступінь відмінності яких може стати ще меншим через малі розміри символу, низький контраст, наявність перешкод тощо.

Уточнення результату розпізнавання, тобто повторне розпізнавання проводиться залежно від результатів першого проходу. Вихідне локалізоване зображення обробляється повторно з іншими параметрами контрастності, зміною масштабу та іншими видами нормалізації залежно від виявленої проблеми.

Міжкадрове злиття результатів розпізнавання та видача остаточного результату. Розпізнавання номера проводиться на всіх кадрах, де його розмір та контраст укладаються в задані межі, а результати передаються в модуль «Простеження» для міжкадрового зіставлення, передбачення траєкторії номера на наступних кадрах відеоряду, а також фінального результату (фіксації номера).

Для стабільно хороших результатів достатньо, щоб номерний знак було видно як мінімум на трьох-п'яти кадрах. Наприклад, одна камера 25-30 кадр/сек, що контролює зону дорожнього полотна в 8 метрів завдовжки, дозволяє отримати чотири і більше кадрів автомобіля, що рухається зі швидкістю 120-180 км/год.

Такої кількості кадрів зазвичай достатньо впевненого розпізнавання всіх символів номерного знаку. Якість та швидкість розпізнавання також залежать від багатьох показників, таких як якість оптики та витримка камери, достатність освітленості в зоні спостереження, процесорна потужність сервера відео аналітики, тощо.

Експертна оцінка проекту та тестування, що передують створенню проекту, дозволяють зберігати показники розпізнавання на рівні 96% – 99,9%.

Остаточний результат розпізнавання видається у вигляді «події», яка доступна всім модулям у складі «SecurOS». Залежно від налаштувань користувача, «SecurOS» зберігає весь фрагмент проїзду ТЗ або його «найкращий кадр», визначений на основі проміжних результатів розпізнавання.

«SecurOS Auto» має кілька режимів видачі результату розпізнавання, які залежать від налаштувань користувача. Два з них, - «дорога» і «парковка», припускають, що камеру встановлено стаціонарно, а третю, - «мобільну», - що камеру встановлено в патрульному автомобілі, що рухається, наприклад.

У режимі «дорога» розпізнаються лише номери, що рухаються, а результат видається, коли номер виїжджає за межі кадру або у разі його «втрати» в кадрі (звичайно в результаті заслонення).

У режимі «парковка» розпізнаються також і нерухомі номери, а результат видається і у разі зупинки транспортного засобу, яке раніше знаходилося в русі. У разі «SecurOS Auto» визначає, коли машина з розпізнаним номером «виїхала з кадру»; ця подія може бути передана в систему керування шлагбаумом. Збереження результатів розпізнавання у базі даних та зіставлення зі списками номерів та перевірка перевищення швидкості.

Результат перевірки, наприклад, присутність даного номера в базі угонів або перевищення дозволеної швидкості руху по даній смузі, також зберігається у власній базі, і також посилаються у вигляді події в «SecurOS». У результаті «тривожний» рядок з'являється в інтерфейсі користувача «SecurOS Auto».

1.3 Тестування існуючих систем розпізнавання автомобільних номерів

У статті [76] описані тести, що були проведені на системах розпізнавання автомобільних номерів, взяті ними з різних джерел. Методика проведення тестів, достатній обсяг тестового матеріалу, присутність під час тестування зацікавлених у використанні даних систем користувачів виключають суб'єктивність чи неоднозначність тлумачення результатів.

Стаття [79] представляє результати аналізу, де вводиться поняття «точності розпізнавання», K. Точність системи пропорційна кількості правильно розпізнаних номерів T і обернено пропорційна загальній кількості зображень - F. Таким чином міра точності розпізнавання може бути обчислена за формулою 1.1:

$$K = \wedge \text{x } 100. \qquad (1.1)$$

У таблиці 1.1 наведено порівняльний аналіз популярних систем, які представлені на ринку програмного забезпечення з розпізнавання автомобільних номерів.

Таблиця 1.1 – Узагальнена таблиця результатів тестування

Назва системи	Кількість правильно розпізнаних номерів (T), %
НОМЕРОК Lite 12	95,45
Vector	90,05
SecurOS Auto	88,69
Auto Trassir	83,07
CVS Авто	82,09
Nomeroff Net	80,33
Overseer Traffic	79,89
MegaCar	77,96

1.4 Алгоритми та методи обробки зображень

В основі кожної системи розпізнавання лежить певний метод обробки зображень. Існує безліч методів та алгоритмів обробки зображень, деякі з них, які були розглянути для потенційного використані в даній роботі, описані далі.

1.4.1 Фільтр Гауса

Фільтр Гауса [12] називається електронний фільтр, у якого функція Гауса є імпульсною перехідною функцією. Фільтр Гауса задуманий так, щоб постійна часу була максимальною і не було перерегулювання у функції переходу. Така поведінка пов'язана з тим, що фільтр Гауса має мінімальну можливу затримку.

Фільтр Гауса зазвичай використовується у цифровому вигляді для обробки двовимірних сигналів (зображень) з метою зниження рівня шуму.

1.4.2 Детектор Кенні

Алгоритм Кенні, виявляч контурів Кенні, оператор Кенні (англ. Canny edge detector) [16] це оператор виявляння контурів, який використовує багатоетапний алгоритм для виявляння широкого спектру контурів на зображеннях. Його розробив Джон Кенні 1986 року. Кенні також розробив обчислювальну теорію виявляння контурів (англ. computational theory of edge detection), яка пояснює, чому ця методика діє.

У комп'ютерному зорі детектор кордонів Кенні служить оператором виявлення меж об'єктів на зображенні. У детекторі Кенні використовується багатоступінчастий алгоритм для того, щоб можна було знайти широкий спектр кордонів зображення.

Хоча його робота була проведена на базі комп'ютерного зору, детектор кордонів Кенні досі є одним із найкращих детекторів. Крім особливих окремих випадків важко знайти детектор, який би працював краще, ніж детектор Кенні.

До основних етапів даного алгоритму можна віднести:

- зглажування;
- пошук градієнтів;
- пригнічення немаксимумів;
- подвійна порогова фільтрація;
- трасування області неоднозначності.

1.4.3 Розпізнавання скелетних образів

Насамперед розпізнаваний символ піддається процедурі скелетизації. Існують різні методи отримання скелета символу, відмінні один від одного [17].

Для кожного зовнішнього і внутрішнього контуру зображення знаходяться початкові верхні ліві точки. Для чергової точки контуру розглядається конфігурація восьми її сусідів. Точка віддаляється, якщо вона не є кінцевою, і якщо після її видалення її сусіди як і раніше утворюватимуть зв'язну множину.

Після аналізу точки і її сусідів і можливого видалення точки здійснюється перехід до наступної точки контуру так, щоб залишитися на межі зображення. Далі крок за кроком віддаляється один шар точок. Шари віддаляються до тих пір, поки не залишаться точки, що тільки не видаляються (рисунок 1.3).

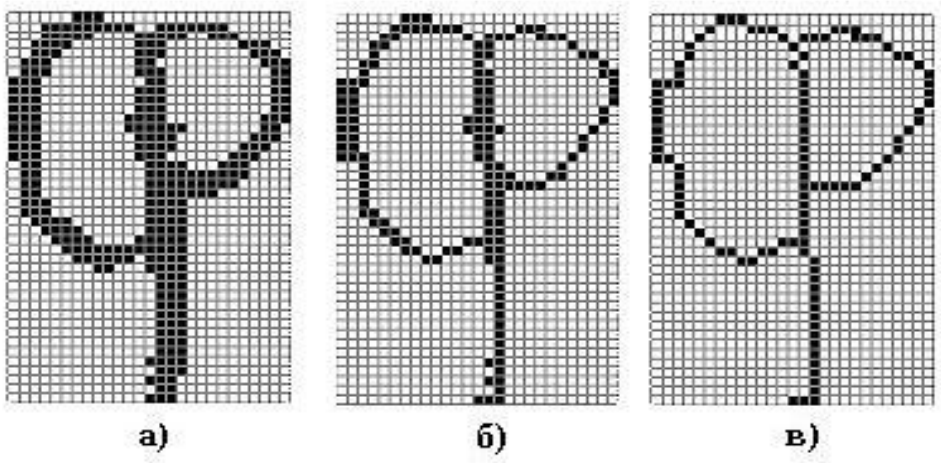


Рисунок 1.3 – Оптимізація точки з'єднання відрізків (деталізація а,б,в)

1.4.4 Скелетизація із застосуванням шаблонів

Для отримання скелетного зображення використовуються шаблони, призначені для видалення зайвих пікселів, де знаком «X» відмічені пікселі будь-якого кольору (рисунок 1.4). У будь-якій області, відповідній одному з шаблонів, віддаляється чорний центральний піксель.

Здійснюється декілька проходів по зображенню, поки не залишиться пікселів, що підлягають видаленню.

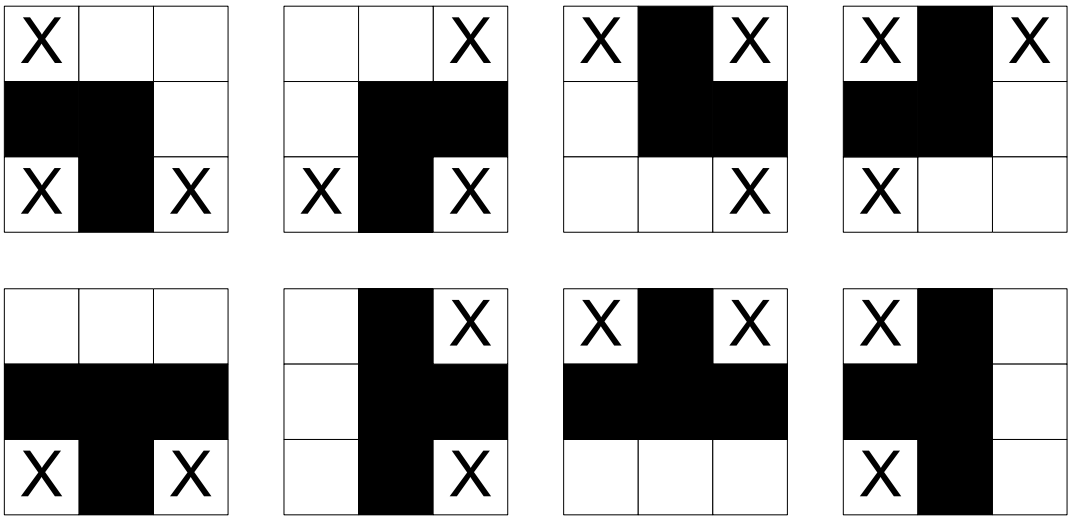


Рисунок 1.4 – Шаблони скелетизації

Для підготовки до етапу розпізнавання в отриманому скелеті виділяємо ключові точки: точки з'єднання три або чотири ребра скелета і кінцеві крапки:

1. Створюємо порожній стек для зберігання координат початку і кінця ребер, точок галуження скелета.

2. Заносимо в нього будь-яку точку скелета.

3. Поки стек не порожній, продовжуємо кроки 4-7.

4. Вибираємо крапку із стека.

5. Будуємо послідовність ребер з вибраної точки зображення, поки не відбудеться галуження скелета, бо не досягнемо кінцевої крапки.

6. Якщо досягли кінцевої крапки або досягли поміченого раніше ребра, то в масив заноситься пройдений шлях.

7. Якщо відбулося галуження скелета, то ми знайшли місце з'єднання ребер, і в масив заноситься послідовність ребер. У стек заносимо точку галуження.

8. Переходимо до пункту 3.

У отриманому описі скелета проводиться передобробка, що огрублює, яка полягає у видаленні коротких ліній і об'єднанні близьких тріодів.

1.4.5 Хвильовий метод

Даний метод [14] полягає в аналізі шляху проходження сферичної хвилі по зображенню (рисунок 1.5). На кожному кроці аналізується зсув центру мас крапок, створюючих нову генерацію хвилі, щодо його попередніх положень.

Метод складається з наступних кроків:

- побудова скелета зображення за допомогою сферичної хвилі;
- оптимізація отриманого скелета.

Відстежування ліній зображення проводиться шляхом відстежування переміщення центру відрізання, що утворюється крайніми точками генерації хвилі. Після відстеження можливе згладжування відрізки.

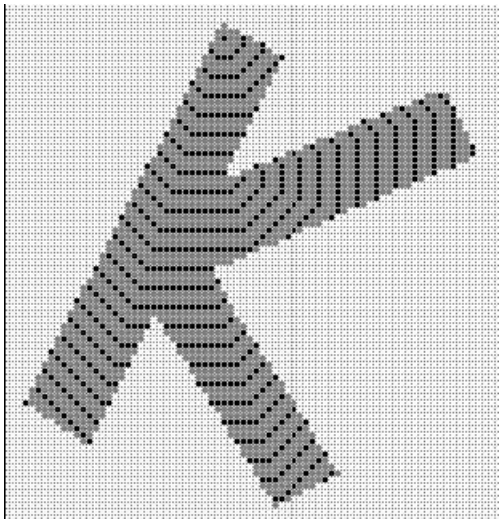


Рисунок 1.5 – Проходження сферичної хвилі по зображенню

Виявлення збільшення «ширини» хвилі і розділення хвилі на дочірнє дозволяє встановити точку передбачуваного з'єднання двох відрізків.

Визначення збільшення «ширини» хвилі проводиться шляхом порівняння «ширини» чергової генерації хвилі і її середнього значення за N попередніх генерацій (N задається заздалегідь). Причому ми отримуємо дві точки (A,B) трасованого відрізання. Після розділення хвилі на дві півхвилі, ми отримуємо ще дві пари крапок (C,D) і (E,F). Точка з'єднання відрізків лежить в шестикутнику ABCDEF і спочатку встановлюється як центр мас багатокутника (рисунок 1.6).

Корекція покладається на оптимізацію скелета зображення.

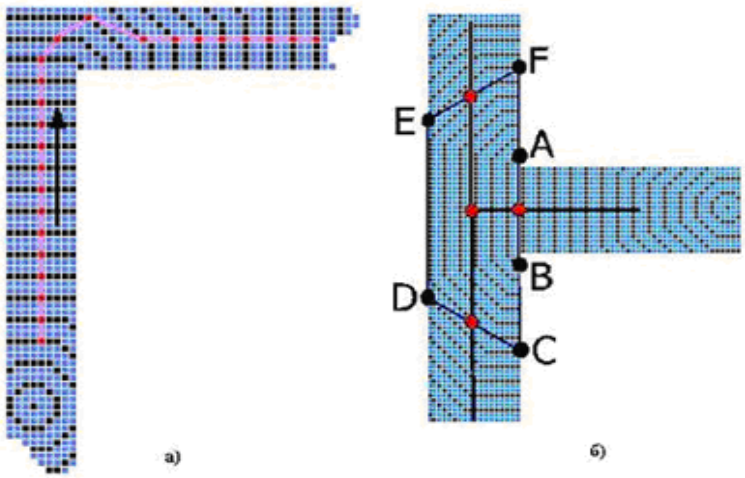


Рисунок 1.6 – Відстеження ліній зображення: (а) визначення місця; (б) з'єднання відрізків.

Отриманий скелет зображення не є оптимальним. Це зв'язано, перш за все з тим, що ми маємо справу з растровим зображенням, а значить, зображення має спотворення тим більші, чим менше розмір зображення в символах.

Для зменшення впливу спотворень на отримуваний скелет необхідно провести його оптимізацію. У отримуваному скелеті можливе представлення одного відрізання деякою послідовністю ребер. Позбавитися від цього можна аналізом послідовності ребер, оцінювання лінії, що виходить, від прямої. У випадку якщо відхилення знаходиться в допустимих межах, необхідно послідовність ребер замінити на одне.

Для оптимізації скелета є видимими околиці виділених точок з'єднання відрізків, тобто таких крапок, де спостерігається розділення хвилі на півхвилі.

Найбільш часто зустрічаються спотворення (рисунок 1.7, а), що виправляються за допомогою аналізу прилеглих до виділеної крапки (А) відрізків (АВ1, В1С1, АВ2, В2С2, АВ3, В3С3). Аналіз полягає в пошуку такої пари відрізків СхВх, СуВу із (В1С1, В2С2, В3С3), що Сх Вх Су Ву максимально корелюються прямої. Тоді необхідно крапку А перемістити в крапку перетину прямих СхСу і АС2, а потім видалити з графа крапки В1,В2,В3 (рисунок 1.7, б).

Іншим варіантом спотворення є випадок з'єднання трьох відрізків в одній крапці (рисунок 1.7, в). В цьому випадку неможливе знаходження пари відрізків корельованих прямою. Точка А має бути перенесена в центр трикутника, що утворюється прямими В1С1, В2С2 і В3С3. Потім крапки В1, В2 і В3 необхідно видалити з графа (рисунок 1.7, г).

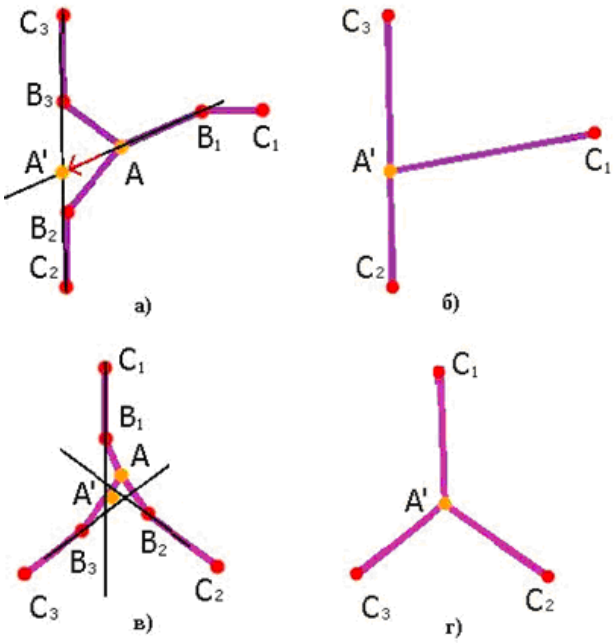


Рисунок 1.7 – Оптимізація точки з'єднання відрізків (деталізація а,б,в,г)

1.4.6 Саморозпізнавання

Для кожної особливої точки отриманого скелетного уявлення обчислюється множина топологічних ознак, основними з яких є:

- нормовані координати особливої точки (вершина графа);
- нормоване направлення з даної крапки на наступну особливу крапку;

- нормований напрям входу в крапку, виходу з крапки;
- довжина ребра до наступної вершини у відсотках від довжини графа;
- кривизна дуги, точніше "ліва" і "права" кривизна дуги, що сполучає особливу крапку з наступною вершиною (кривизна зліва і справа).

Кривизна обчислюється як відношення максимальної відстані від точок дуги (що знаходяться відповідно зліва/справа від прямої) до прямої, що сполучає вершини, до довжини відрізка, що сполучає ті ж вершини.

На рисунку 1.8 умовно показані деякі з топологічних ознак. Граф має п'ять особливих крапок – a0, a1, a2, a3, a4.

При обході графа по маршруту a0 ==> a1 ==> a2... у вершині a1 умовно показані наступні ознаки: вектор R1 – напрям входу в точку, вектор R2 – направлення виходу з крапки, вектор R3 – глобальний напрям на наступну особливу крапку. Двонаправлений вектор h показує величину "лівого" відхилення дуги (a1,a2) від прямої, а "праве" відхилення дорівнює нулю.

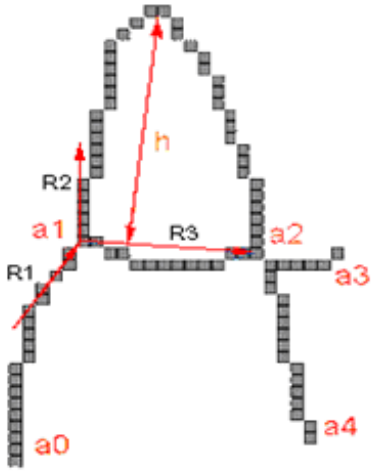


Рисунок 1.8 – Приклад топологічних ознак

Для деяких код число особливих крапок і, відповідно, число топологічних ознак дуже мало. Так, для коду, відповідного символу "0", топологічних ознак взагалі немає, оскільки немає жодної особливої крапки.

Тому можуть обчислюватися і використовуватися наступні додаткові ознаки:

- "чорна" і "біла" ширина верхньої половини символу;
- розміри і положення компонент і дірок;
- модифіковані прямі прогину.

Прогини обчислюються як відстані від точок скелетного уявлення до опуклої оболонки побудованого уявлення. Додатково запам'ятовується положення точок максимального прогину.

Для деяких топологічних кодів число топологічних ознак може бути достатньо велике, що може зажадати дуже великого набору еталонів для навчання, тому у ряді випадків в розпізнаванні використовується частина ознак. Символ визначається після порівняння його опису з кодами з бази даних, при цьому вибирається найближчий топологічний код.

Якщо символ після проходу циклу розпізнавання залишився нерозпізнаним, робиться спроба покращення зображення за допомогою наступних операцій:

- склеїти кінці ліній по напрямах (рисунок 1.9), для цього розглянути напрями всіх кінцевих дуг скелетного уявлення, і якщо напрями яких-небудь ліній схожі (з точністю до знаку), і вказують один на одного, можна спробувати їх з'єднати - можливо, це була суцільна лінія, що розірвалася унаслідок недостатнього рівня сканування або початкових дефектів написання;

- відкинути найкоротшу лінію (дугу графа), зайві короткі дуги (лінії) нерідко виникають при рукописному написанні;

- склеїти точки скелета, що знаходяться на мінімальній відстані одна від одної.

Рисунок 1.9 – Приклад топологічних ознак: а) початкове зображення символу; б) символ із склеєними лініями.

1.4.7 Розпізнавання ABBYY

ABBYY [18] є провідним постачальником програмного забезпечення для розпізнавання документів, збору даних і обробки мови. Було розроблено декілька технологій розпізнавання, які широко використовуються в різних галузях промисловості.

ABBYY FineReader – це програма оптичного розпізнавання символів (OCR), яка перетворює відскановані документи, PDF-файли та зображення у формати, доступні для редагування та пошуку. Він відомий своєю високою точністю розпізнавання тексту та здатністю зберігати компонування та форматування документа.

ABBYY FlexiCapture – це інтелектуальна платформа збору даних і обробки документів. Він автоматизує вилучення даних із різних джерел, таких як паперові документи, форми, електронні листи тощо. FlexiCapture використовує розширені алгоритми машинного навчання для точного вилучення полів даних і маршрутизації інформації до відповідних систем.

ABBYY Mobile Capture SDK дозволяє розробникам інтегрувати OCR і можливості захоплення даних у мобільні програми. Це дає користувачам можливість знімати дані з документів, квитанцій, візитних карток та інших джерел за допомогою камери на своїх мобільних пристроях.

ABBYY Text Analytics SDK надає можливості обробки природної мови (NLP) і аналізу тексту. Завдяки цьому інструменту розробники можуть витягувати ідеї з неструктурованих текстових даних, наприклад для аналізу настроїв, розпізнавання об’єктів і визначення мови.

ABBYY Content Intelligence поєднує в собі OCR, NLP і технології машинного навчання, щоб допомогти розпізнати та обробити величезні обсяги контенту. Ці рішення використовуються для класифікації документів, вилучення даних та аналізу вмісту в таких галузях, як фінанси, охорона здоров’я та право.

ABBYY Recognition Server – це серверне рішення OCR, призначене для конвертації та обробки великих обсягів документів. Він масштабований і може

бути розгорнутий у корпоративному середовищі для автоматизації документообігу та оптимізації бізнес-процесів.

ABBYY Vantage – це платформа на базі штучного інтелекту для інтелектуальної обробки документів. Дана платформа об’єднує OCR, збір даних і можливості автоматизації процесів для оптимізації процесів, орієнтованих на документ, підвищення точності даних і підвищення ефективності роботи.

Загалом, технології розпізнавання ABBYY використовуються компаніями та організаціями по всьому світу для оцифровки документів, автоматизації введення даних і отримання цінної інформації з неструктурованого вмісту. Їхні рішення відіграють вирішальну роль у підвищенні продуктивності, зменшенні ручних зусиль.

