Міністерство освіти і науки України
Харківський національний університет імені В. Н. Каразіна
Факультет комп'ютерних наук
Кафедра теоретичної та прикладної системотехніки

«Затверджую»
Зав. кафедри теоретичної та
прикладної системотехніки
_____ д.т.н., проф. С. І. Шматков
«___» _____ 2024 р

Пояснювальна записка

до кваліфікаційної роботи
бакалавра

на тему: «Модель управління дорожнім рухом на платформі Android»

Захищено на засіданні
Атестаційної комісії № 44
протокол № __ від __.06.2024 р.
Оцінка _____ / _____
Голова Атестаційної комісії
_____ Скоб Ю. О.
(підпис) (прізвище та ініціали)

Виконав:
студент 4 курсу, групи КУ– 41
Галузь знань: 15 – Автоматизація та приладобудування
Спеціальність: 151 – «Автоматизація та комп'ютерно-інтегровані технології»
Станішевський Роман Олександрович

(підпис)

Керівник:
Старший викладач
Павлов Анатолій Миколайович

Рецензент:
канд. техн. наук
Колованова Є. П.

_____
(підпис)

Харків – 2024

АНОТАЦІЯ

Пояснювальна записка до кваліфікаційної роботи бакалавра складається зі вступу, трьох розділів, висновків, списку використаних джерел і чотирьох додатків. Загальний обсяг роботи складає 51 сторінку, із яких 30 сторінок основної частини з 16 рисунками та 12 найменуваннями списку використаних джерел та чотирма додатками.

Метою кваліфікаційної роботи є розробка моделі управління дорожнім рухом на платформі Android з метою оптимізації потоків транспорту, зменшення аварійності та поліпшення загальної безпеки учасників дорожнього руху.

Об'єкт дослідження – процес управління дорожнім рухом, який включає в себе координацію руху транспортних засобів, регулювання дорожнього руху, а також контроль за безпекою на дорогах.

Предмет дослідження – розробка програмного забезпечення для мобільних пристроїв на базі операційної системи Android, спрямованого на покращення управління дорожнім рухом шляхом використання сучасних технологій моніторингу, аналізу та управління транспортними потоками.

Завдання, яке вирішується в кваліфікаційній роботі полягає у розробці алгоритмів для розпізнавання дорожніх об'єктів, використання датчиків та геолокаційних сервісів для збору необхідної інформації, створення інтерфейсу для взаємодії з користувачем, а також тестування та аналіз ефективності розробленої системи. Це завдання має на меті покращення безпеки на дорозі та оптимізації управління дорожнім рухом за допомогою сучасних технологій та програмних рішень.

Область застосування – розробка веб-додатку, який відповідає за дорожню безпеку.

Ключові слова: модель, дорожній рух, архітектура, платформа Android, OpenCv.

ABSTRACT

The explanatory note to the bachelor's thesis consists of an introduction, three chapters, conclusions, a list of references and four appendices. The total volume of the work is 51 pages, including 30 pages of the main part with 16 figures and 12 references and four appendices.

The purpose of the qualification work is to develop a traffic management model on the Android platform in order to optimize traffic flows, reduce accidents and improve the overall safety of road users.

The object of research is the process of traffic management, which includes the coordination of traffic, traffic control, and road safety control.

The subject of research is the development of software for mobile devices based on the Android operating system aimed at improving traffic management through the use of modern technologies for monitoring, analyzing and managing traffic flows.

The task to be solved in the qualification work is to develop algorithms for recognizing road objects, using sensors and geolocation services to collect the necessary information, creating an interface for user interaction, as well as testing and analyzing the effectiveness of the developed system. This task is aimed at improving road safety and optimizing traffic management using modern technologies and software solutions.

The scope is the development of a web application responsible for road safety.

Keywords: model, traffic, architecture, Android platform, OpenCv.

