Вінницький національний технічний університет
(повне найменування вищого навчального закладу)

Факультет інформаційних технологій та комп'ютерної інженерії
(повне найменування інституту, назва факультету (відділення))

Кафедра програмного забезпечення
(повна назва кафедри (предметної, циклової комісії))

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Концепція та засоби побудови програмно-навігаційної системи контролю руху транспортного засобу»

Виконав: студент 5-го курсу,
групи 2ПІ-20м
спеціальності 121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Прокопчук К.І.
(прізвище та ініціали)

Керівник: к.т.н., доц. каф. ПЗ
Рейда О.М.
(прізвище та ініціали)
«___»_____2021 р.

Опонент: д.т.н., професор
Василевський О. М.
(прізвище та ініціали)
«____»_____2021 р.

Допущено до захисту
Завідувач кафедри ПЗ
д.т.н., проф. Романюк О. Н.
(прізвище та ініціали)
«___»_____2021 р.

Вінниця ВНТУ - 2021 рік

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Рівень вищої освіти ІІ-й (магістерський)
Галузь знань 12 – Інформаційні технології

Спеціальність 121 – Інженерія програмного забезпечення

Освітньо-професійна програма – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ

Романюк О. Н.

« 13 » вересня 2021 р.

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Прокопчуку Кирилу Ігровичу

1. Тема роботи – концепція та засоби побудови програмно-навігаційної системи контролю руху транспортного засобу.

Керівник роботи: Рейда Олександр Миколайович, к.т.н., доцент кафедри ПЗ, затверджені наказом вищого навчального закладу від « 24 » вересня 2021 р. № 277.

2. Строк подання студентом роботи
1 грудня 2021 р.

3. Вихідні дані до роботи: базові методи розпізнавання – ме покращення зображення, адаптитивна бінаризація; фреймворк (бібліоте OpenCV, Boost; графічний режим – TrueColor, цифрові зображення; тип оброоки даних – зображення, відео;

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити): загальна характеристика роботи; обґрунтування вибору

методу розробки та постановка задачі дослідження; розробка нових методів та підходів до систем розпізнавання дорожніх знаків; розробка модулів для розпізнавання дорожніх знаків за допомогою технологій покращення зображень; тестування роботи модулю на основі вхідних даних; економічна частина; висновки; додатки.

5. Перелік графічного матеріалу: галузі застосування методів і засобів розпізнавання дорожніх знаків; методи розпізнавання дорожніх знаків за допомогою покращення зображення; програмні засоби для розпізнавання дорожніх знаків; висновки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Романюк О. Н., д.т.н., завідувач кафедри ПЗ		
5	Бурєннікова Н. В., д.е.н., проф. кафедри ЕПВМ		

7. Дата видачі завдання 14 вересня 2021 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Обґрунтування вибору методу розробки та постановка задачі дослідження	15.09.2021-26.09.2021	Вик.
2	Удосконалення методу і моделі розпізнавання дорожніх знаків	27.09.2021-15.10.2021	Вик.
3	Розробка модулю розпізнавання дорожніх знаків	16.10.2021-7.11.2021	Вик.
4	Тестування роботи алгоритмів та інтерфейсу модулю розпізнавання дорожніх знаків	8.11.2021-21.11.2021	Вик.
5.	Економічна частина	22.11.2021-30.11.2021	Вик.

Студент _____ **Прокопчук К.І.**
(підпис) (прізвище та ініціали)

Керівник магістерської кваліфікаційної роботи _____ **Рейда О.М.**
(підпис) (прізвище та ініціали)

Опонент магістерської кваліфікаційної роботи _____ **Василевський О.М**
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

УДК 681.326.3

Прокопчук К. І. Концепція та засоби побудови програмно-навігаційної системи контролю руху транспортного засобу. Магістерська кваліфікаційна робота зі спеціальності 121 – інженерія програмного забезпечення, освітня програма – інженерія програмного забезпечення. Вінниця: ВНТУ, 2021. 119 с.

На укр. мові. Бібліогр.: 38 назв; рис.: 28; табл.: 12.

У магістерській роботі розглянуто методи та засоби побудови програмно-навігаційних систем контролю руху транспортного засобу, розроблено програмний модуль для покращення якості і розпізнавання дорожніх знаків. Запропоновано метод покращення якості зображення за рахунок використання бінарізації з адаптивним порогом для кожного каналу кольорового зображення, що дає можливість підвищити ефективність розпізнавання. Подальшого розвитку отримав метод розпізнавання на основі корелювання шаблону знаку із зображенням. Розроблено програмні компоненти та систему розпізнавання дорожніх знаків. Проведено аналіз методів розробки систем розпізнавання дорожніх знаків. Виконано варіантний аналіз і обґрунтування вибору засобів реалізації і обгрунтування вибору бібліотек і компонентів.

В економічній частині розглянуто комерційний потенціал розробки, витрати на виконання науково-дослідної роботи та конструкторсько-технологічної роботи, комерційні ефекти від реалізації результатів розробки, ефективність вкладених інвестицій та період їхньої окупності.

Ключові слова: розпізнавання дорожніх знаків, цифрові зображення, обробка зображень.

ABSTRACT

Prokopchuk K. I. The concept and instruments for development of navigation system to control driving of the vehicle. Master's thesis in specialty 121 – software engineering, educational program - software engineering. Vinnytsia: VNTU, 2021. 119p.

In Ukrainian language. Bibliogr .: 38 titles; fig .: 28; table: 12.

In the master's thesis, the concepts and tools for building software and navigation systems for vehicle traffic control were developed, namely, a software module was developed to improve the quality and recognition of road signs. A method for improving image quality by using binaryization with an adaptive threshold for each color image channel is proposed, which makes it possible to increase the recognition efficiency. The method of recognition based on the correlation of the sign pattern with the image was further developed. Software components and a system of road sign recognition have been developed. The analysis of methods of development of systems of recognition of road signs is carried out. The variant analysis and substantiation of the choice of means of realization and the substantiation of the choice of libraries and components are executed. The economic part outlines the commercial potential of development, the cost of research and development work, commercial effects of the implementation of development results, the effectiveness of investments and their payback period.

Keywords: road sign recognition, digital images, image processing.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. Проблема безпечного руху транспорту актуальна у зв'язку із постійним збільшенням кількості транспортних засобів. Новітні технології під час руху допомагають зберегти тисячі життів та попередити сотні аварій. Система розпізнавання дорожніх знаків відноситься до технологій забезпечення безпечного руху автотранспорту. Такі системи визначають дорожні знаки для попередження водіїв або корегування руху автотранспорту. Такі системи використовують одні з найвідоміших автовиробників: TOYOTA, Volkswagen, Audi, Opel, BMW, Ford, Mercedes-Benz та багато інших.

Система розпізнавання дорожніх знаків Toyota Safety Sense використовує лазер та камеру для моніторингу дорожніх знаків, а саме: фіксовані обмеження щодо швидкості або обгону, можливості і напрямку руху. Система розпізнавання дорожніх знаків на автомобілях Opel входить до складу системи Opel Eye розпізнає дорожні знаки, має систему динамічного контролю дальності світла та вихід з займаної смуги дороги. Система Speed Limit Assist (MercedesBenz) виявляє неявні і явні знаки обмеження швидкості для інформування водія про максимально дозволену швидкість. Така система включає виявлення умовних обмежень швидкості за допомогою додаткового зчитування оптичних символів. Вирішення проблем безпеки на дорозі за допомогою систем розпізнавання дорожніх знаків є однією з ключових можливостей зниження стану аварійності в Україні та по всьому світу.

В загальному системи розпізнавання дорожніх знаків мають проблеми із вірогідністю визначення знаку у залежності зовнішніх факторів (недостатня освітленість об'єкту, несприятливі погодні умови), а також стану систем відслідковування і деформування або пошкодження знаків. Такі фактори значно зменшують якість і можуть приводити до хибного результату визначення.

Мета та завдання дослідження. Метою роботи є підвищення ефективності навігаційної системи контролю руху транспортного засобу за рахунок підвищення якості зображень дорожніх знаків.

Основними задачами дослідження є:

– провести аналіз існуючих методів розпізнавання дорожніх знаків;

– провести аналіз методів підвищення якості зображень дорожніх знаків;

– запропонувати нові:

  − методи підвищення якості зображень дорожніх знаків;

  − методи розпізнавання дорожніх знаків;

- розробити програмні компоненти та систему розпізнавання дорожніх знаків;
- провести експериментальні дослідження розроблених програмних компонентів розпізнавання дорожніх знаків.

Об’єкт дослідження – процес покращення якості дорожніх знаків для підвищення ефективності розпізнавання.

Предмет дослідження – методи та засоби покращення якості дорожніх знаків та їх розпізнівання.

Методи дослідження. У процесі досліджень використовувались:цифрові обрабоки зображень, математичне моделювання і прогнозування, комп’ютерне моделювання для аналізу та перевірки отриманих теоретичних положень, методи порівняння, методи проектування.

Наукова новизна отриманих результатів.

1. Запропоновано метод покращення якості зображення за рахунок використання бінарізації з адаптивним порогом для кожного каналу кольорового зображення, що дає можливість підвищити ефективність розпізнавання.
2. Подальшого розвитку отримав метод розпізнавання на основі корелювання шаблону знаку із зображенням.

Практична цінність отриманих результатів. Полягає в тому, що на основі отриманих в магістерській кваліфікаційній роботі теоретичних положень запропоновано методи та розроблено програмні засоби розпізнавання дорожніх засобів.

Впровадження.

Зв'язок роботи з науковими програмами, планами, темами. Робота виконувалася згідно плану виконання наукових досліджень на кафедрі програмного забезпечення.

Особистий внесок здобувача. Здобувачем проведено аналіз наукової літератури, розроблено метод розпізнавання дорожніх знаків, оптимізовано метод розпізнавання дорожніх знаків за допомогою бінарізації зображення, проведено розробку та тестування. Усі наукові результати, викладені у магістерській кваліфікаційній роботі, отримані автором особисто.

Апробація матеріалів магістерської кваліфікаційної роботи. Основні положення магістерської кваліфікаційної роботи доповідалися на міжнароднії та всеукраїнськії конференції: Міжнародна науково-практична конференція «Електронні інформаційні ресурси: створення, використання, доступ» (Вінниця, 2021).

Публікації. Основні результати досліджень опубліковано в 1 науковій праці, у тому числі 1 – у матеріалах конференцій.

Структура та обсяг роботи. Відповідно до мети і завдань дослідження структура роботи складається зі вступу, п'яти розділів, висновків та списку використаної літератури. За час роботи опрацьовано 38 літературних джерел. Зміст роботи викладено на 76 сторінках машинописного тексту.

1 ОБҐРУНТУВАННЯ ВИБОРУ МЕТОДУ РОЗРОБКИ ТА ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ

1.1 Опис методів розробки систем розпізнавання дорожніх знаків

Розробка системи розпізнавання знаків дорожнього руху є складним завданням. Існує багато факторів, що можуть стати причиною низької точності, неправильного аналізу чи не повної роботи системи розпізнавання знаків дорожнього руху.

Система визначення дорожніх знаків являє собою набір інструментів для аналізу дорожньої ситуації: камеру, що розташована у фронтальній частині транспортного засобу. Така конструкція дає можливість водіям розібратися в роботі системи. Пристрій знаходить та сканує і розпізнає дорожній знак, аналізує, а потім передає сигнал на екран, дисплей. Також розповсюджена технологія сповіщення за допомогою звукового сигналу. В більшості випадків такі системи розпізнають тільки знаки обмеження швидкості, але існують і такі, що подають сигнал про заборону обгону, контролюють рух в межах дорожньої розмітки. Технології розпізнавання дорожніх знаків постійно оновлюється. Технології Tesla дозволяють користуватися автопілотом, що повність може замінити водія на нескладному відрізку дороги. Точність таких систем цілком залежить від швидкості руху транспортного засобу, кліматичних умов, завантаження шляху, кількості автомобілів, що знаходяться на узбіччі і перешкод. Система розпізнавання знаків на даному етапі не може замінити водія і є додатковою опцією, що асистує на дорозі. Системи вищого класу, встановлені в преміум-автомобілях, показують найвищу швидкість певного типу дороги, мають розширені функції не тільки для асистування, але й управління транспортним засобом. Система не завжди просто пристосовується до погодних умов. Та нові технології допомагають уникнути труднощів, що виникають під

час поганих погодних умов, якщо система відчуває дощик, то відбувається відображення відповідного обмеження швидкості на дисплеї системи навігації. Наявність методу попередження про зміну швидкісного режиму руху дозволяє водієві уникнути неприємних ситуацій зі штрафами перевищення швидкості, їздити в комфорті та безпеці і бути більш обізнаним в дорозі.

Не всі дорожні знаки чи об’єкти можуть ідеально підходити для аналізу чи розпізнавання. Наприклад дорожній знак може бути пошкоджений чи зіпсуватися під супроводом років, а недостатня освітленість об’єкту – є одною з найбільш основних проблем в розробці систем розпізнавання об’єктів, в тому числі і системи розпізнавання елементів дорожнього руху. Для зручності та наглядності дорожні знаки мають різне кольорове забарвлення, що робить їх унікальними та допомагає класифікувати, для кращого вивчення та виділення їх ваги на дорозі. Більшість існуючих алгоритмів розпізнавання елементів дорожнього руху використовують інформацію про колір дорожнього знаку для ідентифікації. Інформація про колір об’єкту є дуже чутлива до умов освітлення та віку знаку, відомо, що за часом на сонці дорожні знаки мають властивість змінювати яскравість забарвлення(див. рисунок 1.1).

Рисунок 1.1 – Праворуч знак «Пішохідний перехід» приклад висвітленого знаку в міських умовах, що втратив яскраві кольори та чіткість.

На дорозі можуть бути розташовані поломані, деформовані або пошкоджені дорожні знаки, що не розпізнаються сучасними алгоритмами розпізнавання елементів дорожнього руху, оскільки в алгоритмах використовується інформація про форму та розмір об’єкту, що може призвести до зовсім неправильної класифікації або ж взагалі до того, що знак не буде розпізнано (див. рис. 1.2). Також бувають випадки коли знак частково закритий іншими об’єктами, наприклад – листям, сміттям або деревом.

Рисунок 1.2 – Приклад деформованого дорожнього знаку

За дощової або туманної погоди, руху у темну пору суток, наявності тіні від інших транспортних засобів є причиною спотворення зображення дорожнього знаку або розмітки. Такі природні явища як град, дощ, сніг, вітер та туман також є серйозними проблемами для сучасних алгоритмів розпізнавання елементів дорожнього руху. Багато існуючих методів пасивної безпеки автомобілів є не зовсім новими, а алгоритми, які в них використовуються –

неефективними. Існуючі системи пасивної безпеки є дуже дорогими, а їхні алгоритми запатентовані та у закритому доступі.

За останній час було запропоновано багато різних методів розпізнавання елементів дорожнього руху. Зазвичай вони складаються з двох послідовних процесів:

- процес розпізнавання зони, тобто області на картинці, де ймовірно може бути елемент дорожнього руху;
- процес ідентифікації елементу, тобто встановлення типу елементу, наприклад дорожній знак, а згодом, встановлення конкретно який це знак.

1.2 Порівняльний аналіз методів розпізнавання дорожніх знаків

Основною ціллю розпізнавання зони знаку – це виділення певних зон зображення, які найбільш схожі на дорожні знаки. Для вирішення цієї задачі можуть використовуватись різні підходи та методи, що використовують основні характеристики елементів дорожнього руху, такі як розмір, форма і колір. В загальному методи розпізнавання області інтересу можна розбити на групи:

- розпізнавання об’єкта за допомогою кольору;
- розпізнавання об’єкта за допомогою форми;
- розпізнавання об’єкта гібридними методами.

Використання інформації про колір об’єкту є одним із найпоширеніших підходів до розпізнавання елементів дорожнього руху. Ця інформація знаходиться за допомогою порогових або розширених методів сегментації і використовується для пошуку областей, де найбільш ймовірно може бути дорожній знак. Однак головним недоліком цього методу – є висока чутливість до нестабільних умов освітлення, тому цей метод є дуже залежним від часу доби та погодних умов. У сучасних системах цей метод використовується з різними типами кольорових просторів, найбільш поширеними є RGB, HSI/HSV, YUV, YCbCr, CIELab та CIECAM97.

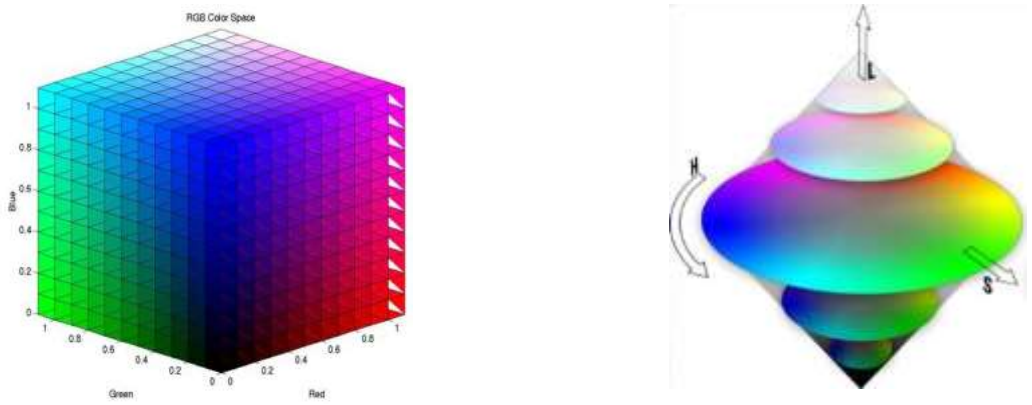
Методи розпізнавання об’єктів на основі колірних характеристик мають високу швидкодію, хорошу стійкість та інші характеристики, які можуть певною мірою поліпшити продуктивність розпізнавання об’єктів на картинці. В таких методах розпізнавання об’єктів зазвичай використовуються кольорові зображення високої роздільної здатності, але, зазвичай, такі методи дають низьку точність на чорно-білих зображеннях.

Для розпізнавання елементів, вхідні картинки розглядають як набір пікселів, в яких кожен піксель має певні колірні характеристики, які визначають його колір, потім набір пікселів ділять на окремі ділянки в яких сусідні пікселі мають схожі характеристики кольору. Після чого знаки дорожнього руху виявляють за допомогою базового кольору та алгоритму розпізнавання. В таких підходах велику роль відіграє вибір колірного простору, тобто рівняння, яке визначає характеристики кольору пікселя. Вхідні картинки системи розпізнавання дорожніх знаків зазвичай трансформують у вибраний колірний простір в якому складові елементи більш виразні. Існуючі алгоритми розпізнавання дорожніх знаків використовують наступні колірні простори:

- RGB (червоний, зелений, синій), (див. рис. 1.3)
- HSI (відтінок, насичення, інтенсивність), (див. рис. 1.3)
- HSV (відтінок, насичення, значення). (див. рис. 1.4).

Найбільш поширеними алгоритмами розпізнавання елементів дорожнього руху з використанням інформації про колір є:

- алгоритм динамічної агрегації кольору.
- алгоритм порогування кольору,
- алгоритм “region growing”,
- алгоритм індексування кольору.



а) б)

Рисунок 1.3 – Колірні системи:

а) – RGB; б) – HIS

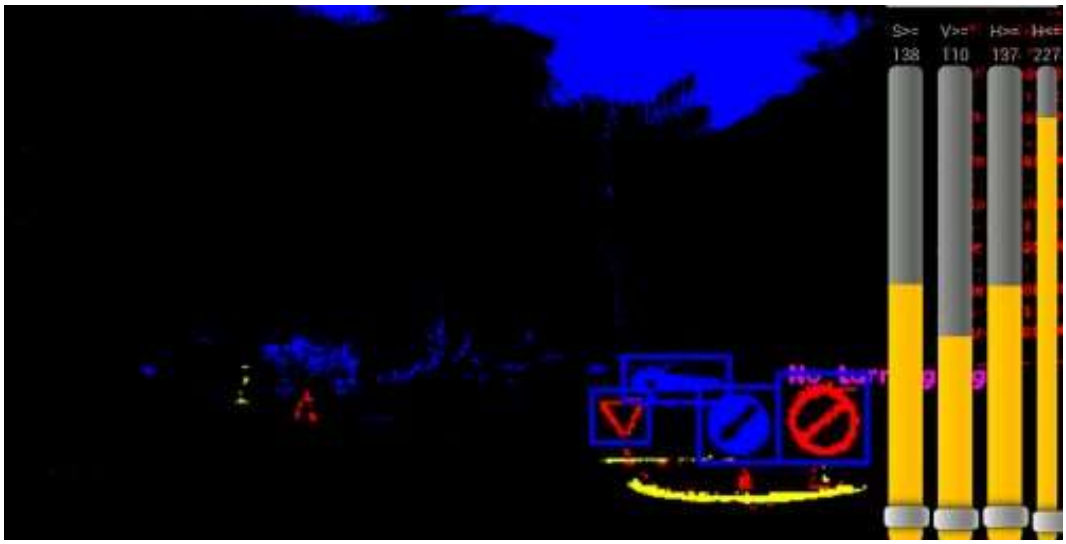


Рисунок 1.4 – Приклад режим налагодження з кольоровим фільтром HSV і регульованими порогами

Алгоритм порогування кольору є вже давно відомою технікою, яка використовується для сегментації зображення. Загальна ідея цієї техніки – це припущення, що суміжні пікселі, значення яких (рівень сірого кольору, значення кольору, текстури тощо) лежить у певному діапазоні, належать до одного класу. Кольорова сегментація була використана на власному наборі тестових даних, близько 2000 зображень, що дало точність розпізнавання лише 82%.

Ефективність даного підходу була вдосконалена в роботі [4] завдяки використанню сегментації кольору за якої слідувало посилення кольору.

Останнім часом, порогування кольору – є дуже поширеним алгоритмом для попередньої обробки зображення [5]. В роботі [6] використано перехід з простору кольорів RGB в простір HSI, в якості додаткового параметру для класифікації дорожніх знаків на білому фоні. В роботах [7, 8] колірні простори HSI/HSV були використані для виявлення дорожніх знаків. Основна перевага простору кольорів HSI над простору кольорів RGB – це те, що в просторі кольорів HSI є лиш два компонента, які задають колір – це відтінок і насичення, які дуже схожі на сприйняття людиною, і вони більш незалежні від умов освітлення, в порівнянні з RGB.

У роботі [7] перехід з простору кольору RGB в HSI використовується для ціле. У роботі [9] простір кольорів HSI був використаний для виявлення елементу, а згодом отримані характеристики використовували у для розпізнавання форми об’єкту, для виявлення форми для підвищення рівня точності. Середня точність склала приблизно 88,4%. Основне обмеження використання трансформації HSV – це сильна залежність відтінку від яскравості зображення. Відтінок – це лише фізична характеристика яскравості кольору, а не сприйнятої яскравості зображення. Таким чином, значення повністю насиченого жовтого та синього кольорів однакове.

«Region growing» – це ще один простий і популярний алгоритм, що використовується для розпізнавання елементів у системах розпізнавання дорожніх знаків (СРДЗ). Метод сегментації зображення на основі пікселів, який починається з вибору початкової точки або початкового пікселя. Потім область розширюються шляхом додавання сусідніх пікселів, які є рівномірними, за певним критерієм відповідності, збільшуючи крок за кроком розмір області. Цей метод використовували для (СРДЗ). Його ефективність була не дуже високою, приблизно 84%. Оскільки ефективність цього методу залежить від початкових

даних, а саме від вибору початкового пікселя. Проблеми можуть виникнути, коли початкові пікселі лежать на краях (якогось об’єкту), невизначеність навколо країв сусідніх областей може бути усунена неправильно.

Алгоритм індексації кольорів – це ще один простий метод, який ідентифікує об'єкти повністю на основі кольору. Він був розроблений Баллардом і Суеном [12] і був використаний дослідниками на початку 1990-х. У цьому методі порівняння будь-яких двох кольорових зображень проводиться шляхом порівняння їх гістограми кольорів. Для заданої пари гістограм I і M, кожна з яких містить n бітів, тоді перетини гістограми визначаються як (формула 1.1):

$$\sum_{j=1}^{n} \min(I_j, M_j) \tag{1.1}$$

Значення відповідності тоді:

$$H(I, M) = \frac{\sum_{j=1}^{n} \min(I_j, M_j)}{\sum_{j=1}^{n} M_j} . \tag{1.2}$$

Перевагою використання кольорових гістограм є їх надійність щодо геометричних змін проектованих об'єктів. Однак індексація кольорів залежить від сегментації. Ефективна та надійна сегментація не може бути виконана до розпізнавання. Таким чином, кольорова індексація характеризується як ненадійний метод.

Мюллер та ін. [14] розробили систему комп'ютерного зору для виявлення об'єктів у зовнішніх сценах. Пічолі та ін. [15] розробили систему виявлення дорожніх знаків шляхом аналізу кольору як апріорної інформації, щоб обмежити можливі місця розташування знаків на зображенні, застосовуючи техніку взаємної кореляції. Нововічова та ін. [16] використовував класифікатор ядра

Лапласа в дереві рішень для класифікації дорожніх знаків, незважаючи на деякі труднощі з розпізнаванням квадратних знаків у міських районах, де на зображенні існує багато горизонтальних і вертикальних ліній, а Юлле та ін. [17] розробили підхід для виявлення лише знаків зупинки шляхом коригування кольору зовнішнього освітлення, визначення меж знаків і відображення знака на передній частині.

Де Ла Ескалера та ін. [18] розглянули підбір кольорів, під час якого вони шукали залежності в певних відношеннях, які відповідають трикутним, прямокутним або круговим знакам. Лозьєр та ін. [19] використовував підхід, заснований на фізиці, для розпізнавання знаків, але цей підхід вимагав збереження в пам'яті змін у параметрі моделі для врахування природних змін освітлення. Методи відповідності шаблонів, які використовувалися, стикаються з деякими проблемами у випадку шуму в аналогових зображеннях або зображеннях у відтінках сірого, неправильного освітлення на зображеннях, шуму дискретизації та квантування та наявності тіней на зображеннях.