Програмісти компанії ABBYY розробили оригінальні технології, які покращують якість розпізнавання [19]. Ідея нового способу зберігання знань про букву: структурно-плямний еталон вперше з'явилася у світ в студентських роботах Д. Яна, К. Анісимовіча і П. Сенаторова. Технологія розпізнавання за допомогою структурно-плямних еталонів отримала назву «Перетворення фонтану» (англ. font – шрифт).

Перетворення фонтану суміщає в собі достоїнства шаблонного і структурного методів і дозволяє уникнути недоліки, властивих кожному з них окремо. У основі цієї технології лежить використання структурно-плямного еталону. Він дозволяє представити зображення у вигляді набору плям, зв'язаних між собою n-парними відношенями, задаючими структуру символу.

Ці відношення (тобто розташування плям один щодо одного) утворюють структурні елементи, складові символ. Так, наприклад, відрізок – це один тип n-парних відношень між плямами, еліпс – другий, дуга – третій. Інші відношення задають просторове розташування створюючих символів елементів.

Наочно це можна уявити собі у вигляді тенісних куль, нанизаних на гумовий джгут (рис 1.10). Кулі можуть зрушуватися щодо один одного.

Таку зв'язку рухомих куль можна "натягнути" на різні зображення одного символу, і система стає менш залежною від шрифтів і дефектів.

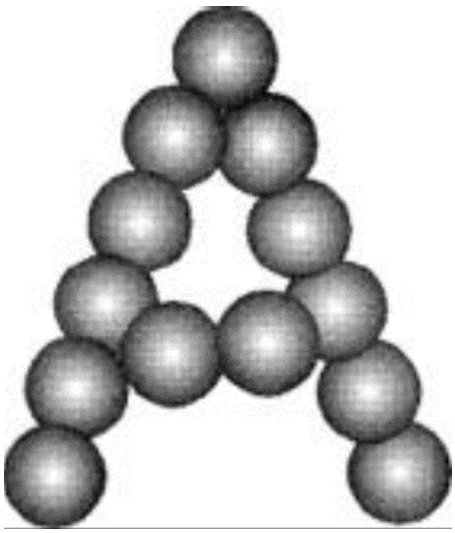


Рисунок 1.10 – Структурно-плямний еталон букви «А»

У еталоні задаються:

- ім'я;
- обов'язкові структурні елементи, що забороняють і необов'язкові;
- стосунки між структурними елементами;
- стосунки, що пов'язують структурні елементи з тим, що описує прямокутником символу;
- атрибути, використовувані для виділення структурних елементів;
- атрибути, використовувані для перевірки стосунків між елементами;
- атрибути, використовувані для оцінки якості елементів і стосунків;
- позиція, з якою починається виділення елементу (стосунки локалізації елементів).

Структурні елементи, що виділяються для класу зображень, можуть бути початковими і складеними. Початкові структурні елементи – це плями, складені - відрізок, дуга, кільце, крапка. Як складові структурні елементи, в принципі, можуть бути узяті будь-які об'єкти, описані в еталоні. Крім того, вони можуть бути описані як через початкових, так і через інші складові структурні елементи.

Як стосунки використовуються зв'язки між структурними елементами, які визначаються або метричними характеристиками цих елементів (наприклад, <довжина більша>), або їх взаємним розташуванням на зображенні (наприклад, <правіше>, <стикається>).

При завданні структурних елементів і стосунків використовуються конкретизуючі параметри, що дозволяють довизначити структурний елемент або відношення при використанні цього елементу в еталоні конкретного класу.

Для структурних елементів, що конкретизують можуть бути, наприклад, параметри, які задають діапазон допустимої орієнтації відрізка, а для відношення – параметри, задаючі граничне допустиме відстань між характерними точками структурних елементів відносно <стикається>.

Побудова і тестування структурно-плямних еталонів для класів розпізнаваних об'єктів, процес складний і трудомісткий. База зображень, яка використовується для відладки описів, повинна містити приклади хороших і поганих (гранично допустимих) зображень для кожної графеми, а зображення бази розділяються на повчальні і контрольні множині.

Розробник опису заздалегідь задає набір структурних елементів (розбиття на плями) і відношення між ними. Система навчання по базі зображень автоматично обчислює параметри елементів і відношень.

Отриманий еталон перевіряється і коректується по контрольній вибірці зображень даної графеми. По контрольній же вибірці перевіряється результат розпізнавання, тобто оцінюється якість підтвердження гіпотез.

Розпізнавання з використанням структурно-плямного еталону відбувається наступним чином. Еталон накладається на зображення, і стосунки між виділеними на зображенні плямами порівнюються із стосунками плям в еталоні.

Якщо виділені на зображенні плями і стосунки між ними задовольняють еталону деякого символу, то даний символ додається в список гіпотез про результат розпізнавання вхідного зображення.

1.4.8 Згорткові нейронні мережі

Згорткова нейронна мережа [22] – це нейронна мережа, яка має особливу архітектуру, орієнтовану на обробку зображень.

У згортковій нейронній мережі [23] відбувається чергування двох типів шарів: згорткових та шарів підвиборки. Згортка нейронна мережа односпрямована, що має безліч шарів.

Навчання у цьому типі нейронної мережі [24] відбувається стандартно, визначення помилки зазвичай використовується метод зворотного поширення помилки. Функція активації може бути будь-якою.

Через операцію згортки архітектура даної нейронної мережі отримала назву «згорткова», суть якої полягає в поелементному множенні частини зображення на матрицю згортки, після чого відбувається підсумовування та запис у ту саму позицію вихідного зображення.

Шар згортки є основним блоком згорткової нейронної мережі. Шар згортки включає фільтр для кожного каналу, ядро згортки якого обробляє попередній шар за фрагментами (підсумовуючи результати матричного добутку для кожного фрагмента). Вага ядер згортки (мала матриця) невідомі та встановлюються у процесі навчання.

Особливістю згортання є відносно невелика кількість параметрів, що задаються в процесі навчання. Наприклад, якщо вихідне зображення має розмірність 100*100 пікселів на трьох каналах (це означає 30 000 вхідних нейронів), а шар згортки використовує фільтри з ядром 3x3 пікселя з виходом 6 каналів, то в процесі навчання визначається лише 9 ваг ядер, але для всіх комбінацій каналів, тобто 9x3x6 = 162, в цьому випадку для цього шару потрібно знайти лише 162 параметри, що значно менше кількості необхідних параметрів повнозв'язкової нейронної мережі.

Шар пулінгу є нелінійним ущільненням карти ознак і групи пікселів (зазвичай розміром 2x2), ущільнених до одного пікселя, що проходить через нелінійне перетворення. Найбільш поширеною є функція максимуму.

Перетворення впливають на прямокутники або квадрати, що не перетинаються, кожен з яких стискається в один піксель, і вибирається піксель з максимальним значенням. Операція об'єднання може значно зменшити об'єм зображення.

Об'єднання інтерпретується так: якщо на попередній операції згортки вже були виявлені деякі ознаки, то для подальшої обробки таке докладне зображення більше не потрібно, і воно ущільнюється до менш детального. Фільтрація непотрібних деталей дозволяє не перевчитися. Шар пулінгу зазвичай вставляється після шару згортки перед наступним шаром згортки.

Крім об'єднання з максимальною функцією можна використовувати інші функції: наприклад, середнє значення або L2-нормалізація. Однак практика показала переваги об'єднання з максимальною функцією, що входить до стандартних систем. Щоб зменшити розмір результуючих уявлень агресивніше, все частіше зустрічається ідея використання менших фільтрів або повної відмови від шарів пулу.

Після декількох проходів згортки зображення та друку за допомогою системи об'єднання проводиться реконструкція від конкретної піксельної сітки з найвищою роздільною здатністю до абстрактніших карток знаків, зазвичай на кожному шарі збільшується кількість каналів і зменшується розмірність зображення в кожному каналі.

У результаті залишається великий набір каналів, що зберігають невеликий обсяг даних (навіть один параметр), які інтерпретуються як абстрактні поняття, виділені з вихідного зображення.

Ці дані об'єднуються і передаються на звичайну повнозв'язкову нейронну мережу, яка також може складатися з кількох шарів. При цьому повнозв'язкові шари вже втрачають просторову структуру пікселів та мають порівняно невелику розмірність. Згорткова нейронна мережа має низку своїх переваг і один головний недолік.

Основними перевагами згорткових нейронних мереж можна назвати:

- зручне розподілення обчислень, а отже, можливість реалізації алгоритмів роботи та навчання мережі на графічних процесорах;
- відносна стійкість до повороту та зсуву зображення, що розпізнається;
- навчання за допомогою методу зворотного розповсюдження помилки;

- згорткова нейронна мережа має найкращу архітектуру для класифікації та розпізнавання образів;
- у порівнянні з повнозв'язковою нейронною мережею, наприклад перцептроном, необхідно налаштувати набагато менше ваг, так як одне ядро ваг використовується цілком для всього зображення, замість того щоб робити для кожного пікселя вхідного зображення свої персональні вагові коефіцієнти.

Це дозволяє при навчанні нейронної мережі узагальнювати інформацію, що демонструється, а не по піксельному запам'ятовувати кожну показану картинку в міріадах вагових коефіцієнтів, як це робить повнозв'язкова нейронна мережа.

До основних недоліків згорткових нейронних мереж відноситься наступне:

- досить багато змінних параметрів нейронної мережі;
- не зовсім зрозуміло, для якого завдання та обчислювальної потужності необхідні якісь параметри.

До змінних параметрів можна віднести кількість шарів, розмір згорткового ядра кожного шару, кількість ядер 35 кожного шару, крок зсуву ядра при обробці шару, потреба субдискретизуючих шарів, ступінь зменшення їх розмірності, функцію зменшення розмірності, передатну функцію нейронів, параметри вихідної повнозв'язкової нейронної мережі на виході згортки.

Ці налаштування мають великий вплив на результат, але вибираються емпірично дослідниками.

1.5 Висновки

У цьому розділі представлено поглиблений аналіз різноманітних доступних на даний момент програмних рішень для розпізнавання номерних знаків. Завдяки детальному аналізу було отримано конкретну інформацію про методи, функціональні можливості та основні технології розпізнавання, які використовуються в кожному продукті.

Маючи дані про основні принципі роботи, обсяг та використовувані методи розпізнавання, цей огляд є цінним ресурсом для розуміння усіх аспектів систем розпізнавання номерних знаків і методів обробки зображень, поширених у галузі.

Ретельна перевірка кожного програмного продукту дає змогу приймати обґрунтовані рішення щодо вибору та впровадження рішень розпізнавання номерних знаків відповідно до їхніх конкретних вимог.

Загалом, цей розділ слугує посібником для навігації серед безлічі варіантів, доступних на ринку розпізнавання промислових номерних знаків, надаючи зацікавленим сторонам знання, необхідні для вибору найбільш прийнятного рішення для своїх програм.

Важливою відмінністю системи «SecurOS Auto» (на відміну від інших представлених на тестування систем) є повна відсутність спеціалізованих налаштувань і регулювань параметрів розпізнавання, що впливають на результати розпізнавання, що дозволяє використовувати її як у світлий, так і в темний час доби, при різних погодних умов.

Для цілей пошуку автомобілів найкращою є система «НОМЕРОК Lite 12»: 95,45% правильно зареєстрованих номерів.

Для попередньої обробки зображень найчастіше використовуються: фільтр Гауса – в основному для видалення шумів, детектор Кенні – для пошуку контурів об'єктів, хвильовий метод – для пошуку об'єкта на зображенні, розпізнавання технологіями ABBYY – для обробки зображень відсканованих документів, саморозпізнавання – для пошуку меж об'єктів. Безліч методів та алгоритмів реалізовано у бібліотеці OpenCV.

В загальному для обробки зображень та розпізнавання символів найчастіше використовують бібліотеку Tesseract, а також середовище для глибинного машинного навчання Caffe.

2 МЕТОД ТА АЛГОРИТМ РОЗПІЗНАВАННЯ НОМЕРНИХ ЗНАКІВ

На основі проведеного теоретичного аналізу та вивчення існуючих методів обробки зображень і розпізнавання цифро-літерних символів було вирішено використати власний метод розпізнавання номерних знаків, та визначити послідовні етапи його виконання.

Попередня обробка включає кроки:

- прочитати вхідне зображення, що містить номерний знак автомобіля;

- перетворити зображення на градації сірого, щоб спростити обробку;

- застосовувати методи покращення зображення, такі як вирівнювання гістограми або фільтрація, щоб покращити видимість пластини.

Локалізація номерного знаку включає кроки:

- використати методи визначення країв (наприклад, Sobel, Canny), щоб виявити краї на зображенні;

- застосувати морфологічні операції (наприклад, розширення, розмивання), щоб покращити та з’єднати краї;

- використовувати алгоритми визначення контурів, щоб визначити регіони-кандидати, які можуть містити номерні знаки, на основі розміру, співвідношення сторін та інших характеристик;

- перевірити регіони-кандидати за допомогою класифікаторів машинного навчання, щоб зменшити помилкові спрацьовування.

Сегментація символів включає кроки:

- для кожної потенційної області номерного знаку витягніть окремі символи за допомогою методів сегментації;

- методи включають аналіз пов’язаних компонентів, сегментацію на основі контурів або підходи ковзного вікна;

- переконатися, що сегментовані символи правильно вирівняні та розділені для точного розпізнавання.

Розпізнавання символів включає кроки:

- навчити модель розпізнавання символів за допомогою алгоритмів машинного навчання;

- витягнути функції з сегментованих символів, наприклад гістограми орієнтованих градієнтів (histogram of oriented gradients, або HOG), значення інтенсивності пікселів або вбудовані технології глибокого навчання;

- використовувати навчену модель, щоб класифікувати кожен символ і розпізнавати буквено-цифрові символи на номерному знаку.

Подальша обробка включає кроки:

- виконати кроки виправлення помилок і перевірки, щоб підвищити точність розпізнавання;

- застосувати правила й обмеження, характерні для форматів номерних знаків (наприклад, розташування символів, тип шрифту), щоб перевірити розпізнані символи;

- використовувати контекстну інформацію, таку як мовні моделі, засоби перевірки орфографії або відомі моделі номерних знаків, щоб уточнити результати розпізнавання.

Як варіант також зберегти розпізнані номерні знаки в базі даних або передати їх іншим системам для подальшої обробки (наприклад, пошуку в базі даних, автоматизованого збору плати за проїзд).

Цей план містить загальну структуру для створення простого алгоритму розпізнавання номерних знаків.

Залежно від вимог і обмежень програми може з’явитися необхідність адаптувати та вдосконалити ці етапи та включити додаткові методи для покращення продуктивності та точності.

2.1 Загальне рішення поставленої задачі

Процес розв'язання задачі розпізнавання автомобільних номерних знаків у загальному вигляді може бути представлений простою блок-схемою з послідовністю кроків, показаних на рисунку 2.1.

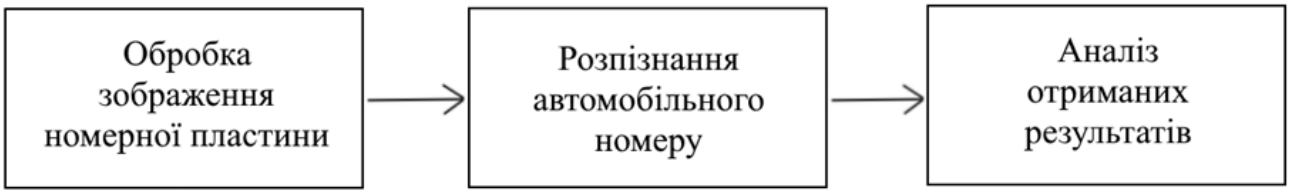


Рисунок 2.1 – Загальна схема розв'язання задачі розпізнавання автомобільного номерного знаку

2.2 Структурна схема та опис алгоритму

Для подальшого виконання поставленої задачі було розроблено структурну схему процесу розпізнавання, яка ґрунтується на методах та кроках, зазначених у підрозділі 2.1.

Структурна схема процесу представлена рисунку 2.2.

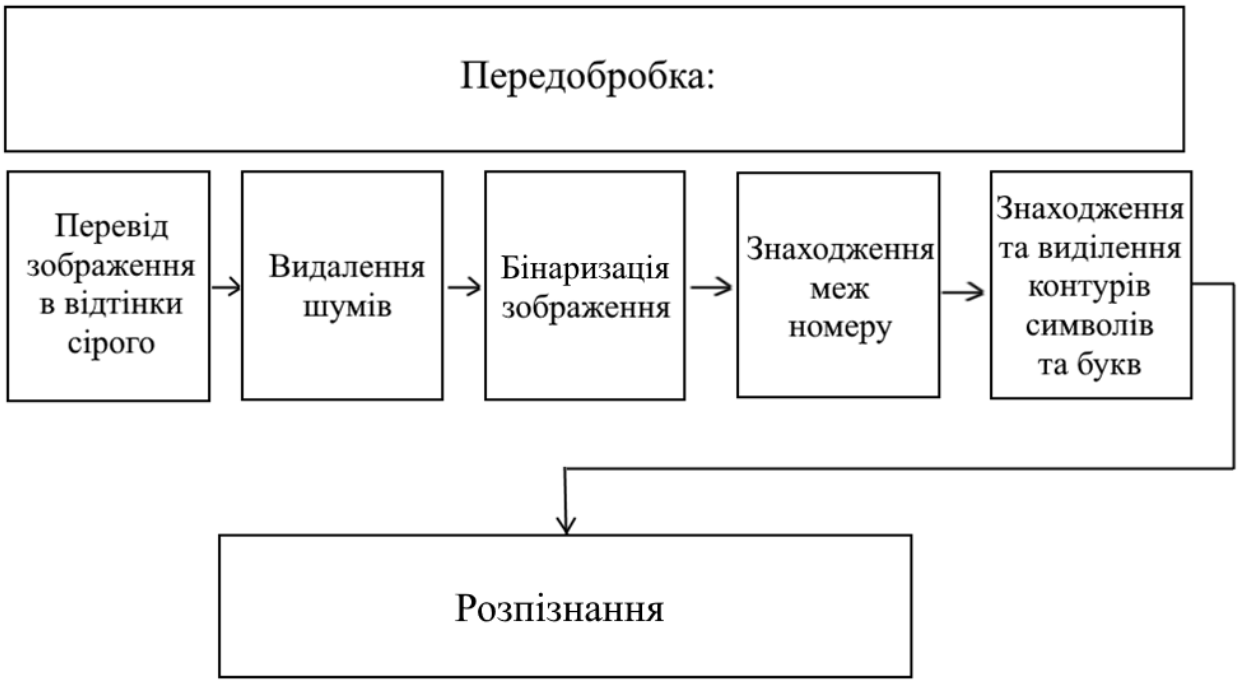


Рисунок 2.2 – Структурна схема процесу розпізнавання

Пропонований алгоритм розпізнавання номерних знаків подається у вигляді послідовності наступних кроків:

Крок 1. Переведення зображення автомобільного номера в градації сірого.

Крок 2. Видалення шумів.

Крок 3. Бінаризація зображення.

Крок 4. Знаходження меж автомобільного номера.

Крок 5. Знаходження та малювання контурів символів та літер.

Крок 6. Сегментація номерної пластини.

Крок 7. Розпізнавання символів та літер.

На рисунку 2.3 показано номерний знак, що буде розглядатися як приклад обробки та розпізнавання зображення.

Рисунок 2.3 – детальне фото автомобільного номерного знаку

2.3 Цифрова обробка номерної пластини

2.3.1 Перетворення на градації сірого

Перетворення зображення в градації сірого [42] є звичайним етапом попередньої обробки в обробці зображень і завданнях комп’ютерного зору. Зображення у відтінках сірого — це простіше представлення кольорових зображень, де кожен піксель представлено одним значенням інтенсивності замість окремих значень для червоного, зеленого та синього каналів.

Перетворення зображення в градації сірого працює наступним чином:

- розрахунок інтенсивності пікселів: на зображенні у градаціях сірого інтенсивність кожного пікселя представляє яскравість відповідної точки на вихідному зображенні. Існують різні методи обчислення цієї інтенсивності, найпоширенішими з яких є методи на основі яскравості.

- обчислення яскравості: одним із методів перетворення кольорових зображень у градації сірого є обчислення яскравості кожного пікселя.

Яскравість представляє сприйману яскравість кольору та зазвичай обчислюється за такою формулою (2.1):

$$Y=0,299\cdot R+0,587\cdot G+0,114\cdot BY=0,299\cdot R+0,587\cdot G+0,114\cdot B, \quad (2.1)$$

де RR, GG і BB — червоний, зелений і синій компоненти кольорового пікселя відповідно. Коефіцієнти 0,299, 0,587 і 0,114 представляють сприйману яскравість кожного колірного каналу.

Формування зображення у градаціях сірого: коли значення яскравості обчислено для кожного пікселя, вони призначаються як інтенсивність відповідного пікселя на зображенні у градаціях сірого. Це призводить до одноканального зображення, де значення кожного пікселя представляє яскравість вихідного зображення.

Більшість бібліотек обробки зображень містять вбудовані функції або методи для перетворення кольорових зображень у градації сірого [43]. Наприклад, у Python такі бібліотеки, як OpenCV і PIL (Python Imaging Library), надають функції для перетворення градацій сірого.

У OpenCV (Python) можна використовувати функцію cv2.cvtColor() із прапорцем cv2.COLOR_BGR2GRAY, щоб перетворити кольорове зображення BGR у градації сірого.

У PIL (Python Imaging Library - бібліотеці зображень Python) можна використовувати метод .convert('L') для об’єкта зображення, щоб перетворити його на градації сірого.

Зображення у відтінках сірого зазвичай використовуються в різних завданнях обробки зображень, включаючи виявлення країв, розпізнавання об’єктів і виділення ознак. Вони спрощують складність обробки, зменшуючи розмірність зображення, зберігаючи важливу інформацію про структуру та контраст зображення [44].

На рисунку 2.4 показано базовий приклад перетворення зображення в градації сірого на Python із використанням OpenCV:

```python
import cv2

# Read a color image
color_image = cv2.imread('input_image.jpg')

# Convert color image to grayscale
grayscale_image = cv2.cvtColor(color_image, cv2.COLOR_BGR2GRAY)

# Save the grayscale image
cv2.imwrite('output_image.jpg', grayscale_image)
```

Рисунок 2.4 – Перетворення зображення на градації сірого на Python

На рисунку 2.5 показано приклад перетвореного на градації сірого зображення номерного знаку автомобіля.

Рисунок 2.5 – Перетворення зображення на градацію сірого

2.3.2 Звільнення від сторонніх шумів

Для зменшення від сторонніх шумів на зображенні використовується фільтрація, яка здійснюється із застосуванням фільтра Гауса (рисунок 2.6). Цей фільтр добре справляється з шумами різних типів, а також даний фільтр дозволяє усунути нерівності контуру, що дуже важливо для подальшого контурного аналізу.

Рисунок 2.6 – Звільнення від сторонніх шумів

2.3.3 Бінаризація зображення

Бінаризація зображення [45] – це фундаментальна техніка обробки, коли зображення у відтінках сірого перетворюється на двійкове зображення, де кожен піксель класифікується як чорний (передній план) або білий (фон). Бінаризація зазвичай використовується для відокремлення цікавих об’єктів або областей від фону, що полегшує аналіз і вилучення інформації із зображень.

Порогове визначення є найпоширенішим методом, який використовується для бінаризації зображення [46]. Він передбачає вибір порогового значення та класифікацію кожного пікселя на зображенні як чорного чи білого залежно від того, чи є його інтенсивність вищою чи нижчою за порогове значення.

Глобальне порогове значення: у глобальному пороговому значенні одне порогове значення застосовується до всього зображення. Для пікселів зі значенням інтенсивності вище порогового значення встановлено білий колір, тоді як для пікселів із значенням інтенсивності нижче порогового значення встановлено чорний колір.

Адаптивне порогове визначення: в адаптивному пороговому визначенні різні порогові значення застосовуються до різних областей зображення, враховуючи зміни в освітленні та контрасті. Це допомагає працювати з зображеннями з нерівномірним освітленням.