.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ І УМОВНИХ ПОЗНАЧЕНЬ

СУДР – система управління дорожнім рухом;

ШІ – штучний інтелект;

IoT – (англ. Internet of Things) Інтернет речей;

GPS – (англ. Global Positioning System) глобальна система позиціонування;

ITC – (англ. Intelligent Transportation System) інтелектуальна транспортна система;

ДРМ – дорожня розмітка;

ІСУДР – інформаційна система управління дорожнім рухом;

HSV – (англ. Hue, Saturation, Value) тон, насиченість, яскравість;

АЗ – апаратне забезпечення;

ПЗ – програмне забезпечення;

API – (англ. Application Programming Interface) інтерфейс програмування додатків;

SDK – (англ. Software Development Kit) набір засобів розробки програмного забезпечення.

ВСТУП

В сучасному світі швидкість розвитку технологій та зростання мобільності населення створюють нагальну потребу в ефективних та інноваційних рішеннях управління дорожнім рухом. З впровадженням смартфонів у повсякденне життя виникає можливість застосування їх як потужного інструменту для вирішення проблем транспортної інфраструктури. Дана дипломна робота присвячена розробці моделі управління дорожнім рухом на платформі Android, яка поєднує в собі переваги мобільних технологій та алгоритмів управління трафіком.

Слід розробити інноваційну систему, яка забезпечить оптимізацію потоків транспорту, покращить безпеку дорожнього руху та зменшить час очікування на дорогах. Застосування мобільних технологій дозволить активно залучати користувачів до участі в оптимізації транспортної інфраструктури, що робить цей проект актуальним та перспективним у контексті сучасних викликів управління міським рухом.

Актуальність теми. По-перше, кількість автомобілів у світі постійно зростає, що призводить до збільшення завантаженості доріг та заторів. По-друге, щільна забудова міст ускладнює рух транспорту, тому важливо мати дієві інструменти для оптимізації транспортних потоків. Крім того, у багатьох містах існує нестача паркувальних місць, що створює додаткові затримки та незручності для водіїв. Транспорт є одним із основних джерел забруднення довкілля, тому впровадження ефективних моделей управління дорожнім рухом сприятиме зменшенню викидів шкідливих речовин. Нарешті, затори та несприятливі дорожні умови призводять до збільшення кількості ДТП, тому моделі управління дорожнім рухом, реалізовані на платформі Android, можуть підвищити рівень безпеки на дорогах. Таким чином, розробка та впровадження моделі управління дорожнім рухом на платформі Android є важливою для

вирішення сучасних проблем міської інфраструктури, покращення екологічної ситуації та підвищення рівня безпеки на дорогах.

Метою дослідження є розробка моделі управління дорожнім рухом на платформі Android з метою оптимізації потоків транспорту, зменшення аварійності та поліпшення загальної безпеки учасників дорожнього руху.

Об'єкт дослідження – процес управління дорожнім рухом, який включає в себе координацію руху транспортних засобів, регулювання дорожнього руху, а також контроль за безпекою на дорогах.

Предмет дослідження – розробка програмного забезпечення для мобільних пристроїв на базі операційної системи Android, спрямованого на покращення управління дорожнім рухом шляхом використання сучасних технологій моніторингу, аналізу та управління транспортними потоками.

Методи дослідження: моделювання, розробка програмного забезпечення, спостереження, тестувальні платформи.

Завдання дослідження

1. Визначити основні вимоги до системи управління дорожнім рухом на платформі Android.
2. Розробити прототип мобільного додатку для платформи Android, що реалізує функціонал управління дорожнім рухом.
3. Провести тестування програмного забезпечення в умовах реального часу.

РОЗДІЛ 1
АНАЛІЗ СУЧАСНИХ СИСТЕМ УПРАВЛІННЯ ДОРОЖНІМ РУХОМ

1.1 Аналіз існуючих систем управління дорожнім рухом

Системи управління дорожнім рухом (СУДР) відіграють важливу роль у забезпеченні безпечного, ефективного та пропускного руху на дорогах. Завдяки збору даних про трафік, аналізу цих даних та динамічному регулюванню світлофорів, дорожніх знаків та інших елементів інфраструктури, СУДР можуть допомогти зменшити затори, покращити час у дорозі та знизити кількість аварій [1].