Іншим підходом до сегментації кольорів є – алгоритм динамічної агрегації пікселів. У цьому способі процес сегментації здійснюється шляхом введення динамічного порогу до процесу агрегації пікселів у кольоровому просторі HSV. Застосований поріг незалежний з точки зору лінійності і його значення визначається як:

$$a = k - \sin(S_{seed}), \qquad (1.3)$$

де, k – параметр нормалізації, а S_{seed} – насичення пікселів.

Основна перевага такого підходу – це зменшення нестабільності відтінку. Однак це не вирішує інші проблеми сегментації зображення, такі як згасання та освітлення. Цей метод був випробуваний на власній базі даних з 620

зображеннями, в результаті чого рівень точності був приблизно від 86,3 до 90,7%.

Таблиця 1.1. Порівняння технік розпізнавання елементів за кольором

Техніка	Метод сегментації	Переваги	Точність
Порогування кольору	RGB сегментація кольору	Простота	~81%
	RGB сегментація з посиленням кольору	Швидкодія та відносно висока точність розпізнавання	
HSI/HSV трансформація	HSI порогування як додаткова характеристика для білих знаків	Сегментує погано освітлені знаки	~88.4 %
	HSI кольорова сегментація	Простота та швидкодія	-
	Перехід від RBG до HSI простору кольорів	Простота та висока точність розпізнавання	-
Region growing	Розширення області від початкових пікселів в залежності від фактору подібності		~84%

Продовження таблиці 1.1

Індексування кольору	Порівняння двох зображень на основі гістограми кольорів	Швидкодія	-
Алгоритм динамічної агрегації кольору	Динамічне порогування кольору в HSV просторі кольорів	Зменшена нестабільність відтінку	~86.3% - 90.7%

На підставі вищенаведеного аналізу та методу визначення оцінок за різними критеріями можна зробити висновок, що ідеальних алгоритмів не існує, а запропоновані можна покращити збільшивши точність та швидкість роботи алгоритмів.

1.3 Аналіз методів розв'язання поставленої задачі розробки системи розпізнавання дорожніх знаків

Ще одною важливою характеристикою елементів дорожнього руху – є форма (кругла, трикутна, восьмикутна або ж прямокутна). В порівнянні з методом розпізнавання об'єкту за допомогою кольору, даний метод не має залежності від погодній умов і змін яскравості завдяки тому, що не використовує інформацію про колір об'єкту, тому багато сучасних систем використовують інформацію про форму об'єкту для його ідентифікації.

Для розпізнавання знаків дорожнього руху за формою знаку використовують різні алгоритми розпізнавання форми, наприклад пошук контурів і їх наближення для отримання остаточного рішення, виходячи з кількості контурів. Даний метод також може використовуватись в якості детектора дорожніх знаків, завдяки тому, що він дозволяє знаходити області

інтересу з вхідної картинки, а потім їх окремо розглядати для подальшої ідентифікації знаку. Проте, такий підхід може бути затратним по пам’яті, якщо вхідна картинка має велику роздільну здатність.

Існує багато різних алгоритмів розпізнавання форми елементу, такі як:

- «Трансформація Хафа»;
- «Виявлення схожості»;
- «Узгодження трансформації відстані»;
- «Виявлення характеристик контурів»;
- «Характеристики, схожі Хаара».

Найбільш поширеним алгоритмом є – «Трансформація Хафа». Алгоритм Хафа може використовуватись для розпізнавання ліній, кіл, прямокутників чи інших кривих. Цей метод вперше був введений в 1962 році, і його першим застосуванням було пошук ліній в послідовності зображень. Основна перевага «Трансформації Хафа» – це стійкість до шумів, масштабування та обертання. Автори в [13] використовували «Трансформацію Хафа» для кіл для розпізнавання круглих дорожніх знаків, а для розпізнавання трикутних знаків автори використали «Трансформацію Хафа» для розпізнавання прямих ліній.

Даний алгоритм базується на основі представлення елемента, якого ми шукаєм у вигляді деякого параметричного рівняння. Масив параметрів рівняння називають «простором Хафа», для кожної точки вхідного зображення розраховується параметричне рівняння і отримуються параметри, які зберігаються в просторі Хафа. Фінальним кроком алгоритму є обхід простору Хафа і вибір максимальних значень, за яке «проголосувало» найбільша кількість пікселів зображення, що дає нам параметри для рівняння пошукового елементу. В основі алгоритму Хафа лежить ідея, що будь-яка точка зображення може бути частиною деякого набору можливих ліній.

Виявлення схожості – алгоритм розпізнавання форми об’єкту. Алгоритм базується на пошуку фактору подібності між набором вже відомих зображень з вхідним зображенням. При цьому вхідна картинка повинна бути добре

сегментована та мати таку ж роздільну здатність. Цей метод був використаний в одній праці, де на своєму наборі зображень знаків дорожнього руху, де автор отримав точність розпізнавання 86.3%. Основна перевага даного методу це простота і прямолінійність, в той час як недолік цього методу це строгі обмеження вхідного зображення, яке повинне бути правильно сегментовано та зжато до потрібного розміру, який зазвичай такий же, як зображення в навчальному наборі даних.

Узгодження трансформації відстані – алгоритм розпізнавання елементу за допомогою форми. В алгоритмі, кожному пікселю, який не є граничним, розраховують відстань до найближчого кутового пікселю (який ймовірно є границею об’єкту), далі за допомогою обчислених параметрів отримують параметри для параметричного рівняння. Алгоритм є схожим до «Виявлення схожості», бо отримані відстані від не граничних пікселів до найближчих кутових пікселів є обернено пропорційні відповідності вхідного зображення та уже відомим шаблонам (навчальному набору).

Характеристики контурів – одна з основних технік обробки зображень для знаходження меж різних предметів на зображені. Цей метод найбільше застосовують в системах обробки зображення та в комп’ютерному зорі для сегментації, коли зображення розділяють на ділянки відповідні різним об’єктам. Наприклад, детектор Кенні був використаний в роботі [4], оскільки цей метод зберігає контури, що є необхідним для детекторів дорожнього руху.

Метод виявлення кутових точок використовує кутовий детектор Харріса для виявлення дорожній знаків. Вони розпізнавали трикутні та квадратні форми знайшовши кути для кожної області, використовуючи кут Харріса. Потім існування кутів перевіряється в шести різних областях управління області пошуку. Кутовий детектор Харріса вперше був введений у 1988 році Крісом Харрісом та Майком Стівенсом [5]. Цей кутовий детектор став популярним, тому що він простий і швидкий. Більше того, кутовий детектор Харріса не залежить від масштабу, обертання та освітлення.

Як відомо методи розпізнавання елементів дорожнього руху за допомогою кольору та форми мають свої переваги та недоліки. Тому багато систем використовують різні комбінації цих методів для досягнення максимальної точності розпізнавання.

Багато систем розпізнавання елементів дорожнього руху включають в себе крок сегментації кольору, за яким слідує етап розпізнавання форми елементу. Фанга в своїй роботі [9] використав характеристику відтінку кольору та метод виявлення кутових точок для розпізнавання форми елементу. [9] є ще одним прикладом використання композиції методів розпізнавання елементів за допомогою кольору та форми.

Алгоритми класифікації елементів дорожнього руху використовуються після локалізації ROI для подальшого визначення вмісту виявлених дорожніх знаків. Розуміння правила дорожнього руху, що задається дорожнім знаком, досягається за допомогою зчитування внутрішньої частини виявленого дорожнього знаку за допомогою методу класифікатора. Алгоритми класифікації не ґрунтуються ні на кольорі, ні на формі. Класифікатор зазвичай приймає певний набір параметрів на вхід, за якими відрізняє кандидатів один від одного. Для класифікації дорожніх знаків використовуються різні алгоритми в яких є свої недоліка та переваги.

Методи класифікації об’єктів:

- метод відповідності шаблонів,
- дерево вибору,
- генетичний алгоритм,
- нейроні мережі,
- AdaBoost класифікатор,
- Опорна векторна машина (SVM).

Метод відповідності шаблонів є загальним методом обробки зображень та розпізнавання шаблонів. Це підхід низького рівня, який використовує заздалегідь задані шаблони для пошуку зображення піксель за пікселем. Основна

перевага даного підходу – це швидкодія та точність (максимально отримана точність було 90% на власному наборі даних). Однак недоліком цього методу є те, що він дуже чутливий до шуму та оклюзій. Крім того, для даного методу потрібен окремий шаблон для різних розмірів зображення об’єкта та орієнтації.

Одним із поширених метод класифікації – дерево вибору. Це метод машинного навчання, який функціонує шляхом побудови безлічі дерев рішень протягом навчального. Цей метод порівнювали з класифікаторами на основі SVM, MLP, класифікаторами на основі гістограми орієнтованого градієнта (HOG), показуючи найвищу швидкість точності та найменший час обчислення. На основі власного набору даних точність становила приблизно 94,2%, тоді як точність SVM становить 87,8%, а в MLP – 89,2%. Щодо обчислювального часу для однієї класифікації, SVM займає 115,87 мс, MLP займає 1,45 мс, а дерево рішень займає 0,15 мс. Незважаючи на високу точність та низький час обчислення, головне обмеження методу дерева рішень полягає в тому, що велика кількість дерев може зробити алгоритм громіздким, що спричинить уповільнення і неефективним для прогнозування в реальному часі.

Генетичний алгоритм – це ще один метод класифікації об’єктів. Він заснований на природному процесі відбору, що імітує біологічну еволюцію, яка була використана на початку цього століття. У цих дослідженнях було доведено, що цей метод ефективний для виявлення дорожнього знаку, навіть якщо дорожній знак має певні деформації форми чи недостаток освітленості. Недоліком генетичного алгоритму є недетермінований час роботи та негарантоване знаходження найкращого рішення. Перші спроби симуляції еволюції були проведені у 1954 році Нільсом Баричеллі на комп'ютері, встановленому в Інституті перспективних досліджень Принстонського університету. Його робота, що була опублікована у тому ж році, привернула увагу громадськості. З 1957 року, австралійський генетик Алекс Фразер[en] опублікував серію робіт з симуляції штучного відбору серед організмів з множинним контролем вимірюваних характеристик. Це дозволило комп'ютерній

симуляції еволюційних процесів та методам, які були описані у книгах Фразера та Барнела та Кросбі, з 1960-х років стати більш розповсюдженим видом діяльності серед біологів. Симуляції Фразера містили усі найважливіші елементи сучасних генетичних алгоритмів. До того ж, Ганс-Іоахім Бремерман в 1960-х опублікував серію робіт, які також приймали підхід використання популяції рішень, що піддаються відбору, мутації та рекомбінації, в проблемах оптимізації. Дослідження Бремермана також містили елементи сучасних генетичних алгоритмів. Також варто відмітити Річарда Фрідберга, Джоржа Фрідмана та Майкла Конрада.

Інший найпоширеніший метод класифікації – це використання штучної нейронної мережі (ANN). Цей метод завоював все більшу популярність в останні роки завдяки просуванню в обчисленні загального призначення технологій графічної обробки (GPU). Крім того, він популярний завдяки своїй надійності, більшої адаптивності до змін, гнучкості та високій швидкості та точності. Ще однією ключовою перевагою цього методу є його здатність розпізнавати та класифікувати об’єкти одночасно, зберігаючи високу швидкість та точність. В одному експерименті, швидкість потрапляння становила 97,6%, а обчислювальний час ~0,2 с. Однак у методах на основі ANN були описані деякі обмеження, такі як їх повільність та нестабільність у навчанні ANN через занадто великий крок. Цей метод порівнювали з методом відповідності шаблонів, роблячи висновок, що для ANN потрібна велика кількість навчальних зразків для реальних програм. Вперше про штучні нейронні мережі заговорили в 1940-х роках. Саме як наукова дисципліна теорія нейронних мереж була відображена в роботі Мак Каллока і Пітса в 1943 році. У даній роботі стверджувалося, що майже будь-яку логічну або арифметичну функцію можна реалізувати за допомогою найпростішої нейронної мережі.

У 1949 році Дональд Гебб сформував закон, який став відправним пунктом для навчання нейронних мереж. Він припустив, що навчання, перше за все полягає в зміні сили синаптичних зв’язків. Його теорія – типовий випадок

самонавчання, де випробувана система навчається виконувати потрібну задачу без втручання експериментатора.

Ф. Розенблат в 1958 році запропонував нейронну мережу, названу персептроном, який був призначений для класифікації об'єктів. При навчанні персептрон отримував повідомлення від «вчителя». Завдяки повідомленню, можна визначити до якого класу належить даний об'єкт. Крім того, навчений персептрон був здатний самостійно класифікувати об'єкти, які раніше не використовувались, роблячи при цьому мізерну кількість помилок.

Період затишшя у розвитку нейронних мереж припав на 1968-1985рр. З появою високопродуктивних персональних комп'ютерів з'явилася можливість моделювати нейронні мережі. Настільною книгою фахівців, які цікавились теорією нейронних мереж, стала робота Філіпа Уоссермена «нейрокомп'ютерна техніка».

Коли з'явилася робота Джона Гопфілда у 1982 році, зацікавленість до нейронних мереж сильно зросла. Гопфілд, ґрунтуючись на правилах навчання Гебба, показав, що задачі з нейронами можуть бути зведені до узагальненого ряду моделей, розроблених на той час у фізиці невпорядкованих систем.

Саме 80-тих роках поступово сформувався міцний теоретичний фундамент, на основі якого сьогодні створюється більшість мереж. Розроблена теорія стала широко застосовуватися в останні два десятиліття для вирішення прикладних завдань. Почали з'являтися компанії, що займаються розробкою програмного забезпечення для конструювання штучних нейронних мереж.

У 90-ті роки нейронні мережі стали використовувати в бізнесі, де вони показали колосальну ефективність при вирішенні багатьох завдань – від передбачення попиту на продукцію до аналізу платоспроможності клієнтів банку.

Джефрі Гінтон у 2007 році створив алгоритми глибокого навчання нейронних мереж. При навчанні нижніх слоїв мережі, Гінтон використовував обмежену машину Больцмана, яка являє собою стохастичну рекурентну

нейронну мережу. Після навчання мережі, отриманий застосунок міг швидко виконувати поставлену задачу (наприклад, пошук лиць на фото). Дану функцію вмонтовано у всі цифрові фотоапарати. Подібна технологія використовується пошуковими системами в інтернеті для класифікації картинок.

За оцінками фахівців, в області проектування нейронних мереж і нейрокомп'ютерів очікується технологічний ріст. Чимало нових можливостей було відкрито за останні роки, а праці в даній області стають важливим внеском в науку, технології, економіку. Не зважаючи на те, що вивчення нейронного моделювання триває вже понад шістдесят років, немає жодної області мозку, де процес обробки інформації був би зрозумілий до кінця. Також немає жодного нейрона, для якого можна було б визначити код передачі інформації у вигляді послідовності імпульсів.

В даний час існує велика кількість конфігурацій нейронних мереж, які відрізняються за принципами функціонування, а отже, спрямовані на виконання різних задач.

Майбутнє нейрокомп’ютерних технологій буде пов’язано з новими відкриттями в області нейронного моделювання – як тільки вдасться розгадати таємницю функціонування хоча б однієї області мозку, відразу ж стане багато чого зрозуміло і про інші його області.

Можна сказати, що штучна нейронна мережа являється математичною моделлю, а також її програмною та апаратною реалізацією, що побудована за принципом, схожим до біологічних нейронних мереж – нервових клітин живого організму. Дане поняття з'явилося при спробі змоделювати процеси, що відбуваються у мозку людини. Дана мережа представляє систему, яка складається з простих процесорів, що з’єднанні та функціонують між собою. Кожен з процесорів системи має справу з сигналами, які періодично надходять або передаються іншими процесорами. Велика мережа здатна вирішувати дуже складні задачі за короткий період часу. З математичної точки зору нейронні мережі являють собою спосіб вирішення нелінійних задач оптимізації. У

кібернетиці використовується теорія нейронних мереж для вирішення задач адаптивного управління, побудови алгоритмів для робототехніки

1.4 Постановка задачі для розробки програмного-модулю розпізнавання дорожніх знаків

Методика розпізнавання образів оперує такими поняттями як:

- клас – множина об'єктів, що мають загальні властивості, класів може бути необмежена кількість;
- класифікація – процес призначення міток класу об'єктів, відповідно до деякого опису властивостей цих об'єктів;
- класифікатор – пристрій, який в якості вхідних даних отримує набір ознак об'єкта, а в якості результату видає мітку класу;
- верифікація – процес зіставлення примірника об'єкта з однією моделлю об'єкта або описом класу;
- ознака – кількісний опис тієї чи іншої властивості досліджуваного предмета або явища;
- простір ознак – це N-мірний простір, визначене для даної задачі розпізнавання, де N – фіксоване число вимірюваних ознак для будь-яких об'єктів.

Вектор з простору ознак, відповідний об'єкту завдання розпізнавання це N-мірний вектор з компонентами (x_1, x_2, ..., x_N), які є значеннями ознак для даного об'єкта.

Таким чином, задача розпізнавання зводиться до виділення істотних ознак для кожного класу і віднесення вхідних даних до одного з них за допомогою виявлення ключових ознак в оригінальному зображенні.

Розпізнавання образів можна розділити на кілька завдань, таких, як:

- отримання вхідних даних, за допомогою сенсорів, відеоспостереження, добірок даних;
- первинна обробка зображень така, як нормалізація даних, фільтрація

шумів, виявлення ознак;

- формування векторів ознак, за допомогою вибору найбільш значущих ознак, за допомогою яких можна виділити непересічні безлічі класів;
- класифікація чи прогнозування на основі отриманих даних про класи.

Метод Speeded Up Robust Features (SURF) – дозволяє виконувати пошук особливих точок на зображенні і вираховувати їх дескриптори, які будуть інваріантні до масштабу і обертанню. Пошук здійснюється з допомогою матриці Гессе. Гессіан зображення, який вираховується як детермінант матриці Гессе, в точках максимального перепаду яскравості досягає екстремуму. Це дозволяє знайти на зображенні такі особливі точки, як кути, краї ліній та інше.

Для обчислення матриці Гессе і фільтру Хаара зручніше перейти до інтегрального представлення зображення. Інтегральне представлення зображення являє собою матрицю, розмірність якої співпадає із розмірністю вхідного зображення.

Метод опорних векторів спочатку відноситься до бінарних класифікаторів, хоча існують способи змусити його працювати і для задач мультикласифікації.

Для покращення точності та оптимізація розпізнавання елементів дорожнього руху необхідно вирішити такі задачі:

- аналіз існуючих алгоритмів для розпізнавання елементів на зображені.
- програмна реалізація інтелектуальної системи.
- точність розпізнавання елементів дорожнього руху повинна бути більше 93%;
- знайдені елементи повинні бути інтуїтивно виділені серед інших елементів;
- можливість використання фото файлу в якості вхідних даних для

системи;

- простота у налагодженні.

1.5 Висновки

У розділі було проведено аналіз існуючих рішень та предметної області. Таким чином, було прийнято рішення модифікувати існуючий метод розпізнавання дорожніх знаків, для підвищення ефективності розпізнавання та розробити програмний модуль для систем детектування дорожніх знаків. Як результат було проведено аналіз та постановку задачі, наведене головне призначення, цілі та задачі розробки.

2 ПРОГРАМНІ МЕТОДИ ТА ЗАСОБИ РОЗРОБКИ ПРОГРАМНОГО ДОДАТКУ

2.1 Варіантний аналіз і обґрунтування вибору засобів реалізації

Для вибору мови програмування використано такі мови програмування, що мають найвищу ефективність для даної розробки є найвищою. Дану парадигму підтримують мови C++ , C#, Java та Python. Розглянемо особливості кожної з них.

C++ – компільована, статично типізована мова програмування, яка поєднує властивості як високорівневих та і низькорівневих мов програмування. Код написаний на C++ компілюється напряму в машинний код, що робить дану мову одною з найшвидших в світі. C++ повністю сумісний з мовою C. Мова C++ широко використовується для розробки прикладних програм, операційних систем, драйверів пристроїв, відеоігор тощо [4].

C++ має такі відмінності порівняно з C:

- підтримка об'єктно-орієнтованого програмування через класи;
- підтримка узагальненого програмування через шаблони;
- доповнення до стандартної бібліотеки;
- додаткові типи даних;
- обробка винятків;
- вбудовані функції;
- перевантаження операторів;
- посилання і оператори управління вільно розподіленою пам'яттю.

З появою нового стандарту C++11 мова програмування доповнилась новими функціями:

- Розширення стандартної бібліотеки в таких областях, як регулярні вирази, хеші, генератори випадкових чисел, інтелектуальні вказівники тощо.
- Підтримка лямбда-виразів і лямбда-функцій, тобто анонімних функцій, які декларуються в місці використання.

- Підтримка списків ініціалізації, тобто передачі структури або масиву у вигляді списку значень.
- Можливість автоматичного призначення типу при вказівці ключового слова «auto».
- Можливість створення шаблону функції, що повертається, тип якого визначається автоматично на підставі іншої функції або виразу.
- Можливість створювати шаблони з перемінною кількістю аргументів.
- Можливість викликати одні конструктори класу з інших конструкторів цього ж класу, що дозволяє створювати конструктори, що використовують інші конструктори без дублювання коду.
- Замість макросу NULL для позначення нульового вказівника введено ключове слово nullptr.

C++ - надзвичайно потужна мова, що містить засоби створення ефективних програм практично будь-якого призначення, від низькорівневих утиліт та драйверів до складних програмних комплексів різного призначення. В C++ підтримуються різні стилі та технології програмування, включаючи традиційне директивне програмування, ООП, узагальнене програмування, метапрограмування (шаблони, макроси).

Передбачуване виконання програм є важливою перевагою побудови систем реального часу. Весь код, що неявно генерується компілятором для реалізації мовних можливостей (наприклад, при перетворенні змінної на інший тип), визначений у стандарті. Також суворо визначено місця програми, у яких цей код виконується. Це дає можливість вимірювати або розраховувати час реакції програми на зовнішню подію. Автоматичний виклик деструкторів об'єктів за її знищенні, причому у порядку, зворотному виклику конструкторів. Це спрощує (досить оголосити змінну) і робить надійнішим звільнення ресурсів (пам'ять, файли, семафори тощо), а також дозволяє гарантовано виконувати переходи станів програми, не обов'язково пов'язані зі звільненням ресурсів (наприклад, запис до журналу).

Користувальницькі функції-оператори дозволяють коротко і об'ємно записувати вирази над типами користувача в природній алгебраїчній формі.

Мова підтримує поняття фізичної (const) та логічної (mutable) константності. Це робить програму надійнішою, оскільки дозволяє компілятору, наприклад, діагностувати помилкові спроби зміни значення змінної. Оголошення константності дає програмісту, який читає текст програми додаткове уявлення про правильне використання класів і функцій, і навіть може бути підказкою оптимізації. Перевантаження функцій-членів за ознакою константності дозволяє визначати зсередини об'єкта цілі виклику методу (константний читання, неконстантний зміни). Оголошення mutable дозволяє зберігати логічну константність при використанні кешів та лінивих обчислень.

Використовуючи шаблони, можна створювати узагальнені контейнери та алгоритми для різних типів даних, а також спеціалізувати та обчислювати на етапі компіляції. Можливість імітації розширення мови для підтримки парадигм, які безпосередньо не підтримуються компіляторами. Наприклад, бібліотека Boost.Bind дає змогу пов'язувати аргументи функцій. Також є можливість створення вбудованих предметно-орієнтованих мов програмування. Такий підхід використовує, наприклад бібліотека Boost.Spirit, що дозволяє задавати EBNF-граматику парсерів у коді C++.

Використовуючи шаблони та множинне спадкування можна імітувати класи-домішки та комбінаторну параметризацію бібліотек. Такий підхід застосований у бібліотеці Loki, клас SmartPrt якої дозволяє, керуючи лише декількома параметрами часу компіляції, згенерувати близько 300 видів розумних покажчиків для управління ресурсами.

Кросплатформенність: стандарт мови накладає мінімальні вимоги на ЕОМ для запуску скомпільованих програм. Для визначення реальних властивостей системи виконання стандартної бібліотеці присутні відповідні можливості (наприклад, std::numeric_limits T). Доступні компілятори для великої кількості платформ, мовою C++ розробляють програми для різних платформ і систем.

Ефективність. Мова спроектована так, щоб дати програмісту максимальний контроль над усіма аспектами структури та порядку виконання програми. Жодна з мовних можливостей, що веде до додаткових накладних витрат, не є обов'язковою для використання — за необхідності мова дозволяє забезпечити максимальну ефективність програми.

Деякі вважають недоліком мови C++ відсутність вбудованої системи складання сміття. З іншого боку, засоби C++ дозволяють реалізувати складання сміття на рівні бібліотеки. Противники складання сміття вважають, що RAII є більш гідною альтернативою. C++ дозволяє користувачеві самому вибирати стратегію управління ресурсами, щ обуло покладенно в основну мету створення мови.

C# – об’єктно орієнтована мова, розроблена компанією Microsoft спеціально для використання можливостей .NET Framework. C# відноситься до мов програмування з C-подібним синтаксисом. Мова має строгу статичну типізацію, підтримує перевантаження операторів, анонімні функції, узагальнені типи і методи, вказівники на члени функції класів. C# не підтримує множинне спадкування класів, на відміну від C++, але підтримує множинне спадкування інтерфейсів [5] C# є дуже близьким родичем мови програмування Java. Мова Java була створена компанією Sun Microsystems, коли глобальний розвиток інтернету поставив завдання розподілених обчислень. Взявши за основу популярну мову C++, Java виключила з неї потенційно небезпечні речі (на зразок вказівників без контролю виходу за межі). Для розподілених обчислень була створена концепція віртуальної машини та машинно-незалежного байт-коду, свого роду посередника між вихідним текстом програм і апаратними інструкціями комп'ютера чи іншого інтелектуального пристрою. Java набула чималої популярності, і була ліцензована також і компанією Microsoft. Але, з часом, Sun почала звинувачувати Microsoft, що та при створенні свого клону Java робить її сумісною виключно з платформою Windows, чим суперечить самій концепції машинно-незалежного середовища виконання і порушує ліцензійну угоду.