Локальні двійкові шаблони [47] це дескриптор, який використовується для бінаризації зображення та аналізу текстури. Він порівнює кожен піксель із сусідніми та призначає двійкові коди залежно від того, чи є інтенсивність сусіднього пікселя більшою чи меншою за інтенсивність центрального пікселя.

Бінаризація Оцу [80]: метод Оцу є популярним методом автоматичного визначення оптимального порогового значення на основі гістограми зображення. Це максимізує поділ між пікселями переднього плану та фону, що призводить до отримання бінарного зображення з мінімальною помилкою.

Більшість бібліотек обробки зображень надають функції бінаризації зображень. Наприклад, у Python такі бібліотеки, як OpenCV і scikit-image, забезпечують функції порогового та адаптивного значення.

Далі зображення проходить процедуру бінаризації (рисунок 2.7).

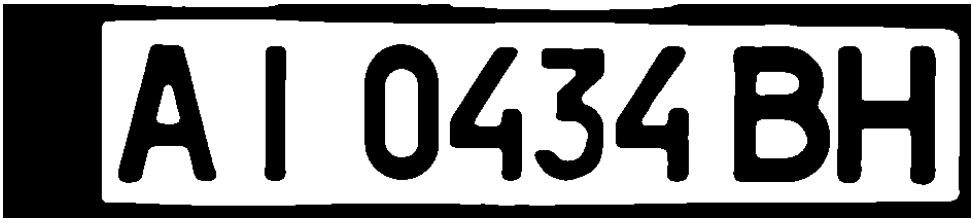


Рисунок 2.7 – Бінаризація зображення номерної пластини

2.3.4 Пошук контурів

Пошук по контуру зображення – це фундаментальна техніка в обробці зображень, яка використовується для ідентифікації та виділення меж об’єктів або областей на зображенні. Контури – це просто контури об’єктів на зображенні, представлені у вигляді серії точок або координат.

Процес пошуку контурів зображення можна описати в декілька етапів.

Виявлення країв: процес пошуку контурів починається з виявлення країв, що виділяє області зі значними коливаннями інтенсивності зображення. До популярних алгоритмів визначення краю належать детектор Кенні, оператор Sobel і оператор Prewitt.

Порогове значення: після виявлення країв зображення може бути обмежено, щоб перетворити його на бінарне зображення, де пікселі класифікуються як передній план (об’єкт) або фон.

Цей крок спрощує процес пошуку контурів, зменшуючи складність зображення.

Виявлення контурів: після попередньої обробки зображення застосовуються алгоритми виявлення контурів для визначення контурів об’єктів на зображенні. Загальні алгоритми виявлення контурів включають алгоритм Дугласа-Пюкера, ланцюговий код Фрімена та алгоритм трасування Мура-Сусіда.

Апроксимація контуру: алгоритми виявлення контуру створюють велику кількість точок, які наближено до межі об’єкта. Методи контурної апроксимації можуть бути застосовані для зменшення кількості точок при збереженні загальної форми контуру.

Характеристики контуру: після того, як контури виявлені та наближені, з контурів можна виділити різні характеристики, щоб охарактеризувати об’єкти на зображенні. Ці функції можуть включати площу, периметр, центроїд, обмежувальну рамку, співвідношення сторін і ексцентриситет.

Фільтрація контурів: контури можна фільтрувати на основі певних критеріїв, щоб видалити небажані або нерелевантні контури. Наприклад, контури можна відфільтрувати за розміром, формою чи орієнтацією, щоб зберегти лише цікаві контури.

Візуалізація контурів: виявлені контури можна візуалізувати на вихідному зображенні, щоб перевірити точність процесу виявлення контурів. Ця візуалізація може допомогти зрозуміти просторові відносини між об’єктами на зображенні.

На рисунку 2.8 відображено результат пошуку всіх контурів, присутніх на зображенні номерної пластини.

Рисунок 2.8 – Результат пошуку контурів

Знайдені замкнуті контури виділяються червоними прямокутними областями наступної сегментації.

На рисунку 2.9 показаний результат виділення замкнутих контурів прямокутними областями.

Рисунок 2.9 – Результат виділення замкнутих контурів

2.3.5 Виключення неінформативних областей

Не всі виділені області цікавлять завдання розпізнавання номерної пластини. Використовуючи відношення довжини до ширини цифро-літерних символів, виключаються області, що не є цифро-буквеними символами.

На рисунку 2.10 показаний результат виділення цільових областей, що підлягають розпізнаванню.

Рисунок 2.10 – Результат виділення цільових областей для розпізнавання

2.3.6 Сегментація номерної пластини

Отримані координати прямокутних областей позначені зеленим кольором дозволяють сегментувати номерну пластину, вирізавши кожен сегмент наступного розпізнавання.

На рисунку 2.11 показаний результат вилучення сегментів із зображення номерної пластини.

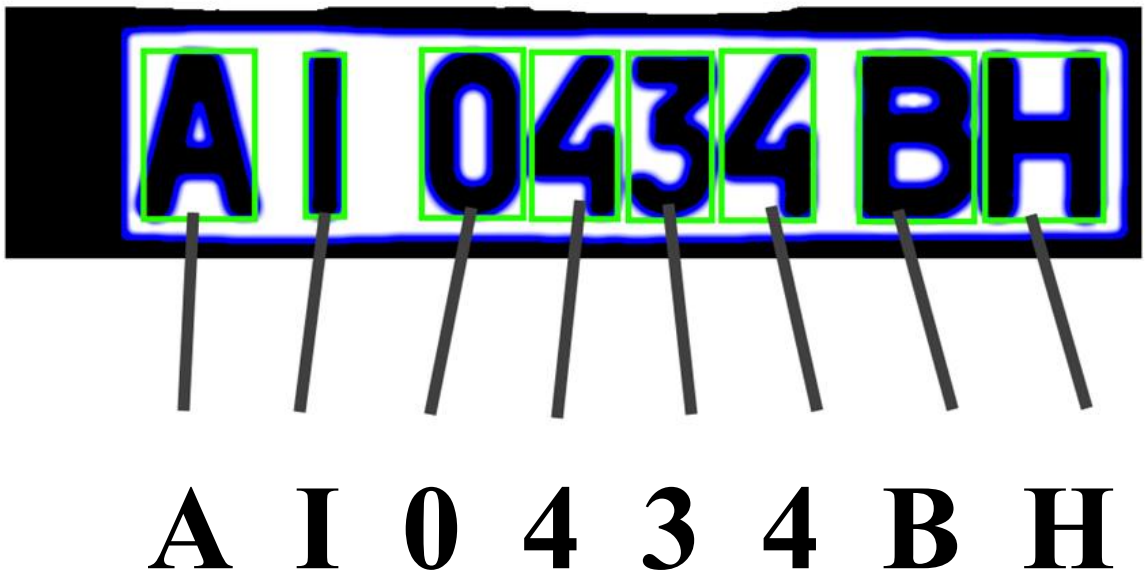


Рисунок 2.11 – Результат вилучення сегментів

Кожен видобутий сегмент наводиться до розміру 20*20 пікселів і зберігається в папку для подальшого розпізнавання. Сегменти нумеруються зліва направо від 1 до 8.

2.4 Бібліотеки для обробки зображень

Для розробки методу розпізнавання номерних знаків було розглянуто декілька бібліотек та оптичних систем розпізнавання символів.

2.4.1 OpenCV

OpenCV [26] це бібліотека програмних функцій, спрямованих в основному на комп'ютерний зір у реальному часі. Розроблена компанією Intel, пізніше підтримувалась Willow Garage, а потім Itseez. Бібліотека є крос-платформною та безкоштовною для використання під ліцензією BSD з відкритим вихідним кодом.

OpenCV [27] надає широкий спектр функціональних можливостей для задач комп’ютерного зору та обробки та форматування зображень різних типів, зокрема:

- обробка зображень (фільтри, трансформації, морфологічні операції);
- зображення та відео I/O (вхід/вихід);
- виявлення та розпізнавання об'єктів;
- виявлення ознак і відповідність;
- калібрування камери та 3D реконструкція.

OpenCV [28] розроблено для роботи на різних платформах, включаючи Windows, Linux, macOS, iOS та Android. Завдяки цьому дана бібліотека є універсальною для широкого спектру застосувань. OpenCV має велике й активне співтовариство розробників і дослідників, які роблять свій внесок у його розвиток. Така підтримка спільноти забезпечує регулярне оновлення, виправлення помилок та вдосконалення бібліотеки.

Бібліотека OpenCV оптимізована для підвищення продуктивності з реалізацією багатьох алгоритмів, які використовують інструкції SIMD для паралельної обробки та підтримки прискорення за допомогою GPU.

OpenCV [29] можна інтегрувати з іншими бібліотеками та фреймворками, такими як NumPy (для Python), CUDA (для прискорення графічного процесора) і різними фреймворками машинного навчання для завдань глибокого навчання.

OpenCV надає прив’язки Python, які значно спрощують її використання у сценаріях, коли використовується Python. Також забезпечує майже всі функції, доступні у C++. Дана бібліотека підтримує фреймворки глибокого навчання – TensorFlow, Torch/PyTorch та Caffe, також її можна вільно використовувати та розповсюджувати як у проектах з відкритим кодом, так і в комерційних проектах.

Загалом, OpenCV – це напрочуд універсальна бібліотека для виконання задач комп’ютерного зору та обробки зображень, яка широко використовується в академічних, дослідницьких і промислових галузях для різноманітних програм – від робототехніки та доповненої реальності до медичних зображень і спостереження.

2.4.2 Tesseract

Tesseract [30] це оптична система розпізнавання символів для різних операційних систем. Це безкоштовне програмне забезпечення, випущене під ліцензією Apache, розробку спонсорували Google з 2006 року.

На той час Tesseract вважалася однією з найточніших бібліотек OCR з відкритим вихідним кодом. Бібліотека Tesseract була спочатку розроблена як програмне забезпечення в Hewlett Packard Labs у Брістолі, Англії, Грилі (Колорадо) між 1985 і 1994 роками, з деякими змінами, внесеними в 1996 році в порт для Windows.

Tesseract є вільним для використання та модифікацій, оскільки його вихідний код є відкритим та доступний на багатьох платформах, наприклад, GitHub. Система доступна для різних платформ, включаючи Windows, macOS і Linux. Віна також має прив’язки для різних мов програмування, включаючи Python, Java та C++. Більшість первісного коду було написано на C, та доробка велася на C++.

Tesseract [31] підтримує понад 100 мов, у тому числі мови з нелатинським шрифтом, такі як арабська, китайська, японська та корейська. Також доступна підтримка розпізнавання різних скриптів, що робить дану систему універсальною для багатомовних завдань OCR з високою точністю розпізнавання тексту, особливо друкованого тексту на зображеннях. Це добре працює в сценаріях, де текст чітко визначений і якість зображення хороша. Такі методи попередньої обробки, як бінаризація зображення, зменшення шуму та виправлення перекосів, можуть значно покращити його продуктивність.

Однією з переваг Tesseract є те, що він дозволяє навчатися власним шрифтам і мовам. Це означає, користувач може точно налаштувати Tesseract, щоб розпізнавати текст, специфічний для домену чи мови конкретної програми. Tesseract можна інтегрувати в різні програми та робочі процеси, оскільки є можливість надати інструменти командного рядка, а також API для інтеграції в програмні програми.

2.4.3 Caffe

Caffe [32] це популярне середовище для глибинного машинного навчання, розроблене Яньцинем Цзи у процесі підготовки та захисту своєї дисертації в Каліфорнійському університеті Берклі. Caffe є доступним для широкого кола користувачів, які розповсюджуються під ліцензією BSD, відкритим ПЗ. Розроблено мовою C++ із підтримкою інтерфейсу мовою програмування Python.

Caffe [33] підтримує різні типи машинного навчання, які націлені в основному на вирішення задач сегментації та класифікації зображень. Також забезпечує роботу з згортковими нейронними мережами, довгу короткострокову пам'ять та пов'язані нейронні мережі. GPU в даному випадку може бути застосований для того, щоб прискорити навчання.

Caffe [34] оперує багатовимірними масивами даних, які інакше називаються блоби. Навчання у згортковій нейронній мережі реалізується як паралельні багатопроцесорні обчислення блобів від шару до шару (прямим та зворотним ходом). Solver (англ. solve – вирішувати) координує весь процес навчання - прямий хід від вихідних до вихідних даних, отримання функції помилок, зворотний хід назад від вихідного шару з використанням градієнтів помилок. При цьому Caffe реалізує різні стратегії навчання для Solver'a.

Як вхідні дані використовуються дані з пам'яті, бази даних або зовнішніх носіїв. Приховані шари використовують традиційні згорткові шари, шари пулла, повнозв'язні шари і шари розгортання. Також передбачено багато інших типів шарів, фільтрів, перетворень даних та функцій помилок.

2.5 Висновки

Розроблений метод розпізнавання номерних знаків представляє комплексну структуру, яка охоплює різні етапи, від попередньої обробки зображень до розпізнавання символів, демонструючи структурований підхід до ефективної ідентифікації цифро-літерних символів на номерних знаках.

Завдяки теоретичному аналізу та вивченню існуючих методів обробки зображень кожен етап був ретельно розроблений, щоб забезпечити надійність і точність.

Початкові етапи попередньої обробки включають перетворення вхідного зображення на градації сірого та застосування методів покращення для оптимізації видимості пластини. Локалізація номерних знаків використовує визначення країв, або інші операції та алгоритми визначення контурів для ідентифікації меж номерних пластин. Сегментація символів використовує різноманітні методи, такі як аналіз зв’язаних компонентів і сегментацію на основі контурів, щоб точно виділити окремі символи. Забезпечення правильного вирівнювання та розділення сегментованих символів закладає основу для точного розпізнавання.

Подальша обробка включає виправлення помилок, етапи перевірки та дотримання специфікацій формату номерних знаків для уточнення результатів розпізнавання. Використання контекстної інформації підвищує точність, дозволяючи розпізнавати номерні знаки з більшою надійністю. Крім того, система що в подальшому буде розроблятися матиме високу універсальність, полегшуючи зберігання бази даних або інтеграцію з іншими системами для безперебійної передачі даних та їх подальшого аналізу.

Незважаючи на те, що цей проект забезпечує надійну основу для розпізнавання номерних знаків, може знадобитися адаптація та вдосконалення для відповідності конкретним вимогам і обмеженням програми. Включення додаткових методів і оптимізація існуючих процесів може ще більше підвищити продуктивність і точність, забезпечуючи ефективність алгоритму розпізнавання в різних сценаріях.

3 РОЗРОБКА АЛГОРИТМУ РОЗПІЗНАВАННЯ НОМЕРНИХ ЗНАКІВ

Для подальшого виконання поставленої задачі було розроблено алгоритм, який ґрунтується на методах та кроках, зазначених у розділі 2, а саме: переведення зображення автомобільного номера в градації сірого, видалення шумів, бінаризація зображення, знаходження меж автомобільного номера, знаходження та малювання контурів символів та літер, сегментація номерної пластини, розпізнавання символів та літер.

3.1 Пошук області автомобільного номера

Для початку потрібно поставити завдання. Наприклад, на зображенні з автомобілем необхідно виділити номерну рамку (рисунок 3.1).

Рисунок 3.1 – Виділена номерна рамка

Класифікатор формується на примітивах Хаара шляхом розрахунку значень ознак [80]. Для навчання на вхід класифікатора спочатку подається набір «правильних» зображень із попередньо виділеною областю автомобільного номера, далі відбувається перебір примітивів та розрахунок значення ознаки. Обчислені значення зберігаються у файлі у форматі xml.

Утиліту для навчання каскаду Хаара вбудовано в пакет OpenCV.

Для навчання збирається колекція файлів, які надалі знадобляться:

- Реальні фотографії об'єкта. Чим більше схоже вибірка буде на те, що розпізнаватиметься, тим краще будуть результати. Наприклад, якщо навчати розпізнавач особи за фотографіями людей зі студії, то на вулиці рівень розпізнавання буде нижчим, ніж у студії. На це впливають як тіні, одяг, так і обличчя.

- Вибір негативних фотографій, на яких немає об'єкта розпізнавання. Фотографії повинні бути зроблені в тому ж середовищі, де буде розпізнавання. Якщо вибірка контрприкладів буде зроблено за фотографіями на північному полюсі, а розпізнавання відбувається за зображеннями зробленими в тропічних джунглях, то нічого не запрацює.

Для детектора осіб, що стабільно працює, це 3000-4000 позитивних прикладів і стільки ж негативних. З 500 позитивних та 500 негативних можливо зробити стабільний детектор номерів. Для даного детектора підібрано 500 позитивних та 500 негативних фотографій. Чим більша і різноманітніша вибірка, тим стабільніше працює і тим довше навчається.

Для початку необхідно сформувати папку з позитивними та негативними фотографіями. Для цієї мети використано програмне забезпечення Picture Cropper. Робота з цим програмним забезпеченням: мишкою виділяється область яка має бути збережена. При натисканні на клавішу "s" відбувається збереження, на клавішу "г" - збереження та перехід до наступного зображення, на "пробіл" - просто перехід до наступного зображення.

Для навчання необхідно мати дві папки з прикладами. "Good" - папка з відповідними зображеннями, "Bad" - з невідповідними. Варто врахувати, що OpenCV відмовляється працювати з точками, пробілами та спеціальними символами, тому не радиться використовувати ці символи в назвах прикладів.

Для кожної папки з прикладами необхідно мати файл опису, в якому описані зображення, що використовуються. Стандартно їх називають "Good.dat" та "Bad.dat".

Файли повинні лежати на одному рівні з папками.

Приклад ієрархії (рисунок 3.2):

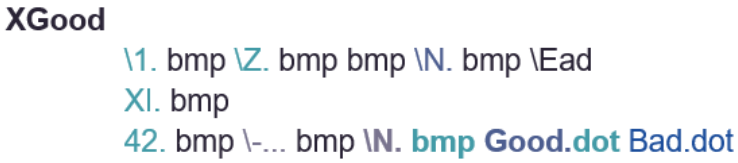


Рисунок 3.2 – Приклад ієрархії

Файли опису для відповідних та невідповідних об'єктів мають різну структуру (рисунок 3.3). Для файлу невідповідних прикладів це просто список відносних шляхів до зображень:

Рисунок 3.3 – Структура невідповідних об'єктів

Для файлів з відповідними прикладами запис трохи складніший. Крім шляху має бути зазначено положення об'єкта, і його розмір (рисунок 3.4).

Рисунок 3.4 – Структура відповідних об'єктів

"Good \0.bmp" - адреса об'єкта щодо файлу опису. "1" - кількість позитивних об'єктів на зображенні. «0 0 414 148» - координати прямокутника на зображенні, в якому знаходиться об'єкт. Якщо об'єктів кілька, то запис набуває вигляду: «Good \0.bmp 2 100 200 50 50 300 300 25 25».

Найзручніше, коли кожен об'єкт є окремим кадром, при цьому координати об'єкта дорівнюють розміру кадру.

Для обробки зображень необхідно запустити Picture Cropper та вибрати папку з відповідними зображеннями. Після завершення роботи з програмним забезпеченням Picture Cropper має з'явитися Good.dat і в папці Good папка Cropper, де зберігаються обрізані зображення.

На цьому робота з файлами опису закінчена, далі відбувається навчання.

Для початку навчання необхідно відкрити командний рядок і перейти до директорії з файлом опису Good.dat:

Для створення набору наведених відповідних зображень запустимо opencv_createsamples через консоль (рисунок 3.5):

```
■opencv_createsamples.exe -info Good.dat -vec samples.vec -w 180 -h 40|
```

Рисунок 3.5 – Створення набору відповідних зображень

Good.dat – файл опису відповідних зображень. Вказується або повна адреса або щодо програми opencv_createsamples.exe. "-vec samples.vec" - файл, в який буде збережено приведену до загального формату базу відповідних зображень. Адреса має бути вказана щодо програми opencv_createsamples.exe (припустимо повний шлях у системі).

«-w 180 -h 40» - розмір шаблону. Повинен приблизно відображати пропорції об'єкта, що шукається. Наприклад, для осіб або для символу автомобільного номера найбільш підходяща пропорція висоти шириною 1 до 3. Для номерів це приблизно 4 до 1. Розмір шаблону повинен бути досить маленьким.

Ідеально ставити його таким, щоб людина сама могла відрізнити зображений об'єкт, але не більше того. Чим більший шаблон, тим довше навчання. Результатом роботи програми є файл samples.vec, в якому будуть лежати всі відповідні зображення у форматі, близькому до bmp і розміром w*h.

Якщо все пройшло успішно, має з'явитися файл samples.vec.

Для підрахунку підсумкового каскаду використовується програма opencv_traincascade.exe, що лежить в тій же папці, що і opencv_createsamples.exe. Навчання каскаду на 500-1000 об'єктів займає майже цілий день.

Приклад навчався близько 24 години (рисунок 3.6):

```
w:\learnMjpencv_traincascade.exe -data haarcascade -vec samples.vec -bg Bad.dat -numstages 16 -minhitrate 0.990
-maxFalseAlarmRate 0.5 -numPos 400 -numNeg 500 -w 180 -h 40 -mode ALL -pre calcValBufSize 256 -precalcIdxBufSize 256]
```

Рисунок 3.6 – Приклад навчання каскаду

«-data haarcascade» - адреса папки, куди класти отримані результати. Відраховується від кореневої папки програми. Необхідно створити заздалегідь.

«-vec samples.vec» - адреса порахованого в минулому пункті файлу з відповідними прикладами.

«-bg Bad.dat» - адреса файлу-опису невідповідних прикладів.

«-numStages 16» - кількість рівнів каскаду, які програма буде навчати. Чим більше рівнів, тим точніше, але довше. Нормальна їхня кількість від 16 до 25.

«-minhitrate 0.999» - коефіцієнт, що визначає якість навчання. По суті це відсоток «правильних» виявлень. Якщо встановлено .999, тобто за вихідною вибіркою буде трохи більше, ніж 1- 0.999 = 0.1% пропусків цілей. Чим вищий коефіцієнт, тим вищий рівень хибних тривог.

Якщо вибірка хороша, можна поставити 0.99-0.999. Якщо погана (об'єктів мало, вони поєднуються з фоном), то слід опускати.

«-maxFalseAlarmRate 0.4» - рівень помилкової тривоги.

«-numPos 400» - кількість відповідних прикладів. Чим нижчий коефіцієнт «minhitrate», тим більше файлів вважатиметься непридатними. Зазвичай досить поставити numPos 80% від наявних відповідних файлів.

«-numNeg 500» - кількість наявних невідповідних прикладів.

«-w 180 -h 40» - розмір примітиву з попереднього пункту.

«-mode ALL» - використовувати повний комплект Хаар-ознак. Від цього залежить швидкість роботи та точність алгоритму.

«-precalcValBufSize 256 -precalcIdxBufSize 256» - пам'ять, що виділяється.

Після того, як алгоритм закінчить роботу, з'явиться папка haarcascade і файл cascade.xml у цій папці, це буде створений каскад.

3.2 Алгоритм нормалізації кута нахилу та масштабу

Після того, як знайдено автомобільний номер, необхідно підготувати його до розпізнавання, для цього проводиться нормалізація кута нахилу та масштабу автомобільного номера (рисунок 3.7).

Рисунок 3.7 - Максимальні кути

Припустимо, що номер може мати діапазон повороту від -10 ° до +10 °. При цьому відбуватиметься обробка кожного разу нового кадру з кроком 0.1°. Обробка кожного кадру відбувається незалежно від попередніх. Яка гіпотеза щодо повороту дасть найкращий результат, та й переможе.

Максимальні кути нахилу наведені рисунку 3.7. Для кожного кадру розраховується нижня межа зображення. Після роботи алгоритму виграє кут, для якого була розрахована найвища межа. Це і буде шуканий кут.

Масштаб зображення (за аналогією з поворотом) має діапазон від 1 до 3 із кроком 0.1.

3.3 Алгоритм пошуку верхньої та нижньої меж автомобільного номера

Після того, як знайдено кут нахилу автомобільного номера, необхідно знайти його межі. Пошук нижньої межі автомобільного номера будується на аналізі гістограми яскравості.

Під час тестів дослідним шляхом встановлено, що гіпотеза з використанням гістограми яскравості для пошуку верхньої межі автомобільних номерів працює в 50% випадків, що неприйнятно для системи, що розробляється. У зв'язку з цим було вирішено навчити каскад Хаара на кожну літеру та проаналізувати верхні межі літер, тим самим можна буде отримати верхню межу номерного знаку.