Існує багато різних типів СУДР, які можна класифікувати за різними критеріями, такими як:

- Масштаб: локальні, міські, регіональні, національні;
- Функціональність: адаптивні, неадаптивні, інтегровані;
- Технологія: традиційні, засновані на датчиках, на основі штучного інтелекту.

Класичні СУДР використовують фіксовані програми світлофорів та дорожніх знаків, які не змінюються залежно від умов трафіку. Ці системи, як правило, є простими та недорогими, але вони не можуть ефективно адаптуватися до мінливих потоків трафіку, що може призвести до заторів та затримок.

Адаптивні СУДР використовують датчики та інші технології для збору даних про трафік в режимі реального часу. Ці дані використовуються для динамічного регулювання світлофорів та дорожніх знаків, щоб оптимізувати потік трафіку. Адаптивні СУДР можуть значно покращити пропускну спроможність та зменшити затори, але вони, як правило, є більш складними та дорогими, ніж класичні СУДР.

Інтегровані СУДР поєднують в собі елементи класичних та адаптивних СУДР, а також інші системи, такі як системи громадського транспорту, системи навігації та системи стягнення плати за проїзд. Ці системи можуть надавати комплексне управління дорожнім рухом, яке враховує всі види транспорту та умови руху. Інтегровані СУДР є найскладнішими та найдорожчими, але вони можуть мати найбільший вплив на пропускну спроможність, безпеку та ефективність дорожнього руху.

Використання штучного інтелекту (ШІ) в СУДР. ШІ стає все більш важливим інструментом для розробки та експлуатації СУДР. Алгоритми ШІ можуть використовуватися для аналізу даних про трафік в режимі реального часу, прогнозування потоків трафіку та прийняття оптимальних рішень щодо управління дорожнім рухом. ШІ може допомогти СУДР стати більш адаптивними, ефективними та проактивними.

Переваги використання СУДР:

- Зменшення заторів та затримок на дорогах.
- Покращення часу в дорозі для користувачів.
- Зниження ризику аварій і підвищення загальної безпеки на дорозі.
- Мінімізація викидів забруднюючих речовин у навколишнє середовище.
- Зменшення витрат на пальне та загальна економія ресурсів.
- Підвищення комфорту під час водіння та зменшення стресу для водіїв.
- Вирішення викликів, пов'язаних із впровадженням СУДР.
- Виявлення високої вартості впровадження та експлуатації систем.
- Необхідність збору та аналізу великих обсягів даних для ефективного функціонування.
- Складнощі інтеграції з іншими існуючими системами на дорозі.
- Потенційні проблеми кібербезпеки, що виникають внаслідок використання технологій управління дорожнім рухом.

СУДР постійно розвиваються з появою нових технологій. ШІ, хмарні обчислення та Інтернет речей (IoT) відіграватимуть все більшу роль у розробці та експлуатації СУДР наступного покоління. Ці системи матимуть можливість надавати ще більш ефективне та проактивне управління дорожнім рухом, що призведе до більш безпечного, економного та зручного досвіду водіння для всіх [2].

1.2. Використання геолокаційних сервісів у дорожньому русі

Геолокаційні сервіси стають все більш важливим інструментом для управління дорожнім рухом. Завдяки цим сервісам транспортні органи, водії та інші зацікавлені сторони можуть отримувати та обмінюватися інформацією про дорожню ситуацію в режимі реального часу [3].

Використання геолокаційних сервісів у дорожньому русі має багато переваг, зокрема:

- Покращення моніторингу дорожньої ситуації: транспортні органи можуть відстежувати рух транспорту, виявляти затори, аварії та інші дорожні події. Ця інформація може бути використана для покращення координації роботи дорожніх служб та прийняття більш ефективних рішень щодо управління дорожнім рухом.
- Надання інформації водіям: водії можуть отримувати інформацію про дорожню ситуацію в режимі реального часу. Це може допомогти їм вибирати оптимальні маршрути, уникати заторів та планувати час поїздки.
- Підвищення безпеки дорожнього руху: водії можуть бути попереджені про небезпечні ділянки дороги, такі як зони з високим рівнем аварійності або ремонтні роботи. Це може допомогти їм їздити більш обережно та знизити ризик ДТП.
- Оптимізація роботи громадського транспорту: геолокаційні сервіси можуть бути використані для покращення координації роботи громадського транспорту. Це може допомогти транспортним

компаніям планувати маршрути, оптимізувати розклади та покращувати якість обслуговування пасажирів.