Microsoft відмовилася піти назустріч вимогам Sun, і тому з'ясування стосунків набуло статусу судового процесу. Суд визнав позицію Sun справедливою, і зобов'язав Microsoft відмовитися від позаліцензійного використання Java.

Мова має строгу статичну типізацію, підтримує поліморфізм, перевантаження операторів, вказівники на функції-члени класів, атрибути, події, властивості, винятки, коментарі у форматі XML. Перейнявши багато що від своїх попередників — мов C++, Delphi, і Smalltalk — C#, спираючись на практику їхнього використання, виключає деякі моделі, що зарекомендували себе як проблематичні при розробці програмних систем, наприклад множинне спадкування класів (на відміну від C++).

Java – об’єктно-орієнтована мова програмування, одна з найпоширеніших мов програмування в світі. Основною особливістю є те, що її програмний код спочатку транслюється в спеціальний байт-код, незалежний від платформи, а згодом байт-код виконується віртуальною машиною JVM (Java Virtual Machine)[6]. Це дозволяє код написаний на Java запускати майже на всіх відомих системах: Windows, Mac, Android тощо. Спочатку мова називалася Oak («дуб») і розроблялася Джеймсом Гослінгом для програмування побутових електронних пристроїв. Згодом вона була перейменована в Java і стала використовуватися для написання клієнтських застосунків і серверного програмного забезпечення. Названа на честь марки кави Java, яка, в свою чергу, отримала найменування однойменного острова (Ява), тому на офіційній емблемі мови зображена чашка з паркою кавою. Існує й інша версія походження назви мови, пов'язана з алюзією на каво-машину як приклад побутового устаткування, для програмування якого спочатку мова створювалася. Але дана мова програмування не є компільованою, як C++, саме через це код написаний на Java працює повільніше. Java розроблялась як платформо-незалежна мова, тому вона має менше низькорівневих можливостей для роботи з апаратним забезпеченням. За необхідності таких дій java дозволяє викликати підпрограми, написані іншими мовами програмування. Мова значно запозичила синтаксис із C і C++. Зокрема,

взято за основу об'єктну модель C++, проте її модифіковано. Усунуто можливість появи деяких конфліктних ситуацій, що могли виникнути через помилки програміста та полегшено сам процес розробки об'єктно-орієнтованих програм. Ряд дій, які в C/C++ повинні здійснювати програмісти, доручено віртуальній машині. Передусім Java розроблялась як платформо-незалежна мова, тому вона має менше низькорівневих можливостей для роботи з апаратним забезпеченням, що в порівнянні, наприклад, з C++ зменшує швидкість роботи програм. За необхідності таких дій Java дозволяє викликати підпрограми, написані іншими мовами програмування.

Python – високорівнева мова програмування, як призначена для створення додатків різного типу: веб-додатки, відеоігри, настільні програми, утиліти для роботи з базами даних тощо. Велике розповсюдження Python отримав в області машинного навчання та штучного інтелекту. Код написаний на Python інтерпретується байт-код який переводиться віртуальною машиною в набір команд, які виконуються операційною системою. Python, подібно іншим інтерпретованим мовам програмування, які не застосовують, наприклад, JIT-компілятори, має суттєвий недолік — порівняно невисоку швидкість виконання програм. Однак, у випадку з Python цей недолік компенсується зменшенням часу розробки програми. У середньому, програма, написана на Python, в 2-4 рази компактніша, ніж її аналог на C++ або Java. Збереження байт-коду (файли .pyc і .pyo) дозволяє інтерпретатору не витрачати зайвий час на перекомпіляцію коду модулів при кожному запуску, на відміну, наприклад, від мови Perl. Крім того, існує спеціальна JIT-бібліотека psyco (проте призводить до збільшення споживання оперативної пам'яті). Ефективність psyco в значній мірі залежить від архітектури програми.

Проаналізувавши мови програмування, було складено таблицю (табл. 2.1), яка демонструє відмінності Java, Python, C#, C++.

Таблиця 2.1 Функціональні характеристики мов програмування

	Java	Python	C#	C++
Висока продуктивність	1	0	0	1
Статична типізація	1	0	1	1
Багатоплатформність	1	1	0	1
Швидкість виконання	0	1	1	1
Мова високого рівня	1	1	1	1
Безпечність	1	0	0	1
Сумарний коефіцієнт	5	3	3	6

Отже, проаналізувавши відмінні ознаки Java, Python, C# та C++, було прийнято рішення, що C++ найбільше підходить для виконання поставленої задачі.

Вибір середовища програмування, є дуже важливим етапом у створенні програмного модулю. Щоби вибрати потрібно виділити основні вимоги. Отже, що нам потрібно від середовища розробки? Набір функцій різних середовищ може відрізнятися, але є набір базових речей, що спрощують програмування: збереження файлів. Важливо щоб IDE або редактор надавали можливості зберегти роботу і пізніше все відкрити в тому ж стані, в якому воно було під час закриття, а ящо вони цього не забезпечують, то не така вже це і IDE. Підтримка налагодження, можливість крок за кроком виконати код є базовою функцією всіх IDE і більшості хороших редакторів коду. Підсвічування та виділення синтаксису. Не забуваємо про можливість швидко знайти ключові слова, змінні та інше робить читання і розуміння коду на порядок простіше. Автоматичне форматування коду, максимально зручна та корисна функція, що зберігає час та робить написання коду цікавим. Потрібно зазначити, що IDE (або інтегроване середовище розробки) - це програма, призначена для розробки програмного забезпечення. як випливає з назви, IDE об'єднує кілька інструментів, спеціально

призначених для розробки. Ці інструменти зазвичай включають редактор, призначений для роботи з кодом (наприклад, підсвічування синтаксису і автодоповнення) та інструменти збірки, виконання та налагодження чи і певну форму системи управління версіями.

Для порівняння використано такі середовища як, Qt Creator, C++ Builder, Eclipse.

Qt Creator – інтегроване середовище розробки, призначене для створення крос-платформових застосунків з використанням бібліотеки Qt (див. рис. 2.1.). Підтримується розробка як класичних програм мовою C++, так і використання мови QML, для визначення сценаріїв, в якій використовується JavaScript, а структура і параметри елементів інтерфейсу задаються CSS-подібними блоками. Qt Creator може використовувати GCC або Microsoft VC++ як компілятор і GDB як зневаджувач. Для Windows версій бібліотека комплектується компілятором, заголовними і об'єктними файлами MinGW [7].

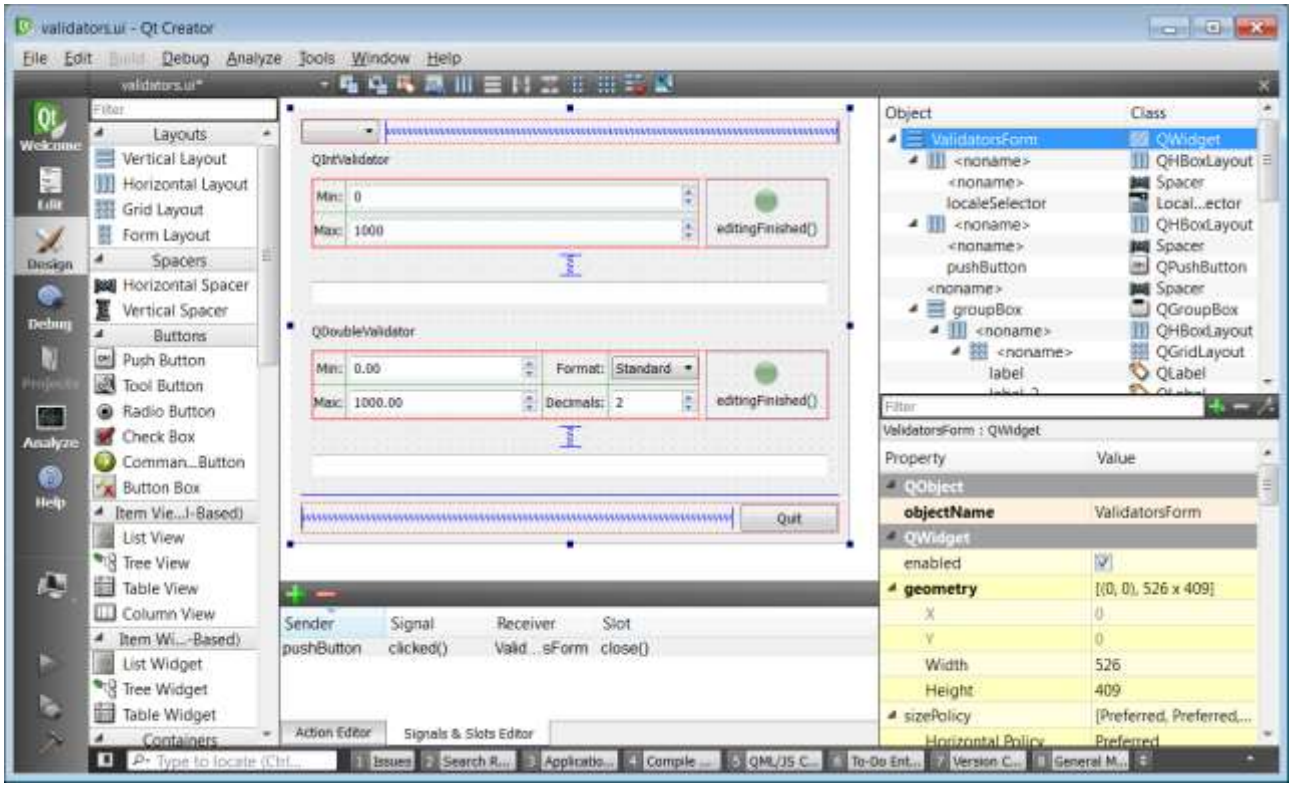


Рисунок 2.1 – Інтерфейс програмного додатку «Qt Creator»

Спочатку розроблявся компанією Borland Software, а потім її підрозділом CodeGear, який сьогодні належить компанії Embarcadero Technologies.

C++ Builder (див. рис. 2.2) об'єднує в собі комплекс об'єктних бібліотек (STL, VCL, CLX, MFC та ін.), компілятор, зневаджувач, редактор коду та багато інших компонентів. Цикл розробки аналогічний Delphi. Більшість компонентів, розроблених в Delphi, можна використовувати і в C++ Builder без модифікації, але зворотнє твердження не вірне. Розробники та команди розробників ПЗ, наступні ідеології Agile, зможуть писати код швидше та краще за допомогою сучасних практик ООП, надійних фреймворків C++Builder та функціонального інтегрованого середовища розробки. Використання Code Insight ™ забезпечує code-completion на основі коду та використаних бібліотек, щоб допомогти швидко та точно писати код, а також налаштувати IDE відповідно до вашого стилю кодування. Доступний високошвидкісний прямий доступ до СУБД InterBase, SQLite, MySQL, SQL Server, Oracle, PostgreSQL, DB2, SQL Anywhere, Advantage DB, Firebird, Access, Informix, MongoDB та багато інших.

C++ Builder дозволяє використовувати популярні бібліотеки, такі як Boost, Eigen та ZeroMQ, а також широкий набір інструментів та бібліотек спільноти розробників. Проекти можуть бути інтегровані у систему контролю версій, включаючи Git, Subversion та Mercurial

За допомогою інтегрованого середовища розробки C++Builder можна налагоджувати програми, що працюють дистанційно під операційними системами Windows, macOS, iOS і Android.

Швидші цикли розробки не обов'язково призводять до погіршення якості. C++Builder включає безліч функцій, що дозволяють впровадити передові методи при написанні коду, знизити дублювання та допомогти у розробці.

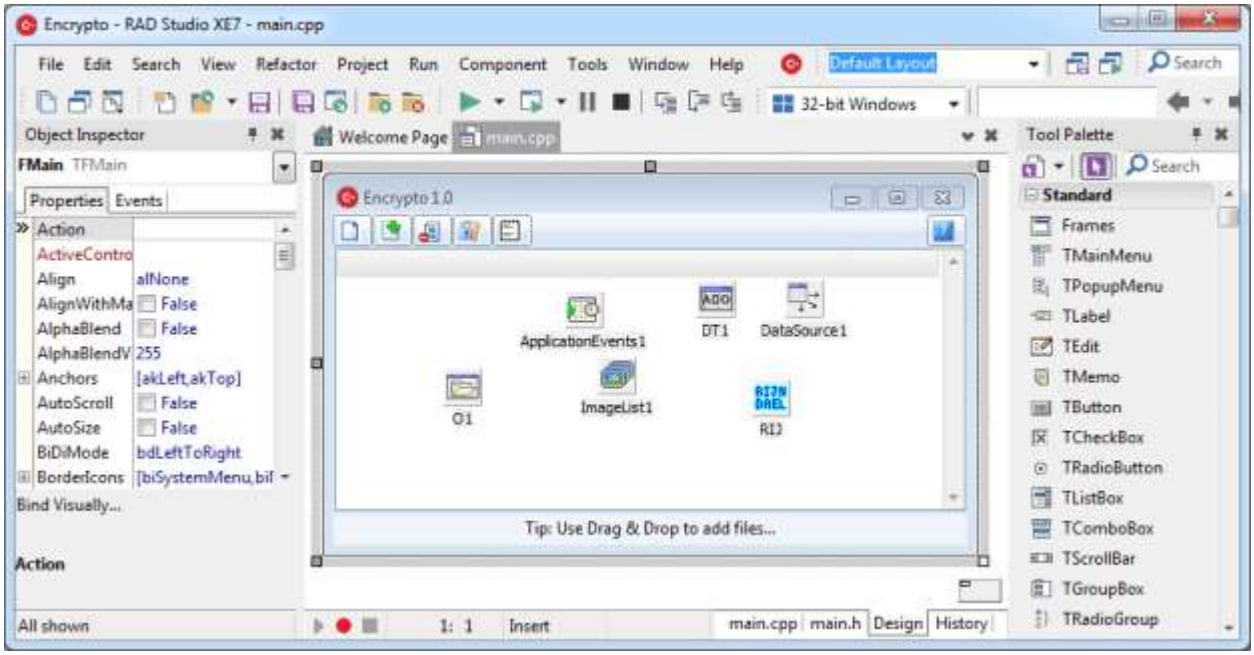


Рисунок 2.2 – Інтерфейс програмного додатку «C++ Builder»

Eclipse – вільне модульне інтегроване середовище розробки програмного забезпечення (див. рис. 2.3). Розробляється і підтримується Eclipse Foundation і включає проєкти, такі як платформа Eclipse, набір інструментів для програмістів на мові Java, системи контролю версій, конструктори GUI тощо. Будучи доступним для Linux, Windows і OS x, Eclipse де-факто є open-source IDE для розробки на Java. Існує безліч розширень і аддонів, які роблять Eclipse корисним для різного роду завдань. Одним з таких розширень є CDT, що надає інтерактивну консоль C++ і можливості для налагодження і автодоповнення коду. Встановити його досить просто: потрібно запустити Eclipse, вибрати help → Eclipse marketplace, потім знайти CDT, натиснути «install» і при необхідності перезапустити Eclipse. Написаний в основному на Java, може бути використаний для розробки застосунків на Java і, за допомогою різних плагінів, на інших мовах програмування, включаючи Ada, C, C++, C#, COBOL, Fortran, Perl, PHP, Python, R, Ruby (включно з каркасом Ruby on Rails), Scala, Clojure та Scheme. Середовища розробки зокрема включають Eclipse ADT (Ada Development Toolkit) для Ada, Eclipse CDT для C/C++, Eclipse JDT для Java, Eclipse PDT для PHP.

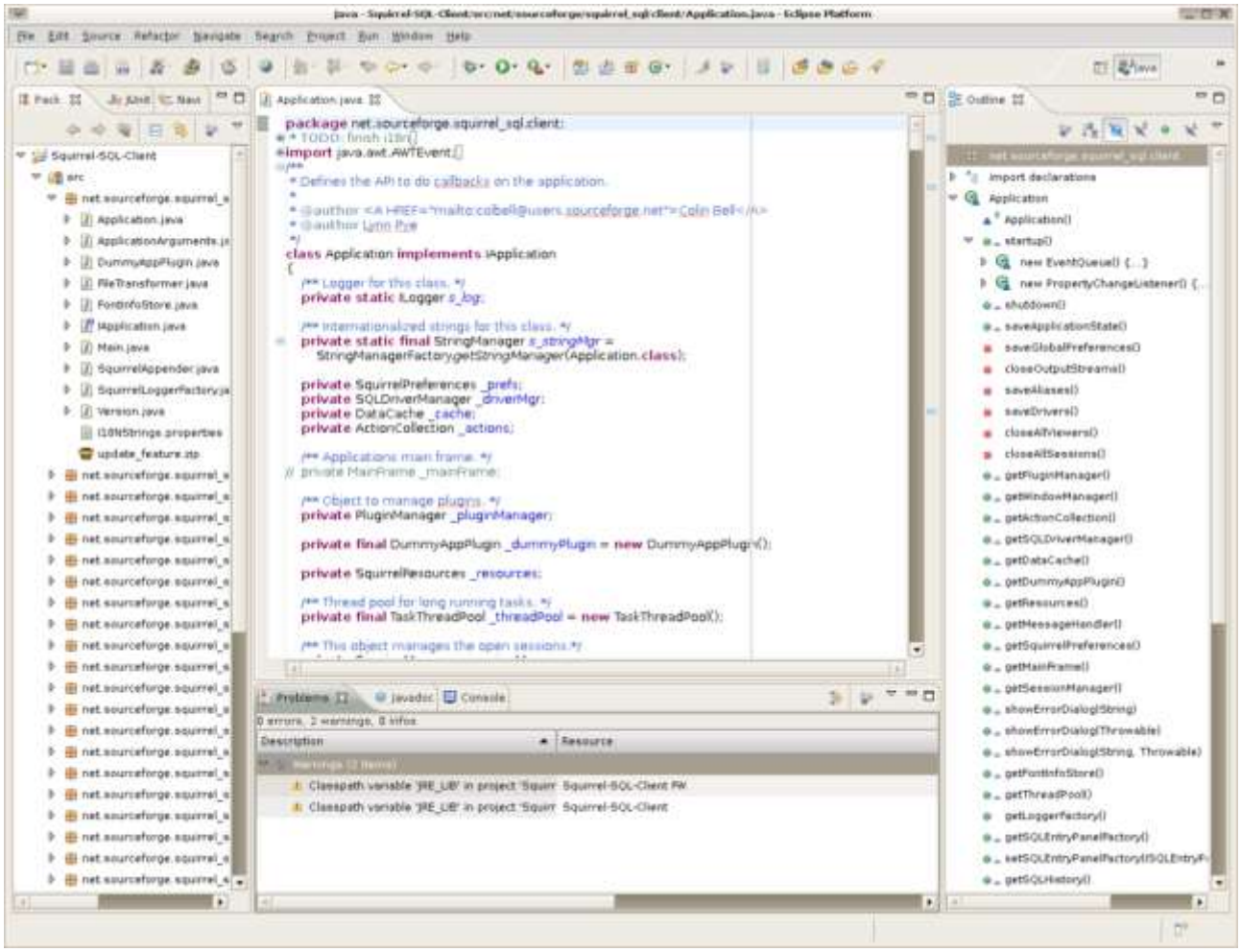


Рисунок 2.3 – Інтерфейс програмного додатку «Eclipse»

Початок коду йде від IBM VisualAge, він був розрахований на розробників Java, складаючи Java Development Tools (JDT). Але користувачі могли розширяти можливості, встановлюючи написані для програмного каркаса Eclipse плагіни, такі як інструменти розробки під інші мови програмування, і могли писати і вносити свої власні плагіни і модулі.

Випущена на умовах Eclipse Public License, Eclipse є вільним програмним забезпеченням. Він став одним з перших IDE під GNU Classpath і без проблем працює під IcedTea.

Microsoft Visual Studio 2019 – багатофункціональний інструмент для розробки програмного забезпечення, який представляє собою

кроссплатформенную IDE, яка має функції тестування, інструментами допомагають при розробці, і величезній кількості інших зручний функцій, покликаних підвищити якість написаного коду [5]. Сьогодні є актуальною версія 19.

Модуль CodeLens дозволяє провести аналіз коду, зрозуміти, як саме вплинуть на майбутній програмний продукт внесені у код зміни, провести модульне тестування. Користувач у будь-який момент може переглянути потрібні посилання, авторів, журнал фіксацій та інші дані. Процес відлагодження коду займає значну частину часу при написанні програмного продукту. Microsoft Visual Studio Professional 2019 забезпечує швидкий пошук помилок та їх виправлення. Щоб провести аналіз помилки, програміст може зупинити виконання коду у потрібному місці за допомогою відповідних методів та точок для зупинки. Тестування програмного продукту - необхідний та важливий етап у розробці якісного програмного продукту. У середовищі Microsoft Visual Studio Professional 2019 для тестування використовується набір тестів для впорядкування даних, проводять аналіз об'єму коду, який необхідно протестувати.

В сучасних умовах злагоджена спільна робота команди є необхідною для створення якісного продукту. Командну роботу в Visual Studio 2019 забезпечує функціонал Live Share. Функції командної роботи допомагають керувати процесом розробки, проводити спільне редагування коду, його відлагодження та тестування. Редактор містить спеціалізовані параметри, щоб індивідуально налаштовувати сеанси, забезпечувати спільний стиль написання коду. Microsoft Visual Studio Professional 2019 містить готові шаблони для різних типів застосунків і локальні емулятори Azure (навіть не маючи облікового запису Azure). Не виходячи з середовища Visual Studio розробник може налаштувати бази даних SQL.

Переваги: швидкі навігація, написання та виправлення коду; проста налагодження, профілювання і діагностика коду; комплексні інструменти

тестування допомагають писати якісний код; тисячі розширень дозволять вам налаштувати IDE під себе.

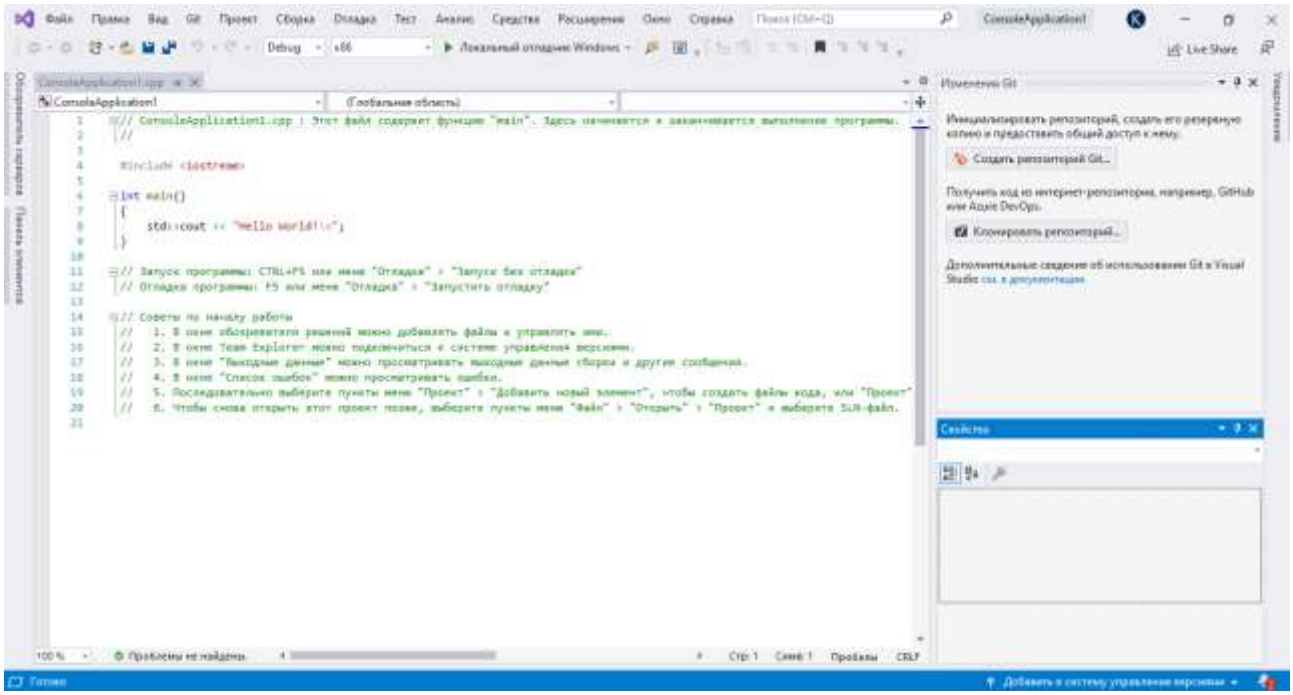


Рисунок 2.4 – Інтерфейс програмного додатку «Microsoft Visual Studio»

Для наочної демонстрації відмінностей даних середовищ розробки було створено таблицю порівняння (табл. 1.3)

Таблиця 2.2 - Функціональні характеристики середовищ розробки

	Qt Creator	C++ Builder	Eclipse	Microsoft Visual Studio
Зручний та зрозумілий інтерфейс	1	1	1	1
Легкий в налаштуванні	1	0	0	1
Багатоплатформність	1	0	1	1
Можливість використання додаткових модулів	1	1	1	1
Активно підтримується розробником	0	1	1	1
Сумарний коефіцієнт	4	1	4	5

Отже, в результаті проведеного аналізу середовищ розробки, для створення додатку було обрано середовище Microsoft Visual Studio.

2.2 Обгрунтування вибору бібліотек і компонентів

OpenCV (Open Source Computer Vision Library) являє собою бібліотеку з ліцензією BSD програмного забезпечення для комп'ютерного зору та комп'ютерного навчання з відкритим вихідним кодом, що включає декілька сотень алоритмів комп'ютерного бачення. OpenCV створена для забезпечення загальної інфраструктури додатків для комп'ютерного зору і прискорення використання сприйняття машини. Будучи повністю ліцензованим BSD продуктом, OpenCV спрощує бізнес для використання і модифікації коду [30].