У деяких випадках алгоритм з використанням каскаду Хаара може не дати результату, це характерно для зображень з дуже низькою роздільною здатністю. Для таких зображень застосовується як альтернативний алгоритм, пошук кордону з використанням гістограми яскравості.

3.4 Алгоритм пошуку бічних кордонів автомобільного номера

На даному етапі вже обрізаний автомобільний номер по верхньому та нижньому кордоні, залишається визначити бічні межі автомобільного номера. В даному випадку застосовується метод побудови гістограми яскравості, але з'являється проблема: колір автомобіля. Якщо автомобіль білого кольору, то після бінаризації зображення по краях матиме білий колір, а якщо автомобіль чорного кольору (темних тонів), то краї будуть чорні, а сам номер білим.

Звідси випливає, що слід використовувати дві гіпотези на кожну сторону, одна для пошуку кордону на білій (світлій) машині та друга для пошуку на чорній (темній) машині. Виграє та гіпотеза результатом дослідження, якою є найближчий до центру автомобільного номера.

3.5 Використання згорткової нейронної мережі для розпізнавання символів

Згорткова нейронна мережа (Convolutional Neural Network або коротко CNN) – це тип глибокої нейронної мережі, спеціально розроблений для обробки структурованих сіткових даних, наприклад зображень. CNN широко використовуються в задачах комп’ютерного зору, включаючи класифікацію зображень, виявлення об’єктів, сегментацію тощо.

У принципі роботи згорткової нейронної мережі лежать згорткові шари, що є її основними будівельними блоками. Згортковий шар складається з набору навчальних фільтрів (також їх називають ядрами), які пересуваються по вхідному зображенню, виконуючи поелементне множення та підсумовуючи результати для створення карт функцій.

Кожен фільтр фіксує різні характеристики вхідного зображення, наприклад краї, текстури або форми. Коли фільтри ковзають по вхідному зображенню, вони виділяють дедалі складніші характеристики в різних просторових масштабах.

Шари об’єднання використовуються для зменшення дискретизації карт функцій, створених згортковими шарами, зменшуючи просторові розміри, зберігаючи важливу інформацію. Загальні операції об’єднання включають максимальне об’єднання, де зберігається максимальне значення в кожній області об’єднання, де обчислюється середнє значення.

Об’єднання допомагає зробити представлення інваріантним до невеликих перекладів і спотворень у вхідному зображенні, а також зменшує складність обчислень.

Функції активації вводять нелінійність у мережу, дозволяючи їй вивчати складні відображення між входом і виходом. Найпоширенішою функцією активації є Rectified Linear Unit (ReLU), яка замінює від’ємні значення нулем. Інші функції активації, такі як sigmoid і tanh, також можуть використовуватися в певних мережевих архітектурах.

Після кількох згорткових шарів та шарів об’єднання карти функцій зводяться в одновимірний вектор і пропускаються через один або більше пов’язаних шарів.

Повністю підключені рівні виконують міркування високого рівня та приймають рішення на основі витягнутих ознак, відображаючи вхідні дані на вихідні класи або мітки.

Як правило, функція активації softmax використовується у вихідному рівні для завдань багатокласової класифікації, тоді як функція активації sigmoid використовується для двійкової класифікації.

У більшості випадків згорткові нейронні мережі навчаються за допомогою алгоритмів зворотного поширення та градієнтного спуску, щоб мінімізувати функцію втрат, таку як втрата перехресної ентропії, між прогнозованими та фактичними результатами.

Навчальні дані, що складаються з вхідних зображень і відповідних міток, використовуються для оновлення параметрів мережі (ваги та зміщення) ітеративно через кілька епох.

Під час навчання мережа вчиться автоматично витягувати з вхідних зображень функції, які відповідають поставленому завданню, наприклад, розпізнавати об’єкти чи візерунки.

Техніка, за якої попередньо підготовлені моделі навчені на великих наборах даних, таких як ImageNet, налаштовані для конкретних завдань за допомогою менших, специфічних завдань наборів даних це трансферне навчання.

Завдяки використанню функцій, отриманих із попередньо навченої моделі, трансферне навчання забезпечує швидшу конвергенцію та покращує продуктивність, особливо коли навчальні дані обмежені.

Згорткової нейронні мережі здійснили революцію в галузі комп’ютерного зору та досягли найсучасніших результатів у різних контрольних наборах даних і додатках. Їхня здатність автоматично вивчати ієрархічні представлення візуальних даних зробила їх незамінними для широкого спектру завдань обробки зображень.

Якщо в алгоритмах машинного навчання необхідно було формувати хороший вектор ознак, щоб отримати прийнятну якість, то в нейронах мережах цього робити не потрібно. Головне – спроектувати гарну архітектуру мережі.

Для створеної архітектури мережі було введено такі позначення:

- input це вхідний шар, зазвичай це пікселі зображення;
- conv це шар згортки;
- pool це шар під вибори;
- fully-conn це повнозв'язковий шар;
- output це вихідний шар, що видає передбачуваний клас зображення.

Для завдання класифікації зображень основною є наступна архітектура нейронної мережі (рисунок 3.8).

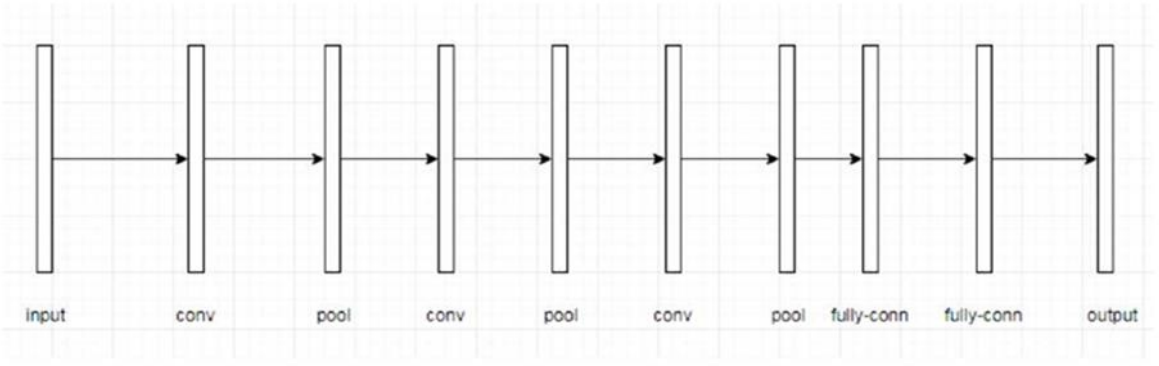


Рисунок 3.8 – Архітектура нейронної мережі [49]

Кількість (conv => pool) шарів зазвичай може бути різною, але для навчання нейронної мережі необхідно не менше двох таких шарів. Водночас, кількість шарів fully-conn має бути не менше одного, і хоча можна додати більше, це не завжди приведе до кращого результату.

3.6 Висновки

Підсумовуючи, алгоритм, розроблений для розпізнавання номерних знаків, містить ретельне поєднання теоретичних основ і практичних стратегій. Завдяки систематичному підходу, що включає попередню обробку, локалізацію, сегментацію, розпізнавання та подальшу обробку, алгоритм демонструє надійну структуру для точної ідентифікації та вилучення цифро-літерних символів з номерних знаків.

В даному алгоритмі підкреслюється важливість кожного етапу, від етапів попередньої обробки, які оптимізують якість зображення, до сегментації та розпізнавання символів. Використовуючи набір методів обробки зображень, алгоритмів машинного навчання та методів точної перевірки, алгоритм демонструє цілісний підхід до вирішення складних завдань, притаманних задачам розпізнавання номерних знаків.

Здатність даного алгоритму розпізнавання розвиватися та адаптуватися до різноманітних сценаріїв у поєднанні з інтеграцією в більш широкі системи підкреслює його корисність у багатьох сферах, включаючи спостереження, керування трафіком та безпеку.

У сфері розпізнавання символів у системах розпізнавання номерних знаків згорткові нейронні мережі стали часто використовуваним інструментом. Їхня здатність автоматично вивчати ієрархічні функції з необроблених даних робить їх особливо придатними для таких завдань, як розпізнавання символів. Використовуючи згорткові нейронні мережі, алгоритм має змогу адаптуватися до безлічі нюансів і варіацій, присутніх у символах номерних знаків, підвищуючи точність розпізнавання навіть у складних умовах, таких як різне освітлення, ракурси та фон.

Крім того, гнучкість архітектури згорткових нейронних мереж дозволяє інтегрувати додаткові рівні або тонко налаштовувати існуючі, забезпечуючи безперервне вдосконалення продуктивності розпізнавання з часом. Таким чином, алгоритм матиме ще більшу ефективність і актуальність у сфері розпізнавання номерних знаків.

Оскільки технологія машинного навчання продовжує розвиватися, даний метод та алгоритм можуть служити основоположним для розробників та дослідників, які орієнтуються в складнощах розробки програмного забезпечення, що виконує задачу розпізнавання як номерних знаків, так і розпізнавання будь-яких цифро-літерних та інших символів, а також обробки зображень у загальному плані.

4 ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ СИСТЕМИ РОЗПІЗНАВАННЯ НОМЕРНИХ ЗНАКІВ

4.1 Розробка структурної схеми системи розпізнавання номерних знаків

Для опису загального принципу роботи системи ідентифікації номерних знаків транспортних засобів, було розроблено структурну схему (рисунок 4.1).

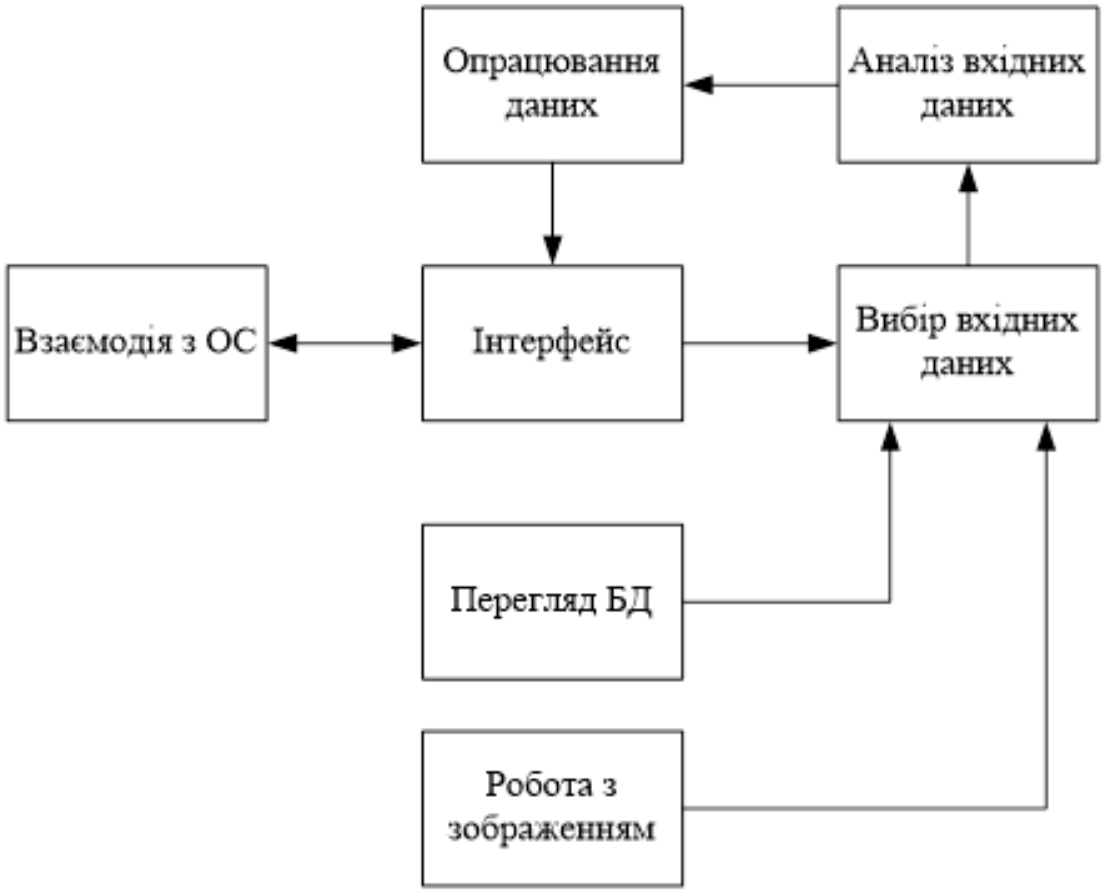


Рисунок 4.1 – Структурна схема ПЗ ідентифікації номерних знаків

Принцип функціонування структурної схеми полягає в тому, що користувач після запуску програми обирає задачу, яку йому необхідно виконати: чи це буде робота з базою даних , чи це буде ідентифікація чергового транспортного засобу. Після цього проводиться вибір вхідних даних (фільтри для пошуку по БД, чи зображення відповідно). Після цього отримані результати виводяться на інтерфейс, де користувач зможе їх бачити, аналізувати та проводити подальші дії.

4.2 Розробка алгоритму функціонування програми ідентифікації транспортних засобів

Після запуску програми, користувачеві надається вибір подальших його дій. Він має можливість переглядати наявну базу даних, або обрати нове зображення для ідентифікації наявного на ньому транспортного засобу.

Якщо користувач обирає перегляд бази даних, йому пропонується обрати один з варіантів: пошук за реєстраційним номером та пошук за датою розпізнавання. При пошуку за номером одразу формується SQL-запит з метою отримання наявних у базі номерних знаків. Після вибору потрібного користувачеві номеру формується список усіх зображень, де фігурує вказаний номер. Обравши зображення зі списку, воно виводиться в окремому вікні. При пошуку за датою користувачеві пропонується обрати довільну дату, після чого формується SQL-запит і виводяться дані ідентифікації проведеної в цей день, а саме: точний час ідентифікації та реєстраційний номер.

За умови, що користувач вибере варіант ідентифікації чергового транспортного засобу – зображення завантажується відображається у вікні. У фоновому режимі проводиться процедура бінаризації, тобто переведення зображення у чорно-біле. Це проводиться в зв’язку з тим, що більшість алгоритмів комп’ютерного зору адаптовані для роботи з чорно-білими зображеннями. Після цього проводиться пошук прямокутних областей.

Процес пошуку прямокутних областей включає в себе адаптування (полягає у масштабуванні) зображення за нижнім порогом, знаходження областей у кожній кольоровій площині зображення, застосування оператора Кенні на нульовому пороговому рівні, що дозволить захопити області з градієнтом затінення.

Згодом проводиться перевірка обраної області на предмет співпадіння пропорції сторін області з пропорцією сторін номерного знаку. Після того, як буде знайдено підходящу за критеріями область, вона передається для аналізу в бібліотеку Tesseract OCR. На цьому етапі виникає проблема з наявністю шумів на картинці, так як дана бібліотека чутлива до шумів.

Для того, щоб позбавитись від шумів необхідно бінаризувати зображення і застосувати операції ерозії та дилатації, які відносяться до морфологічних операції. Ерозія робить об’єкти більш тонкими, а дилатація – навпаки робить їх більш товстими. Після цього шум забирається і залишаються лише вагомі деталі. Згодом проводиться цикл по контурах і вибір букв з них, з подальшим їх розпізнаванням і формуванням рядка символів.

Результат роботи виводиться у відведене для нього місце у головному вікні.

4.3 Опис використовуваних технологій

Було вирішено розробити два варіанти програмного забезпечення для реалізації алгоритму розпізнавання номерних знаків, удосконалену версію створено для порівняння та оцінки отриманих результатів.

Перше програмне забезпечення, що реалізує алгоритм, розроблено мовою програмування C++ з використанням бібліотек OpenCV і Tesseract. Для обробки зображення використовується бібліотека OpenCV, а для розпізнавання символів – Tesseract. Програмне забезпечення включатиме модулі для отримання зображень або кадрів відео з джерел як камери, файли або потоки.

Попередня обробка за допомогою OpenCV: дана бібліотека використовуватиметься для завдань попередньої обробки, таких як покращення зображення, зменшення шуму та сегментація. Ці операції допомагають покращити якість зображення та виділити область номерного знаку.

Локалізація номерного знаку: даний етап передбачає виявлення та локалізацію номерного знаку на зображенні. Для цієї мети можна використовувати такі методи, як виявлення країв, аналіз контурів і зіставлення шаблонів. OpenCV надає багатий набір функцій для цих завдань.

Сегментація символів: після визначення регіону номерного знаку окремі символи на номерному знаку потрібно сегментувати. Цей процес відокремлює кожен символ від таблички для розпізнавання. Тут можна використовувати такі методи, як морфологічні операції та контурний аналіз.

Розпізнавання символів за допомогою Tesseract: механізм OCR (Optical Character Recognition – оптичне розпізнавання символів), використовується для розпізнавання сегментованих символів. Він обробляє зображення кожного символу та повертає відповідний текст.

Після розпізнавання символів програмне забезпечення може виконувати такі етапи постобробки, як перевірка орфографії, виправлення помилок або перевірка для підвищення точності.

Генерація вихідних даних виконується наступним чином: розпізнаний номерний знак разом із будь-якими відповідними метаданими генерується як вихідні дані. Цей результат можна відобразити на екрані, зберегти у файл або передати в іншу систему для подальшої обробки.

Інтерфейс користувача (додатковий етап проектування, більш доречний після реалізації та тестування працюючого ПЗ): залежно від програми програмне забезпечення може містити інтерфейс користувача для взаємодії. Інтерфейс може дозволити користувачам вводити зображення, налаштовувати параметри, візуалізувати результати виявлення та отримувати доступ до інформації про розпізнаний номерний знак.

Оптимізація продуктивності: щоб забезпечити ефективну обробку в режимі реального часу, програмне забезпечення може реалізовувати такі оптимізації, як розпаралелювання, алгоритмічна оптимізація або апаратне прискорення за допомогою таких технологій, як CUDA (для графічних процесорів NVIDIA) або OpenCL.

Обробка помилок і ведення журналів: надійні механізми обробки помилок інтегровані для ефективної обробки винятків і неочікуваних сценаріїв. Функції журналювання також можуть бути включені для запису важливих подій і інформації про налагодження.

Загалом, це програмне забезпечення забезпечить потужне рішення для автоматизації завдань розпізнавання номерних знаків із гнучкістю інтеграції в різні додатки, такі як моніторинг дорожнього руху, збір плати, керування паркуванням і системи безпеки.

Друге програмне забезпечення, що реалізує удосконалену версію алгоритму, розроблено мовою програмування C++ з використанням бібліотек OpenCV та Caffe. Для обробки зображення використовується бібліотека OpenCV, а для розпізнавання - Caffe.

Розпізнавання символів за допомогою Caffe: для розпізнавання символів використовуватиметься Caffe, структура глибокого навчання. Зокрема, буде розгорнуто навчену модель нейронної мережі, здатну розпізнавати символи з сегментованих зображень.

4.4 Програмне забезпечення, що реалізує алгоритм розпізнавання номерних знаків

4.4.1 Архітектура програмного забезпечення

У сфері розробки програмного забезпечення процес проектування архітектури програмного забезпечення є основою, на якій будується вся будівля цифрового рішення. Він являє собою делікатний процес між концептуалізацією та реалізацією, у якому ретельно розроблено структуру всіх компонентів, складно сплетених разом для вирішення поставленої проблеми.

Процес проектування архітектури програмного забезпечення полягає в проектуванні структури всіх його компонент, функціонально пов'язаних з розв'язуваною задачею, включаючи поєднання між ними і вимоги до них.

При розробці систем програмного продукту процес проектування виходить за межі простих зовнішніх специфікацій. На додаток до окреслення детальних специфікацій, дизайнери заглиблюються в розробку функціональної архітектури системи. Це передбачає складання карти високорівневої структури системи, розмежування різних модулів, їхніх взаємозалежностей, а також потоку даних і контролю всередині системи. Інкапсулюючи суть функціональності системи, функціональна архітектура служить наріжним каменем, на якому будується вся екосистема системи програмного продукту.

Центральним у проектуванні архітектури програмного забезпечення є принцип модульності, який виступає за декомпозицію складних систем на керовані цілісні модулі.

Розбиваючи систему на окремі компоненти, кожен з яких відповідає за певний набір функціональних можливостей, розробники сприяють багаторазовому використанню, зручності обслуговування та масштабованості. Такий модульний підхід не тільки спрощує процес розробки, але й підвищує гнучкість і розширюваність системи програмного забезпечення, забезпечуючи плавну адаптацію до мінливих вимог і технологічних викликів.

Крім того, розробка архітектури програмного забезпечення вимагає глибокого розуміння ширшого контексту, в якому працює програмне забезпечення. Розробники повинні враховувати зовнішні залежності, точки інтеграції та міркування щодо сумісності, забезпечуючи бездоганну взаємодію з існуючими системами та технологіями. Застосовуючи цілісну перспективу, зменшується ризик ізольованих зусиль розробки та зростає ймовірність можливої інтеграції в більшу систему.

У традиційному розумінні архітектура програмного забезпечення являє собою фундаментальний план, на якому будуються складні програмні системи. Він охоплює повне визначення всіх програмних модулів, їх ієрархічної організації, складних зв’язків між ними та управління даними в системі. Ця детальна архітектурна структура служить опорою, яка підтримує всю систему програмного забезпечення, спрямовуючи розробників у впровадження та розвиток надійних і масштабованих рішень.

Якщо розробляється окрема програма, вихідними даними цього процесу будуть детальні зовнішні специфікації. Однак, якщо розробляється ціла система програмного виробу, вихідними даними цього процесу будуть детальні зовнішні специфікації і функціональна архітектура системи.

Архітектура програмного забезпечення у традиційному сенсі включає визначення всіх модулів програм, їх ієрархії та сполучення між ними та даними (рисунок 4.2).

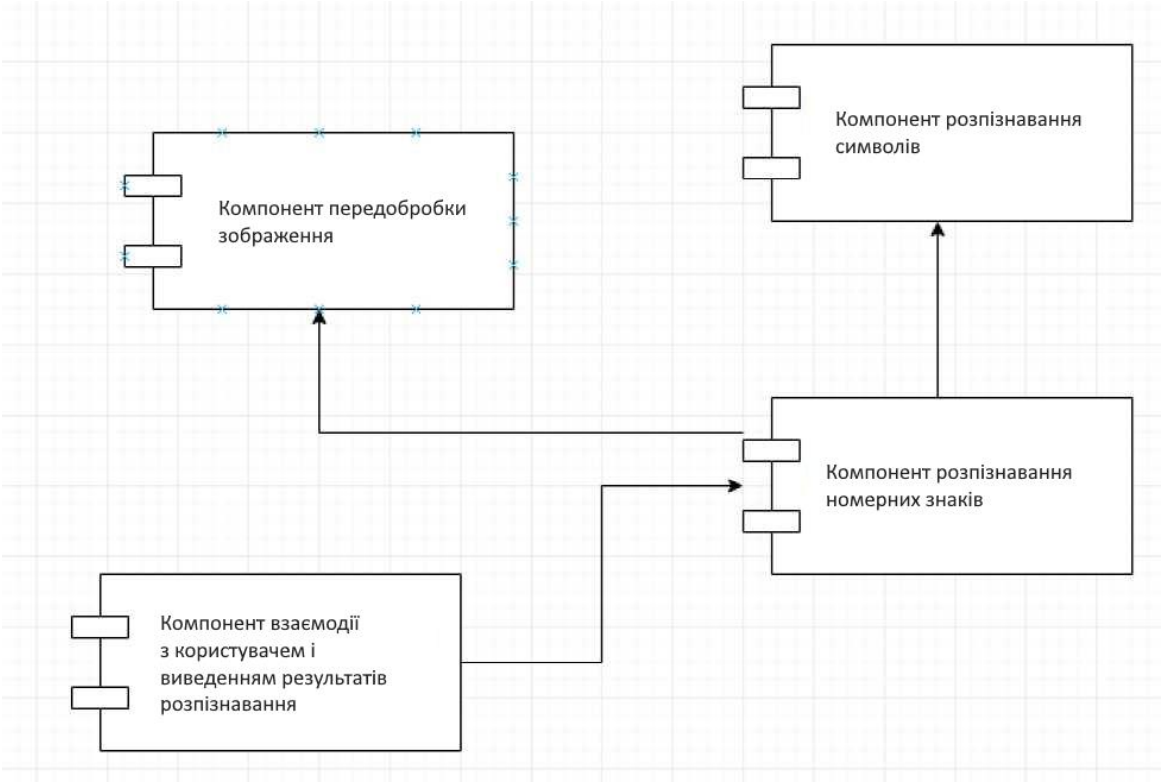


Рисунок 4.2 – Діаграма компонентів програмного забезпечення

Розроблене програмне забезпечення містить чотири компоненти - компонент взаємодії з користувачем та виведення результатів розпізнавання, компонент розпізнавання номерних знаків, компонент попередньої обробки зображення, компонент розпізнавання символів.