- Зниження забруднення довкілля: водії можуть вибирати більш економні маршрути, використовувати екологічні транспортні засоби та знижувати викиди парникових газів.

Незважаючи на численні переваги, використання геолокаційних сервісів у дорожньому русі також має певні недоліки, які необхідно враховувати:

1. Точність:

- Точність геолокаційних даних може варіюватися залежно від типу використовуваного пристрою, мережевого покриття та навколишнього середовища.
- У деяких випадках дані можуть бути неточними або застарілими, що може призвести до невірних рекомендацій щодо маршруту або завищених оцінок часу в дорозі.

2. Приватність:

- Збір та зберігання геолокаційних даних користувачів викликає занепокоєння щодо конфіденційності.
- Існує ризик витоку або несанкціонованого використання цих даних, що може призвести до порушення конфіденційності та особистої безпеки.

3. Залежність:

- Занадто сильна залежність від геолокаційних сервісів може призвести до того, що водії не будуть звертати увагу на дорожні знаки, розмітку та інші сигнали, що може призвести до небезпечних ситуацій.

4. Перевантаження мережі:

- Масове використання геолокаційних сервісів може призвести до перевантаження мереж мобільних операторів, що може призвести до зниження швидкості передачі даних та збоїв у роботі.

5. Маніпулювання:

- Існує ризик маніпулювання геолокаційними даними з метою створення штучних заторів або для того, щоб змусити водіїв використовувати певні маршрути.

6. Недоступність:

- Геолокаційні сервіси можуть бути недоступні в деяких районах з поганим мережевим покриттям або в країнах з обмеженим доступом до Інтернету.

7. Вартість:

- Використання деяких геолокаційних сервісів може бути платним, що може стати додатковим тягарем для користувачів.

8. Необхідність додаткового обладнання:

- Для використання деяких геолокаційних сервісів може знадобитися додаткове обладнання, таке як GPS-навігатори або смартфони з підтримкою GPS.

9. Несумісність:

- Різні геолокаційні сервіси можуть використовувати різні формати даних, що може ускладнити обмін інформацією між ними.

10. Недосконалість алгоритмів:

- Алгоритми, які використовуються в геолокаційних сервісах для розрахунку маршрутів та прогнозування часу в дорозі, не завжди є досконалими.
- Вони можуть не враховувати всі фактори, що впливають на дорожній рух, що може призвести до неточних результатів.

Важливо зазначити, що переваги використання геолокаційних сервісів у дорожньому русі переважають їх недоліки. З розвитком технологій та покращенням точності, конфіденційності та надійності геолокаційних сервісів їх роль у дорожньому русі буде лише зростати.

Існує багато різних способів використання геолокаційних сервісів у дорожньому русі, зокрема:

- Моніторинг трафіку: транспортні органи використовують GPS-датчики та інші технології для відстеження руху транспорту на дорогах.
- Навігаційні програми: водії використовують навігаційні програми на своїх смартфонах або інших пристроях, щоб отримувати покрокові інструкції щодо проїзду.
- Додатки для спільних поїздок: додатки для спільних поїздок дозволяють людям знаходити попутників для подорожей.
- Системи керування паркуванням: системи керування паркуванням допомагають водіям знайти вільні паркувальні місця.

Геолокаційні сервіси є основою для розвитку ITC, які поєднують транспортні технології, інфраструктуру та дані для покращення мобільності та безпеки на дорогах. ITC можуть використовуватися для:

- Динамічного управління світлофорами.
- Відображення інформації про дорожню ситуацію на інформаційних панелях змінної інформації.
- Впровадження електронних систем оплати проїзду.
- Розробки автономних транспортних засобів.