Основною метою для розробки даної бібліотеки було підвищення ефективності в додатках реального часу. Функціонал бібліотеки реалізований на мові С але вона спокійно підходить для роботи з іншими мовами програмування, такими як: Python, C#, Java, Ruby. Фреймворк OpenCV має можливість використовувати багатоядерні процесори в своїй роботі. Можливе додаткове придбання та інтеграція з бібліотекою IPP, при необхідності автоматичної оптимізації на платформах Intel.

Взагалі до складу OpenCV бібліотеки входить більше 3000 оптимізованих алгоритмів, які включають в себе насичений набір класичних і сучасних алгоритмів машинного навчання та комп'ютерного зору. Ці алгоритми можуть використовуватися для безліч процесів виявлення і розпізнавання осіб чи ідентифікації об'єктів, класифікації дій людини на відео, розпізнавання домашніх чи диких тварин, відстеження рухомих об'єктів і рухів самої камери і багато інших об’єктів. OpenCV має величезну популярність серед користувачів, а кількість завантажень перевищує 18 мільйонів. Бібліотека дуже широко використовується в компаніях, дослідницьких групах і в урядових органах.

В доповнення, серед таких відомих компаній як Google, Yahoo, Microsoft, Intel, IBM, є безліч різноманітних стартапів, таких як Applied Minds, VideoSurf і Zeitera, які так само використовують функціонал OpenCV.

Бібліотека має інтерфейси для Python, C++,C# , Java і підтримує операційні системи Windows, Linux, Android і Mac OS. OpenCV використовує в основному додатки в режимі реального часу і використовує переваги команд MMX і SSE, якщо вони доступні. В даний час активно розвиваються повнофункціональні інтерфейси CUDA і OpenCL.

Поряд з добре відомими компаніями, такими як Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota, які використовують бібліотеку, є багато стартапів, таких як Applied Minds, VideoSurf і Zeitera, які широко використовують OpenCV[3]. Розгорнуті застосування OpenCV охоплюють діапазон від простих зображень streetview до виявлення вторгнень у відео спостереження в Ізраїлі, моніторингу шахтного обладнання в Китаї, допомоги роботам в навігації і підборі об'єктів в гаражі Willow, виявлення нещасних випадків з утопленям в басейнах, запуску інтерактивного мистецтва в Іспанії і Нью-Йорку, перевірки злітно-посадкових смуг на наявність сміття в Туреччині, перевірки етикеток на продуктах на заводах по всьому світу до швидкого виявлення особи в Японії [4].

Крім того, велика кількість аналізу зображень і відео зводиться до максимального спрощення джерела. Це майже завжди починається з перетворення в градації сірого, але це також може бути кольоровий фільтр, градієнт або їх комбінація. Звідси ми можемо робити всі види аналізу та перетворення джерела. Як правило, відбувається трансформація, потім аналіз, потім будь-які накладення, які ми хочемо застосувати, застосовуються назад до вихідного джерела, тому часто можна побачити «готовий продукт» можливого розпізнавання об’єктів чи обличчя. відображається на повнокольоровому зображенні або відео. Однак рідко дані фактично обробляються в необробленому вигляді.

Будова OpenCV зображена на (рис. 2.5).

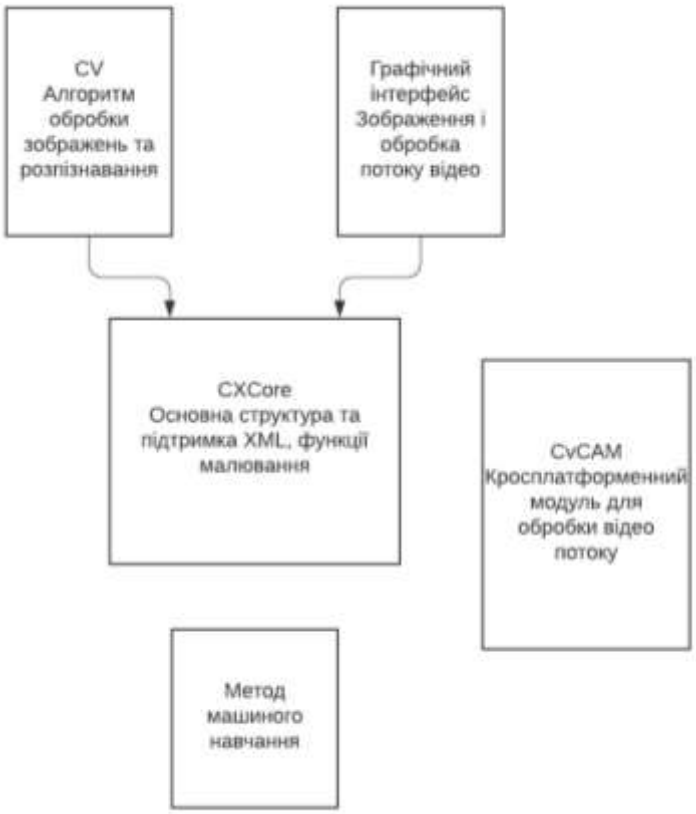


Рис. 2.5 Структурна схема OpenCV

Ядро CX CORE виконує такі функції:

- основні операції над багатовимірними масивами;
- математична обробка зображень;
- базова функціональність 2D графіки.

Ядро CM використовується для обробки зображень які будуть використовуватись для комп'ютерного зору:

- основні операції над зображеннями, структурування, математичні перетворення;
- аналіз зображення;
- алгебраїчний аналіз (описання форм);

- виявлення об’єктів, зокрема тексту.

Cvaux – модуль містить такі основні функції:

- об'ємний зір;
- пошук відповідностей;
- пошук тексту;
- пошук прихованих Ейлерових векторів [6].

Бібліотека Boost це набір фреймворків для C++, що прошли експертну оцінку та повністю поширюються під ліцензією Boost Software License. В основному використовується для проєктів з відкритим кодом, але також може бути використана для проєктів з закритим кодом. Наразі Boost вулючає в собі більше 150 бібліотек, що значно розширюють потенціал C++ як мови програмування та створює комфортний функціонал для потреб розробника. Головною метою даної бібліотеки є підняти якість функціонального програмування, представленням розробникам просунутих інструментів для розробки, доповненням бази інструментів C++ та включити нові можливості C++ так, щоб вони стали простими та зручними в користуванні. Метапрограмування з активним використанням шаблонів ті узагальнене програмування спрямовує C++ на розширюваність та широкі дослідження за допомогою цієї бібліотеки. Завдяки чіткому контролю якості та роботі незалежних експертів до фреймворку були включені високо продуктивні та надійні пакети. Boost – є кросплатформним і це забезпечує роботу розробки на основних операційних системах Windows, Linux и MacOS. «Boost Function» розширює взаємодію з функціями та коллбеками. Приклади бібліотек, що входять до Boos:

- «Boost Interprocess» дає можливість взаємодіяти на рівні процесів, тобто дозволяє налагоджувати взаємодію різних програм між собою.
- «Boost Pool» налагоджує роботу з пулами.

• «Boost Program_Options» додає можливість працювати з опціями програми.

• «Boost Python» дозволяє інтегрувати можливості Python у ваш проект.

• «Boost Regex» розширює можливості взаємодії з регулярними виразами.

Як ми бачимо, Boost складається з багатьох бібліотек і перевірка часом, каже, що він є дуже популярним, простим в використанні, а головне надійним.

Tesseract – це програмний рухок з відкритим вихідним кодом, який дозволяє розпізнати символи з підтримкою кодування Unicode і розпізнати більше 130 мов, а також мати можливість доповнення для розпізнавання інших мов. Eseract спочатку був розроблений в Hewlett-Packard Laboratories Bristol і Hewlett-Packard Co, Greeley Colorado між 1985 і 1994 роками, з деякими змінами, внесеними в 1996 році для перенесення на Windows, і деякими кодуванням C++ в 1998 році. У 2005 році Tesseract був відкритий. джерело HP. З 2006 по листопад 2018 року його розробляла Google. Програмне рішення Tesseract (рос. Тессеракт) з відкритим вихідним кодом від компанії Google призначено для розпізнавання тексту (англ. OCR). Програма розпространяється безкоштовно і доступна для використання за ліцензією Apache 2.0. Програмне забезпечення Tesseract може використовуватися безпосередньо або за допомогою API, що дозволяє витягувати друкований текст із зображень. Програмний рух Tesseract не має вбудованого графічного інтерфейсу (GUI), але є кілька доступних варіантів графічного інтерфейсу з третіх сторін. Система підтримує широкий спектр мов для розпізнавання – більше 130.

Ibvips — це бібліотека обробки зображень з горизонтальними потоками, орієнтована на попит. У порівнянні з аналогічними бібліотеками, libvips працює

швидко і використовує мало пам’яті. libvips ліцензовано відповідно до LGPL 2.1+.

Він має близько 300 операцій, що охоплюють арифметику, гістограми, згортки, морфологічні операції, частотну фільтрацію, колір, повторну вибірку, статистику та інші. Він підтримує широкий діапазон числових типів, від 8-бітового int до 128-бітного комплексного. Зображення можуть мати будь-яку кількість смуг. Він підтримує широкий спектр форматів зображень, включаючи JPEG, JPEG2000, JPEG-XL, TIFF, PNG, WebP, HEIC, AVIF, FITS, Matlab, OpenEXR, PDF, SVG, HDR, PPM / PGM / PFM, CSV, GIF, Analyze, NIfTI, DeepZoom і OpenSlide. Він також може завантажувати зображення через ImageMagick або GraphicsMagick, дозволяючи йому працювати з такими форматами, як DICOM.

Він поставляється з прив’язками для C, C++ та командного рядка. Повні прив’язки доступні для Ruby, Python, PHP, C# / .NET, Go і Lua. libvips використовується як движок обробки зображень Sharp (на node.js), bimg, Sharp для Go, Ruby on Rails, carrierwave-vips, mediawiki, PhotoFlow та іншими. Офіційним графічним інтерфейсом libvips є nip2, дивна комбінація електронної таблиці та редактора фотографій.

VIGRA – це бібліотека комп’ютерного зору, яка робить основний акцент на гнучких алгоритмах, оскільки алгоритми представляють основне ноу-хау в цій галузі. Отже, бібліотека була побудована з використанням загального програмування, введеного Степановим і Мюссером, і на прикладі бібліотеки стандартних шаблонів C++. Написавши кілька адаптерів (ітераторів зображень і засобів доступу), ви можете використовувати алгоритми VIGRA поверх ваших структур даних у вашому середовищі. Крім того, ви також можете використовувати структури даних, надані в VIGRA, які можна легко адаптувати до широкого кола застосувань. Гнучкість VIGRA надається майже безкоштовно: оскільки дизайн використовує поліморфізм під час компіляції (шаблони), продуктивність скомпільованої програми наближається до традиційного,

налаштованого вручну, негнучкого рішення. DCMTK — це набір бібліотек і програм, що реалізують великі частини стандарту DICOM. Він включає програмне забезпечення для перевірки, створення та перетворення файлів зображень DICOM, роботи з офлайн-медіа, надсилання та отримання зображень через мережеве підключення, а також демонстраційне зберігання зображень і сервери робочих списків. DCMTK написаний на суміші ANSI C і C++. Він поставляється в повному вихідному коді і доступний як програмне забезпечення з відкритим вихідним кодом.

DCMTK використовувався на численних демонстраціях DICOM для забезпечення центрального, незалежного від постачальника серверів зберігання зображень і робочих списків (CTN - Central Test Nodes). Він використовується лікарнями та компаніями по всьому світу для широкого спектру цілей, починаючи від інструменту для тестування продуктів і закінчуючи будівельним блоком для дослідницьких проектів, прототипів та комерційних продуктів.

Програмне забезпечення DCMTK можна зібрати під Windows і широкий спектр операційних систем Unix, включаючи Linux, Solaris, FreeBSD, OpenBSD, MacOS X і NetBSD. Усі необхідні скрипти конфігурації та make-файли проекту надаються.

ITK — це кросплатформна бібліотека з відкритим вихідним кодом, яка надає розробникам широкий набір програмних засобів для аналізу зображень. Розроблена за допомогою екстремальних методологій програмування, ITK спирається на перевірену просторово-орієнтовану архітектуру для обробки, сегментації та реєстрації наукових зображень у двох, трьох чи більше вимірах.У 1999 році Національна медична бібліотека США Національного інституту охорони здоров’я уклала шість трирічних контрактів на розробку інструментарію реєстрації та сегментації з відкритим вихідним кодом, який з часом став відомий як Insight Toolkit (ITK) і склав основу Консорціум програмного забезпечення Insight. Менеджером проекту NIH/NLM ITK був доктор Террі Ю, який координував роботу шести головних підрядників, що

складали консорціум Insight. Ці члени консорціуму включали трьох комерційних партнерів — GE Corporate R&D, Kitware, Inc. і MathSoft — і трьох академічних партнерів — Університету Північної Кароліни (UNC), Університету Теннессі (UT) (Росс Вітакер згодом переїхав до Університету Юти) та Університет Пенсільванії (UPenn). Основними дослідниками для цих партнерів були, відповідно, Білл Лоренсен з GE CRD, Уілл Шредер з Kitware, Вікрам Чалана з Insightful, Стівен Ейлворд з Луїсом Ібаньєсом з UNC, Росс Вітакер з Джошем Кейтсом з UT та Дімітрі Метаксас з UPenn. Крім того, декілька субпідрядників доповнили консорціум, включаючи Пітера Райту з Бригамської і жіночої лікарні, Селіну Імелінську та Пета Молхолта з Колумбійського університету, Джима Джі з лабораторії Grasp Lab UPenn і Джорджа Стеттена з Університету Піттсбурга.

У 2002 році був доступний перший офіційний публічний випуск ITK. Крім того, Національна медична бібліотека уклала тринадцять контрактів з кількома організаціями для розширення можливостей ITK. Протягом багатьох років NLM фінансувала підтримку набору інструментів, і в липні 2010 року було розпочато значне фінансування, яке завершилося випуском ITK 4.0.0 у грудні 2011 року. Спільнота ITK завершила значну модернізацію інтерфейсу інструментарію, вдосконалення архітектури для покращення продуктивності та пакети Python для швидкого створення прототипів із ITK 5.0.0, випущеним у травні 2019 року.

TinyEXIF — це крихітна, легка бібліотека C++ для аналізу метаданих, що існують у файлах JPEG. Для аналізу даних EXIF не потрібні сторонні залежності, однак для доступу до даних XMP потрібна бібліотека TinyXML2. TinyEXIF простий у використанні, просто скопіюйте два вихідні файли у вашому проекті та передайте дані JPEG до класу EXIFInfo. Наразі витягується така поширена інформація, як марка/модель камери, вихідна роздільна здатність, часова мітка, фокусна відстань, інформація про об’єктив, F-зупинка/час експозиції,

інформація GPS тощо, вбудована в метадані EXIF/XMP. Але легко розширити його та додати будь-які відсутні або нові поля EXIF/XMP. OpenEXR забезпечує специфікацію та довідкову реалізацію формату файлу EXR, формату зберігання зображень професійного рівня в кіноіндустрії. Метою формату є точне та ефективне представлення даних лінійного зображення сцени з високим динамічним діапазоном та пов'язаних з ними метаданих із сильною підтримкою багатокомпонентних, багатоканальних випадків використання. Бібліотека широко використовується в програмному забезпеченні хоста, де точність має вирішальне значення, наприклад, фотореалістична візуалізація, доступ до текстур, композитинг зображень, глибокий композитинг і DI.

OpenEXR є проектом Academy Software Foundation і є частиною довідкової платформи VFX. OpenEXR спочатку був розроблений Industrial Light & Magic (ILM) і вперше випущений у 2003 році. Weta Digital, Walt Disney Animation Studios, Sony Pictures Imageworks, Pixar Animation Studios, DreamWorks та інші студії, компанії та окремі особи зробили значний внесок у розвиток. кодова база.

Метою проекту OpenEXR є збереження формату надійним і сучасним і збереження його місця як бажаного формату зображення для створення розважального контенту. Великі редакції нечасті, і нові функції будуть ретельно зважуватися з підвищеною складністю.

Основними пріоритетами проекту є:

- міцність, надійність, безпека;
- зворотна сумісність, довговічність даних;
- продуктивність - час читання/запису/стиснення/декомпресії;
- простота, зручність використання, ремонтопридатність.

Широке поширення, підтримка кількох платформ – Linux, Windows, macOS та інші

OpenEXR призначений виключно для 2D-даних. Він не підходить для зберігання об’ємних даних, кешованих або освітлених тривимірних сцен або більш складних 3D-даних, таких як світлові поля.

Основною цільовою аудиторією OIIO є студії VFX і розробники таких інструментів, як засоби візуалізації, композитори, засоби перегляду та інше програмне забезпечення, пов’язане з зображеннями, яке ви знайдете у виробництві.

OpenImageIO складається з:

Простих але потужних API-інтерфейсів ImageInput і ImageOutput, які забезпечують абстракцію для читання та запису файлів зображень майже будь-якого формату, при цьому програмі, що викликає, не потрібно знати будь-які деталі цих форматів файлів, і навіть без того, щоб програма, що викликає, знати, який доступні формати.

Бібліотека, яка керує підкласами ImageInput і ImageOutput, які реалізують введення/виводу з певних форматів файлів, при цьому реалізація кожного формату файлу зберігається як плагін. Таким чином, програма, яка використовує API OpenImageIO, може читати та записувати будь-який файл зображення, для якого можна знайти плагін під час виконання. Плагіни, що реалізують введення/виводу для кількох популярних форматів файлів зображень, включаючи TIFF, JPEG/JFIF, OpenEXR, PNG, HDR/RGBE, ICO, BMP, Targa, JPEG-2000, RMan Zfile, FITS, DDS, Softimage PIC, PNM, DPX , Cineon, IFF, OpenVDB, Ptex, Photoshop PSD, Wavefront RLA, SGI, WebP, GIF, DICOM, HEIF/HEIC/AVIF, багато форматів цифрових камер «RAW» і різноманітні формати відео (читаються як окремі кадри). Весь час розробляється все більше.

Кілька інструментів для зображень командного рядка на основі цих класів, включаючи oiiotool (перетворення формату командного рядка та обробка зображень), iinfo (друк детальної інформації про зображення), iconvert

(перетворення форматів, типів даних або змінення метаданих), idiff (порівняння зображень), igrep (пошук зображень для відповідних метаданих) та iv (переглядач зображень). Оскільки ці інструменти засновані на ImageInput/ImageOutput, вони працюють з будь-якими форматами зображень, для яких доступні плагіни ImageIO.

Клас ImageCache, який прозоро керує кеш-пам'яттю, щоб він міг отримати доступ до дійсно величезної кількості даних зображень (десятки тисяч файлів зображень на загальну суму кілька ТБ) дуже ефективно, використовуючи лише крихітну кількість (максимум десятки мегабайт) оперативної пам'яті.

Клас TextureSystem, який забезпечує відфільтрований пошук текстур MIP-карти на вершині гарної поведінки кешування ImageCache. Це використовується в комерційних програмах візуалізації, а також у багатьох великих VFX та анімаційних фільмах.

Функції ImageBuf і ImageBufAlgo – простий клас для зберігання та керування цілими зображеннями в пам’яті, а також набір найкорисніших обчислень, які ви можете виконати, включаючи ці зображення, включаючи багато операцій обробки зображень.

Прив’язки Python для всіх основних API.

2.3 Висновки

В даному розділі було проведено аналіз мов програмування. В результаті було прийнято рішення використовувати одну мову для програми – C++, а середовищем розробки обрано MS Visual Studio. Розглянуто бібліотеки методи OpenCV. Використано функціональний (структурний) метод як такий, що максимально задовольняє вимогам.

3 МЕТОДИ ОБРОБКИ ЗОБРАЖЕНЬ ДОРОЖНІХ ЗНАКІВ

3.1 Методи просторового покращення зображень дорожніх знаків

Існуючі підходи вирішення задач покращення цифрового зображення дорожніх знаків та відновлення його структури поділяються на дві категорії:

1) методи обробки в просторовій області (просторові методи), засновані на прямому маніпулюванні пікселями зображення;

2) методи обробки в частотній області (частотні методи) ґрунтуються на модифікації (фільтрації) сигналу, що формується шляхом застосування до зображення перетворення Фур'є.

Просторова обробка застосовується, коли єдиним джерелом викривлень є адитивний шум, що виникає у результаті впливу погодних умов. Частотна фільтрація може використовуватися для нечітких зображень з дефектами освітлення, вона також враховує і шум. Тому така обробка є універсальним і поширеним методом поліпшення якості цифрового зображення дорожніх знаків.

Методика обробки зображень дорожніх знаків в частотній області полягає у представленні його як двовимірної функції f(x,y), де x та y – координати в просторі. Значення f(x,y) в будь-якій точці, що задається парою координат (x, y), називається інтенсивністю або рівнем яскравості у цій точці в колірному просторі.

Загально визнаним є твердження, що будь-яка функція, що періодично повторює свої значення може бути представлена у вигляді суми синусів та косинусів різних частот, помножених на деякі коефіцієнти. Таке представлення функції називається представленням у вигляді ряду Фур’є. Коли функція не є періодичною, проте площа під її графіком є кінцевою, то ми маємо справу з перетворенням Фур’є.

Функція, що задана перетворенням Фур’є, може бути повністю без втрати інформації відновлена за допомогою алгоритму перетворення. Ця властивість є

важливою, оскільки дозволяє працювати в «Фур'є-просторі», а потім повернутися в початкову область визначення функції без втрати інформації.

Оскільки цифрові зображення описуються двовимірними дискретними функціями, то будемо розглядати дискретне перетворення Фур'є (ДПФ) саме для таких функцій.

Нехай *f(x,y),* при *x = 0,1,2..., M-1* і *y = 0,1,2..., N-1,* позначає зображення *M*x*N*. Двовимірне дискретне перетворення Фур'є зображення *f(x,y)*, яке позначається *F(u,v),* задається рівнянням (2.1).

$$F(u,v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) e^{-j2\pi(ux/M + \upsilon y/N)}$$

(2.1)

де, *u=0,1,2...,M-1* і *v=0,1,2...,N-1; M* і *N* парні числа.

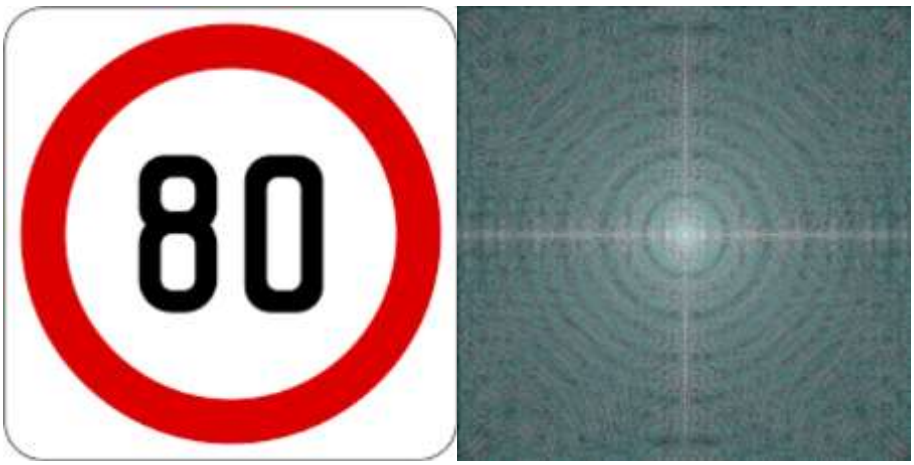


Рисунок 3.1 – Зображення знаку обмеження швидкості і його спектр

Координатна система, задаючи аргументи *F(u,v)* частотними змінними *u* і *v,* називається частотною областю. У даному випадку можна виявити аналогію із завданням аргументів *f(x,y)* просторовими змінними *x* і *y*. Прямокутну область розміру *M*x*N* при *u=0,1,2...,M-1* і *v=0,1,2...,N-1*, прийнято називати частотним прямокутником. Він має ті ж розміри, що і початкове зображення.

Перетворення Фур’є являється комплексним. Основний метод візуального аналізу цього перетворення полягає в обчисленні його спектру (тобто абсолютної величини *F(u,v))* і його відображення на дисплеї. Нехай *R(u,v)* і *I(u,v)* позначають дійсну і уявну компоненти *F(u,v)*, тоді спектр Фур’є задається виразом (2.2).

$$\left| F(u,v) \right| = \left[R^2(u,v) + I^2(u,v) \right]^{1/2}$$

(2.2)

Кожен елемент фур’є-образ *F(u,v)* містить всі відліки функції *f(x,y)*, помножені на значення експоненціальних членів, тому зазвичай неможливо встановити пряму відповідність між характерними деталями зображення і його образом. Проте деякі спільні твердження щодо взаємозв’язку частотних складових фур’є-образу і просторових характеристик зображення можуть бути зроблені. Наприклад, оскільки частота прямо пов’язана із швидкістю зміни сигналу, то інтуїтивно зрозуміло, що частоти у перетворення Фур’є пов’язані з варіацією яскравості на зображенні. Найбільш повільно змінювана (постійна) частотна складова *(u=v=0)* збігається з середньою яскравістю зображення. Низькі частоти, що відповідають точкам поблизу початку координат фур’є-перетворення, відповідають повільно змінним компонентам зображення. На зображенні кімнати, наприклад, вони можуть відповідати плавним змінам яскравості стін і підлоги. По мірі віддалення від початку координат, вищі частоти починають відповідати все більшим змінам яскравості деталей зображення та їх граней.

Алгоритм фільтрації в частотній області складається з таких кроків:

1. Початкове зображення множиться на $(-1)^{x+y}$, відповідно до виразу (2.3). Це робиться для того, щоб його перетворення Фур’є виявилось центрованим, тобто початок координат для образу функції буде знаходитися у центрі частотного прямокутника у точці *(M/2;N/2*);

$$\xi\left[f(x,y)(-1)^{x+y}\right] = F(u - M/2, \nu - N/2) \qquad (2.3)$$

2. Обчислюється пряме ДПФ *F(u, v)* зображення, отриманого після кроку 1;
3. Функція *F(u, v)* множиться на деяку функцію фільтру *H(u, v);*
4. Обчислюється зворотне ДПФ від результату кроку 3;
5. Виділяється потрібна частка результату кроку 4;
6. Результат кроку 5 множиться на $(-1)^{x+y}$.