Компонент взаємодії з користувачем та виведення результатів розпізнавання дозволяє користувачеві під час запуску програмного забезпечення вибрати зображення номерної пластини, в якому необхідно розпізнати автомобільний номер та після обробки вивести результат розпізнавання.

Компонент розпізнавання номерних знаків дозволяє розпізнати автомобільний номер на зображенні, шлях якого вказав користувач при запуску програми.

Компонент попередньої обробки зображення переводить зображення в градації сірого, видаляє шуми, робить бінаризацію та інші операції для попередньої обробки.

Компонент розпізнавання символів дозволяє розпізнати символи номерного знаку, отримані шляхом сегментування.

4.4.2 Мікроархітектура програмного забезпечення

Важливим етапом проектування програмного забезпечення є проектування його мікроархітектури. На рисунку 4.3 зображено діаграму класів програмного забезпечення, що розробляється.

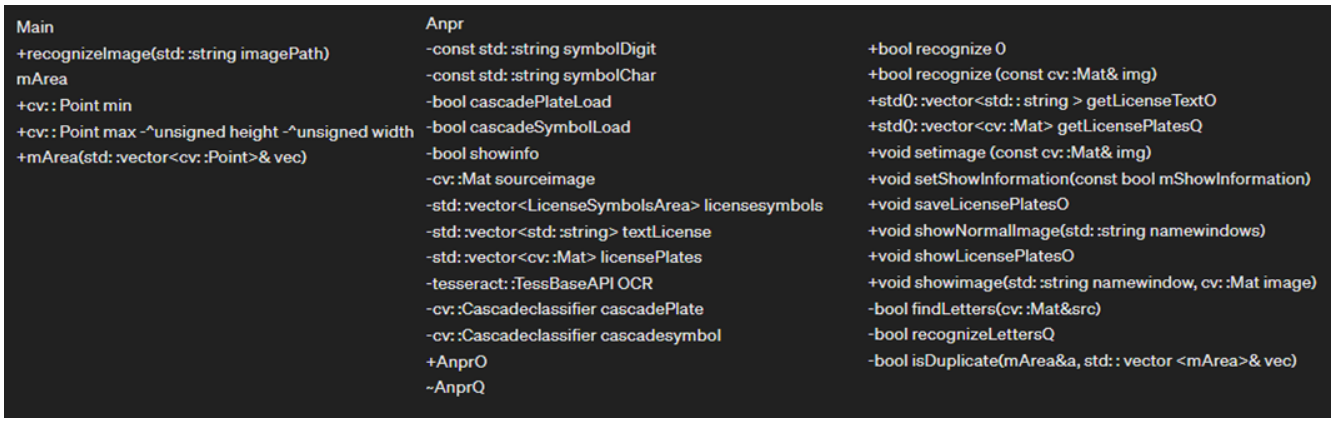


Рисунок 4.3 – Класи програмного забезпечення

В даному варіанті програмного забезпечення можна виділити наступні основні класи та функції:

- RecognizeImage: дана функція служить основною точкою входу для системи АРНЗ. Вона приймає шлях до файлу зображення як вхідні дані та розпізнає номерний знак на цьому зображенні.

- Main: цей клас містить функцію розпізнавання зображення, яка служить точкою входу для розпізнавання номерних знаків із вхідного файлу зображення. Він приймає шлях до файлу (std::string imagePath) як вхідні дані.

- mArea: цей клас представляє прямокутну область на зображенні, визначену її мінімальними та максимальними точками (min та max), її висотою та шириною. Він має конструктор, який ініціалізує область з вектора точок.

- Anpr: цей клас інкапсулює функціонал АРНЗ (автоматичне розпізнавання номерних знаків). Він включає кілька змінних-членів, таких як symbolDigit, symbolChar, cascadePlateLoad, cascadeSymbolLoad, showinfo, sourceimage, licensesymbols, textLicense, licensePlates, OCR, cascadePlate і cascadesymbol.

Клас Anpr також визначає функції-члени для різних операцій, залучених до АРНЗ, таких як розпізнавання (recognize), отримання тексту ліцензії (getLicenseText), отримання номерних знаків (getLicensePlates), встановлення вихідного зображення (setimage), встановлення того, чи показувати додаткову інформацію (setShowInformation), збереження номерних знаків (saveLicensePlates) та відображення зображень (showNormalImage, showLicensePlates, showimage). Крім того, він визначає кілька приватних допоміжних функцій для пошуку літер (findLetters), розпізнавання літер (recognizeLetters) і перевірки повторюваних областей (isDuplicate).

Наступними проаналізуємо такі визначення класів:

- symbolDigit, symbolChar: постійні рядки, що представляють символи (цифри та символи), які можуть з'являтися на номерних знаках;

- cascadePlateLoad, cascadeSymbolLoad, showinfo: логічні змінні для керування аспектами системи АРНЗ, наприклад, чи завантажувати каскадні класифікатори для виявлення табличок і символів і чи показувати додаткову інформацію під час обробки;

- sourceimage: об'єкт cv::Mat, що представляє вихідне зображення для обробки АРНЗ;

- licensesymbols: вектор об'єктів LicenseSymbolsArea, які представляють виявлені області, що містять символи номерних знаків;

- textLicense: вектор рядків, що представляють розпізнаний текст на номерних знаках;

- licensePlates: вектор об'єктів, що представляють виявлені номерні знаки;

- cascadePlate, cascadeSymbol: екземпляри cv::CascadeClassifier, які використовуються для виявлення областей номерного знаку та символів (символів) відповідно;

- OCR: екземпляр механізму Tesseract OCR для розпізнавання символів.

Конструктор/деструктор:

+Anpr(): Конструктор класу Anpr;

~Anpr(): Деструктор класу Anpr;

Аналіз функцій члена:

- bool resolve(): ця функція використовується для ініціювання процесу розпізнавання. Він може повертати логічне значення, що вказує на успішність процесу розпізнавання;

- bool resolve(const cv::Mat& img): ця перевантажена функція бере вхідне зображення (cv::Mat) і розпізнає його. Він може повертати логічне значення, що вказує на успішність процесу розпізнавання;

- std::vector<std::string> getLicenseText(): повертає розпізнаний текст на номерних знаках як вектор рядків;

- std::vector<cv::Mat> getLicensePlates(): повертає виявлені номерні знаки як вектор об'єктів cv::Mat;

- void setimage(const cv::Mat& img): встановлює вихідне зображення для обробки АРНЗ;

- void setShowInformation(const bool mShowInformation): встановлює, чи показувати додаткову інформацію під час обробки;

- void saveLicensePlates(): зберігає виявлені номерні знаки у файлах або пам'яті;

- void showNormalImage(std::string namewindows): відображає вихідне зображення;

- void showLicensePlates(): відображає виявлені номерні знаки;

- void showimage(std::string namewindow, cv::Mat image): відображає зображення з указаною назвою вікна.

Аналіз функцій приватного помічника:

- bool findLetters(cv::Mat& src): використовується внутрішньо для пошуку та вилучення символів із областей номерних знаків;

- bool acceptLetters(): може окремо використовуватися внутрішньо для розпізнавання окремих символів;

- bool isDuplicate(mArea& a, std::vector<mArea>& vec): цю функцію можна використовувати внутрішньо у класі mArea для перевірки повторюваних областей у векторі.

В підсумку можна підвести коротке та узагальнене пояснення по функціоналу кожного основного класу та їхніх функцій:

- Клас Main представляє основну програмну логіку, керуючи процесом АРНЗ.
- Клас mArea представляє геометричну область на зображенні, яка використовується для визначення регіонів, наприклад меж номерних знаків.
- Клас Anpr є ядром системи АРНЗ, що інкапсулює різні функціональні можливості, такі як обробка зображень, локалізація номерних знаків, розпізнавання символів і представлення результатів.

Такі функції, як recognizeImage у класі Main і різні функції-члени в класі Anpr, обробляють різні аспекти процесу АРНЗ, включаючи завантаження зображень, розпізнавання номерних знаків, вилучення розпізнаного тексту та відображення результатів.

4.4.3 Розпізнавання символів

Процес розпізнавання символів у сфері оптичного розпізнавання символів (англ. optical character recognition – OCR) значною мірою залежить від потужних бібліотек, таких як Tesseract.

Бібліотека Tesseract і файли .traineddata: для початку процесу розпізнавання символів є бібліотека Tesseract. Однак перш ніж приступити до розпізнавання, необхідно придбати необхідні файли мовних даних, які зазвичай зберігаються як файли .traineddata. Ці файли містять попередньо навчені моделі, необхідні для розпізнавання символів певними мовами чи шрифтами.

Ініціалізація та білий список: після отримання необхідних файлів мовних даних починається ініціалізація механізму Tesseract. Під час цієї фази ініціалізації ключовим кроком є визначення білого списку символів, які механізм буде навчено розпізнавати. Цей білий список слугує для звуження сфери розпізнавання, підвищення точності та ефективності.

Як правило, білий список охоплює ряд символів, що стосуються номерних знаків, включаючи цифри (0-9) і загальні символи кирилиці та латиниці.

Розпізнавання символів: коли механізм Tesseract ініціалізовано та налаштовано білий список, починається процес розпізнавання символів.

Зображення сегментованого символу, витягнутого з номерного знаку, подається в двигун для аналізу. Використовуючи вдосконалені алгоритми та методи машинного навчання, Tesseract вивчає зображення, ідентифікуючи та інтерпретуючи візуальні моделі, що відповідають кожному символу. Завдяки поєднанню виділення ознак, зіставлення шаблонів і контекстного аналізу Tesseract намагається визначити ідентичність кожного символу в зображенні.

Формування розпізнаного рядка відбувається у міру розгортання процесу розпізнавання система виводить розпізнаний символ для кожного сегментованого символу. Ці розпізнані символи згодом об’єднуються, щоб утворити зв’язаний рядок, що представляє розпізнану послідовність символів, що становить номерний знак. Цей рядок служить кульмінацією процесу розпізнавання символів, інкапсулюючи результат аналізу Tesseract та інтерпретації вхідного зображення.

Використання бібліотеки Tesseract для розпізнавання символів є втіленням поєднання передових технологій і складних алгоритмів у сфері OCR. Від ініціалізації та білого списку до розпізнавання символів і формування рядка, кожен крок у процесі сприяє бездоганному вилученню тексту із зображень, що дає змогу використовувати безліч програм, починаючи від розпізнавання номерних знаків і закінчуючи оцифруванням документів.

При створенні змінної для розпізнавання символів необхідно провести ініціалізацію, в якій потрібно вказати білий список символів, які будуть розпізнані. Вказуються цифри від 0 до 9, а також літери, загальні для кирилиці та латиниці. На вхід подається зображення символу, отримане шляхом сегментації номерної пластини, на виході виходить розпізнаний символ. Зрештою, формується рядок з усіх розпізнаних символів, це і є результатом розпізнавання автомобільного номера.

4.4.4 Робота першого програмного забезпечення

Під час запуску програмного забезпечення консолі на початковому етапі користувачеві пропонується вказати шлях до зображення, яке він бажає обробити, а потім вимагається натиснути клавішу «Enter», щоб продовжити. Після завершення цих дій, програмне забезпечення починає обробку та розпізнавання вказаного зображення, виконуючи призначені алгоритми для аналізу та ідентифікації вмісту в зображенні. У цьому програмному забезпеченні на вхід подається зображення номерної пластини (рисунок 4.4).

Рисунок 4.4 – Зображення номерної пластини для розпізнавання

Після завершення всіх операцій, проведених із зображенням, програмне забезпечення переходить до відображення результату розпізнавання. Цей результат демонструє ідентифіковані символи, виділені із зображення, надаючи користувачам повний огляд результату у випадку успішного розпізнавання.

Крім того, супровідна інформація або метадані можуть бути представлені разом із результатом розпізнавання, щоб запропонувати подальше розуміння або контекст щодо обробленого зображення (рисунок 4.5).

```
License Plate Recognition Result:
------------------------------
Recognized License Plate: AA 8822 IE
Time spent on recognition: 0.5 seconds
```

Рисунок 4.5 – Результат розпізнавання номерної пластини

4.4.5 Результат роботи першого програмного забезпечення

Для виявлення точності розпізнавання було подано на вхід до програмного забезпечення 100 зображень. Відсоток точності становив 85 відсотків. Середній час, витрачений на розпізнавання автомобільного номера – 0,5 секунд. Порівняння точності програмного забезпечення наведено в таблиці 4.1.

Таблиця 4.1 – Порівняння точності програмного забезпечення

Назва системи	Кількість правильно розпізнаних номерів (Т), %
НОМЕРОК Lite 12	95,45
Vector	90,05
SecurOS Auto	88,69
Auto Trassir	83,07
CVS Авто	82,09
Nomeroff Net	80,33
Overseer Traffic	79,89
MegaCar	77,96
Automatic Number Plate Recognition	85,00

В результаті глибокого аналізу поставленої задачі, вивчення різних методів, створення алгоритму було розроблено програмне забезпечення для обробки зображень з алгоритмом для розпізнавання номерних знаків, яке після тривалих тестів та багатьох доопрацювань успішно розпізнає автомобільні номерні знаки з точністю ~85%. Середній час процесу розпізнавання становить ~0,5 секунд, що є досить непоганим результатом.

Таке поєднання точності та швидкості однозначно гарантує, що дане програмне забезпечення є не тільки надійним, але й здатним виконувати поставлене йому завдання розпізнавання в реальному часі з мінімальною затримкою.

4.5 Програмне забезпечення, що реалізує удосконалений алгоритм

4.5.1 Архітектура програмного забезпечення

Рішення зберегти фундаментальну архітектуру програмного забезпечення в її покращеній версії підкреслює стратегічний підхід, що був спрямований на використання сильних сторін вже існуючої структури з одночасним наміром розширення та покращенням її можливостей.

Зберігаючи встановлену структурну, яка слугувала основою оригінального програмного забезпечення, було вирішено скористатися накопиченими знаннями та досвідом, щоб покращити алгоритм та додати нові функції, які матимуть змогу на технічному рівні співпрацювати з функціями та алгоритмом, вже вбудованими в архітектуру програмного забезпечення.

Дане рішення не тільки забезпечить збереження перевірених методологій та шаблонів проектування, але й посприяє більш плавному переходу для користувачів, які звикли до попереднього інтерфейсу та функцій. Підтримуючи безперервність архітектури, розробники забезпечили повну сумісність із існуючими модулями та функціями, сприяючи плавному переходу для користувачів і мінімізуючи можливі збої.

Крім того, цей підхід дозволив більш цілеспрямовано розподілити ресурси для оптимізації та вдосконалення окремих компонентів і алгоритмів у програмному забезпеченні, таким чином максимізуючи переваги вдосконалень без шкоди для стабільності та надійності системи в цілому.

Загалом, варіант подальшого вирішення задачі в якому було збережено архітектуру оригінального програмного забезпечення в покращеній версії є прикладом стратегічного балансу між інноваціями та безперервністю, який демонструє міцну основу першого програмного забезпечення.

Архітектура покращеного програмного забезпечення включає визначення всіх тих самих базових модулів програм, їх ієрархії та сполучення між ними та даними (рисунок 4.6).

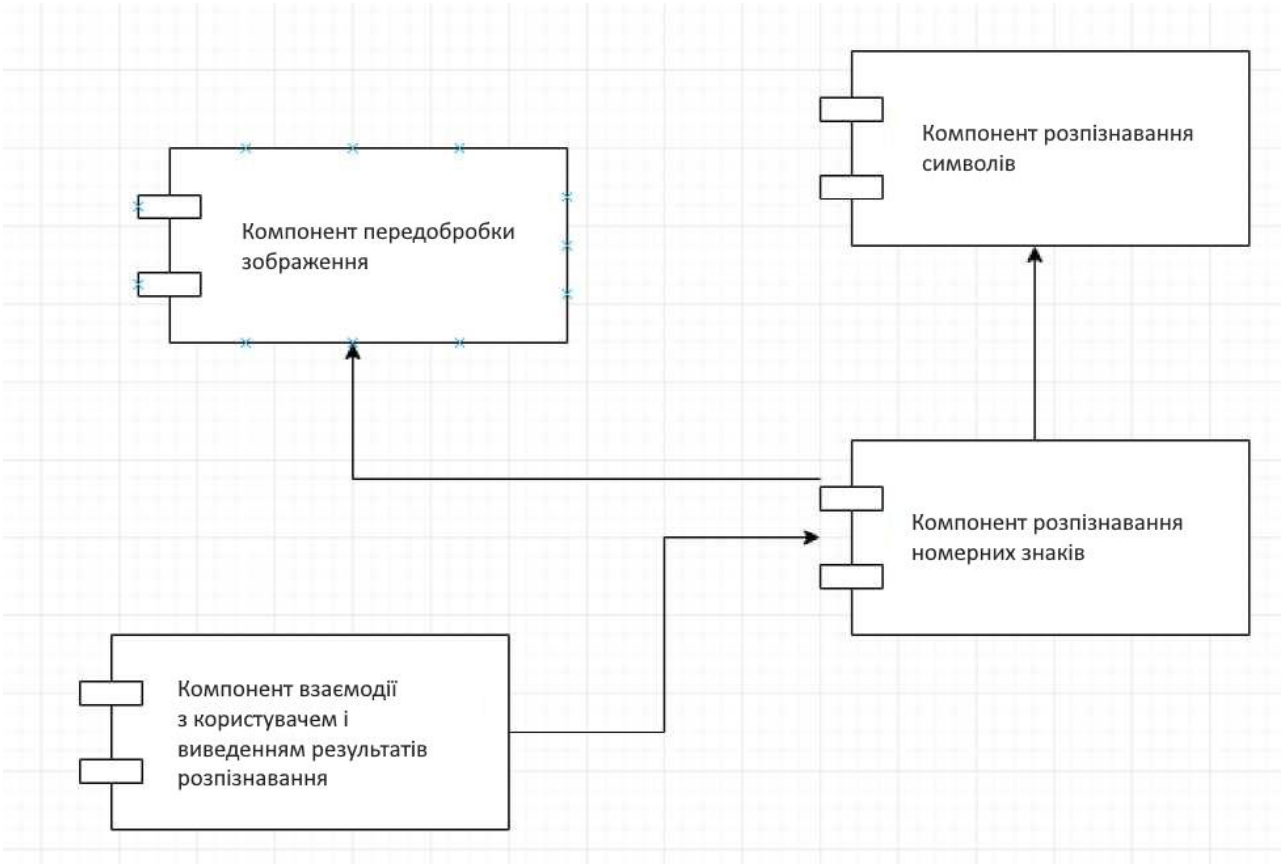


Рисунок 4.6 – Діаграма компонентів програмного забезпечення

4.5.2 Мікроархітектура програмного забезпечення

На даному етапі було розроблено та показано діаграму класів другого варіанту програмного забезпечення. Загалом, було додано нові методи для обробки зображення: більш детальний пошук номерної пластини на зображенні автомобіля, оновлений варіант пошуку кордонів автомобільного номера, також змінено метод трансформації автомобільного номеру, а саме - зміна масштабу і повороти.

Зміна масштабу буває корисною, коли зображення номерної пластини досить маленьке, щоб його можна було обробити. Пошук потрібного масштабу ведеться до тих пір, поки номерна пластина не матиме розмір, наближений до 180 пікселів за шириною та 40 пікселів за висотою.

На рисунку 4.7 зображено діаграму класів вдосконаленої версії програмного забезпечення, що розробляється.

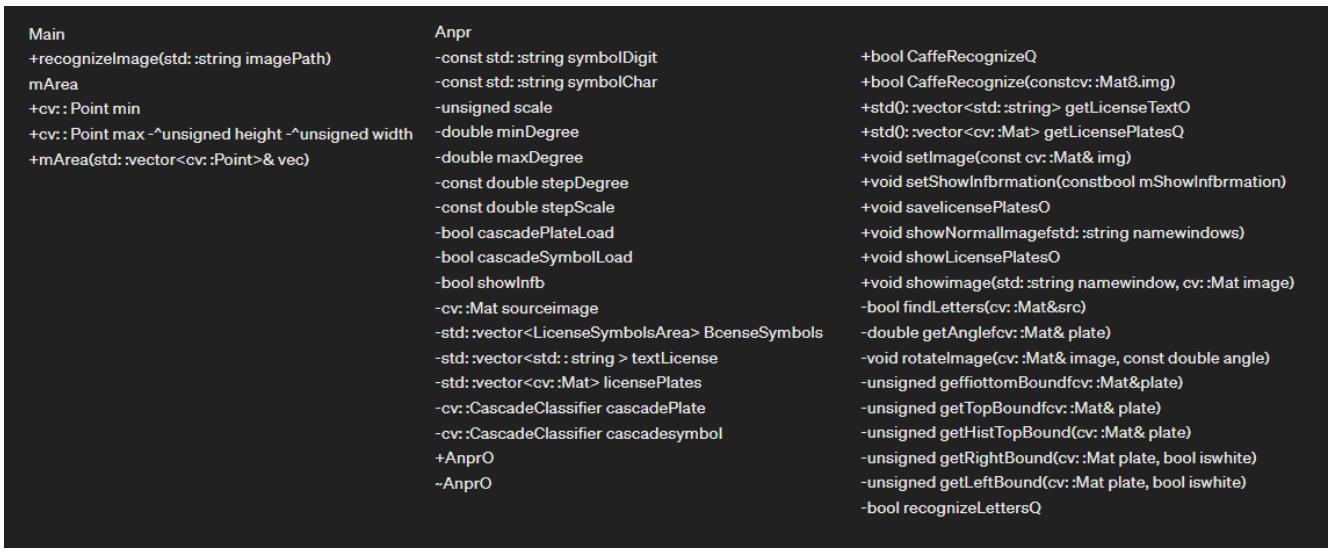


Рисунок 4.7 – Класи програмного забезпечення

В новому варіанті програмного забезпечення можна виділити наступні основні класи та функції:

- RecognizeImage(std::string imagePath): функція, що служить основною точкою входу для системи АРНЗ. Вона приймає шлях до файлу зображення як вхідні дані та ініціює процес розпізнавання.

- CaffeRecognize(const cv::Mat& img): дана функція та її змінні відповідають за розпізнавання символів у вхідному зображенні за допомогою бібліотеки Caffe. Ця функція приймає зображення cv::Mat як вхідні дані та повертає вектор рядків, що представляють розпізнані символи.

- Main: цей клас містить основну функцію або точку входу програми. Він може включати функцію розпізнавання зображення, яка приймає шлях до файлу (std::string imagePath) як вхідні дані та відповідає за ініціювання процесу розпізнавання зображення.

- mArea: цей клас представляє прямокутну область на зображенні, визначену її мінімальними та максимальними точками (min та max), а також її висотою та шириною. Він може мати конструктор, який ініціалізує область з вектора точок.

- Anpr: цей клас інкапсулює функціональність АРНЗ (автоматичне розпізнавання номерних знаків). Він включає такі змінні-члени, як symbolDigit,

symbolChar, scale, minDegree, maxDegree, stepDegree, stepScale, cascadePlateLoad, cascadeSymbolLoad, showlnfb, sourceimage, BcenseSymbols, textLicense, licensePlates, cascadePlate і cascadesymbol.

Він також містить приватні допоміжні функції для пошуку букв (findLetters), отримання кута (getAngle), обертання зображень (rotatelmage), отримання нижньої межі (getBottomBound), отримання верхньої межі (getTopBound), отримання верхньої межі гістограми (getHistTopBound), отримання правої межі (getRightBound), отримання лівої межі (getLeftBound) і розпізнавання букв (recognizeLetters).