Причина, по якій множник *H(u,v)* називається фільтром полягає в тому, що він зменшує величину певних частот перетворення, залишаючи при цьому інші майже без зміни. Питання знаходження передаточної функції фільтру і є ключовим, адже воно визначає метод фільтрації і вказує які саме частоти будуть відфільтровуватися [3].

Нехай *f(x,y)* позначає вхідне зображення після кроку 1, а *F(u, v)* є його фур'є-образ. Тоді фур'є-образ вихідного зображення визначається виразом (2.4).

$$G(u,\nu) = H(u,\nu) \cdot F(u,\nu) \qquad (2.4)$$

Множення функцій двох змінних *H* і *F* здійснюється поелементно. Фільтроване зображення отримується обчисленням зворотного перетворення Фур'є від фур'є-образу $F(u,\nu)$ обчислюється за формулою (2.5). Покращене зображення:

$$\xi^{-1}\left[G(u,\nu)\right] \qquad (2.5)$$

Шукане зображення виходить виділенням дійсної частини з останнього результату і множення на $(-1)^{x+y}$, щоб компенсувати ефект від множення вхідного зображення на ту ж величину.

Недоліком і предметом досліджень всіх методів фільтрації у частотній області є неможливість створення ідеального фільтру, що відкидав би всі «зайві» частоти, відновлюючи при цьому якість зображення.

Нелінійні методи фільтрації належать до одного із виду методів обробки зображень по частотній області. Клас нелінійних цифрових фільтрів є дуже широким для того, щоб проводити їх опис в загальному вигляді. Розглянемо одні з найбільш відомих методів із сімейства нелінійних цифрових фільтрів.

При зображень обмеженого розміру виникає гранична проблема обчислення оцінок у точках нульового рядка і нульового стовпчика. Природним рішенням є використання одномірної калмановської фільтрації.

Уолкап і Чоенс запропонували використовувати вінерівську фільтрацію для боротьби із періодичним високочастотним шумом, що може бути викликаний снігом, дощем, градом і т.і.) в моделі системи зображення, що описує формула (6).

$$\tilde{y}_{i,j} = y_{i,j} + \alpha\left[y_{i,j}\right]^{1/3} n_{i,j} \qquad (2.6)$$

де, α – постійна величина.

Для цієї моделі була отримана частотна характеристика фільтра, що реставрує, відповідному випадку безупинного зображення, що описує рівністю (2.7).

$$\mathrm{H}_R(\omega_x,\omega_y) = \frac{W_{F_1}(\omega_x,\omega_y)}{W_{F_1}(\omega_x,\omega_y) + \alpha^2 E\left\{\left[F_1(\omega_x,\omega_y)\right]^{2/3}\right\}} \qquad (2.7)$$

де, $W_{F1}(\varpi_x,\varpi_y)$ – енергетичний спектр ідеального зображення, Е-позначення математичного сподівання.

Цвейг і інші розробили евристичний нелінійний метод реставрації малоконтрастних зображень із метою послаблення шуму. При використанні

цього методу вхідне зображення розгортається з високою роздільною здатністю, а кожний його елемент квантується великим числом рівнів. Потім одержують зображення зниженої чіткості, об’єднуючи елементи у фрагменти, що не перетинаються, розміром 2×2. Звичайно, чітке зображення має більш різкі границі, ніж зображення зі зниженою чіткістю, проте дисперсія шуму останнього виявляється меншою. У випадку білого шуму дисперсія нечіткого зображення в чотири рази менша, ніж для чіткого зображення, що є наслідком просторового усереднення елементів. Усереднене зображення повторно квантується з використанням рівномірної шкали, причому крок квантування вибирається рівним 4х значенню СКВ шуму. Завдяки такому вибору забезпечується помилка квантування 5% при гаусовом шумі. Отримані квантовані елементи нечіткого зображення досліджують в окрузі розміром 3×3 елементів. Може виявитися, що усі вісім периферійних елементів проквантовані з тим самим рівнем, а центральний елемент – з іншим рівнем. У цьому випадку вважають, що ізольований центральний елемент містить помилку, обумовлену шумом, і приписують йому середній рівень периферійних елементів. Простий алгоритм полягає в тому, що елементу, який відповідає високій роздільній здатності, приписують рівень одного з чотирьох пов'язаних.

3.2 Методи адаптитивної бінаризації для покращення зображень дорожніх знаків

В даний час системи комп'ютерного зору досить широко і успішно використовуються в різних областях для таких операцій, як контроль та відстеження різних об'єктів, їх класифікації на вигляд, розпізнавання складових об'єктів, в системах OCR і в багатьох інших систем.

Одним із основних методів обробки зображень у системах комп'ютерного зору є бінаризація. Під бінаризацією мають на увазі виділення об'єкта, що

містить інформацію про його форму, для подальшого її аналізу та вирішення завдання сегментації.

На відміну від представлення границь об'єкта результат бінаризації передається у вигляді двомірної матриці з розмірами, рівними вихідному зображенню. Даний вид представлення результату процесу бінаризації є інтуїтивно зрозумілим і може бути перетворений на заданий вид представлення операціями над матрицею. Таким чином, завдання бінаризації зводиться до перетворення кольорового, включаючи градації сірого, растрового зображення в чорнобіле зображення.

При розв'язанні задачі сегментації об'єктів велику важливість грає вибір алгоритму бінаризації, так як результати роботи алгоритму бінаризації використовуються в алгоритмах маркування пов'язаних областей, що є невід'ємною частиною розв'язання задач сегментації. Вибір алгоритму бінаризації впливає на алгоритми, що застосовуються для аналізу зображення. Правильний вибір алгоритму бінаризації дозволить використовувати більш прості та ефективні топології та класифікації, швидші методи сегментації та суттєво вплине на точність результату аналізу зображення.

Виділяють такі види алгоритмів бінаризації зображення:

- адаптивна бінаризація;
- обробка з постійним порогом;
- методи бінаризації на основі гістограм (як правило, гістограми містять функцію яскравості від кількості точок з відповідною яскравістю);
- на основі нейронних мереж.

Методи, що лежать в основі адаптивної бінаризації та використовуються на практиці:

- метод Саувола;
- метод Бернсен;
- метод Ніблека;
- метод порогу Гатоса;

- метод максимальної ентропії.

Метод Саувола. Складаються два допоміжні «інтегральні» зображення. Перше є результатом обробки квадратною апертурою всього зображення, де результуюче зображення виходить як середньоарифметичне значення яскравостей точок у квадратній апертурі. Друге зображення створюється подібним чином, але на відміну середньоарифметичного значення будується сума квадратів яскравостей точок в апертурі, і від цього значення віднімається квадрат середньоарифметичного значення аналогічної апертури з першого зображення. В основному циклі проводиться аналіз точок, якщо колір точки менше глобального мінімуму (заданого початку роботи алгоритму), то результатом бінаризації даної точки є білий, інакше - чорний. Якщо колір точки залишився між глобальними порогами, відбувається робота наступного алгоритму: для цієї точки знаходяться відповідні значення першого і другого допоміжних зображень, і відповідно до формули обчислюється поріг для поточної точки. На основі вибраного порогу для цієї точки проводиться бінаризація.

Метод Бернсен полягає у виборі квадратної апертури, як правило, з непарним числом точок і, починаючи з лівого верхнього кута, пересувається по всьому зображенню. На кожному кроці в апертурі знаходиться значення з мінімальною яскравістю точки (позначається як Min) і з максимальним значенням яскравості точки (позначається як Max). Знаходиться середнє Avg = (Min + Max) /2. Для кожної точки зображення обчислюється яскравість, і якщо яскравість точки більша за відповідне значення Avg плюс E (деяка константа, задана користувачем), то результатом бінаризації цієї точки є чорний, інакше – білий. Однак якщо Avg менший за поріг контрасту, то поточний піксель стає того кольору, який був обраний для кольору «сумнівного пікселя».

Метод Ніблека робота даного методу полягає у варіюванні порога яскравості від точки до точки на підставі локального значення стандартного відхилення. Поріг яскравості в точці (x, y) розраховується так: B(x,y) = μ(x,y) +

k*s(x,y), де μ(x, y) – середнє, а s(x, y) – СКВ вибірки для деякої околиці точки. Розмір апертури повинен бути вибраний так, щоб зберегти локальні деталі зображення і одночасно знизити вплив шуму. Значення k визначає, яку частину межі об'єкта взяти як самий об'єкт.

Метод порогу Гатоса базується на використанні двох допоміжних зображень: перше зображення є бінаризованим за алгоритмом Ніблека і служить для визначення фону та розташування об'єктів; друге зображення будується з використанням першого за принципом ідентичності оригінальному, зображені всі точки, а відповідні точки зі значенням білого з першого зображення замінюються на інтерполовані значення кольору сусідніх точок. У результаті виходить фонове зображення. Вибирається кожна точка вихідного зображення, і якщо різниця між кольором точки та кольором відповідної точки з фонового зображення більша за поріг, що обчислюється за певною формулою, то результатом бінаризації даної точки є білий, а в іншому випадку – чорний.

Метод максимальної ентропії базується на використанні гістограми на основі яскравостей точок зображення та кількості точок з певною яскравістю. Гістограма нормалізується, і будується кумулятивна гістограма, для кожного значення якої будуються з використанням нормалізованої гістограми ще дві гістограми: по ентропії чорного кольору та по ентропії білого кольору. На основі гістограм, побудованих за ентропіями чорного та білого кольору, визначається значення максимальної суми [5], і вибирається відповідний поріг бінаризації. Далі відбувається порогова бінаризація для знайденого значення порога (див. рис. 3.2).

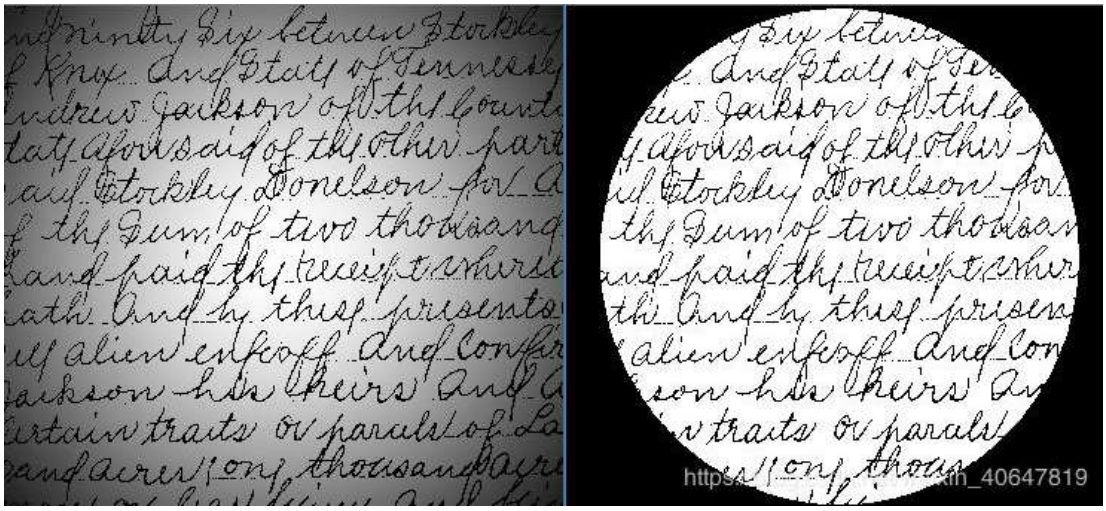


Рисунок 3.2 – Бінарізація методом максимальної ентропії

Метод адаптивного порога полягає не у обчисленні порога глобального зображення, а у обчисленні локального порогу відповідно до розподілу яскравості різних областей зображення. Тому для різних областей зображення різні пороги можуть обчислюватися адаптивно. Для обчислення локального порогу обчислюється середнє значення, медіанне значення, середньозважене значення за Гаусом (фільтрація за Гаусом) для окола з визначення порога. У випадку коли локальний поріг використовується локальне середнє значення, його часто називають методом ковзного.

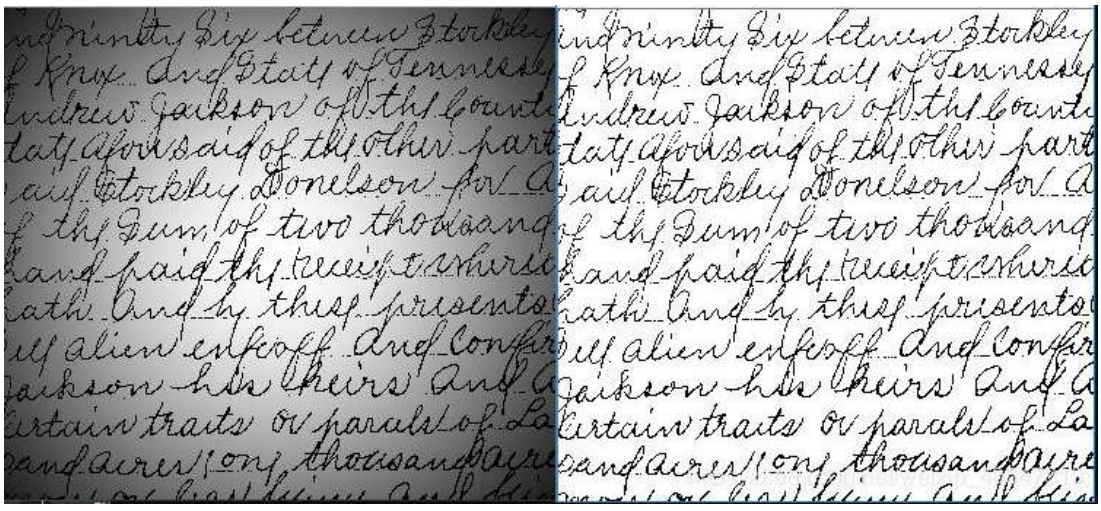


Рисунок 3.3 – Бінарізація методом адаптивного порога

3.3 Розробка основних програмних елементів модулю розпізнавання дорожніх знаків

Використані класи або елементи зв’язуємо в одну систему з зображених на рисунку 2.1. Діаграма показує, вміст класів, елементи, методи, змінні. Після отримання всієї інформації від одних класів та алгоритму, формується результат роботи програмного модулю. В діаграмі представлені основні класи, що використовуються для розпізнавання та класифікації дорожніх знаків на зображеннях та відео.

В файлі main.cpp йде підлючення основних бібліотек та об'єднання хедер файлів, можна побачити на рисунку 3.2. Хедер файли, включені в програму за допомогою директиви #include, рекурсивно проходять стадію препроцессингу і включаються у файл, що випускається. Проте, кожен хедер може бути відкритий під час припроцессингу кілька разів, тому зазвичай використовуються спеціальні припроцесорні директиви, що оберігають від циклічної залежності.

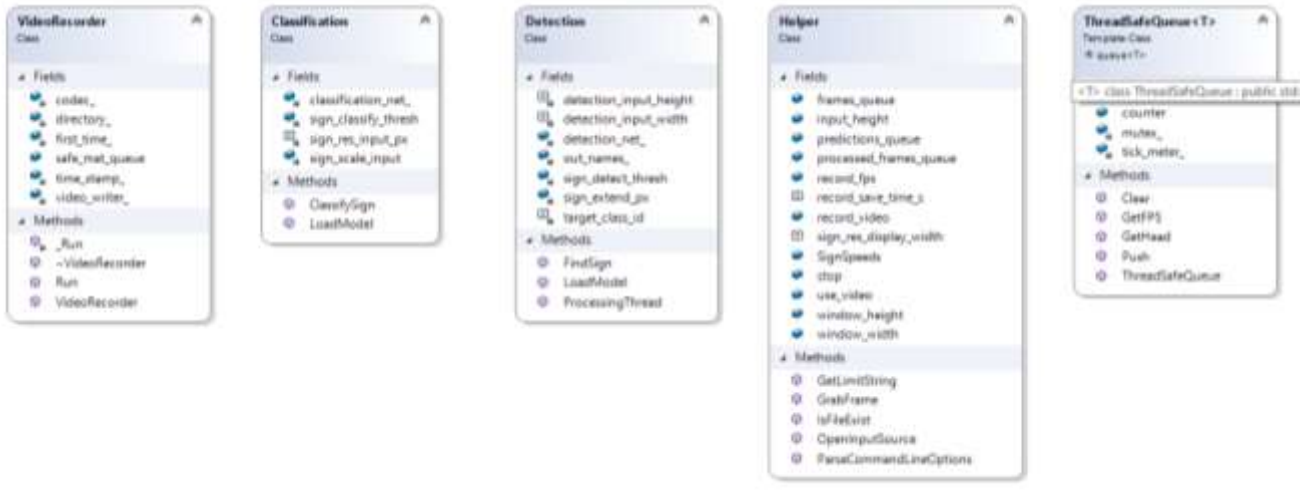


Рисунок 3.4 - Діаграма класів

```
#include <opencv2/imgproc.hpp>
#include <opencv2/highgui.hpp>
#include <opencv2/core/utils/logger.hpp>

#include <thread>
#include <queue>
#include <iostream>

#include "safe_queue.h"
#include "helper.h"
#include "detection.h"
#include "classification.h"
#include "video_recorder.h"
```

Рисунок 3.5 – Підключення основних бібліотек та хедер файлів

Далі створюється вікно в якому вказані параметри руху, ілюструється зображення та відбувається аналіз отриманих даних.

Клас ThreadSafeQueue є реалізацією структури даних черги з імплементацією безпечної роботи з потоками. Тобто реалізована черга дозволяє одночасне звернення до себе декількох потоків модулю з уникненням таких ситуацій як стану гонитви даних, повернення застарілих даних і забезпечує атомарністб операцій.

```
class ThreadSafeQueue : public std::queue<T> {
public:
	ThreadSafeQueue();
	void Push(const T& to_push);
	T GetHead();
	float GetFPS();
	void Clear();
	unsigned int counter;
private:
	cv::TickMeter tick_meter_;
	std::mutex mutex_;};
```

Рисунок 3.6 – Опис класу ThreadSafeQueue

Клас Helper містить допоміжні змінні та функції-утиліти, що використовуються компонентами модулю. Собою являє сховище корисних

функцій, збереження всіх файлів та імпортувати в текст програми для подальшого імпортування та використання.

```
class Helper {
	public:
		static bool use_video;
		static bool record_video;

		static const int sign_res_display_width = 128;
		constexpr static float input_height = 440.0f;
		constexpr static float window_width = 800.0f;
		constexpr static float window_height = 480.0f;
		static const int record_save_time_s = 60;
		static int record_fps;
		static ThreadSafeQueue<Mat> frames_queue;
		static ThreadSafeQueue<Mat> processed_frames_queue;
		static ThreadSafeQueue<std::vector<Mat>> predictions_queue;
		static volatile sig_atomic_t stop;
```

Рисунок 3.7 - Опис класу Helper

Клас Detection виконує обробку зображення та знаходить на ньому області в яких розміщені дорожні знаки. Обробка виконується за допомогою вже навченої моделі комп'ютерного зору. Отримані області зображення класифікуються за допомогою класу Classification.

```
	class Detection {
		static Net detection_net_;
		static std::vector<String> out_names_;
		static const int target_class_id = 0;
		static const int detection_input_width = 300;
		static const int detection_input_height = 300;
		constexpr static float sign_detect_thresh = 0.90f;
		constexpr static float sign_extend_px = 0.008f;

	public:
		static bool LoadModel();
		static bool FindSign(Mat& frame,
const std::vector<Mat>& outs, Mat& sign_img);
static void ProcessingThread();};
```

Рисунок 3.8 - Опис класу Detection

Клас Classification відповідає за класифікацію дорожніх знаків.

```
class Classification {
```

```
	static Net classification_net_;
	constexpr static float sign_classify_thresh = 0.99f;
	constexpr static float sign_scale_input = 0.003921f;
	static const int sign_res_input_px = 32;
public:
	static bool LoadModel();
	static void ClassifySign(const Mat& sign_classify, Mat& sign_img,
std::string& limit_str);};
```

Рисунок 3.9 - Опис класу Classification

Клас VideoRecorder використовується для перетворення відео-потоку в окремі зображення для їхньої подальшої обробки та класифікації.

```
class VideoRecorder {
	std::string directory_;
	int codec_ = VideoWriter::fourcc('M', 'J', 'P', 'G');
	bool first_time_ = true;
	std::string time_stamp_;
	VideoWriter video_writer_;

	void _Run(const Mat& m);

public:
	ThreadSafeQueue<Mat> safe_mat_queue;

	VideoRecorder(std::string directory);
	~VideoRecorder() = default;
	void Run(); };
```

Рисунок 3.10 – Опис класу VideoRecorder

На рисунку 3.11 зображено початок обробки кадру отриманого з відеопотоку. Якщо кадр не було отримано, то завершаємо обробку. Коли ми маємо джерело, ми розміщуємо зображення в чергу.

```
/* Захоплюємо кадр */
if (!Helper::GrabFrame(video_capture, frame)){
	break;
}
```

Рисунок 3.11 – Перевірка захоплення кадру

На рисунку 3.12 Якщо черга кадрів для обробки не пуста, то беремо кадр з черги. Покращуємо зображення отриманого кадру та розпізнаємо дорожні

знаки. Якщо було знайдено хоча б один дорожній знак, то проводимо класифікацію для кожного із розпізнаних на попередньому кроці знаків.

```
if (!Helper::predictions_queue.empty()) {
                    /* Перевіряємо чи було успішно виявлено знак */
                    std::vector<Mat> outs = Helper::predictions_queue.GetHead();
                    Mat pred_frame = Helper::processed_frames_queue.GetHead();
                    Mat sign_classify;
                    Detection::FindSign(pred_frame, outs, sign_classify);

                    /* Робимо класифікацію  */
                    if (!sign_classify.empty()) {
                          Classification::ClassifySign(sign_classify, sign_img,
limit_str);}

                    detection_fps_str = format("Inference: %d FPS",
int(Helper::predictions_queue.GetFPS()));}
```

Рисунок 3.12 – Розпізнавання знаків

Після завершення розпізнавання та класифікації кадру формуємо візуальне представлення результатів. З початкового зображення вирізаються області на яких було розпізнано дорожній знак та виводяться користувачу з відповідною назвою, отриманою за допомогою класифікатора.

```
Rect roi_sign(Point(0, frame.size().height - Helper::sign_res_display_width),
sign_img.size());
            sign_img.copyTo(frame(roi_sign));
            int x_add = int((Helper::window_width - frame.size().width) / 2);
            Rect roi_frame(Point(x_add, 0), frame.size());
            Mat display_frame(int(Helper::window_height),
int(Helper::window_width), CV_8UC3, Scalar::all(255));
            frame.copyTo(display_frame(roi_frame));
                  input_fps_str       =       format("Video:       %d       FPS",
int(Helper::frames_queue.GetFPS()));
```

Рисунок 3.13 – Візуалізація результатів

Якщо черга кадрів для обробки пуста, то завершити виконання роботи модулю, інакше повторити попередні кроки для наступного кадру в черзі.

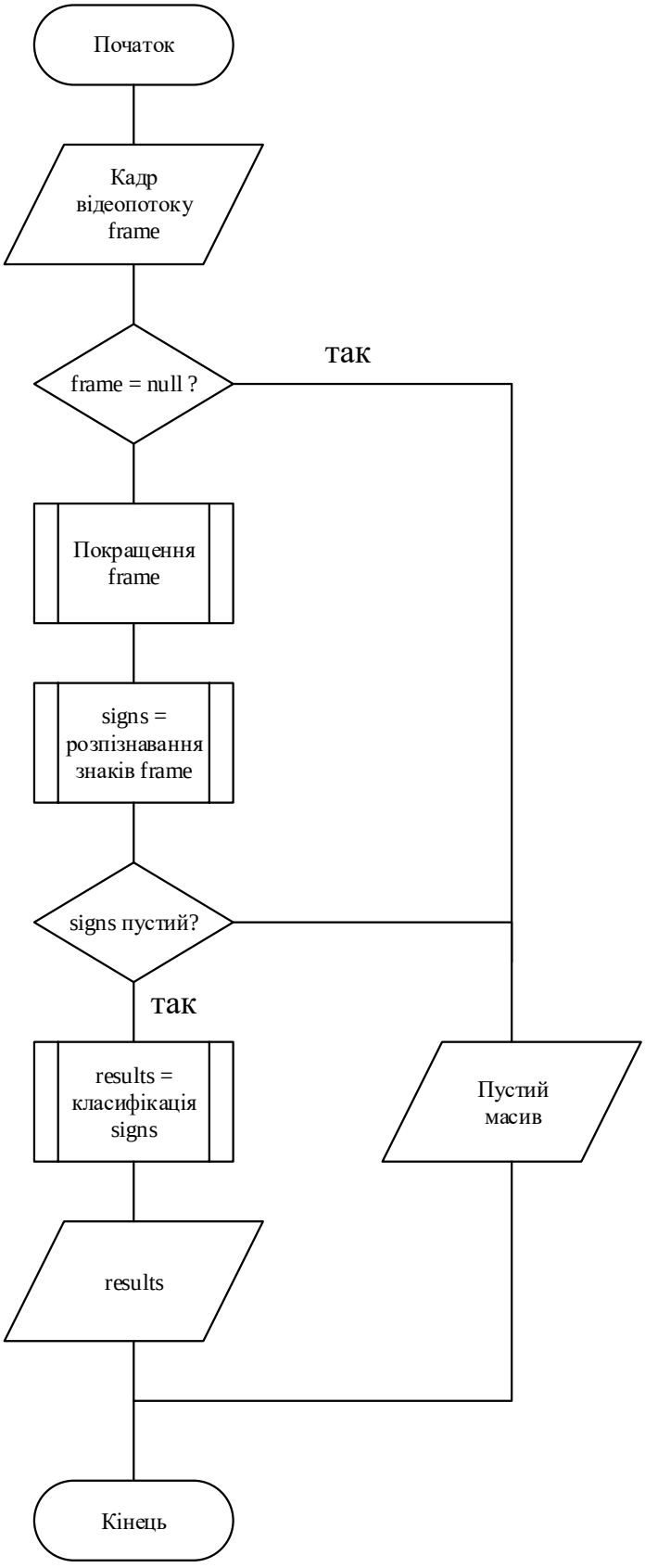


Рисунок 3.14 – Блок-схема програмного модулю

3.4 Висновки

В даному розділі, було проаналізовано та розглянуто декілька популярних мов програмування, а саме C, C++, Java, Python, Swift. Було наведено плюси та мінуси цих мов. Серед всіх найкращою для досягнення поставлених цілей в даній магістерській кваліфікаційній роботі було обрано мову програмування C++. Також було проаналізована середовища програмування для мови програмування Python, а саме Visual Studio, Eclipse,Visual studio Code, Sublime text та PyCharm. Серед яких було обрано середовище програмування Visual Studio, тому що воно є зручним та ефективним для досягнення поставлених цілей.

Обгрунтовано вибір бібліотек та фреймворків для програмного модуля. А саме, обрано бібліотеку, котра містить в собі більше 150 фреймворків, що суттєво покращують можливості та роботу c++, а саме Boost та OpenCV (зручна та необхідна бібліотека для роботи з зображеннями, має близько 3000 оптимізованих алгоритмів, які включають в себе насичений набір класичних і сучасних алгоритмів машинного навчання та комп'ютерного зору). Наведенно декілька прикладів бібліотек, що входять до складу Boost, а саме: Boost Interprocess, Boost Pool та інші.

Описано основні класи роботи програмного модулю, а саме VideoRecorder – що відповідає за запис відео; Classification – відповідає за визначення знаку; Detection – що виконує обробку зображень та знаходження знаків; ThreadSafeQueue – який відповідає за черги з багатопотоковою безпекою та клас Helper котрий містить в собі набір основні операції.

Розроблено діаграму програмного модулю в якій представлено основні класи програми.

Виконано розробку програмного модулю розпізнавання дорожніх знаків та продемонстровано декілька програмних елементів.

4 ТЕСТУВАННЯ

4.1 Аналіз методів тестування

Приступаючи до тестування програми розглянемо, що собою являє тестування.

Тестування програмного забезпечення - це процес, що використовується для виміру якості розроблюваного програмного забезпечення або процес технічного дослідження, спрямований на виявлення та отримання інформації про якість продукту відносно сфери, в якій він має використовуватись. Тестування розглядає коректність роботи інтерфейсу та його зв’язку з базою даних. Також коректність виведених даних, щоб текст та пояснення були зрозумілими для користувача.

Швидкість і ефективність розробки ПЗ залежить від того наскількипроцес тестування вписується в загальний життєвий цикл розробки ПЗ і від ефективності використання технології тестування.

Тестування - одна з функцій перевірки роботи та якості , що включає в себе чітке планування робіт (TestManagement), проектуванню тестів (TestDesign), виконанню тестування (TestExecution) і аналізу отриманих результатів (TestAnalysis). План тестування включає в себе інформацію всього обсягу тестування, це план в якому виділяються основні пункти перевірки, починаючи з планування обсягу роботи та закінчуючи потрібним обладнанням. Проектування тестів – це моделювання успішного та неуспішного виконання завдань, які проводяться відповідно до установлених критеріїв контролю та якості. Під час виконання аналізу тестування підбиваються підсумки. Необхідними умовами для тестування є наявність :

- що тестувати, а саме об’єкт доступний для проведення іспитів;
- ті хто буде це виконувати (залежно від виду проведених іспитів це може бути програма, людина, або машина чи їх комбінації).

Серед методів тестування виділяють: статичне тестування, динамічне тестування, метод «білої скриньки», метод «чорної скриньки».

Діяльність пов’язана з обробкою та аналізом результатів розробки ПЗ є статичним тестуванням. Тут передбачається перевірка програм, програмного коду та перевірка продукту без запуску на операційній системі. Включає в себе повну перевірку всієї документації, що була отримана поки існувала робота над ПЗ. Звичайно, все перевіряється по стандартам ведення документації, стандартам програмування. Як результат отримуємо продукт високої якості, який був перевірений по всім критеріям та порівняний з заданим ТЗ. Динамічне тестування це вже експлуатація готового продукту, безпосереднє виконання програми. Коректність перевіряється засобом безлічі тестувань, введенням набор готових вхідних даних.

Тестування «білої скриньки» полягає в вивченні внутрішньої поведінки програми. До перевірки підлягає всі внутрішні частини програми та перевіряється коректність їх взаємодії між собою. До переваг даного тестування відносяться: можливість точного виявлення недоліків та проблем в основі чи периферії програми; іноді результат тесту залежить від внутрішніх станів програми. Серед недоліків варто відзначити маштабність маршрутів та велику ймовірність пропуску шляхів, через велику масивність тесту.

Тестування «чорної скриньки» відбувається з самим інтерфейсом програми. Повністю ігнорується внутрішня будова програми, головним є швидкість та коректність роботи програми. Особливість методу дозволяє виконати повне тестування та отримати максимум аналітичної інформації по роботі програми, працюючи лише з комбінаціями вхідних та вихідних данних. До плюсів методу можна віднести швидкість, простоту та наглядність методу. Мінусом є низький полігон перевірки, тестування цим методом не допоможе знайти проблеми, які лежать в основі програми.

В залежності від потреб тестування процес слід організовувати відповідним чином. Отже, можна виділити 4 види тестування ПЗ:

- Функціональне тестування (Functional testing).
- Нефункціональне тестування (Non-functional testing).
- Структурне тестування (Structural testing).
- Тестування змін (Change related testing).

Функціональне тестування. Досить важливий вклад створює можливість функціонального тестування, саме ця перевірка спрямована на тестування всього основного функціоналу системи для підтвердження, що кожна функція програми працює відповідно до ТЗ.

Елементи функціонального тестування:

- Перевірка тест-кейсів.
- отримання результату на основі специфікації;
- підготовка тестових даних виходячи з описаної документації;

Функціональне тестування проводиться відповідно до специфікації, а також на основі бізнес-плану, тобто до даних що є в системі.

До переваги функціонального тестування можна віднести:

- в межах тестування виконується імітація безпосереднього використання системи;
- умови тестування, в основному, є близькими до реальних.

Серед недоліків недоліки:

- є значна ймовірність не помітит кілька помилок логіки ПЗ під час повної перевірки функціоналу програми.

Нефункціональне тестування. В основі тестування лежить перевірка якості та парцездатності розробленої системи. Цей тип тестування направлено на перевірку тсаме аспектів ПЗ, які можуть бути описані в документації, але можуть бути не віднесені до повних функцій програмних продуктів. Серед основних методів: інсталяційне тестування, конфігураційне та тестування продуктивності.

Деструктивне тестування. Набір спроб привести розроблювану

систему до збою. В основі перевірки лежить перевірка ПЗ під час не правильного використування, введення некоректних даних.

Тестування швидкодії. Як зрозуміло з назви, перевірка швидкості системи, її частин при цьому виконуючи певне навантаження. Серед основних напрямів тестування швидкодії виділяють стрес та стабільність.

Тестування зручності використання. Створено з метою визначення зручності користування ПЗ для його майбутнього застосування. Цей метод заснований на залученні користувачів та їх оцінки зручності продукту у використанні.

Тестування що виконується на функціональному рівні чи інших певних розділів коду називають – модульним. Цей методі більш відноситься до стилю «білої скриньки», виконується розробником в момент написання для перевірки правильності написаного коду чи працює він як планувалось. Для перегляду повної кількості випадків користування коду, до однієї функції можна створити декілька тестів. Основна мета – це гарантія коректної роботи основних блоків програмного забезпечення.

Системне тестування. Тестування інтегрованої системи перевірки відповідності до всіх вимог. Також, системне тестування ПЗ повинно давати гарантії, що програма буде працювти так, як очікується, а також, що її не можна зруйнувати чи пошкодити її робоче середовище, що призведе процеси в цьому середовищі, що зроблять систему не робочою. Системне інтеграційне тестування перевіряє, чи може система інтегруєтися в іншу зовнішню систему відповідно до системних вимог.

Досить популярним є альфа-тестування — що являє собою, уподібнення роботи системи зі розробниками, котрі займаються тестуванням та керують процесом. Частіше за все альфа-тестування може проводитися в ранніх стадіях розробки продукту, але у певних випадках, має можливість бути застосованим для закінченого продукту, як внутрішнього тестування. Нерідко альфа-тестування виконується з використанням середовища, це допомагає швидко

виявляти помилки. Знайдені неточності можуть бути передані в відділ тестування для повного додаткового дослідження серед умов в яких буде використовуватися програма.

Бета-тестування — популярний вид тестування, в більшості випадків відбувається поширення не повної версії ПЗ з деякими обмеженнями, на приклад скорочений функціонал чи розміри для спеціальної групи осіб, з тим щоб переконатися, що продукт містить достатньо мало помилок. Також бета-тестування може виконуватися для того, щоб отримати прибуток, якщо продукт знаходиться на хвилі популярності чи має достатню рекламованість[32].

4.2 Тестування системи

Система тестувалася в світлий та вечірній час доби на смартфоні Xiaomi Redmi 4 Prime/Pro. Підраховано наступні дані: кількість знайдених знаків кількість помилкових спрацьовувань на етапі локалізації, кількість вірно класифікованих, кількість незнайдених знаків, набір невірно класифікованих. Всього на шляху проходження було 15 знаків, тільки знаки, присутні в навчальній базі. Результати наведені у таблиці 3.1 та 3.2.

Таблиця 3.1 – Результати тестування

Точність локалізації	Не виявлені	Хибні	Точність	Невірно класифіковані
15	1	1	90%	3

Щоб продемонструвати результати було порівняно декілька систем, а саме: OpelEye, RoadAR, Speed limit assist, Road sign information (таблиця 3.2).

Таблиця 3.2 – Порівняння систем

	OpelEye	RoadAR	Speed limit assist	Road sign information	Розроблений алгоритм
Заявлена точність розпізнавання	90%	95%	95%	96%	93%
Розпізнавання знаків обмеження швидкості	+	+	+	+	+
Розпізнавання інших забороняють знаків	+	+	+	+	+
Реальна точність розпізнавання	85%	75%	70%	75%	93%

З таблиці можемо зробити висновки, що подібних програмних продуктів дуже мало. В вільному доступі майже не існує аналогів, більшість зразків опираються на GPS систему і дуже часто не показують знаки чи взагалі попереджують про неіснуючі знаки. В основному більшість технологій та систем є вбудованими в транспортні засоби за заводськими опціями виробників автомобілів. Таким чином, комерційна вартість є дуже високою. Майже всі подібні представники мають високий відсоток точності розпізнавання, але він може впасти, ще й дуже сильно при нахилі, частковому перекритті або забрудненості знаку, його викривлені, так як зображення зазнають проектні та афінні спотворення. При випробуваннях надійно розпізнаються лише чисті знаки високої контрастності, але й розпізнаються не зовсім ідеальні, але відсоток є не високим.

Так як класифікатори працюють із зображеннями, отриманими безпосередньо з камери, то якщо об'єкт не буде знайдений на одному кадрі, він може бути знайдений на наступному, саме завдяки цьому дана точність прийнятна. Середній час, котрий потрібен одному каскаду для обробки зображення не зовсім якісного розміру 1280 × 720 на смартфоні становить 230-

550 мс. Зважаючи на те, що обчислення розглядаються на мобільній та портативній платформі, зазначений час задовільний, проте для роботи в режимі реального часу цього недостатньо. Тому було можливо оптимізувати даний метод уточненням зони інтересів, це дозволило скоротити до 210-290 мс.

Рисунок 4.1 – Зображення обмежуючого знаку із засвітленням

Рисунок 4.2 – Зображення обмежуючого знаку у хмарну погоду

Рисунок 4.3 – Зображення обмежуючого знаку із тінню

4.3 Висновки

Було розглянено що таке тестування, як воно відбувається і від яких речей залежить. Також необхідними умовами для тестування є наявність: об’єкту дослідження та наглядача (перевіряючого).

Виконано тестування програмного модулю в різних режимаї роботи (онлайн та офлайн) та за різних погодніх умов. Тестування показало коректну роботу усіх складових розроблюваного програмного додатку. Зроблено порівняльну характеристику, де було підтвердженно якість та ефективність розробленого модулю.

5 ЕКОНОМІЧНА ЧАСТИНА

5.1 Оцінювання комерційного потенціалу розробки.

Метою проведення технологічного аудиту є оцінювання комерційного потенціалу розробки, створеної в результаті науково-технічної діяльності.

Результатом магістерської кваліфікаційної роботи «Концепція та засоби побудови програмно-навігаційної системи контролю руху транспортного засобу» є модуль розпізнавання дорожніх знаків.

Для проведення технологічного аудиту залучено трьох незалежних експертів: Рейда О.М. (к.т.н., доцент каф. ПЗ ВНТУ), Катєльніков Д.І. (к.т.н., доцент каф. ПЗ ВНТУ) та Майданюк В.П. (к.т.н., доцент каф. ПЗ ВНТУ).

Оцінювання комерційного потенціалу буде здійснене за критеріями, що наведені в таблиці 5.1.

Таблиця 5.1 - Критерії оцінювання комерційного потенціалу розробки бальна оцінка

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Кри-терії.	0	1	2	3	4
Технічна здійсненність концепції:					
	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність продукту в реальних умовах

Продовження таблиці 5.1

Ринкові переваги (недоліки):					
	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Критерії	0	1	2	3	4
	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів

Продовження таблиці 5.1

	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					

Продовження таблиці 5.1

	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування

Продовження таблиці 5.1

0	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
1	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років

Продовження таблиці 5.1

2	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів	Необхідно отримання великої кількості дозвільних документів, що вимагає значних коштів та часу	Процедура отримання дозвільних документів у вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання комерційного потенціалу експертами розробки зведено в таблицю 5.2.

Таблиця 5.2 - Результати оцінювання комерційного потенціалу розробки

Критерії	Прізвище, ініціали, посада експерта		
	Іванчук	Катєльніков	Черноволик
	Бали, виставлені експертами:		
1	3	3	4
Ринкові переваги (недоліки):			
2	3	3	2
3	3	4	3
4	3	3	4
5	3	4	3
Ринкові перспективи			
6	3	2	4
7	3	3	3

Продовження таблиці 5.2

Практична здійсненність			
8	4	3	4
9	3	4	3
10	4	4	4
11	4	4	3
12	4	3	3
Сума балів	$СБ_1$=40	$СБ_2$=40	$СБ_3$=40
Середньоарифметична сума балів $\overline{СБ}$	40		

За даними таблиці 5.3 можна зробити висновок, щодо рівня комерційного потенціалу розробки. Зважимо на результат й порівняємо його з рівнями комерційного потенціалу розробки, що представлено в таблиці 5.3.

Таблиця 5.3 – Рівні комерційного потенціалу розробки

Середньоарифметична сума балів $\overline{СБ}$, розрахована на основі висновків експертів	Рівень комерційного потенціалу розробки
0 – 10	Низький
11 – 20	Нижче середнього
21 – 30	Середній
31 – 40	Вище середнього
41 – 48	Високий

Рівень комерційного потенціалу розробки, становить 40 балів, що відповідає рівню «вище середнього».

В першому розділі роботи були розглянуті переваги та недоліки таких аналогів, як: RoadAsist, ARRoad.

В якості аналога для розробки було обрано aLoader.

Основними недоліками аналога є: версія програми доступна тільки для ПК з ОС Windows або MacOS. Також до недоліків можна віднести те, що в аналозі не реалізований сканер штрих- та QR-кодів.

У таблиці 5.4 наведені основні технічні показники аналога і нового програмного продукту.

Виходячи з результатів отриманих в таблиці 5.4 слід зауважити те, що розроблений програмний продукт випереджає аналог майже по всіх критеріях, що підтверджує його конкурентоспроможність на ринку програмного забезпечення.

Розробка є модифікацією існуючих розробок даної категорії з якісно новим функціональним наповненням. Розробка технічно готова на 100%. Проведене тестування програми довело повну працездатність даного програмного продукту. Розроблено інструкцію користувача.

Таблиця 5.4 - Основні технічні показники аналога і нового програмного продукту

Параметри	Аналог, %	Розробка, %	Примітка
Функціональність	60	85	Переваги у розробки
Мобільність	50	100	Переваги у розробки
Сумісність	70	100	Переваги у розробки
Зручність використання	80	90	Переваги у розробки
Дружність інтерфейсу	70	90	Переваги у розробки

Запропонована програмна реалізація будуть корисним для компаній, особливо за умов, що їх діяльність пов’язана галуззю пожежної безпеки.

На ринку праці наявні фахівці відповідної кваліфікації для обслуговування та підтримки програмного продукту, регламентні обмеження відсутні і немає необхідності отримання дозвільних документів.

Комерціалізація розробки – інвестором проекту є компанія яка виготовляє прилади пожежної безпеки, розроблений додаток є суміжним програмним продуктом із окремими приймально-контрольними пожежними приладами.

Дохід від продукту може бути реалізований у залученні рекламодавців у розміщенні таргетингової реклами.

5.2 Прогнозування витрат на виконання науково-дослідної та конструкторсько –технологічної роботи

Прогнозування витрат на виконання науково-дослідної, дослідно-конструкторської та конструкторсько-технологічної роботи може складатися з таких етапів:

1. Розрахунок витрат, які безпосередньо стосуються виконавців даного розділу роботи.
2. Розрахунок загальних витрат на виконання даної роботи.
3. Прогнозування загальних витрат на виконання та впровадження результатів даної роботи.
4. Розрахунок витрат, які безпосередньо стосуються виконавців даного розділу роботи.

Виконаємо розрахунок витрат, які безпосередньо стосуються виконавця даного розділу роботи, приймаючи до уваги те, що для розробки програми було залучено одного розробника.

Основна заробітна плата розробника (дослідника) $З_о$:

$$З_о = \frac{М}{Т_р} \cdot t, [грн], \quad (5.1)$$

де М - місячний посадовий оклад конкретного розробника, М = 17000 грн;

$Т_р$ – кількість робочих днів у місяці, $Т_р$ = 21 день;

t - число днів роботи розробника, t = 63 днів.

$$З_о = \frac{17000,00}{21} \cdot 63 = 51000,00 \text{ (грн).}$$

Результати розрахунків зведемо до таблиці 5.5.

Таблиця 5.5 – Основна заробітна плата розробників

Найменування посади	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату, грн.
Розробник	17000,00	809,50	63	51000,00
Керівник проекту	19000,00	904,70	3	2714,30
Всього				53744,30

Додаткова заробітна плата розраховується як 12 % від суми основної заробітної плати за формулою:

$$З_д = 0,12 \cdot З_о \text{ [грн].} \quad (5.2)$$

$$З_д = 0,12 \cdot 53744 = 6445,70 \text{(грн).}$$

Нарахування (ЄСВ) на заробітну плату становлять 22%:

$$H_{зп} = (З_о + З_р) \cdot \frac{\beta}{100}, \text{ [грн],} \quad (5.3)$$

де Зо – основна заробітна плата розробників, грн;

Зд – додаткова заробітна плата всіх розробників, грн;

β – ставка єдиного соціального внеску на загальнообов'язкове державне соціальне страхування, %.

$$Н_{зп} = (53714,30+6445,70) \cdot 0,22 = 13235,20 \text{ (грн)}.$$

Амортизація обладнання та приміщення, яке використовувалось для проведення розробки, розраховується за формулою:

$$A = \frac{Ц \cdot H_a}{100} \cdot \frac{T}{12} \text{ [грн]}, \quad (5.4)$$

де, Ц – балансова вартість обладнання, грн.;

$Н_а$ – річна норма амортизаційних відрахувань;

Т – термін використання під час розробки, місяців.

Норма амортизації розраховується за формулою:

$$H_a = \frac{B_п - B_л}{B_п \cdot T_{кв}} \cdot 100 \text{ [грн]}, \quad (5.5)$$

де $В_п$ і $В_л$ – відповідно первісна та ліквідаційна вартість основних фондів;

$Т_{кв}$ – строк корисного використання, роки.

Норма амортизаційних витрат становитиме:

$$H_a = \frac{20000 - 2000}{20000 \cdot 5} \cdot 100 = 18\% \text{ (грн)}.$$

Розрахуємо амортизаційні витрати на ноутбук, балансова вартість якого становить 28000 грн, а термін використання – 3 місяці:

$$А = \frac{28000 \cdot 18}{100} \cdot \frac{3}{12} = 1260 \text{ (грн)}.$$

Зроблені розрахунки наведено у таблиці 4.5.

Таблиця 4.5 – Амортизаційні відрахування

Найменування	Балансова вартість, грн	Термін використання, р	Фактична трив. використання, міс.	Величина амортизаційних відрахувань, грн
Ноутбук	28000,00	5	3	1260,00
Офісне приміщення	350000,00	20	3	5400,00
Всього:				**6850,00**

Інформацію про витрати на матеріали, що використані при розробці внесено до таблиці 5.6.

Таблиця 5.6 – Витрати на матеріали, що були використані для розробки продукту.

Найменування матеріалу	Одиниці виміру	Ціна за одиницю, грн	Витрачено, шт	Вартість витрачених матеріалів, грн
Папір	уп.	100	1	100,00
Ручка	шт.	15	2	30,00
Олівець	шт.	10	1	10,00
Загальна сума витрат за статтею				140,00

Розрахунок витрат на силову електроенергію ($В_е$) здійснюється за формулою:

$$В_е = В \cdot П \cdot Ф \cdot К_п, [грн], \quad (5.6)$$

де В – вартість 1 кВт-год. електроенергії, становить 1,68 грн./кВт.

П – установлена потужність комп’ютера, кВт;

Ф – фактична кількість годин роботи обладнання.

$К_п$ – коефіцієнт використання потужності, становить 0,3.

$$В_е = 1{,}68 \cdot 0{,}5 \cdot 504 \cdot 0{,}3 = 127{,}1 \text{ (грн)}.$$

Інші витрати $В_{ін}$ охоплюють: витрати на управління організацією, Інтернет, оплату службових відряджень, витрати на утримання, ремонт та експлуатацію основних засобів, витрати на опалення, освітлення, водопостачання, Інтернет послуги. Інші витрати $І_в$ можна прийняти як 150% від суми основної заробітної плати розробника:

$$В_{ін} = 150\% \cdot (З_р) \text{ [грн]}. \quad (5.7)$$

$$В_{ін} = 1{,}5 \cdot 53714 = 80571{,}45 \text{ (грн)}.$$

Сума всіх попередніх статей витрат дає витрати на виконання даної частини (розділу, етапу) роботи – В:

В=53714,30+6445,70+13235,20+7050,00+140+127,1+80571,45=161283,75(грн).

Розраховуємо загальні витрати на виконання даної роботи $В_{заг}$ за формулою:

$$B_{заг} = \frac{B}{\alpha} \text{ [грн]}, \quad (5.8)$$

де α – частка витрат, які безпосередньо здійснює виконавець даного етапу роботи, у відн. одиницях (α= 1.)

$$В_{заг.} = \frac{161283.75}{1} = 161283.75 \text{ (грн)}.$$

Визначаємо загальні витрати на виконання та впровадження результатів виконаної наукової роботи (ЗВ) за формулою:

$$ЗВ = \frac{В_{заг}}{\beta} [\text{ грн}], \qquad (5.9)$$

де β – коефіцієнт, який характеризує етап (стадію) виконання даної роботи. Так, якщо розробка знаходиться:

- на стадії науково-дослідних робіт, то $\beta \approx 0{,}1$;
- на стадії технічного проектування, то $\beta \approx 0{,}2$;
- на стадії розробки конструкторської документації, то $\beta \approx 0{,}3$;
- на стадії розробки технологій, то $\beta \approx 0{,}4$;
- на стадії розробки дослідного зразка, то $\beta \approx 0{,}5$;
- на стадії розробки промислового зразка, $\beta \approx 0{,}7$;
- на стадії впровадження, то $\beta \approx 0{,}9$.

Отже, підставимо дані в формулу й отримаємо результат:

$$ЗВ = \frac{161283.75}{0{,}7} = 230404{,}30(\text{грн}).$$

Витрати на виконання наукової роботи та впровадження її результатів становитиме 230404,30 грн.

5.3 Прогнозування комерційних ефектів від реалізації результатів розробки

В умовах ринку узагальнюючим позитивним результатом, що його отримує підприємство від впровадження результатів тієї чи іншої розробки, є збільшення чистого прибутку підприємства.

Виконання даної наукової роботи та впровадження її результатів складає приблизно 1 рік. Позитивні результати від впровадження розробки очікуються вже в перші місяці після впровадження.

Проведемо детальніше прогнозування позитивних результатів та кількісне їх оцінювання по роках.

Обчислимо збільшення чистого прибутку підприємства $\Delta П_i$ для кожного із років, протягом яких очікується отримання позитивних результатів від впровадження розробки, розраховується за формулою:

$$\Delta П_i = \sum_1^n (\Delta П_я \cdot N + П_я \cdot \Delta N)_i \ [грн], \qquad (5.10)$$

де $\Delta П_я$ – покращення основного якісного показника від впровадження результатів розробки у даному році;

N – основний кількісний показник, який визначає діяльність підприємства у даному році до впровадження результатів наукової розробки;

ΔN – покращення основного кількісного показника діяльності підприємства від впровадження результатів розробки;

$П_я$ – основний якісний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки;

n – кількість років, протягом яких очікується отримання позитивних результатів від впровадження розробки.

Припустимо, що внаслідок впровадження результатів наукової розробки чистий прибуток підприємства збільшиться на 50,00 грн, а кількість одиниць реалізованих послуг збільшиться: протягом першого року – на 1000 од., протягом другого року – ще на 2000 од., протягом третього року – ще на 3000 од.