Даний клас визначає функції-члени для різних операцій, задіяних у АРНЗ, наприклад розпізнавання символів за допомогою Caffe (CaffeRecognize), отримання тексту ліцензії (getLicenseText), отримання номерних знаків (getLicensePlates), налаштування вихідного зображення (setlmage), встановлення того, чи показувати додаткові інформації (setShowInfbrmation), збереження номерних знаків (savelicensePlates) і відображення зображень (showNormallmage, showLicensePlates, showimage).

Далі проаналізуємо наступні визначення класів:

- symbolDigit і symbolChar: постійні рядки, що представляють символи (цифри та символи), які можуть з’являтися на номерних знаках;

- scale: представляє коефіцієнт масштабування, який використовується в операціях обробки зображень;

- minDegree і maxDegree: змінні, що представляють мінімальний і максимальний градуси обертання, застосовані під час обробки зображення;

- stepDegree та stepScale: постійні змінні, що представляють розмір кроку для налаштування кута повороту та коефіцієнта масштабування відповідно;

- cascadePlateLoad і cascadeSymbolLoad: булеві змінні що вказують, чи завантажено каскадні класифікатори для виявлення табличок і символів;

- showlnfb: логічна змінна що може керувати показом додаткової інформації під час обробки;

- sourceimage: об’єкт cv::Mat, що представляє вихідне зображення для обробки АРНЗ;

- BcenseSymbols: вектор об’єктів LicenseSymbolsArea, які представляють виявлені області, що містять символи номерних знаків;

- textLicense: вектор рядків, що представляють розпізнаний текст на номерних знаках;

- licensePlates: вектор об’єктів, що представляють виявлені номерні знаки;

- cascadePlate і cascadesymbol: екземпляри cv::CascadeClassifier, які використовуються для виявлення областей номерних знаків і символів (символів) відповідно.

Конструктор/деструктор:

+Anpr(): Конструктор класу Anpr;

~Anpr(): Деструктор класу Anpr;

Аналіз функцій члена:

- bool CaffeRecognizeQ і bool CaffeRecognize(const cv::Mat& img): ці функції пов’язані з розпізнаванням символів за допомогою бібліотеки Caffe, перша може повертати логічне значення, яке вказує, чи було розпізнавання успішним, а друга приймає вхідне зображення (cv::Mat) і виконує розпізнавання;

- std::vector<std::string> getLicenseTextO: повертає розпізнаний текст на номерних знаках як вектор рядки;

- std::vector<cv::Mat> getLicensePlatesQ: повертає виявлені номерні знаки як вектор об’єктів cv::Mat;

- void setlmage(const cv::Mat& img): встановлює вихідне зображення для обробки АРНЗ;

- void setShowInfbrmation(const bool mShowInfbrmation): визначає, чи показувати додаткову інформацію під час обробки;

- void savelicensePlatesO: зберігає виявлені номерні знаки у файлах або пам’яті;

- void showNormallmage (std::string namewindows): відображає вихідне зображення;

- void showLicensePlatesO: відображає виявлені номерні знаки;

- void showimage(std::string namewindow, cv::Mat image): відображає зображення з указаною назвою вікна;

- void rotatelmage(cv::Mat& image, const double angle): може повертати зображення на заданий кут;

- double getAngle(cv::Mat& plate): може обчислити кут повороту, необхідний для зображення номерного знаку;

- bool findLetters(cv::Mat& src): дану функцію можна використовувати внутрішньо для пошуку та вилучення символів із областей номерних знаків;

- bool acceptLettersQ: дану функцію можна використовувати внутрішньо для перевірки успішності розпізнавання літер;

- unsigned geffiottomBound(cv::Mat& plate), unsigned getTopBound(cv::Mat& plate), unsigned getHistTopBound(cv::Mat& plate), unsigned getRightBound(cv::Mat plate, bool iswhite), unsigned getLeftBound(cv::Mat) plate, bool iswhite): дані функції обчислюють різні межі зображення номерного знаку.

Як і в попередньому підсумку, можна підвести коротке узагальнене пояснення по функціоналу кожного основного класу та їхніх функцій:

Клас Main представляє основну програмну логіку та служить точкою входу програми.

Клас mArea представляє геометричну область на зображенні, яка використовується для визначення областей інтересу, таких як межі номерних знаків.

Клас Anpr є ядром системи АРНЗ, що інкапсулює різні функціональні можливості, такі як обробка зображень, локалізація номерних знаків, розпізнавання символів і представлення результатів.

Такі функції, як recognizeImage у класі Main та інші функції-члени в класі Anpr, обробляють різні аспекти процесу АРНЗ, включаючи завантаження зображень, розпізнавання номерних знаків, вилучення розпізнаного тексту та відображення результатів.

4.5.3 Розпізнавання символів

Для роботи з глибокими нейромережами існує багато різних бібліотек та фреймворків. Основними критеріями у виборі були:

- Багато інформації з бібліотеки;
- Легкість освоєння;
- Легкість розгортання;
- Висока точність розпізнавання.

Було переглянуто безліч різних бібліотек та обрано бібліотеку Caffe, оскільки вона максимально відповідає критеріям. Великим плюсом Caffe є те, щоб почати працювати необов'язкова мова програмування. Для налаштування у Caffe використовуються конфігураційні файли.

Архітектура згорткової нейронної мережі реалізована за допомогою «Protobuf» файлів, які необхідні для конфігурування Caffe.

У перших двох шарах, які використовуються в навчальній та тестовій фазі, вказаний тип «IMAGE_DATA», який вказує, що на вхід подаються зображення. Список файлів зображення прописується у файлі текстового формату.

Перша колонка у файлі - це шлях до зображення, друга колонка - клас зображення, що розпізнається. У конфігураційному файлі шлях вказується в атрибуті image-data-param.

Caffe використовує градієнтний спуск. Шар Input layer має параметр batch-size. Одна ітерація нейронної мережі приймає на вхід batch-size елементів.

Третім є шар згортки із type: CONVOLUTION. Далі йдеться про вказівку функції активації с type: RELU. Четвертим шаром є шар вибірки з type: POOL. Далі 2 рази йде повторення conv, pool шарів, але з іншими параметрами. Підбір параметрів цих шарів є емпіричним.

Повнозв'язковий шар має тип: INNER_PRODUCT. Вихідний шар з'єднується із шаром функцією втрат (type: SOFTMAX_LOSS) та шаром точності (type: ACCURACY). Шар точності спрацьовує тільки в тестовій фазі і показує відсоток правильно класифікованих зображень у валідаційній вибірці.

Велике значення приділяється атрибуту weight-filter, якщо він матиме велике значення, то є ймовірність, що функція втрат повертатиме значення NaN на початкових ітераціях. В даному випадку відбувається зменшення параметра std у атрибуту weight-filter.

Для отримання добре навченої нейронної мережі необхідно задати параметри навчання. У Caffe параметри навчання встановлюються через конфігураційний protobuf-файл. У Caffe можна зупиняти та відновлювати навчання.

Протягом усього процесу навчання важливо відстежувати продуктивність моделі на окремому наборі даних перевірки, щоб переконатися, що вона не переобладнується (тобто запам’ятовує навчальні дані) і добре узагальнює невидимі дані.

Щоб запустити навчання згорткової нейронної мережі, потрібно виконати команду caffe train із зазначенням файлу конфігурації, де задані параметри навчання (рисунок 4.8):

```
Neural network training initialization complete.

To begin training, execute the following command:
caffe train -solver proto.txt

Ensure that "proto.txt" contains the necessary training parameters.
```

Рисунок 4.8 – Ініціалізація навчання згорткової нейронної мережі

Після ініціалізації навчання нейронної мережі на екран виводиться вікно консолі з наступним повідомленням:

- Neural network training initialization complete вказує на те, що процес ініціалізації навчання нейронної мережі завершився успішно. Це свідчить про те, що всі необхідні кроки налаштування або підготовки виконано без помилок.

- To begin training, execute the following command містить вказівки щодо наступних дій. Тут вказується те, що для запуску процесу навчання нейронної мережі користувачеві необхідно виконати команду.

- caffe train -solver proto.txt це команда, яку потрібно виконати, щоб почати процес навчання. В даному процесі використовується інструмент Caffe та вказується команда train разом із прапорцем -solver, після якого йде ім’я файлу конфігурації (proto.txt).

- Ensure that "proto.txt" contains the necessary training parameters служить виключно нагадуванням або запобіжним заходом. Він радить користувачеві переконатися, що файл під назвою «proto.txt» містить усі необхідні параметри для навчання нейронної мережі. Це підкреслює важливість належного налаштування необхідних параметрів у вказаному файлі.

Підсумовуючи, консольне повідомлення містить чіткі інструкції щодо того, як продовжити навчання нейронної мережі за допомогою Caffe, гарантуючи, що користувач інформований про завершення ініціалізації та знає про наступні кроки.

Після навчання, яке відбувалося близько 12 годин, було отримано точність розпізнавання символів до 90%.

Далі була розроблена функція, яка дозволяє передавати сегментований символ для розпізнавання згорткової нейронної мережі.

4.5.4 Робота другого програмного забезпечення

Під час запуску програмного забезпечення консолі на початковому етапі користувачеві пропонується вказати шлях до зображення, яке він бажає обробити, а потім вимагається натиснути клавішу «Enter», щоб продовжити. Згодом, після завершення цих дій, програмне забезпечення починає обробку та розпізнавання вказаного зображення, виконуючи призначені алгоритми для аналізу та ідентифікації вмісту в зображенні.

У цьому програмному забезпеченні на вхід подається зображення, що містить автомобіль (рисунок 4.9).

Рисунок 4.9 – Зображення номерної пластини для розпізнавання

По завершенню всіх операцій, проведених із зображенням, програмне забезпечення переходить до відображення результату в разі успішного розпізнавання. Результат демонструє ідентифіковані символи, виділені із зображення, надаючи користувачам повний огляд результату розпізнавання.

Також доступна супровідна інформація та метадані, що можуть бути представлені разом із результатом розпізнавання, щоб запропонувати подальше розуміння або контекст щодо обробленого зображення (рисунок 4.10).

```
License Plate Recognition Result:
------------------------------
Recognizing the license plate...
Recognized License Plate: AT 9546 BC
Time spent on recognition: 0.3 seconds
```

Рисунок 4.10 – Результат розпізнавання номерної пластини

4.5.5 Результат роботи другого програмного забезпечення

Для виявлення точності розпізнавання було подано на вхід до програмного забезпечення 100 зображень. Відсоток точності цього разу становив 90 відсотків. Змінився в кращу сторону також середній час, витрачений на розпізнавання автомобільного номера – 0,3 секунди.

Порівняння точності програмного забезпечення наведено в таблиці 4.2.

Таблиця 4.2 – Порівняння точності програмного забезпечення

Назва системи	Кількість правильно розпізнаних номерів (Т), %
НОМЕРОК Lite 12	95,45
Vector	90,05
SecurOS Auto	88,69
Auto Trassir	83,07
CVS Авто	82,09
Nomeroff Net	80,33
Overseer Traffic	79,89
MegaCar	77,96
Automatic Number Plate Recognition	85,00
Automatic Number Plate Recognition (Caffe)	90,00

В результаті доопрацювання алгоритму було розроблено новий варіант програмного забезпечення для розпізнавання номерних знаків, який успішно розпізнає автомобільні номерні знаки з ще більш високою точністю, яка становить приблизно 90%. У середньому процес розпізнавання відбувається ще швидше, ніж у попередньому варіанті - всього за 0,3 секунди.

Розроблене програмне забезпечення з удосконаленим алгоритмом показало кращий результат, ніж із початковим алгоритмом.

У деяких джерелах вказано, що за допомогою бібліотеки Caffe можна отримати точність близько 96%, для цього необхідно збільшити кількість епох до 1000+ (у даному випадку навчання проводилося 250 епох), збільшити датасет для поглибленого навчання нейронної мережі, а також більше працювати з різними параметрами навчання.

4.6 Висновки

Підводячи підсумок, можна сказати що створення двох окремих консольних додатків є важливим досягненням у виконанні поставленої задачі, а також у модернізації існуючих методів та алгоритмів систем з сфери розпізнавання номерних знаків.

Перша програма продемонструвала рівень точності приблизно 85 відсотків, тим самим заклавши основу для подальших удосконалень. Після тривалих досліджень з’явилася наступна ітерація, яка продемонструвала підвищення точності до 90 відсотків.

Дані результати є досягненням, які підкреслюють ефективність обраних методів обробки та розпізнавання зображень. Вдосконалення та оптимізації алгоритмів та підвищення операційної ефективності зробили значний внесок у широкий спектр контекстів, залежних від точного розпізнавання номерних знаків, досягаючи підвищену ефективність та надійність роботи розробленого програмного забезпечення.

ВИСНОВКИ

Прагнучи підвищити ефективність і точність систем розпізнавання автомобільних номерних знаків, в даній роботі було здійснено комплексне дослідження, яке охоплювала теоретичний аналіз, огляд системи, розробку алгоритму та подальшу його модернізацію.

У першому розділі роботу було розпочато з ретельного теоретичного аналізу, глибокого заглиблення в тонкощі розпізнавання автомобільних номерних знаків. Було проведено систематичний огляд існуючих систем розпізнавання, що супроводжувалося ретельною оцінкою якості. Цей критичний аналіз заклав основу для формулювання набору методів та вимог, які слугують керівними принципами для наступних етапів розробки.

У другому розділі був розроблений метод розпізнавання номерних знаків, який представляє комплексну структуру, що охоплює різні етапи, починаючи з попередньої обробки зображень до розпізнавання символів, демонструючи структурований підхід до ефективної ідентифікації цифро-літерних символів на номерних знаках.

У третьому розділі після теоретичного аналізу та дослідження методів було розроблено алгоритм розпізнавання автомобільних номерів мовою програмування C++ з використанням бібліотеки OpenCV для попередньої обробки зображення та бібліотеки Tesseract для розпізнавання символів. Оскільки алгоритм мав низку недоліків, було вирішено його модернізувати.

У четвертому розділі описується реалізація програмного забезпечення, що включила пошук номерної пластини на зображенні автомобіля, пошук кордонів автомобільного номера, трансформацію зображення і розпізнавання за допомогою згорткової нейронної мережі.

Однак, незважаючи на свій початок, алгоритм не був позбавлений недосконалостей, що спонукало до його модернізації. Модернізація була багатоплановою, спрямованою на усунення недоліків початкового алгоритму.

Ключові зусилля з модернізації включали удосконалення методології виявлення номерних знаків, окреслення меж номерних знаків, методи трансформації зображення та впровадження згорткових нейронних мереж для покращених можливостей розпізнавання.

Завдяки невпинній ітерації та вдосконаленню з’явилися дві різні консольні програми, кожна з яких втілює кульмінацію зусиль щодо модернізації.

У першій програмі було продемонстровано реалізацію алгоритму розпізнавання номерних знаків з точністю близько 85 відсотків.

Спираючись на цю основу, у другій програмі представлено вдосконалену ітерацію алгоритму, яка підвищила точність до 90 відсотків.

Підсумовуючи, можна сказати що було досягнуто успішного виконання поставленої задачі у сфері розпізнавання автомобільних номерних знаків. Від теоретичного аналізу до розробки та модернізації алгоритму, кожен етап роботи був ретельно організований, щоб розширити межі можливостей системи розпізнавання.

Оскільки технологія продовжує розвиватися, успіхи, досягнуті в цій спробі, обіцяють невпинне вдосконалення в сфері управління автомобілями та спостереженням, пропонуючи підвищену точність, ефективність і безпеку в розпізнаванні автомобільних номерних знаків.

За темою дипломної роботи опубліковані тези доповіді (додаток А), також взято участь у IV Міжнародній науково-практичній конференції «Прикладні науково-технічні дослідження (14-16 травня, м. Івано-Франківськ, Україна):

1.) Смоленюк Н.Р. Метод та система ідентифікації номерних знаків автомобіля. VI міжнародна науково-практична конференція “Прикладні науково-технічні дослідження” – Івано-Франківськ, Україна, 2024, с. 236-238.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Номерний знак транспортного засобу. Wikipedia. URL:https://uk.wikipedia.org/wiki/Номерний_знак_транспортного_засобу (дата звернення: січень 2024р.).

2. Класифікація транспортних засобів. Wikipedia. URL:https://uk.wikipedia.org/wikiТранспортний_засіб (дата звернення: січень 2024р.).

3. Номерні знаки України. Wikipedia. URL:https://uk.wikipedia.org/wiki/Номерні_знаки_України. (дата звернення: січень 2024р.).

4. Автомобільні номерні знаки України. Розшифровка по областях. Dexpens. URL:https://www.dexpens.com/Article/10702/avtomobilni-nomerni-znaki-ukrayini-rozshifrovka-po-oblastyakh. (дата звернення: лютий 2024р.).

5. Тетяна Власюк. Автомобільні номери в Україні: коди та серії. URL:https://www.ukrstrahovanie.com.ua/uk/article-ua/avtomobilni-nomeri-v-ukraini. (дата звернення: лютий 2024р.).

6. Ангеліна Гройсман. Номери автомобілів за областями. URL:https://meta.ua/uk/news/society/88996-rozshifrovka-nomernih-znakiv-ukrayini-regioni-seriyi-kolori-avtonomeriv. (дата звернення: лютий 2024р.).

7. Автомобільні коди регіонів України: повний гід. Chas News. URL:https://chas.news/not/avtomobilni-kody-regioniv-ukraini-povnii-gid. (дата звернення: лютий 2024р.).

8. Secur.ua. Автоматична система розпізнавання номерів НОМЕРОК Lite 12. URL:https://secur.ua/videonablyudenie/programnoeobespechenie/nomerok/pz-dlia-rozpiznavannia-avtomobilnix-nomernix-znakiv-nomerok-lite-12. (дата звернення: лютий 2024р.).

9. Автоматична система розпізнавання автомобільних номерів Vector. URL:https://vector.org.ua/raspoznavanie-avtomobilnih-nomerov. (дата звернення: лютий 2024р.).

10. SecurOS® Auto License Plate Recognition (LPR/ANPR). URL:https://issivs.com/securos-auto. (дата звернення: лютий 2024р.).

11. Вовк С.М., Гнатушенко В.В., Бондаренко М.В. Методи обробки зображень та комп'ютерний зір. *Навчальний посібник.* 2016. №5. С. 144-148. URL:https://it.nmu.org.ua/ua/scientific_method_materials/MOZKZ.php.

12. Burger W., Burge M.J. Digital Image Processing: An Algorithmic Introduction. *Springer Nature.* 2022. P. 936-945. URL:https://books.google.com.ua/books?hl=uk&lr=&id=_kB9EAAAQBAJ.

13. Parker J.R. Algorithms for Image Processing and Computer Vision. *John Wiley & Sons*. 2020. P. 505-512. URL:https://books.google.com.ua/books?hl=uk&lr=&id=BK3oXzpxC44C.

14. Kurban R. Gaussian of Differences: A Simple and Efficient General Image Fusion Method. *Entropy.* 2023. 25(8). P. 1215-1234. URL:https://www.mdpi.com/1099-4300/25/8/1215.

15. Ramesh G., Logeshwaran J., Gowri J., Ajay M. The management and reduction of digital noise in video image processing. *Journal on Image and Video Processing.* 2019. Vol. 13(1). P. 2797-2801. URL:https://www.researchgate.net/publication/365559938_The_management_and_reduction_of_digital_noise_in_video_image_processing_by_using_transmission_based_noise_elimination_scheme.

16. Weibin R., Zhanjing L., Wei Z., Lining S. An improved Canny edge detection algorithm. *International Conference on Artificial Intelligence, Management Science and Electronic Commerce (AIMSEC)*. Tianjin, China, 2021. P. 69-72. URL:https://ieeexplore.ieee.org/abstract/document/6885761.

17. Куцак В.В., Паржин Ю.В., Галкин С.О. Дослідження алгоритмів скелетизації зображень рукописних символів для їх розпізнавання нейронними мережами. *Теоретичні та практичні дослідження молодих вчених*, 2020. С. 1-2. URL:https://repository.kpi.kharkov.ua/server/api/core/bitstreams/5dc84ea4-43aa-4ff7-aec4-f2d2be83008b/content.

18. Vlada M., Babiy I., Ivanescu O. ABBYY recognition technologies – ideal alternative to manual data entry. *Encyclopedia of Information Science and Technology*. Third Edition. 2015. P. 21-22.

19. Ahmad P., Ahmadreza B., Mehdi A., Hamid R.A., Zeyun Y., Peissig P. OCR as a Service: An Experimental Evaluation of Google Docs OCR, Tesseract, ABBYY FineReader, and Transym. *Lecture Notes in Computer Science.*Vol. 10072. 2016. P. 735-746. URL:https://link.springer.com/chapter/10.1007/978-3-319-50835-1_66.

20. Zewen L., Fan L., Wenjie Y., Shouheng P., Jun Z. A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects. *IEEE Transactions on Neural Networks and Learning Systems.* Vol. 33, Issue:12. 2022. P. 699-702. URL:https://ieeexplore.ieee.org/abstract/document/9451544.

21. Jiuxiang G., Kuen J., Lianyang M., Bing S., Xingxing W., Jianfei C., Recent advances in convolutional neural networks. *Pattern Recognition*, Vol. 77. 2018. P. 354-377. URL:https://www.sciencedirect.com/science/article/abs/pii/S0031320317304120.

22. Krizhevsky A., Sutskever I., Hinton G.E. ImageNet Classification with Deep Convolutional Neural Networks. *Advances in Neural Information Processing Systems*. Vol. 25, 2012. P. 33-36.

23. Ciresan D., Meier U., Masci J., Luca M., Gambardella L.M, Schmidhuber J. Flexible, High Performance Convolutional Neural Networks for Image Classification. *International Joint Conference on Artificial Intelligence*. Galleria, Manno-Lugano, Switzerland. 2024, P. 1237-1241. URL:https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=5a47ba057a858f8c024d2518cc3731fc7eb40de1.

24. Leiyu C., Shaobo L., Qiang B., Jing Y., Sanlong J., Yanming M. Review of Image Classification Algorithms Based on Convolutional Neural Networks. *Conference of Mechanical Engineering*. Guizhou University, Guiyang, China. 2021. Vol. 13(22), 4712. P. 1011-1018. URL:https://www.mdpi.com/2072-4292/13/22/4712.

25. Namatēvs I. Deep Convolutional Neural Networks: Structure, Feature Extraction and Training. *Information Technology and Management Science*. Vol. 20, No 1. 2017. P. 40-47. URL:https://itms-journals.rtu.lv/article/view/itms-2017-0007.

26. Brahmbhatt S. Advanced Computer Vision Problems and Coding Them in OpenCV. *Practical OpenCV, Technology in action series.* Apress, 2013. P. 140-144. URL:https://books.google.com.ua/books?hl=uk&lr=&id=_5sQAwAAQBAJ.

27. Kaehler A., Bradski G. Learning OpenCV 3: Computer Vision in C++ with the OpenCV Library. *O'Reilly Media, Inc*. 2016. P. 1020-1024. URL:https://books.google.com.ua/books?hl=uk&lr=&id=LPm3DQAAQBAJ.

28. Guobo X., Wen L. Image Edge Detection Based On Opencv. *International Journal of Electronics and Electrical Engineering*. 2023. Vol. 1. No. 2. P. 104-106. URL: https://www.ijeee.net/uploadfile/2013/0702/20130702104409134.pdf.

29. R. Smith. An Overview of the Tesseract OCR Engine. *Ninth International Conference on Document Analysis and Recognition.* Curitiba, Brazil. 2017. P. 23-26. URL:https://ieeexplore.ieee.org/abstract/document/4376991.

30. Revathi A.S., Nishi A.M. Comparative Analysis of Text Extraction from Color Images using Tesseract and OpenCV. *9th International Conference on Document Analysis and Recognition.* New Delhi, India. 2017. P. 90-117. URL:https://ieeexplore.ieee.org/abstract/document/9441128.