Орієнтовно: реалізація продукції до впровадження результатів наукової розробки складала 1 шт., а прибуток, що його отримувало підприємство на одиницю послуги до впровадження результатів наукової розробки – 45,00 грн.

Потрібно спрогнозувати збільшення чистого прибутку підприємства від впровадження результатів наукової розробки у кожному році відносно базового.

Збільшення чистого прибутку підприємства $\Delta П_1$ протягом першого року складе:

$$\Delta П_1=50\cdot 1+(60+45)\cdot 1500=157550{,}00 \text{ (грн)}.$$

Обчислимо збільшення чистого прибутку підприємства $\Delta П_2$ протягом другого року:

$$\Delta П_2=50\cdot 1+(60+45)\cdot (1500+2000)=367550{,}00 \text{ (грн)}.$$

Збільшення чистого прибутку підприємства $\Delta П_3$ протягом третього року становитиме:

$$\Delta П_3=50\cdot 1+(60+45)\cdot (1500+2000+3000)=682550{,}00 \text{ (грн)}.$$

Розрахунки показують, що комерційний ефект від впровадження розробки виражається у щорічному збільшенні чистого прибутку підприємства.

5.4 Розрахунок ефективності вкладених інвестицій та період їх окупності

Розрахований комерційний ефект від можливого впровадження розробок ще не означає, що ця розробка реально буде впроваджена. Якщо збільшення прогнозованого прибутку від впровадження результатів наукової розробки є вигідним для підприємства (організації), то це ще не означає, що інвестор погодиться фінансувати дану розробку.

Інвестор погодиться вкладати кошти у реалізацію даної наукової розробки тільки за певних умов.

Основними показниками, які визначають доцільність фінансування наукової розробки певним інвестором, є абсолютна і відносна ефективність вкладених інвестицій та термін їх окупності. Розрахунок ефективності вкладених інвестицій передбачає проведення таких робіт:

Розрахунок ефективності вкладених інвестицій передбачає:

1-й крок. Розрахунок теперішньої вартості інвестицій PV, що вкладаються в наукову розробку. Такою вартістю ми можемо вважати прогнозовану величину загальних витрат ЗВ на виконання та впровадження результатів НДДКР, тобто ЗВ = PV = 230404,30 (грн).

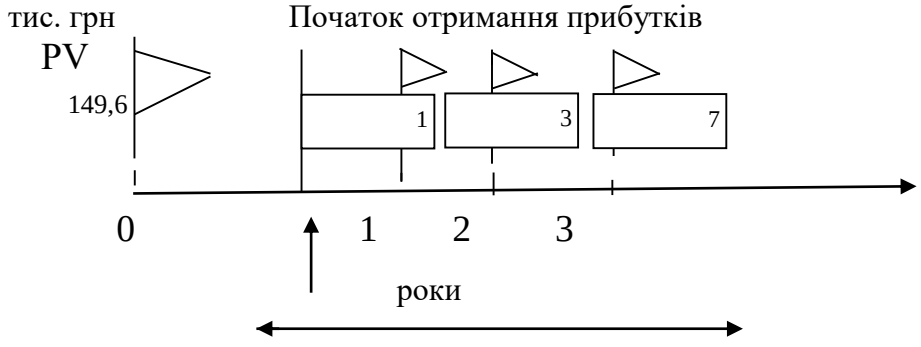
2-й крок. Розраховуємо очікуване збільшення прибутку $\Delta П_i$, що його отримає підприємство (організація) від впровадження результатів наукової розробки, для кожного із років, починаючи з першого року впровадження.

3-й крок. Будуємо вісь часу, на якій відображаємо всі платежі (інвестиції та прибутки), що мають місце під час виконання науково-дослідної роботи та впровадження її результатів (рис. 5.1).

Платежі показуємо у ті терміни, коли вони здійснюються.

Результати вкладених у наукову розробку інвестицій почнуть виявлятись протягом трьох років.

У першому році підприємство отримає збільшення чистого прибутку на 157550,00 грн відносно базового року, у другому році – збільшення чистого прибутку на 367550,00 грн (відносно базового року), у третьому році – збільшення чистого прибутку на 682550,00 грн (відносно базового року).



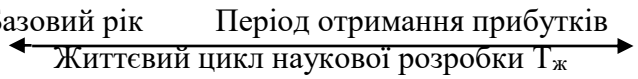


Рисунок 5.1 – Вісь часу з фіксацією платежів, що мають місце під час розробки та впровадження результатів НДДКР

Розраховуємо абсолютну ефективність вкладених інвестицій $Е_{абс}$. за формулою:

$$Е_{абс} = (ПП - PV), [грн], \quad (5.11)$$

де ПП – приведена вартість всіх чистих прибутків, що їх отримає підприємство (організація) від реалізації результатів наукової розробки, грн.;

PV – теперішня вартість інвестицій PV = ЗВ, грн.

У свою чергу, приведена вартість всіх чистих прибутків ПП розраховується за формулою:

$$ПП = \sum_{1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, [\text{ грн}], \quad (5.12)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДДКР, грн;

т – період часу, протягом якого виявляються результати впровадженої НДДКР, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник знаходиться на рівні 0,1;

t – період часу (в роках) від моменту отримання чистого прибутку до точки 0.

Розрахуємо вартість чистого прибутку:

$$ПП = \frac{157550}{(1+0{,}1)^1} + \frac{367550}{(1+0{,}1)^2} + \frac{682550}{(1+0{,}1)^3} = 959797{,}52(\text{грн}).$$

Розраховуємо абсолютної ефективності вкладених інвестицій $E_{абс}$:

$$E_{абс} = 959797{,}52 - 230404{,}30 \ = 729393{,}22(\text{грн.})$$

Оскільки $E_{абс} > 0$, то результат від проведення наукових досліджень та їх впровадження принесе прибуток, а вкладання коштів у даний проект є доцільним.

Розраховуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $E_в$ за формулою:

$$E_в = \sqrt[T_ж]{1+\frac{E_{абс}}{PV}} - 1 \qquad (5.13)$$

де $E_{абс}$ – абсолютна ефективність вкладених інвестицій, грн;

PV – теперішня вартість інвестицій PV = ЗВ, грн;

$T_ж$ – життєвий цикл наукової розробки, роки.

$$E_в = \sqrt[3]{1+\frac{729393{,}22}{230404{,}30}} - 1 = \sqrt[3]{4{,}16} - 1 = (1{,}63 - 1) = 0{,}609 \text{ або } 61{,}9\%$$

Порівнюємо відносну ефективність $E_в$ з мінімальною (бар'єрною) ставкою дисконтування $\tau_{мін}$, яка визначає ту мінімальну дохідність, нижче за яку інвестиції вкладатися не будуть. У загальному вигляді мінімальна (бар'єрна) ставка дисконтування $\tau_{мін}$ визначається за формулою:

$$\tau = d + f, \qquad (5.14)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; d = 0,2;

f – показник, що характеризує ризикованість вкладень; зазвичай, величина f = (0,05...0,1), але може бути і значно більше, у нашому випадку f = 0,05.

$$\tau = 0{,}2 + 0{,}05 = 0{,}25$$

Оскільки $E_в$ = 61% > $\tau_{мін}$ = 0,25 = 20%, то інвестор буде зацікавлений вкладати гроші в дану наукову розробку, адже він отримає значно більші прибутки, ніж якщо просто покладе свої гроші на депозит у комерційному банку.

Визначаємо термін окупності вкладених інвестицій за формулою:

$$T_{ок} = \frac{1}{E_в} [\text{року}]. \qquad (5.15)$$

$$\mathrm{T_{ок}} = \frac{1}{0{,}61} = 1{,}64 \text{ (року)}.$$

Термін окупності інвестицій $T_{ок}$ =1,64 < 3…5 років і свідчить, що фінансування даної наукової розробки є доцільним.

5.5 Висновок

В розділі було виконано оцінювання комерційного потенціалу розробки. Визначено, що рівень комерційного потенціалу розробки є вище середнього. Проведено технологічний аудит з залученням трьох незалежних експертів. Аналіз комерційного потенціалу розробки показав, що нова розробка за своїми характеристиками випереджає аналоги, що підтверджує її перспективність. Вона має кращі функціональні показники, а тому є конкурентоспроможним товаром на ринку.

Згідно із розрахунками всіх статей витрат на виконання науково-дослідної, дослідно-конструкторської та конструкторсько-технологічної роботи загальні витрати на розробку складають 220404,30 грн.

Розрахована абсолютна ефективність вкладених інвестицій в сумі 729393,22 грн свідчить про отримання прибутку інвестором від комерціалізації розробки.

Відносна ефективність вкладених в наукову розробку інвестицій складає 61%, що вище за мінімальну бар'єрну ставку дисконтування, яка складає 25%. Це означає потенційну зацікавленість інвесторів у фінансуванні розробки. Термін окупності вкладених у реалізацію проекту інвестицій становить 1,64 року, що також свідчить про доцільність фінансування нової розробки.

ВИСНОВКИ

У магістерській роботі було опрацьовано концепції та засоби побудови програмно-навігаційних систем контролю руху транспортного засобу, а саме розроблено програмний модуль для покращення якості і розпізнавання дорожніх знаків.

У процесі виконання роботи розвязано такі задачі:

– проведено аналіз існуючих методів розпізнавання дорожніх знаків;

– проведено аналіз методів підвищення якості зображень дорожніх знаків;

– запропонувано: метод підвищення якості зображень дорожніх знаків; метод розпізнавання дорожніх знаків;

- розроблено програмні компоненти та систему розпізнавання дорожніх знаків;

- проведено експериментальні дослідження розроблених програмних компонентів розпізнавання дорожніх знаків

Проведено аналіз методів розробки систем розпізнавання дорожніх знаків. Варіантний аналіз і обґрунтування вибору засобів реалізації і обгрунтування вибору бібліотек і компонентів.

Розглянуто методи просторового покращення зображень дорожніх знаківі адаптитивна бінаризації для покращення зображень дорожніх знаків.

У першому розділі проведено аналіз існуючих рішень та предметної області. Таким чином, було прийнято рішення модифікувати існуючий метод розпізнавання дорожніх знаків, для підвищення ефективності розпізнавання та розробити програмний модуль для систем детектування дорожніх знаків. Як результат було проведено аналіз та постановку задачі, наведене головне призначення, цілі та задачі розробки.

В другому розділі було проведено аналіз мов програмування. В результаті було прийнято рішення використовувати одну мову для програми – C++, а середовищем розробки обрано MS Visual Studio. Розглянуто бібліотеки методи OpenCV. Використано функціональний (структурний) метод як такий, що максимально задовольняє вимогам

В третьому розділі, проведено аналіз та розглянуто декілька популярних мов програмування, а саме C, C++, Java, Python, Swift. Було наведено плюси та мінуси цих мов. Серед всіх найкращою для досягнення поставлених цілей в даній магістерській кваліфікаційній роботі було обрано мову програмування C++. Також було проаналізована середовища програмування для мови програмування Python, а саме Visual Studio, Eclipse,Visual studio Code, Sublime text та PyCharm. Серед яких було обрано середовище програмування Visual Studio, тому що воно є зручним та ефективним для досягнення поставлених цілей.

Обгрунтовано вибір бібліотек та фреймворків для програмного модуля. А саме, обрано бібліотеку, котра містить в собі більше 150 фреймворків, що суттєво покращують можливості та роботу c++, а саме Boost та OpenCV (зручна та необхідна бібліотека для роботи з зображеннями, має близько 3000 оптимізованих алгоритмів, які включають в себе насичений набір класичних і сучасних алгоритмів машинного навчання та комп'ютерного зору). Наведенно декілька прикладів бібліотек, що входять до складу Boost, а саме: Boost Interprocess, Boost Pool та інші.

В чертвертому розділі проведено тестування програмного модулю в різних режимаї роботи та за різних погодніх умов. Тестування показало коректну роботу усіх складових розроблюваного програмного додатку. Зроблено порівняльну характеристику, де було підтвердженно якість та ефективність розробленого модулю.

ПЕРЕЛІК ПОСИЛАНЬ

1. S. Singh, Critical reasons for crashes investigated in the national motor vehicle crash causation survey. Chicago, 2015. 373 c.

2. Liu, C., Chang, F., Chen, Z., Liu, D. Fast traffic sign recognition via high-contrast region extraction and extended sparse representation. IEEE Trans. Intell. Transp. Syst. New York, 2016, 153 c.

3. Varun, S., Singh, S., Kunte, R.S., Samuel, R.S., Philip, B. A road traffic signal recognition system based on template matching employing tree classifier. *International Conference on Computational Intelligence and Multimedia Applications ICCIMA*. (Sivakasi, India, 13–15 December 2007). C. 360–365.

4. Ruta, A., Li, Y., Liu, X. Detection, tracking and recognition of traffic signs from video input. *11th International IEEE Conference on Intelligent Transportation Systems*. (ITSC 2008, Beijing, China, 12–15 October 2008). C. 55–60.

5. Jiang, Y., Zhou, S., Jiang, Y., Gong, J., Xiong, G., Chen, H. Traffic sign recognition using ridge regression and Otsu method. *2011 IEEE Intelligent Vehicles Symposium (IV)*. (Baden-Baden, Germany, 5–9 June 2011). C. 613–618.

6. Vázquez-Reina, A., Lafuente-Arroyo, S., Siegmann, P., Maldonado-Bascón, S., Acevedo-Rodríguez, F. Traffic sign shape classification based on correlation techniques. *5th WSEAS International Conference on Signal Processing, Computational Geometry & Artificial Vision*. (Alcalá de Henares, Malta, 15–17 September 2005). C. 149–154.

7. Maldonado-Bascón,S., Lafuente-Arroyo,S., Gil-Jimenez,P., GómezMoreno,H., López-Ferreras, F. Road-sign detection and recognition based on support vector machines. IEEE Trans. Intell. Transp. Syst.
2007, c. 264–278.

8. Jiménez, P.G., Bascón, S.M., Moreno, H.G., Arroyo, S.L., Ferreras, F.L. Traffic sign shape classification and localization based on the normalized FFT of the signature of blobs and 2D homographies. Signal Process.

2008, c. 2943–2955.

9. Wu, J.-Y.; Tseng, C.-C.; Chang, C.-H.; Lien, J.-J.J.; Chen, J.C.; Tu, C.T. Road sign recognition system based on GentleBoost with sharing features. *2011 International Conference on System Science and Engineering (ICSSE)*, Macao, China, 8–10 June 2011; c. 410–415.

10. Nicchiotti, G.; Ottaviani, E.; Castello, P.; Piccioli, G. Automatic road sign detection and classification from color image sequences. In Proceedings of the 7th International Conference on Image Analysis and Processing, 1994; pp. 623–626.

11. Priese, L.; Rehrmann, V. On hierarchical color segmentation and applications. *1993 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, New York, NY, USA, 15–17 June 1993; c. 633–634

12. Swain, M.J., Ballard, D.H. Color indexing. *Int. J. Comput. Vis.* 1991, C. 11–32.

13. Fan, J., Yau, D.K., Elmagarmid, A.K., Aref, W.G. Automatic image segmentation by integrating color-edge extraction and seeded region growing. IEEE Trans. Image Process. 2001. C. 1454–1466.

14. Mueller, R., Steck, M., 2003. ―Road Sign Recognition‖, Term Paper, Computer Perception with Artificial Intelligence, University of Applied Sciences, Biel, Switzerland

15. Piccioli, G., De Micheli, E., Parodi, P., Campani, M., 1996. ―Robust Method for Road Sign Detection and Recognition‖, Image and Vision Computing 14, pp.208-223.

16. Novovicova, J., Paclik, P., Pudil, P., and Somol, P., 2000. "Road Sign Classification Using Laplace Kernel Classifier," Pattern Recognition Letters 21, pp. 1165-1173

17. Yuille, A. L., Snow, D., and Nitzberg,M., 1998. ― Using Color to Detect, Localize and Identify Informational Signs‖, Proc. International Conference on Computer Vision ICCV98, Bombay, India, pp. 628-633.

18. De la Escalera, A., Moreno, L., Salichs, M.A., and Amingol, J.M., 1997. "Road Traffic Sign Detection and Classification," IEEE Transactions Industrial Electronics, 44, pp. 848-859.

19. Lauziere, Y., Gingras, D., Ferrie, F., 2001. ―A Model-based Road Sign Identification System‖, Proc. IEEE Computer Conference on Computer Vision and Pattern Recognition, pp. 1163-1170.

20. Крисилов В.А. Представление исходных данных в задачах нейросетевого программирования / Одесса: ОНПУ. 2003.

21. База и генератор образовательных ресурсов [Електронний ресурс] // МГТУ им. Н.Э. Баумана, кафедра САПР. 2003-2015. Режим доступу:

22. Шитиков В.К., Розенберг Г.С., Зинченко Т.Д. Методы системной идентификации / Тольятти: ИЭВБ РАН. 2003. 463 с.

23. Творошенко І. С. Конспект лекцій з дисципліни «Цифрова обробка зображень» для студентів 4 курсу денної форми навчання напряму 6.080101 – Геодезія, картографія та землеустрій / І. С. Творошенко ; Харків. нац. ун-т міськ. госп-ва ім. О. М. Бекетова. – Харків : ХНУМГ ім. О. М. Бекетова, 2017. – 75 с.

24. Майданюк В. П. Навчальний посібник. Кодування зображень / В. П. Майданюк. — Вінниця: ВДТУ, 2000. — 62 с

25. LeCun, Yann, et al. “Gradient-based learning applied to document recognition.” Proceedings of the IEEE 86.11 (1998): 2278-2324.

26. Електронний ресурс – [https://image-net.org]

27. Кус М.К., Гьокмен М., Етанер-Уяр А.С.: Розпізнавання дорожніх знаків за допомогою масштабного інваріантного перетворення ознак і класифікації кольорів. InProc. Int. Симп. Комп’ютерні інформаційні науки (2008)

28. The german traffic sign recognition benchmark (GTSRB) dataset: веб-сайт URL:https://sid.erda.dk/public/archives/daaeac0d7ce1152aea9b61d9f1e19370/published-archive.html

29. OpenCV library: веб-сайт. URL:[https://opencv.org/]

30. C++ vs Java vs Python a comparison. URL: https://www.javatips.net/blog/c-vs-java-vs-python-a-comparison

31. Офіційний сайт [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Тестування_програмного_забезпечення

32. Sauvola J., Petikainen M. Adaptive document image binarization, Pattern recognition 33 (2000) 225-236 2. Bernsen J. Dynamic thresholding of grey-level images, Proceedings of the Eighth ICPR, 1986, pp. 1251-1255.

33. Gatos B., Pratikakis I., Perantonis S. J. Anadaptive binarisation technique for low quality historical documents In: IAPR Workshop on Document Analysis Systems (DAS2004), Lecture Notes in Computer Science (3163), September 2004, pp. 102–113 (2004)

34. Офіційний сайт Draw.io – [Електронний ресурс]. – Режим доступу: https://app.diagrams.net

35. Фурман Я. А., Юрьев А. Н. , Яншин В. В. Цифровые методы обработки и распознавания бинарных изображений.— Красноярск: Изд-во Краснояр. ун-та, 1992г-248 с.

36. Kapur J. N., Sahoo P. K. and Wong A.K.C., A New Method for Gray-Level Picture Thresholding Using the Entropy of the Histogram, CVGIP, (29), pp.273-285, 1985

ДОДАТОК А.
ТЕХНІЧНЕ ЗАВДАННЯ

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ
д.т.н., проф. О. Н. Романюк
« 13 » вересня 2021 р.

Технічне завдання
на магістерську кваліфікаційну роботу
«Концепція та засоби побудови програмно-навігаційної системи контролю руху транспортного засобу»
за спеціальністю
121 – Інженерія програмного забезпечення

Керівник магістерської кваліфікаційної роботи:
к.т.н., доц. О.М. Рейда
"____"_____2021 р.
Виконав:
студент гр.2ПІ-20м К. І. Прокопчук
"____"_____2021 р.

Вінниця – 2021 рік

1. Найменування та галузь застосування

Магістерська кваліфікаційна робота: «Концепція та засоби побудови програмно-навігаційної системи контролю руху транспортного засобу».

Галузь застосування – системи комп'ютерного зору.

2. Підстава для розробки.

Підставою для виконання магістерської кваліфікаційної роботи (МКР) є індивідуальне завдання на МКР та наказ ректора по ВНТУ № 277 від « 24 » вересня 2021 р. про закріплення тем МКР.

3. Мета та призначення розробки.

Метою роботи є роботи розробка програмного модулю розпізнавання дорожніх знаків та покращення реалізації алгоритму розпізнавання.

Призначення роботи – розробка методів та засоби реалізації процесу розпізнавання дорожніх знаків для підвищення швидкодії розпізнавання за допомогою покращення зображення.

4. Вихідні дані для проведення МКР

Перелік основних літературних джерел, на основі яких буде виконуватись МКР.

1 S. Singh, Critical reasons for crashes investigated in the national motor vehicle crash causation survey. Chicago, 2015. 373 c.

2 Ruta, A., Li, Y., Liu, X. Detection, tracking and recognition of tra c signs from video input. 11th International IEEE Conference on Intelligent Transportation Systems. (ITSC 2008, Beijing, China, 12–15 October 2008). C. 55–60.

3 Стаття О. М. Рейда, к. т. н., ст. вкл. каф; Ю. В. Олійник; А. О. Панчук; М.Л. Синенький «Методи поліпшення цифрового зображення та відновлення його структури».

5. Технічні вимоги

Базові методи обробки зображення такі як низькочастотна і високочастотна фільтрація, бінірізація, методи кореляції, вихідні дані – кінцеве зображення, опрацьоване розробленим алгоритмом

6. Конструктивні вимоги.

Конструкція пристрою повинна відповідати естетичним та ергономічним вимогам, повинна бути зручною в обслуговуванні та керуванні.

Графічна та текстова документація повинна відповідати діючим стандартам України.

7. Перелік технічної документації, що пред'являється по закінченню робіт:

– пояснювальна записка до МКР;

– технічне завдання;

– лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації та ДСТУ.

9. Стадії та етапи розробки:

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи
1	Аналіз методів розв'язання поставленої задачі розробки системи розпізнавання дорожніх знаків	15.09.2021-26.09.2021
2	Порівняльний аналіз методів розпізнавання дорожніх знаків	27.092021-15.10.2021
3	Розробка методів розпізнавання дорожніх знаків	16.10.2021-7.11.2021
4	Розробка модулю розпізнавання дорожніх знаків	8.11.2021-21.11.2021
5	Економічна частина	22.11.2021-30.11.2021

10. Порядок контролю та прийняття.

Виконання етапів магістерської кваліфікаційної роботи контролюється керівником згідно з графіком виконання роботи.

Прийняття магістерської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком.