31. Agung Y.S., Kendricko A., Tanuwijaya K., Suryaningrum K.M. Extracting Information from Vehicle Registration Plate using OCR Tesseract. *Procedia Computer Science*. Volume 227. 2023. P. 932-938. URL:https://www.sciencedirect.com/science/article/pii/S1877050923017672.

32. Yangqing J., Shelhamer E., Donahue J., Long J., Girshick R., Guadarrama S., Darrell T. Caffe: Convolutional Architecture for Fast Feature Embedding. *ACM international conference on Multimedia.* Orlando, Florida, USA. 2024. P. 675-678. URL:https://dl.acm.org/doi/abs/10.1145/2647868.2654889.

33. Ammar A.A., Hamidouche K., Hashmi J.M., Dhabaleswar K.P. S-Caffe: Co-designing MPI Runtimes and Caffe for Scalable Deep Learning on Modern GPU Clusters. P*roceedings of the 22nd ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*. 2017. P. 193-205. URL:https://dl.acm.org/doi/abs/10.1145/3018743.3018769.

34. Jaspin K., Ajitha E., Dafni J.R., Sherin K. Intelligent Traffic Light Control System using Caffe Model: A Deep Learning Strategy. *Winter Summit on Smart Computing and Networks (WiSSCoN)*. 2023. P. 15-17. URL:https://ieeexplore.ieee.org/abstract/document/10133847.

35. Chia-Chi T., Jiun-In G. IVS-Caffe: Hardware-Oriented Neural Network Model Development. *Transactions on Neural Networks and Learning Systems*. 2023. Vol. 33. P. 60-68. URL:https://ieeexplore.ieee.org/abstract/document/9496282.

36. Sarbjit K. An Automatic Number Plate Recognition System under Image Processing. *Intelligent Systems and Applications*. 2016, Vol. 3. P. 14-25. URL:https://www.mecs-press.org/ijisa/ijisa-v8-n3/IJISA-V8-N3-2.pdf.

37. Sarbjit K., Sukhvir K. An Efficient Approach for Number Plate Extraction from Vehicles Image under Image Processing. *International Journal of Computer Science and Information Technologies*. 2014. Vol. 5 (3). P. 2954-2959.

38. Coetzee C., Botha C., Weber D. PC based number plate recognition system. *IEEE International Conference on Vehicular Electronics and Safety*. 2019. P. 359-365. URL:https://ieeexplore.ieee.org/abstract/document/711680.

39. Sutar G.T., Shah A.V. Number Plate Recognition Using an Improved Segmentation. *International Journal of Innovative Research in Science, Engineering and Technology*. 2014. Vol. 3. P. 12360-12368. URL:https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=d2f7c995362aafc36c94bdce44c6a6ac24bd8973.

40. Nayak V., Sushma P.H., AkshayaKumar K.M., Gururaj C. Automatic number plate recognition. *International Journal of Advanced Trends in Computer Science and Engineering*. 2020, P. 3783-3787.

41. Матвієнко С. Згорткові нейронні мережі. *Інформаційний ресурс про електроніку та програмуванняю* 2022. С. 456-465. URL: https://itmaster.biz.ua/programming/vision/cnns1.html.

42. Saravanan C. Color Image to Grayscale Image Conversion. *International Conference on Adaptive Science & Technology (ICAST)*. 2019. Vol. . P. 196-199. URL:https://ieeexplore.ieee.org/abstract/document/5445596.

43. Ahn H., Lee Y.H. Vehicle Classification and Tracking Based on Deep Learning. *Journal of Web Engineering*. 2020. Vol. 1. P. 1283-1294. URL:https://journals.riverpublishers.com/index.php/JWE/article/view/6969.

44. Журавель І.М. Метод бінаризації металографічних зображень з оптимальним порогом. *Міжнародна конференція фізико-механічних наук.* Інститут ім. Г.В. Карпенка, м. Львів, 2022. С. 142-147.

45. Sauvola J., Pietikäinen M. Adaptive document image binarization. *Pattern Recognition - Journals & Books*. Vol. 33, Issue 2. 2020. P. 225-236. URL:https://www.sciencedirect.com/science/article/abs/pii/S0031320399000552.

46. Gatos B., Pratikakis I., Perantonis S.J. Adaptive document image binarization. *Pattern Recognition - Journals & Books*. Vol. 34, Issue 1. 2021. P. 323-326. URL:https://www.sciencedirect.com/science/article/abs/pii/S0031320305003821.

47. Xiaohong W., Rongchun Z. A new method for image normalization. *Pattern Recognition - Journals & Books*. Vol. 33, Issue 2. 2020. P. 175-186. URL:https://ieeexplore.ieee.org/abstract/document/925407.

48. Парамуд Я.С., Яркун В.І. Метод розпізнавання символів на зображенні на основі згорткової нейронної мережі. *Computer Systems and Networks.* No 881, Lviv, Ukraine. 2018. P. 96-105.

49. Дідич О.Д., Мартинюк Т.Б., Очкуров М.А. Розпізнавання символів текстових документів із використанням нейронної мережі. *Computer Systems and Networks.* No 881, Lviv, Ukraine. 2018. 52-55.

50. Ptucha R., Felipe P.S., Brockler F., Singh V., Hutkowski P. Intelligent character recognition using fully convolutional neural networks. *Pattern Recognition - Journals & Books*. Vol. 88. 2019. P. 604-613. URL:https://www.sciencedirect.com/science/article/abs/pii/S0031320318304370.

51. Tang, J. Automatic number plate recognition (ANPR) in smart cities: A systematic review on technological advancements and application cases. *Pattern Recognition - Journals & Books*. Vol. 88. 2019. P. 228-234.

52. Mohammad A.B., Suneetha M., Muqeet M.A. An Efficient Method for Vehicle theft and Parking rule Violators Detection using ANPR. *Pattern Recognition - Journals & Books*. Vol. 90. 2020. P. 550-555.

53. Yu, Y. Identifying traffic clusters in urban networks based on graph theory using license plate recognition data. *Statistical Mechanics and its Applications*. 2024. Vol. 591. P.747-750.

54. Vashishtha, H. Vehicle Owner Identification from Number Plate. *International Conference on Recent Trends in Computing*. Springer, Singapore. 2022. P. 131-138. URL:https://link.springer.com/chapter/10.1007/978-981-16-7118-0_12.

55. Jain, P., Arya, D., Singh, A.K. Vehicle License Plate Detection and their Count for Traffic Management. *International Journal of Progressive Research in Science and Engineering*. 2022. Vol. 3(05). P. 170-175.
URL:https://journal.ijprse.com/index.php/ijprse/article/view/583.

56. Kumar M. High-Security Registration Plate Detection using OpenCV and Python. *7th International Conference on Communication and Electronics Systems (ICCES)*. Coimbatore, India. 2022. P. 835-839.
URL:https://ieeexplore.ieee.org/document/9835830.

57. Amrutha J.M., Nandini S., Anjali T. Real-Time Litter Detection System for Moving Vehicles Using YOLO. *4th International Conference on Smart Systems and Inventive Technology (ICSSIT)*. Tirunelveli, India. 2022. P. 1311-1315.
URL:https://ieeexplore.ieee.org/document/9716512.

58. Bernabe T.C. License Plate and Owner Recognition of Over speeding Vehicles using LabVIEW. *ASU International Conference in Emerging Technologies for Sustainability and Intelligent Systems (ICETSIS)*. Manama, Bahrain. 2022. P. 247-251.
URL:https://ieeexplore.ieee.org/document/9888887.

59. Maheswari S. Open CV Based Gate Plaza Control Using Rasp Berry Pi. *Second International Conference on Artificial Intelligence and Smart Energy (ICAIS)*. Coimbatore, India. 2022. P. 708-713.
URL:https://ieeexplore.ieee.org/document/9743098.

60. Gadgil A. Employing AI for Development of a Smart Entry Log System. *Communication and Intelligent Systems*. Springer, Singapore. 2021. P. 139-156. URL:https://link.springer.com/chapter/10.1007/978-981-19-2130-8_12.

61. Khan S. U. IoT-Enabled Vehicle Speed Monitoring System. *Electronics*. Vol. 614. P. 1234-1239. URL:https://www.mdpi.com/2079-9292/11/4/614.

62. Suthir S. Conceptual approach on smart car parking system for industry internet of things assisted networks. *Measurement: Sensors*. 2023. Vol. 10. P. 40-47. URL:https://www.sciencedirect.com/science/article/pii/S2665917422001088.

63. Gaur P.K. Computer vision based vehicle tracking as a complementary and scalable approach to RFID tagging. *Computer Vision and Pattern Recognition*. 2022. №1. P. 80-85. URL:https://arxiv.org/abs/2209.05911.

64. Hossain S.N. Automatic License Plate Recognition System using Deep Neural Network. *International Conference on Big Data, IoT, and Machine Learning*. Springer, Singapore. 2022. Vol. 95. P. 91-102. URL:https://link.springer.com/chapter/10.1007/978-981-16-6636-0_8.

65. Sabóia C.M. Brazilian Mercosur License Plate Detection and Recognition Using Haar Cascade and Tesseract OCR on Synthetic Imagery. *International Conference on Intelligent Systems Design and Applications*. Springer, Cham. 2022. P. 849-858. URL:https://link.springer.com/chapter/10.1007/978-3-030-96308-8_79.

66. Al-Mheiri M., Kais O., Bonny T. Car Plate Recognition Using Machine Learning. *Advances in Science and Engineering Technology International Conferences IEEE*. Dubai, United Arab Emirates.2022. P. 101-106. URL:https://ieeexplore.ieee.org/document/9734830.

67. Li W., Pun C.M. A single-target license plate detection with attention. *In International Workshop on Advanced Imaging Technology*. SPIE. 2022. Vol. 12177. P. 396-401.

68. Zou, Y. License plate detection and recognition based on YOLOv3 and ILPRNET. *Signal, Image and Video Processing*. 2022. Vol. 16(2). P. 473-480. URL:https://link.springer.com/article/10.1007/s11760-021-01981-8.

69. Yu H. Research on license plate location and recognition in complex environment. *Journal of Real-Time Image Processing.*2022. Vol. 19. P. 823–837. URL:https://link.springer.com/article/10.1007/s11554-022-01225-z.

70. Kaur. P. Automatic license plate recognition system for vehicles using a CNN. *CMC-Computers, Materials & Continua*. 2022. Vol. 71(1). P. 35-50. URL:https://www.techscience.com/cmc/v71n1/45359.

71. Sharma V. License Plate Detection and Recognition Using OpenCV–Python. *Lecture Notes in Electrical Engineering.* 2022. Vol. 832. P. 251-261. URL:https://link.springer.com/chapter/10.1007/978-981-16-8248-3_20.

72. Kaur P., Kumar Y., Gupta S. Artificial Intelligence Techniques for the Recognition of Multi-Plate Multi vehicle Tracking Systems: A Systematic Review. *Archives of Computational Methods in Engineering*. 2022. Vol. 29. P. 4897–4914. URL:https://link.springer.com/article/10.1007/s11831-022-09753-4.

73. Kumari P. L. Automatic License Plate Detection using CNN and Convolutional Neural Network. *6th International Conference on Computing Methodologies and Communication (ICCMC).* Erode, India. 2022. P. 1634-1639. URL:https://ieeexplore.ieee.org/document/9753706.

74. Pattanaik A., Balabantaray, R.C. Licence Plate Recognition System for Intelligence Transportation Using BR-CNN. *In Advances in Data Computing, Communication and Security.* Springer, Singapore. 2022. P. 659-668. URL:https://link.springer.com/chapter/10.1007/978-981-16-8403-6_60.

75. Yakovlev A., Lisovychenko O. Automated License Plate Recognition Process Enhancement with Convolutional Neural Network Based Detection System to Improve the Accuracy and Reliability of Vehicle Recognition. *In International Conference on Computer Science, Engineering and Education Applications*. Springer, Cham. 2022. P. 248-259. URL:https://link.springer.com/chapter/10.1007/978-3-031-04812-8_21.

76. Maheswari V.U., Aluvalu R., Mudrakola S. An Integrated Number Plate Recognition System using Threshold based methods and CNN. *International Conference on Decision Aid Sciences and Applications (DASA).* Coimbatore, India. 2021. P. 493-497. URL:https://doi.org/10.1109/DASA54658.2022.%209765218.

77. Srividhya S.R. A Machine Learning Algorithm to Automate Vehicle Classification and License Plate Detection. *Wireless Communications and Mobile Computing*. 2022. №1. P. 6323–6338. URL:https://www.hindawi.com/journals/wcmc/2022/9273233/.

78. Li X., Wen Z., Hua Q. Vehicle License Plate Recognition Using Shufflenetv2. *Dilated Convolution for Intelligent Transportation Applications in Urban Internet of Things.* 2022. №1. P. 2830–2842. URL:https://www.hindawi.com/journals/wcmc/2022/3627246/.

79. Valdeos M. Methodology for an automatic license plate recognition system using Convolutional Neural Networks. *IEEE Latin America Transactions*. 2022. Vol. 20(6). P. 1032-1039. URL:https://doi.org/10.1109/TLA.%202022.9757747.

80. Sunil L.B., Dubal A., Pallavi S.B. Otsu's Method For Image Thresholding. *International Journal of Applied Engineering Research*. ISSN 0973-4562. Vol. 10, №9. 2015. P. 21777-21783.

81. Zhang M., Yu W., Su J., Li W. Design of license plate recognition system based on machine learning. *4th International Conference on Image, Vision and Computing (ICIVC).* Xiamen, China. 2019. P. 518–522.

82. Смоленюк Н.Р. Метод та система ідентифікації номерних знаків автомобіля. *VI міжнародна науково-практична конференції "Прикладні науково-технічні дослідження"*. Івано-Франківськ, Україна. 2024. С. 236-238.

ДОДАТОК А

(обов’язковий)

КОПІЯ ТЕЗ ДОПОВІДІ НА МІЖНАРОДНІЙ КОНФЕРЕНЦІЇ

1.) Смоленюк Н.Р. Метод та система ідентифікації номерних знаків автомобіля. *Матеріали VI міжнародної науково-практичної конференції “Прикладні науково-технічні дослідження”.* Івано-Франківськ, Україна. 2024. С. 236-238..

Метод та система ідентифікації номерних знаків автомобіля

Смоленюк Н.Р.

Хмельницький національний університет, м. Хмельницький, Україна

I. Вступ

Завдання розпізнавання автомобільних номерних знаків користується попитом у програмному забезпеченні для контролю в'їзду та виїзду транспортних засобів з території підприємств, паркувань, контролю потоку автотранспорту. Для даної задачі необхідне програмне забезпечення, що може бути розміщене в автосервісах, контрольно-пропускних пунктах, пунктах контролю швидкості [1].

Центральним у цьому дослідженні є спроба вдосконалити технологію, що лежить в основі розпізнавання автомобільних номерних знаків, спираючись на обробку зображень і алгоритми розпізнавання символів. Прийнята методологія передбачає формулювання алгоритму, розробленого для вирішення проблем, пов’язаних із умовами поганої видимості, такими як похиле розміщення номерів або затемнені межі табличок. Розробка цього алгоритму доповнюється створенням програмного забезпечення, готового використовувати його можливості.

II. Аналіз останніх досліджень і публікацій

У роботі [2] описана структура глибокого навчання, що базується на дослідженні згорткових нейронних мереж (CNN), які є ключовими в задачах розпізнавання зображень. Дослідження заглиблюється в складну роботу caffe, сприяючи прийому даних, навчанню моделі та висновку. Крім того, розроблено оптимізовану конфігурацію caffe, адаптовану до конкретних вимог завдання, що підвищує її ефективність і доступність. Ця оптимізована установка використовує економічно ефективні, але надійні компоненти, забезпечуючи практичність без шкоди для продуктивності.

Також у статті [2] корисність CNN у поєднанні з caffe досліджується у сфері класифікації зображень, особливо зосереджуючись на ідентифікації об’єктів. Дослідження ретельно оцінює переваги цієї комбінації, підкреслюючи її досконалість у розрізненні складних візуальних візерунків.

Основними функціональними блоками CNN є згорткові шари. Згортковий шар складається з набору навчальних фільтрів (ядер), які рухаються по вхідному зображенню, виконуючи поелементне множення та підсумовуючи результати для створення карт функцій.

У статті [3] описані згорткові нейронні мережі. У першій частині автор пояснює, чому згорткові нейронні мережі краще підходять для обробки зображень, ніж стандартні нейронні мережі. Згорткові нейронні мережі використовують фільтри для вилучення ознак із зображень. Операція фільтрації полягає в накладанні фільтра на зображення, множенні відповідних елементів та підсумовуванні добутків. Це зменшує розмір зображення, але захоплює важливі деталі.

На основі дослідження та вивчення статті [1] та у результаті проведеного аналізу було вирішено реалізувати розпізнавання символів за допомогою Caffe: для розпізнавання символів використовуватиметься Caffe, структура глибокого навчання. Caffe підтримує різні типи машинного навчання, які націлені в основному на вирішення задач сегментації та класифікації зображень. Caffe забезпечує згорткові нейронні мережі, довгу короткострокову пам'ять та пов'язані нейронні мережі. Зокрема, буде розгорнуто навчальну модель нейронної мережі, здатну розпізнавати символи з сегментованих зображень.

III. Виклад основного матеріалу

Для досягнення поставленої мети виконано наступні завдання – досліджено методи розпізнавання тексту на зображеннях, розроблено алгоритм для розпізнавання номерних знаків автотранспортних засобів за умов слабкої видимості, розроблено програмне забезпечення, що використовує цей алгоритм.

Алгоритм процесу розв'язання задачі розпізнавання автомобільних номерних знаків у загальному вигляді може бути представлений простою блок-схемою з послідовністю кроків, показаних на рисунку 1.

Рисунок 1 - Алгоритм розпізнавання номерних знаків.

На рисунку 2 зображено діаграму компонентів програмного забезпечення.

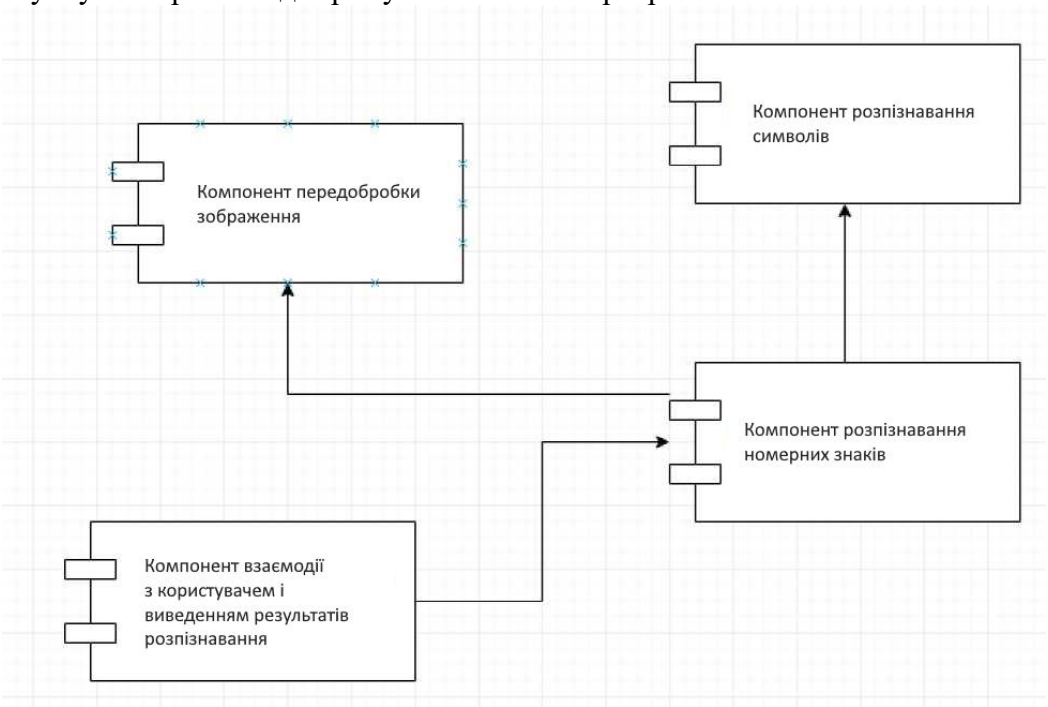


Рисунок 2 - Діаграма компонентів програмного забезпечення.

Розроблене програмне забезпечення містить чотири компоненти - компонент взаємодії з користувачем та виведення результатів розпізнавання, компонент розпізнавання номерних знаків, компонент попередньої обробки зображення, компонент розпізнавання символів.

Процес проектування архітектури програмного забезпечення полягає в проектуванні структури всіх його компонентів, функціонально пов'язаних з розв'язуваною задачею, включаючи поєднання між ними і вимоги до них.

Центральним у проектуванні архітектури програмного забезпечення є принцип модульності, який виступає за декомпозицію складних систем на керовані цілісні модулі. Розбиваючи систему на окремі компоненти, кожен з яких відповідає за певний набір функціональних можливостей, дизайнери сприяють багаторазовому використанню, зручності обслуговування та масштабованості. Цей модульний підхід не тільки спрощує процес розробки, але й підвищує гнучкість і розширюваність системи програмного забезпечення, забезпечуючи плавну адаптацію до мінливих вимог і технологічних ландшафтів.

На рисунку 3 зображено результат роботи програмного забезпечення.

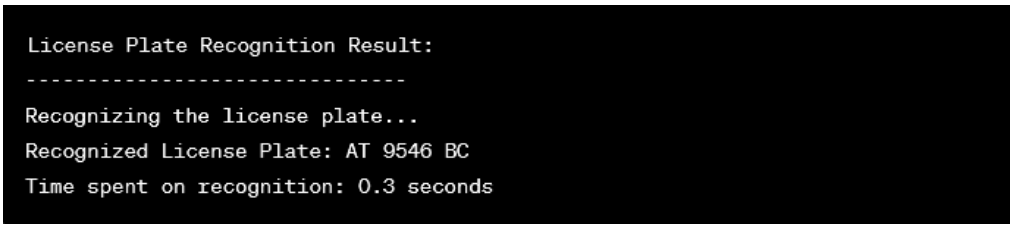


Рисунок 3 - Результат розпізнавання номерної пластини.

В результаті розроблено програмне забезпечення для розпізнавання номерних знаків, яке розпізнає автомобільні номерні знаки з точністю до 90%. У середньому розпізнавання відбувається за 0,3 секунди.

Розроблене програмне забезпечення з доопрацьованим алгоритмом показало кращий результат, ніж із початковим алгоритмом.

IV. Висновки

Після теоретичного аналізу було розроблено алгоритм розпізнавання автомобільних номерів мовою програмування C++ з використанням бібліотеки OpenCV для попередньої обробки зображення та бібліотеки Caffe для розпізнавання символів. Оскільки алгоритм мав низку недоліків, було вирішено його удосконалити. Удосконалення алгоритму базувалося на результатах пошуку номерної пластини на зображенні автомобіля, пошуку меж автомобільного номера, трансформацію зображення і розпізнавання за допомогою згорткової нейронної мережі.

Література

[1]. Yangqing Jia, Evan Shelhamer, Jeff Donahue, Jonathan Long, Ross Girshick, Sergio Guadarrama, Trevor Darrell. Caffe: Convolutional Architecture for Fast Feature Embedding. [Електронний ресурс] URL: https://dl.acm.org/doi/abs/10.1145/2647868.2654889 (дата звернення: квітень 2024р.)

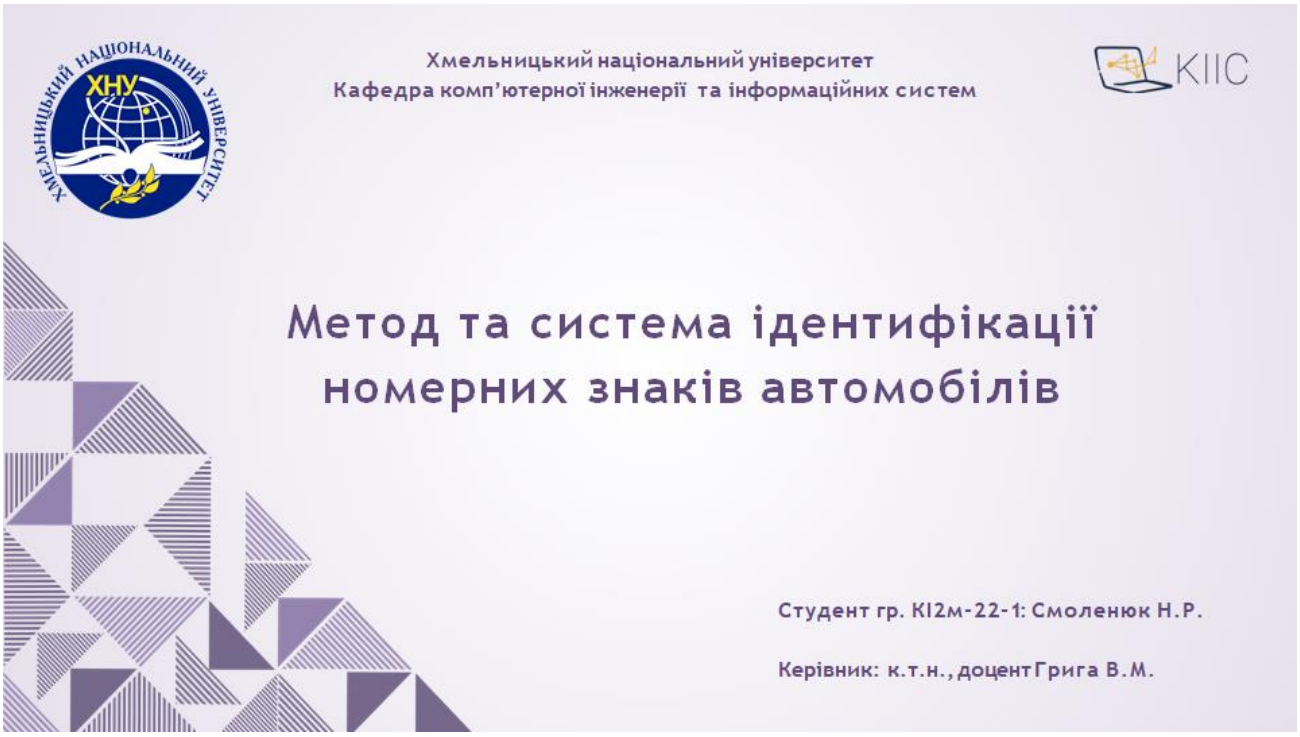
[2]. Ivars Namatēvs. Deep Convolutional Neural Networks: Structure, Feature Extraction and Training. [Електронний ресурс] URL: https://itms-journals.rtu.lv/article/view/itms-2017-0007 (дата звернення: квітень 2024р.)

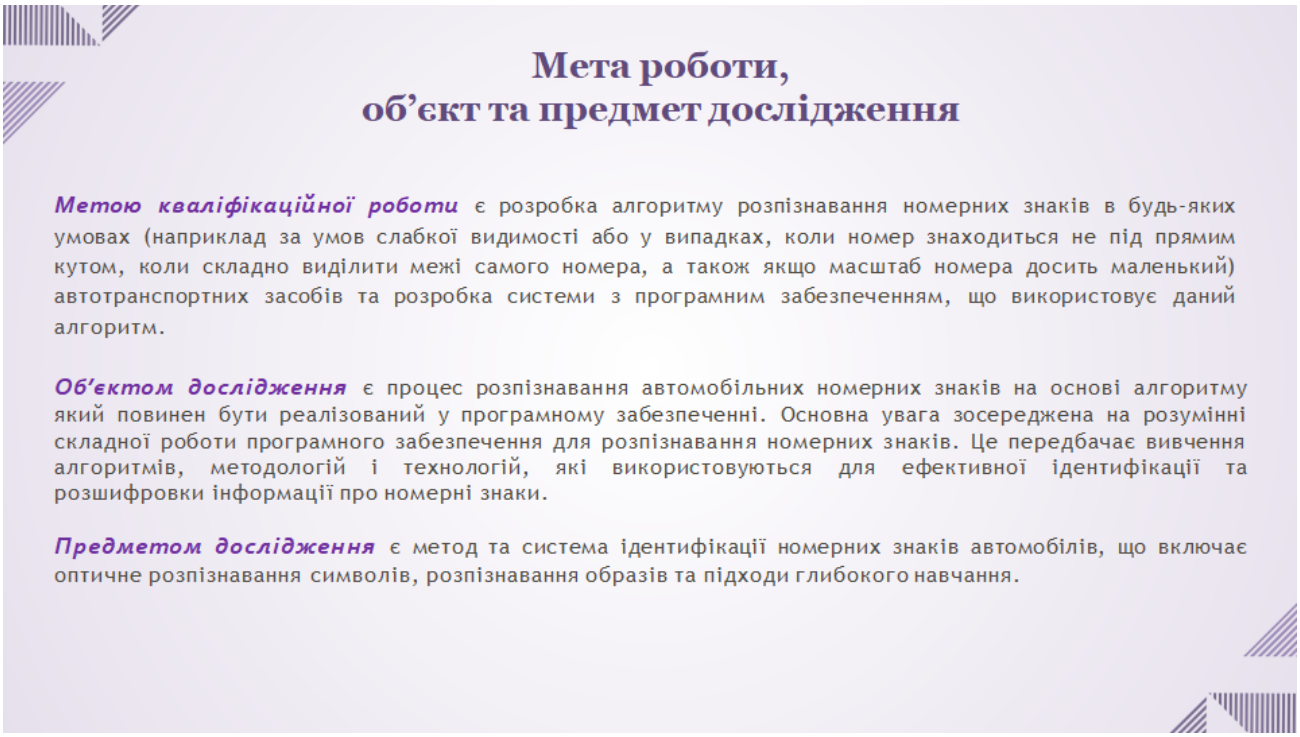
[3]. Сергій Матвієнко. Згорткові нейронні мережі. [Електронний ресурс] URL: https://itmaster.biz.ua/programming/vision/cnns1.html (дата звернення: квітень 2024р.)

ДОДАТОК Б

(обов'язковий)

КОПІЯ ПРЕЗЕНТАЦІЇ ДО ЗАХИСТУ КВАЛІФІКАЦІЙНОЇ РОБОТИ





Наукова новизна роботи

Наукова новизна отриманих результатів:

– набув подальшого розвитку метод розпізнавання номерних знаків на основі згорткових нейронних мереж з використанням бібліотеки Caffe, який відрізняється від інших своєю високою ефективністю та універсальністю, що дозволяє успішно розпізнавати автомобільні номерні знаки з точністю приблизно 85%;

– було удосконалено мікроархітектуру програмного забезпечення системи ідентифікації номерних знаків, яка на відміну від інших, дозволяє успішно розпізнавати автомобільні номерні знаки за короткий час до 0,3 секунди та із більш високою точністю, яка становить приблизно 90%.

Практична значущість отриманих результатів

Практична значущість отриманих результатів: розроблений метод та система мають безпосереднє практичне застосування в різних областях, включаючи правоохоронні органи, управління дорожнім рухом, збір плати та управління паркуванням.

Правоохоронні органи можуть використовувати цю систему для автоматичного виявлення транспортних засобів, причетних до злочинних дій або порушень правил дорожнього руху, для підвищення громадської безпеки. В управлінні дорожнім рухом система полегшить моніторинг руху транспортних засобів у режимі реального часу, дозволяючи оптимізувати дорожній потік, виявляти затори та ефективно контролювати дотримання правил дорожнього руху.

Системи збору плати за проїзд матимуть перевагу від здатності системи швидко ідентифікувати транспортні засоби, забезпечуючи безперебійні та ефективні процеси оплати, а засоби керування паркуванням можуть використовувати систему для моніторингу паркувальних місць, виявлення несанкціонованих транспортних засобів і оптимізації паркувальних операцій у міських районах. Загалом практичне значення отриманих результатів виходить за межі академічних досліджень, впливаючи на різні сектори та сприяючи розвитку суспільства.

Публікації по кваліфікаційній роботі

Публікації. За темою кваліфікаційної роботи опубліковані тези у VI Міжнародній науково-практичній конференції «Прикладні науково-технічні дослідження»:

1) Смоленюк Н.Р. Метод та система ідентифікації номерних знаків автомобіля. *Матеріали VI міжнародної науково-практичної конференція "Прикладні науково-технічні дослідження"*. Івано-Франківськ, Україна. 2024. с. 236-238.

Актуальність роботи, постановка задачі дослідження

Завдання розпізнавання автомобільних номерних знаків користується попитом у програмному забезпеченні для контролю в'їзду та виїзду транспортних засобів з території підприємств, паркувань, контролю потоку автотранспорту. Система може бути розміщене в автосервісах, контрольно-пропускних пунктах, пунктах контролю швидкості.
Задача роботи стоїть у розробці та реалізації алгоритму розпізнавання автомобільних номерів. В основному розпізнавання відбувається у три етапи: передобробка зображення, сегментація та безпосередньо розпізнавання символів.

Отже, для ***досягнення поставленої мети*** необхідно:

- ознайомитись з методами розпізнавання тексту на зображеннях;
- розробити алгоритм для розпізнавання номерних знаків автотранспортних засобів;
- розробити програмне забезпечення, що використовує цей алгоритм.

Загальне рішення задачі розпізнавання номерних знаків

Процес розв'язання задачі розпізнавання автомобільних номерних знаків у загальному вигляді може бути представлений блок-схемою з простою послідовністю кроків.

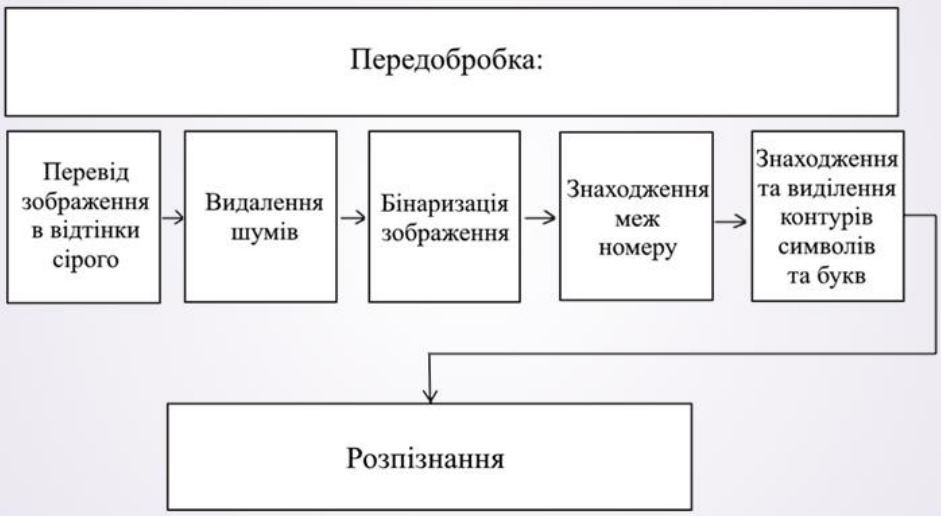
Алгоритм вирішення задачі

Пропонований алгоритм розпізнавання номерних знаків подається у вигляді послідовності наступних кроків:

Крок 1. Переведення зображення автомобільного номера в градації сірого.
Крок 2. Видалення шумів.
Крок 3. Бінаризація зображення.
Крок 4. Знаходження меж автомобільного номера.
Крок 5. Знаходження та малювання контурів символів та літер.
Крок 6. Сегментація номерної пластини
Крок 7. Розпізнавання символів та літер.

Схема запропонованого алгоритму розпізнавання показана на наступному слайді.

Схема процесу розпізнавання



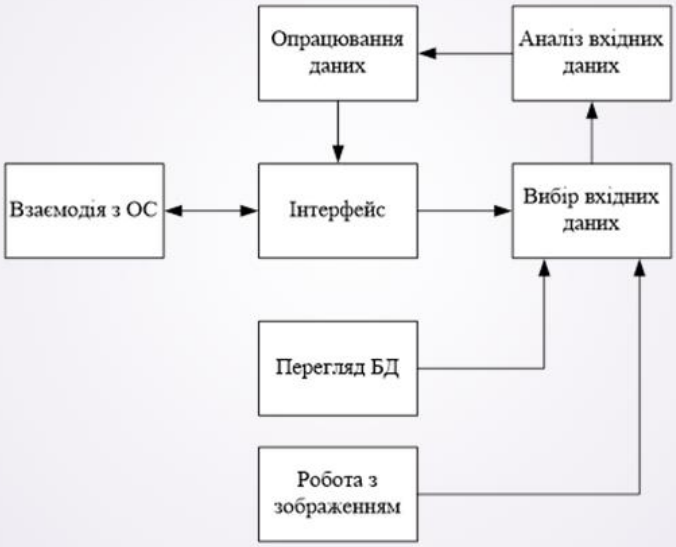
Проектування та реалізація системи розпізнавання номерних знаків

Для опису загального принципу роботи системи ідентифікації номерних знаків транспортних засобів, було розроблено структурну схему.

Принцип функціонування структурної схеми полягає в тому, що користувач після запуску програми обирає задачу, яку йому необхідно виконати: чи це буде робота з базою даних, чи це буде ідентифікація чергового транспортного засобу.

Після цього проводиться вибір вхідних даних (фільтри для пошуку по БД, чи зображення відповідно). Отримані результати виводяться на інтерфейс, де користувач зможе їх бачити, аналізувати та проводити подальші дії.

Структурна схема системи розпізнавання номерних знаків

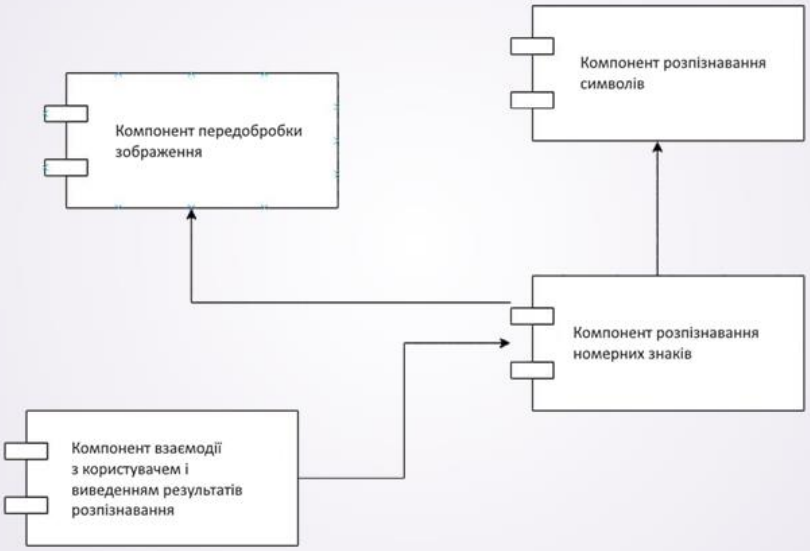


Компоненти програмного забезпечення

Розроблене програмне забезпечення містить чотири компоненти:

- компонент ***взаємодії з користувачем та виведення результатів*** розпізнавання (дозволяє під час запуску програмного забезпечення вибрати зображення номерної пластини, в якому необхідно розпізнати автомобільний номер та після обробки вивести результат розпізнавання);
- компонент ***розпізнавання номерних знаків*** (дозволяє розпізнати автомобільний номер на зображенні, шлях якого було вказано при запуску системи);
- компонент ***попередньої обробки зображення*** (переводить зображення в градації сірого, видаляє шуми, проводить бінаризацію та інші операції для попередньої обробки);
- компонент ***розпізнавання символів*** (дозволяє розпізнати символи номерного знаку, отримані шляхом сегментування).

Діаграма компонентів програмного забезпечення



Принцип роботи розробленого програмного заезпечення

Розроблене програмне забезпечення працює наступним чином:

Під час запуску програмного забезпечення на початковому етапі користувачеві пропонується вказати шлях до зображення, яке він бажає обробити.

Далі починається обробка та розпізнавання вказаного зображення, що виконує призначені алгоритми для аналізу та ідентифікації вмісту в зображенні.

По завершенню всіх операцій, проведених із зображенням, програмне забезпечення переходить до відображення результату в разі успішного розпізнавання. Результат демонструє ідентифіковані символи, виділені із зображення, а також час що був витрачений на розпізнавання.

Результат розпізнавання номерної пластини

Результат розпізнавання номерної пластини показаний у вигляді повідомлення у консолі:

```
License Plate Recognition Result:
------------------------------
Recognizing the license plate...
Recognized License Plate: AT 9546 BC
Time spent on recognition: 0.3 seconds
```

Для виявлення точності розпізнавання на вхід до програмного забезпечення було подано 100 зображень. Відсоток точності програмного забезпечення становив 85%. Після удосконалення алгоритму роботи точність зросла до 90 відсотків. Також змінився середній час, витрачений на розпізнавання з 0,5 до 0,3 секунд. У підсумку можна сказати, що розроблене програмне забезпечення з удосконаленим алгоритмом показало кращий результат, ніж з початковим алгоритмом.

Висновки

В даній роботі було здійснено комплексне дослідження методів розпізнавання автомобільних номерних знаків, яке охоплює теоретичний аналіз, огляд існуючих систем, розробку алгоритму та подальшу його модернізацію.

У ***першому розділі*** роботу було розпочато з ретельного теоретичного аналізу, було проведено систематичний огляд існуючих систем розпізнавання, що супроводжувалося ретельною оцінкою якості. Цей аналіз заклав основу для формулювання набору методів та вимог, які слугують керівними принципами для наступних етапів розробки.

У ***другому розділі*** був досліджений метод розпізнавання номерних знаків, що охоплює різні етапи, починаючи з попередньої обробки зображень до розпізнавання символів, демонструючи структурований підхід до ефективної ідентифікації цифро-літерних символів на номерних знаках.

У ***третьому розділі*** було розроблено алгоритм розпізнавання мовою програмування C++ з використанням бібліотеки OpenCV для попередньої обробки зображення та бібліотеки Tesseract для розпізнавання символів. Оскільки алгоритм мав низку недоліків, було вирішено його модернізувати.

У ***четвертому розділі*** описується реалізація програмного забезпечення, що включила пошук номерної пластини на зображенні автомобіля, пошук кордонів автомобільного номера, трансформацію зображення і розпізнавання за допомогою згорткової нейронної мережі.


Дякую за увагу!

UNICHECK
by Turnitin

Ім'я користувача:
Кафедра КІ

ID перевірки:
1016270954

Дата перевірки:
21.05.2024 19:46:13 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
21.05.2024 20:27:27 EEST

ID користувача:
100005591

Назва документа: Смоленюк_Метод та система ідентифікації номерних знаків автомобілів

Кількість сторінок: 100 **Кількість слів:** 19128 **Кількість символів:** 155949 **Розмір файлу:** 2.83 MB **ID файлу:** 1016060503

14.5%
Схожість

Найбільша схожість: 1.85% з Інтернет-джерелом (https://ela.kpi.ua/bitstream/123456789/51668/1/Kostiaiev_magistr.pdf)

14.2% Джерела з Інтернету	832	Сторінка 102
2.4% Джерела з Бібліотеки	90	Сторінка 113

0.2% Цитат

Цитати	3	Сторінка 114

Не знайдено жодних посилань

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи	23

Anti-Plagiarism v-15.257

Максимальне співпадіння з одним документом 2.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 12%**

ID: 126852 Назва: МКР Метод та система ідентифікації номерних знаків автомобілів Додано в БД: 2024-05-21 Автора: Смоленюк Н.Р. Керівники: Грига В.М. Консультанти: Опоненти:	Документ		Сумарний збіг по Бази Даних	
	Символи	Лексеми	Символи	Лексеми
	132175	1132	3802 (3%)	59 (5%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Дипломник: Смоленюк Назарій Русланович

Тема: Метод та система ідентифікації номерних знаків автомобілів

Спеціальність: 123 «Комп'ютерна інженерія»

Обсяг кваліфікаційної роботи:

Кількість сторінок записки 83.

1. Короткий зміст роботи та прийнятих рішень: Метою роботи є розробка алгоритму розпізнавання номерних знаків автотранспортних засобів та створення системи з програмним забезпеченням, що використовує даний алгоритм.

2. Висновок про відповідність роботи дипломному завданню: Кваліфікаційна робота магістра відповідає виданому завданню.

3. Характеристика виконання кожного розділу, ступінь використання останніх досягнень науки і техніки і передових методів роботи: У першому розділі було проведено систематичний огляд існуючих систем розпізнавання, що супроводжувалося ретельною оцінкою якості. У другому розділі був розроблений метод розпізнавання номерних знаків, який представляє комплексну структуру, що охоплює різні етапи, починаючи з попередньої обробки зображень до розпізнавання символів. У третьому розділі після теоретичного аналізу та дослідження методів було розроблено алгоритм розпізнавання автомобільних номерів мовою програмування C++ з використанням бібліотеки OpenCV для попередньої обробки зображення та бібліотеки Tesseract для розпізнавання символів. У четвертому розділі описується реалізація програмного забезпечення, що включила пошук номерної пластини на зображенні автомобіля, пошук кордонів автомобільного номера, трансформацію зображення і розпізнавання за допомогою згорткової нейронної мережі.

4. Позитивні сторони роботи: Отримано два пункти наукової новизни.

5. Негативні сторони роботи: Було допрацьовано чи удосконалено вже існуючі методи та рішення задачі, замість створення власних.

6. Оцінка графічного оформлення та пояснювальної записки роботи: Пояснювальна записка оформлена коректно, згідно діючих стандартів оформлення документації.

7. Відгук про роботу в цілому: В загальному робота була виконана на задовільному рівні.

8. Інші зауваження: Відсутні.

9. Оцінка дипломної роботи: Розглянувши позитивні та негативні сторони представленої кваліфікаційної роботи вважаю, що робота заслуговує оцінки «задовільно» (D).

Рецензент (прізвище, ім'я, по батькові, посада, місце роботи) Мартинюк Валерій Володимирович, зав. каф. АКІТтаР

"23" 05 2024 р. (підпис)

Завідувачу кафедри КІІС
д-р.техн.наук, проф. Говорущенко Т.О.

Смоленюк Назарій Русланович
ПІБ здобувача вищої освіти

ФІТ, 2 курс, групи КІ2м-22-1

ЗАЯВА

З правилами чинного Положення «Про систему забезпечення академічної доброчесності у Хмельницькому національному університеті» від 01.07.2022, згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування заходів дисциплінарної та академічної відповідальності, ознайомлений (а). Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на плагіат оповіщений(а) та надаю свою згоду на обробку та збереження університетом моєї роботи в інституційному репозитарії університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-технічних засобів (Unicheck та Anti-Plagiarism) та використання роботи для виявлення плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення плагіату в текстах робіт.

Робота для перевірки університетом надається в друкованому та електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

15 травня 2024 року

РІШЕННЯ ЕКСПЕРНОЇ КОМІСІЇ

КАФЕДРИ КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОМАЦІЙНИХ СИСТЕМ

ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: Метод та система ідентифікації номерних знаків автомобілів

Автор: Смоленюк Назарій Русланович

Спеціальність: 123 – Комп'ютерна інженерія

Освітня програма: освітньо-наукова

Науковий керівник: Грига В.М. к.т.н., доцент

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	відповідає
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) запозичення розміщені в розділах є збіг зі звітом з науково-дослідної практики автора Смоленюка Назарія Руслановича "Метод та система ідентифікації номерних знаків автомобілів", який було додано в репозитарій ХНУ 21 березня 2024 року;
2) усі запозичення фрагментарні, або мають належним чином оформленні посилання;
3) окремі виявлені збіги є загальновживаними фразами або виразами, про що свідчить посилання системи на збіг з джерелами на один фрагмент речення;
4) У якості запозичень у деяких місцях системою зафіксовано послідовності чотирирозрядних двійкових кодів та поєднання латинських символів українськими скороченнями та індексами у формулах. Такі модифікації не можуть бути розглянуті як переробка тексту, вони відносяться до використання спеціальних символів та форматування, яке є типовим для математичних та технічних виразів.

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ ідентичності/схожості Unicheck, складає 14.5% і адресується до 832 першоджерела, та системою Anti-Plagiarism складає 2% що, з урахуванням наведених обгрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи.

Керівник роботи _____ В.М. Грига

Гарант ОП _____ О.С. Савенко

Завідувач кафедри КІІС _____ Т.О. Говорущенко