ДОДАТОК Б.
ІЛЮСТРАТИВНИЙ МАТЕРІАЛ

Рисунок Б.1. Плакат №1

Рисунок Б.2. Плакат №2

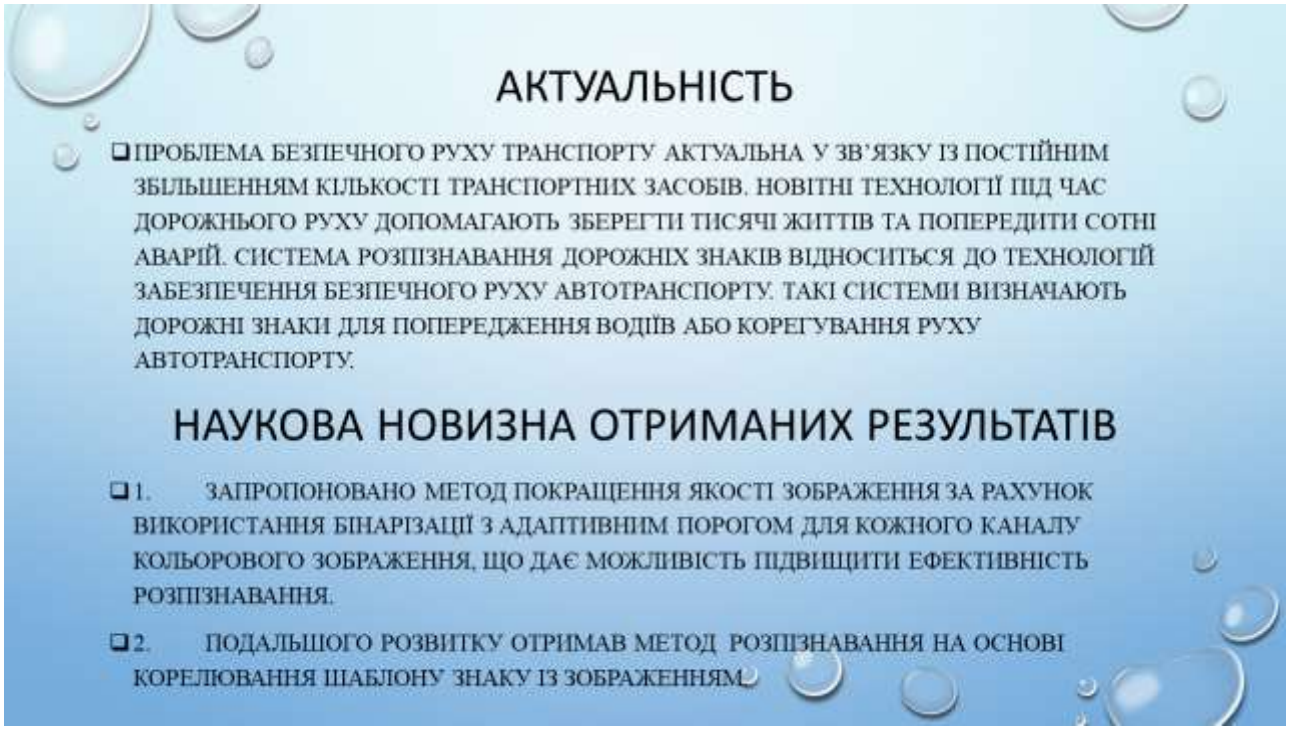


Рисунок Б.3. Плакат №3

Рисунок Б.4. Плакат №4

Рисунок Б.5. Плакат №5

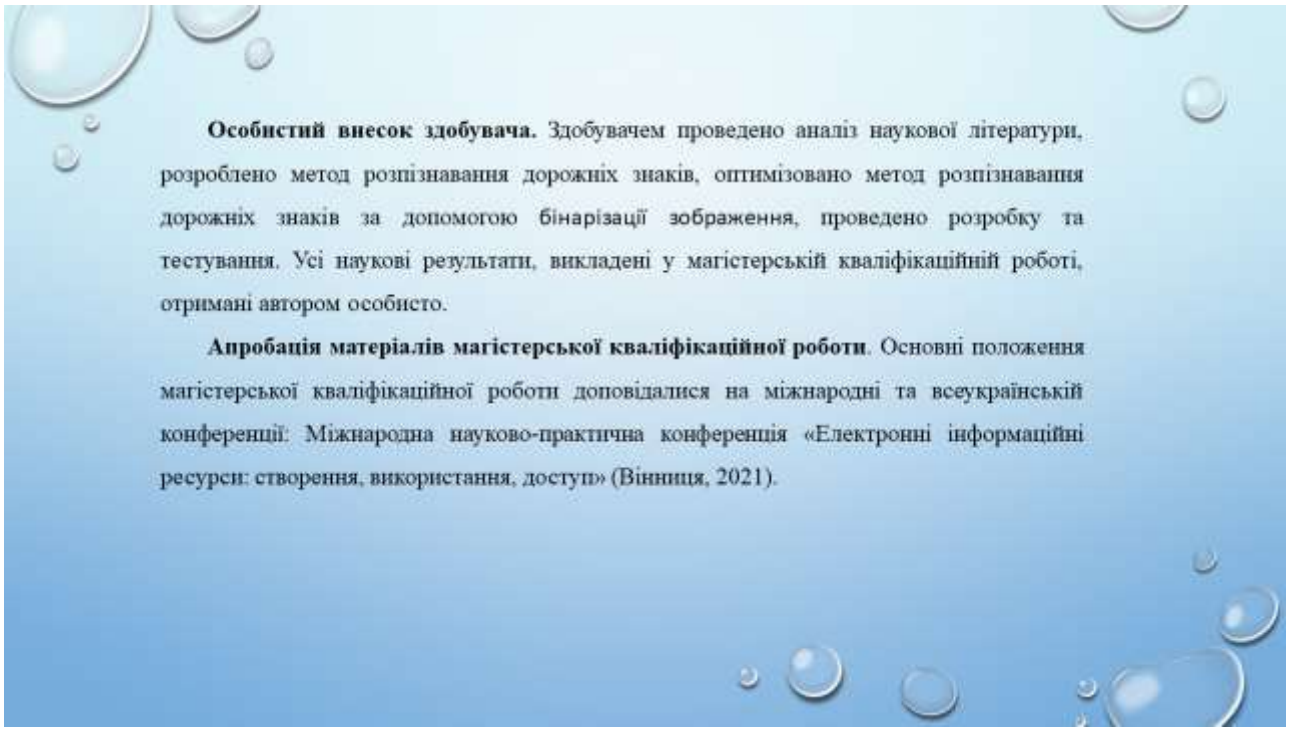


Рисунок Б.6. Плакат №6

Рисунок Б.7. Плакат №7

Рисунок Б.8. Плакат №8

Рисунок Б.9. Плакат №9

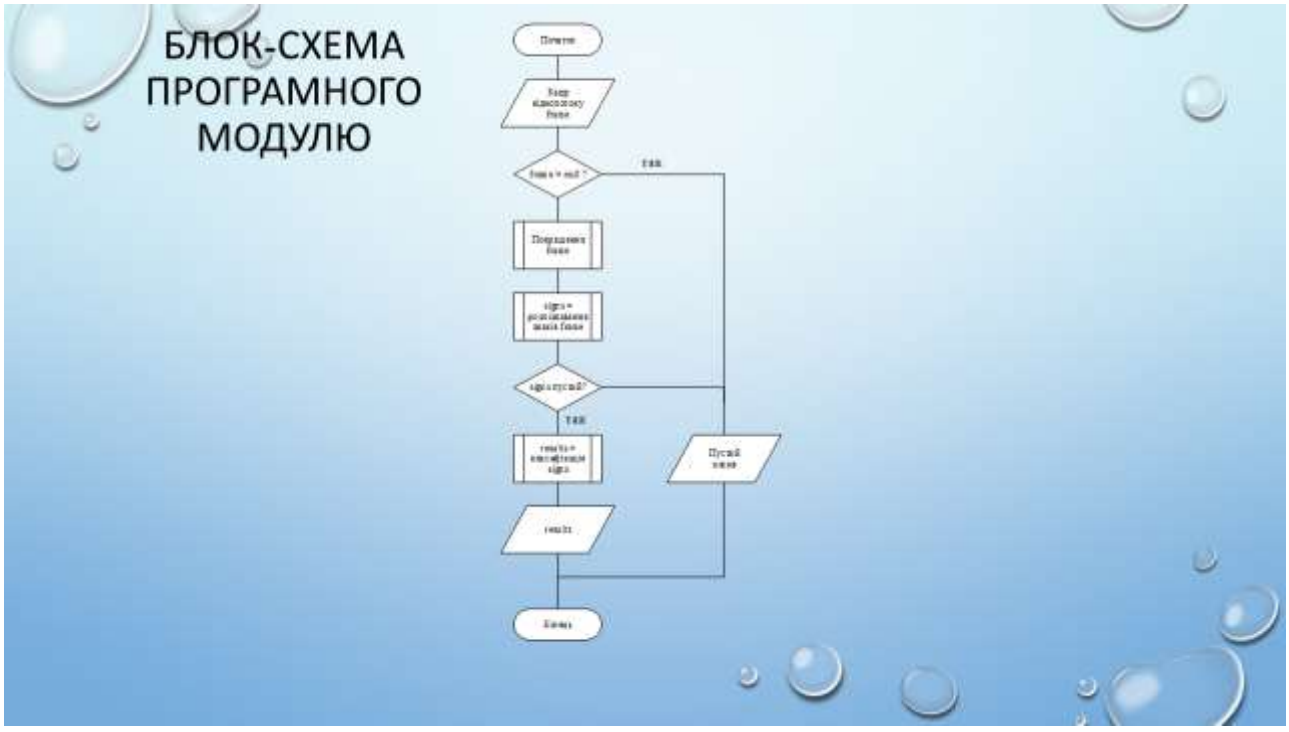


Рисунок Б.10. Плакат №10

Рисунок Б.11. Плакат №11

Рисунок Б.12. Плакат №12

Рисунок Б.13. Плакат №13

ДОДАТОК В
ЛІСТИНГ КОДУ

В.1 Лістинг коду прототипу програми

В.1.1 Файл main.cpp

```
#include <opencv2/imgproc.hpp>
#include <opencv2/highgui.hpp>
#include <opencv2/core/utils/logger.hpp>

#include <thread>
#include <queue>
#include <iostream>

#include "safe_queue.h"
#include "helper.h"
#include "detection.h"
#include "classification.h"
#include "video_recorder.h"

using namespace cv;
using namespace dnn;

/* Ініціюємо статичні змінні */
Net Detection::detection_net_;
Net Classification::classification_net_;
std::vector<String> Detection::out_names_;
```

```
ThreadSafeQueue<Mat> Helper::frames_queue;
ThreadSafeQueue<Mat> Helper::processed_frames_queue;
ThreadSafeQueue<std::vector<Mat>> Helper::predictions_queue;

volatile sig_atomic_t Helper::stop;
bool Helper::record_video = false;
bool Helper::use_video = false;
std::vector<int> Helper::SignSpeeds = { 20, 30, 50, 60, 70, 80, 0, 100, 120 };
int Helper::record_fps = 30;

void run_program(VideoCapture video_capture,
std::unique_ptr<VideoRecorder>& recorder){
Mat frame, sign_img(Helper::sign_res_display_width,
Helper::sign_res_display_width, CV_8UC3, Scalar::all(255));
std::string input_fps_str = "Video: 0 FPS", detection_fps_str = "Inference: 0
FPS", limit_str = "Limit: - km/h";
std::cout << "Program started..." << std::endl;

while (true){
/* Захоплюємо рамку */
if (!Helper::GrabFrame(video_capture, frame)){
break;
}

/* Демонструємо результат */
if (!Helper::predictions_queue.empty()) {
/* Перевіряємо чи було успішно виявлено знак */
std::vector<Mat> outs = Helper::predictions_queue.GetHead();
Mat pred_frame = Helper::processed_frames_queue.GetHead();
```

```
        Mat sign_classify;
        Detection::FindSign(pred_frame, outs, sign_classify);

        /* Робимо класифікацію */
        if (!sign_classify.empty()) {
        Classification::ClassifySign(sign_classify, sign_img, limit_str);
        }

        detection_fps_str = format("Inference: %d FPS",
int(Helper::predictions_queue.GetFPS()));
        }

        /* Створюємо інтерфейс та демонструємо його*/
        Rect roi_sign(Point(0, frame.size().height - Helper::sign_res_display_width),
sign_img.size());
        sign_img.copyTo(frame(roi_sign));
        int x_add = int((Helper::window_width - frame.size().width) / 2);
        Rect roi_frame(Point(x_add, 0), frame.size());
        Mat display_frame(int(Helper::window_height), int(Helper::window_width),
CV_8UC3, Scalar::all(255));
        frame.copyTo(display_frame(roi_frame));
        input_fps_str = format("Video: %d FPS",
int(Helper::frames_queue.GetFPS()));

        putText(display_frame,
        limit_str,
        Point(x_add, frame.size().height + 25),
        FONT_HERSHEY_COMPLEX_SMALL,
        1,
```

```
Scalar::all(0));

putText(display_frame,
input_fps_str,
Point(x_add + 220, frame.size().height + 25),
FONT_HERSHEY_COMPLEX_SMALL,
1,
Scalar::all(0));

putText(display_frame,
detection_fps_str,
Point(x_add + 410, frame.size().height + 25),
FONT_HERSHEY_COMPLEX_SMALL,
1,
Scalar::all(0));

imshow("OpenCV Traffic Sign Recognition", display_frame);

/* Відсилаємо кадр на відеореєстратор */
if (Helper::record_video){
recorder->safe_mat_queue.Push(display_frame);
}

/* При натисненні клавіші вихід */
if (waitKey(1) > 0){
std::cout << "User key pressed, exit!" << std::endl;
break;
}
}
```

```
}

int main(int argc, char** argv)
{
utils::logging::setLogLevel(utils::logging::LogLevel::LOG_LEVEL_SILENT);

std::unique_ptr<VideoRecorder> video_recorder;

/* Аналізуємо команди */
std::cout << "*** OpenCV Traffic Sign Recognition" << std::endl;
program_options::variables_map variables_map;
Helper::ParseCommandLineOptions(variables_map, argc, argv);

/* Завантаження моделей */
if (!Detection::LoadModel() || !Classification::LoadModel()) {
std::cerr << "Error loading required models, exit.." << std::endl;
exit(EXIT_FAILURE);
}

/* Відкриваємо вхід*/
VideoCapture video_capture;
bool success;
if (Helper::use_video) {
std::string video_path = variables_map["video"].as<std::string>();
std::cout << "Trying to open video file: " << video_path << ".." << std::endl;
if (!Helper::IsFileExist(video_path.c_str())) {
std::cerr << "Input video does not exist, exit!" << std::endl;
exit(EXIT_FAILURE);
}
```

```
        success = Helper::OpenInputSource(video_capture, video_path);
        }
        else {
        std::cout << "Trying to open camera.." << std::endl;
        success = Helper::OpenInputSource(video_capture);
        }
        if (!success) {
        exit(EXIT_FAILURE);
        }

        /* Запускаємо всі потоки */
        Helper::stop = false;
        std::vector<std::thread> threads;
        if (Helper::record_video) {
        video_recorder = std::make_unique<VideoRecorder>("./record/");
        threads.emplace_back(std::thread(&VideoRecorder::Run,
video_recorder.get()));
        }
        threads.emplace_back(Detection::ProcessingThread);

        /*Запуск програми */
        run_program(video_capture, video_recorder);

        /* Чистка */
        Helper::stop = true;
        for (auto& th : threads) {
        th.join();
        }
        destroyAllWindows();
```

```
exit(EXIT_SUCCESS);
}
```

B.1.2 Файл video_recorder.h

```
#pragma once

#include <string>
#include <utility>

#include <opencv2/opencv.hpp>
#include <opencv2/videoio.hpp>
#include <opencv2/core.hpp>

#include <boost/filesystem.hpp>
#include <boost/date_time/posix_time/posix_time.hpp>

#include "helper.h"
#include "safe_queue.h"

using namespace cv;
using namespace std::chrono;
namespace file_system = boost::filesystem;

class VideoRecorder {
std::string directory_;
int codec_ = VideoWriter::fourcc('M', 'J', 'P', 'G');
bool first_time_ = true;
std::string time_stamp_;
VideoWriter video_writer_;
```

```
        void _Run(const Mat& m);

        public:
        ThreadSafeQueue<Mat> safe_mat_queue;

        VideoRecorder(std::string directory) : directory_(std::move(directory))       {
        if (!file_system::is_directory(directory_) || !file_system::exists(directory_)) {
        file_system::create_directory(directory_);
        }
        };

        ~VideoRecorder() = default;
        void Run();
        };

        inline void VideoRecorder::_Run(const Mat& m) {
        static auto save_timer = high_resolution_clock::now();

        if (!m.empty())      {
        if (first_time_) {
        time_stamp_ = to_iso_string(boost::posix_time::second_clock::local_time());
        std::cout << "Created first video file: " << directory_ << time_stamp_ <<
".avi" << std::endl;
        video_writer_.open(directory_ + time_stamp_ + ".avi",
        codec_,
        Helper::record_fps,
        Size(int(Helper::window_width),
        int(Helper::window_height)), true);
```

```
        first_time_ = false;
        }

        Mat frame_out;
        resize(m, frame_out, Size(int(Helper::window_width),
int(Helper::window_height)));
        video_writer_.write(frame_out);

        if (duration_cast<seconds>(high_resolution_clock::now() - save_timer).count()
>= Helper::record_save_time_s) {
        time_stamp_ = to_iso_string(boost::posix_time::second_clock::local_time());
        std::cout << "Saving video to file: " << directory_ << time_stamp_ << ".avi"
<< std::endl;
        video_writer_.open(directory_ + time_stamp_ + ".avi",
        codec_,
        Helper::record_fps,
        Size(int(Helper::window_width),
        int(Helper::window_height)), true);

        save_timer = high_resolution_clock::now();
        }
        }
        }

        inline void VideoRecorder::Run() {
        try     {
        while (!Helper::stop) {
        if(!safe_mat_queue.empty()) {
        Mat t = safe_mat_queue.GetHead();
```

```
_Run(t);
}
}
}
catch (...){}
}
```

B.1.3 Файл Classification.h

```
#pragma once

#include <opencv2/dnn/dnn.hpp>
#include <opencv2/imgproc.hpp>

#include "helper.h"

/* визначаємо відносний шлях моделі */
constexpr auto classification_model_path = ("./models/classifier.pb");

using namespace cv;
using namespace dnn;

class Classification {
static Net classification_net_;
constexpr static float sign_classify_thresh = 0.99f;
constexpr static float sign_scale_input = 0.003921f; // нормалізувати до 1/255
static const int sign_res_input_px = 32;
public:
static bool LoadModel();
```

```
        static void ClassifySign(const Mat& sign_classify, Mat& sign_img,
std::string& limit_str);
        };

        inline bool Classification::LoadModel() {
        if (!Helper::IsFileExist(classification_model_path)) {
        return false;
        }

        classification_net_ = readNetFromTensorflow(classification_model_path, "");
        classification_net_.setPreferableBackend(DNN_BACKEND_DEFAULT);
        classification_net_.setPreferableTarget(DNN_TARGET_CPU);

        return true;
        }

        inline void Classification::ClassifySign(const Mat& sign_classify, Mat&
sign_img, std::string& limit_str) {
        /* зменшуємо розмір до helper::sign_res_input_px та обробляємо в сірий */
        Mat blob;
        cvtColor(sign_classify, blob, COLOR_BGR2GRAY);
        resize(blob, blob, Size(sign_res_input_px, sign_res_input_px));
        blobFromImage(blob,
        blob,
        sign_scale_input,
        Size(sign_res_input_px, sign_res_input_px),
        Scalar(), false, false
        );
```

```
        classification_net_.setInput(blob);
        Mat prob = classification_net_.forward();

        Point class_id_point;
        double confidence;
        minMaxLoc(prob.reshape(1, 1), nullptr, &confidence, nullptr,
&class_id_point);
        int class_id = class_id_point.x;

        if ((confidence >= sign_classify_thresh) && (class_id >= 0) && (class_id <
9)) {
        limit_str = Helper::GetLimitString(class_id);
        sign_img = sign_classify;
        }
        }
```

В.1.4 Файл detection.h

```
        #pragma once

        #include <iostream>

        #include <opencv2/core/base.hpp>
        #include <opencv2/dnn/dnn.hpp>
        #include <opencv2/imgproc.hpp>

        #include "helper.h"

        /* Визначення відносного шляху моделі */
        constexpr auto detection_model_path = ("./models/detection.pb");
```

```
constexpr auto detection_config_path = ("./models/detection.pbtxt");

using namespace cv;
using namespace dnn;

class Detection {
static Net detection_net_;
static std::vector<String> out_names_;
static const int target_class_id = 0;
static const int detection_input_width = 300;
static const int detection_input_height = 300;
constexpr static float sign_detect_thresh = 0.90f;
constexpr static float sign_extend_px = 0.008f;

public:
static bool LoadModel();
static bool FindSign(Mat& frame, const std::vector<Mat>& outs, Mat&
sign_img);
static void ProcessingThread();
};

inline bool Detection::LoadModel(){
if (!Helper::IsFileExist(detection_config_path) ||
!Helper::IsFileExist(detection_model_path)) {
return false;
}

detection_net_ = readNetFromTensorflow(detection_model_path,
detection_config_path);
```

```
detection_net_.setPreferableBackend(DNN_BACKEND_DEFAULT);
detection_net_.setPreferableTarget(DNN_TARGET_CPU);
out_names_ = detection_net_.getUnconnectedOutLayersNames();

return true;
}

inline bool Detection::FindSign(Mat& frame, const std::vector<Mat>& outs, Mat& sign_img) {
for (const auto& out : outs) {

const auto data = reinterpret_cast<float*>(out.data);
for (size_t i = 0; i < out.total(); i += 7) {
const float confidence = data[i + 2];
const int class_id = static_cast<int>(data[i + 1]) - 1;

/* Перевіра збігання порогового і цільового класів */
if ((confidence > sign_detect_thresh) && (class_id == target_class_id)) {
const float left =   data[i + 3];
const float top =    data[i + 4];
const float right =  data[i + 5];
const float bottom = data[i + 6];

const int img_width = frame.size().width;
const int img_height = frame.size().height;

const auto box_left =   static_cast<int>(std::max(int((left - sign_extend_px) * img_width), 0));
```

```
const auto box_top =    static_cast<int>(std::max(int((top - sign_extend_px) *
img_height), 0));
const auto box_right =  static_cast<int>(std::min(int((right + sign_extend_px)
* img_width), img_width - 1));
const auto box_bottom = static_cast<int>(std::min(int((bottom +
sign_extend_px) * img_height), img_height - 1));

/* заміна розміру знака */
sign_img = frame(Rect(box_left, box_top,
abs(box_right - box_left),
abs(box_bottom - box_top))).clone();

resize(sign_img, sign_img,
Size(Helper::sign_res_display_width, Helper::sign_res_display_width),
INTER_CUBIC);
return true;
}
}
}
return false;
}

inline void Detection::ProcessingThread() {
Mat blob;
std::cout << "Processing Thread started..." << std::endl;
while (!Helper::stop) {
/* Отримання кадру */
Mat frame;
{
```

```
if (!Helper::frames_queue.empty()){
frame = Helper::frames_queue.GetHead();
/* пропуск решти карів */
Helper::frames_queue.Clear();
}
}

/* Висновок */
if (!frame.empty()) {
blobFromImage(frame,
blob, 1.0,
Size(detection_input_width, detection_input_height),
Scalar(),
true
);

detection_net_.setInput(blob);

std::vector<Mat> outs;
detection_net_.forward(outs, out_names_);
Helper::predictions_queue.Push(outs);
Helper::processed_frames_queue.Push(frame);
}
}
}
```

В.1.5 Файл helper.h

```
#pragma once
```

```
#include <csignal>
#include <fstream>

#include <boost/program_options.hpp>

using namespace cv;
namespace program_options = boost::program_options;

/* Зміна id камери */
constexpr auto camera_id = 0;

class Helper {
public:
static bool use_video;
static bool record_video;

static const int sign_res_display_width = 128;
constexpr static float input_height = 440.0f;
constexpr static float window_width = 800.0f;
constexpr static float window_height = 480.0f;
static const int record_save_time_s = 60;
static int record_fps;
static ThreadSafeQueue<Mat> frames_queue;
static ThreadSafeQueue<Mat> processed_frames_queue;
static ThreadSafeQueue<std::vector<Mat>> predictions_queue;
static volatile sig_atomic_t stop;

static bool GrabFrame(VideoCapture& video_capture, Mat& frame);
```

```
        static bool IsFileExist(const char* file_name);
        static bool OpenInputSource(VideoCapture& video_capture, const std::string&
input_src);
        static void ParseCommandLineOptions(program_options::variables_map&
variables_map, int argc, char** argv);
        static std::vector<int> SignSpeeds;
        static std::string GetLimitString(int class_id);
        };

        inline bool Helper::GrabFrame(VideoCapture& video_capture, Mat& frame) {
        /* захват кадру та добавлення до черги */
        video_capture >> frame;
        if (!frame.empty()) {
        Helper::frames_queue.Push(frame.clone());
        }
        else if (Helper::use_video) {
        std::cout << "End of video reached, exit.." << std::endl;
        return false;
        }
        else {
        std::cerr << "Error connecting to camera, exit.." << std::endl;
        return false;
        }

        /* Зменшення вхідного зображення для відображення */
        const float ratio = Helper::input_height / frame.size().height;
        resize(frame, frame, Size(), ratio, ratio, INTER_LINEAR);
        return true;
        }
```

```
inline bool Helper::IsFileExist(const char* file_name) {
std::ifstream infile(file_name);
return infile.good();
}

inline bool Helper::OpenInputSource(VideoCapture& video_capture, const
std::string& input_src = "") {
try {
/* Відкриття камери */
if (input_src.empty()) {
video_capture.open(camera_id);
record_fps = int(video_capture.get(CAP_PROP_FPS));
int cam_width =  int(video_capture.get(CAP_PROP_FRAME_WIDTH));
int cam_height = int(video_capture.get(CAP_PROP_FRAME_HEIGHT));
std::cout << "Opened camera with resolution " << cam_width << "x" <<
cam_height << " and FPS: " << record_fps << ".." << std::endl;
}

/* Відкрити відео */
else {
video_capture.open(input_src);
int video_width = int(video_capture.get(CAP_PROP_FRAME_WIDTH));
int video_height = int(video_capture.get(CAP_PROP_FRAME_HEIGHT));
std::cout << "Opened video with resolution " << video_width << "x" <<
video_height << ".." << std::endl;
}
}
catch (...) {
```

```
        std::cerr << "Unable to open our input source, exit!" << std::endl;
        return false;
        }

        if (!video_capture.isOpened()) {
        std::cerr << "Unable to open our input source, exit!" << std::endl;
        return false;
        }

        return true;
        }

        inline void
Helper::ParseCommandLineOptions(program_options::variables_map&
variables_map, int argc, char** argv) {
        try {
        program_options::options_description desc("Allowed options");
        desc.add_options()
        ("help", "produce help message")
        ("video,v", program_options::value<std::string>(), "use test video instead of
camera")
        ("record,r", "record output to video file");

        store(parse_command_line(argc, argv, desc), variables_map);
        notify(variables_map);

        /* Допомога */
        if (variables_map.count("help")) {
        std::cout << desc << "\n";
```

```
exit(EXIT_FAILURE);
}

/* замість камери використовуємо тест-відео */
if (variables_map.count("video")) {
std::cout << "Using video instead of camera input.." << std::endl;
use_video = true;
}

/* запис зображення в файл */
if (variables_map.count("record")) {
std::cout << "Recording video with resolution " << window_width << "x" <<
window_height <<
" and " << record_fps << " FPS.." << std::endl;
record_video = true;
}
}
catch (boost::program_options::error& error) {
std::cerr << "ERROR: " << error.what() << std::endl;
exit(EXIT_FAILURE);
}
}

inline std::string Helper::GetLimitString(const int class_id) {
const int speed = SignSpeeds[class_id];
if (speed != 0) {
return "Limit: " + std::to_string(speed) + " km/h";
}
```

```
return "Limit: - km/h";
}
```

B.1.6 Файл safe_queue.h

```
#pragma once

#include <queue>
#include <mutex>

#include <opencv2/core/utility.hpp>

template <typename T>
class ThreadSafeQueue : public std::queue<T> {
public:
ThreadSafeQueue() : counter(0) {}

void Push(const T& to_push)    {
std::lock_guard<std::mutex> g(mutex_);

std::queue<T>::push(to_push);
counter++;

/* рахунок з другого кадру, тест */
if (counter == 1) {
tick_meter_.reset();
tick_meter_.start();
}
}
```

```
T GetHead() {
std::lock_guard<std::mutex> g(mutex_);
T entry = this->front();
this->pop();
return entry;
}

float GetFPS() {
tick_meter_.stop();
const double fps = counter / tick_meter_.getTimeSec();
tick_meter_.start();
return static_cast<float>(fps);
}

void Clear() {
std::lock_guard<std::mutex> g(mutex_);
while (!this->empty()) {
this->pop();
}
}

unsigned int counter;

private:
cv::TickMeter tick_meter_;
std::mutex mutex_;
};
```