Геолокаційні сервіси можуть використовуватися для збору даних про дорожній рух, які можуть бути використані для аналізу та прогнозування транспортних потоків.

Використання геолокаційних сервісів у дорожньому русі викликає певні проблеми конфіденційності. Важливо, щоб дані про місцезнаходження людей збиралися та використовувалися з дотриманням етичних принципів та відповідно до чинного законодавства [4].

1.3 Методи прогнозування та аналізу дорожньої ситуації

Прогнозування та аналіз дорожньої ситуації є важливими завданнями для забезпечення безпечного, ефективного та пропускного руху на дорогах. Завдяки знанню про поточний стан та динаміку зміни дорожньої ситуації

можна вживати заходів для оптимізації дорожнього руху, запобігання заторів, зниження аварійності та покращення загальної пропускної спроможності дорожньої мережі [5].

Існує багато різних методів прогнозування та аналізу дорожньої ситуації, які можна поділити на кілька категорій:

1. Тип даних:
    - Дані про трафік з датчиків: Ці дані збираються з датчиків, встановлених на дорогах, які вимірюють такі параметри, як швидкість руху, щільність трафіку та час очікування на перехрестях.
    - Дані про GPS: Ці дані збираються з GPS-пристроїв, встановлених у транспортних засобах або смартфонах, які надають інформацію про місцезнаходження, швидкість та напрямок руху транспортних засобів.
    - Дані з соціальних мереж: Ці дані збираються з соціальних мереж, таких як Twitter та Facebook, де люди публікують інформацію про дорожню ситуацію, затори та аварії.
    - Дані з відеокамер: Ці дані збираються з відеокамер, встановлених на дорогах, які можуть використовуватися для відстеження потоку трафіку та виявлення інцидентів.
2. Методи аналізу:
    - Статистичні методи: Ці методи використовують статистичні моделі для аналізу даних про трафік та прогнозування майбутніх умов руху.
    - Методи машинного навчання: Ці методи використовують алгоритми машинного навчання для виявлення закономірностей у даних про трафік та прогнозування майбутніх умов руху.
    - Фізичні моделі: Ці моделі ґрунтуються на фізичних законах для моделювання потоку трафіку та прогнозування його поведінки.
3. Застосування:

- Маршрутизація: Прогнозування дорожньої ситуації може використовуватися для розробки оптимальних маршрутів для водіїв, враховуючи затори, час очікування на перехрестях та інші фактори.
- Управління світлофорами: Прогнозування дорожньої ситуації може використовуватися для динамічного регулювання світлофорів, щоб оптимізувати потік трафіку та зменшити затори.
- Попередження про затори: Прогнозування дорожньої ситуації може використовуватися для попередження водіїв про затори та інші небезпеки на дорозі.
- Планування дорожньої інфраструктури: Прогнозування дорожньої ситуації може використовуватися для планування нової дорожньої інфраструктури та покращення існуючої.

Переваги використання методів прогнозування та аналізу дорожньої ситуації:

- Зменшення заторів та затримок.
- Покращення часу в дорозі.
- Зниження ризику аварій.
- Зменшення викидів забруднюючих речовин.
- Економія палива.
- Підвищення комфорту водіння.

Виклики використання методів прогнозування та аналізу дорожньої ситуації:

- Необхідність збору та аналізу великих обсягів даних.
- Складність розробки точних та надійних моделей прогнозування.
- Необхідність інтеграції з іншими системами, такими як системи управління світлофорами та системи навігації.
- Потенційні проблеми кібербезпеки.

Завдяки поєднанню різних методів можна отримати більш точні та всебічні прогнози та аналізи дорожньої ситуації. Важливо зазначити, що область прогнозування та аналізу дорожньої ситуації постійно розвивається. З появою нових технологій та збором все більших обсягів даних методи прогнозування та аналізу стають все більш точними та надійними [6].

1.4 Принципи побудови інформаційних систем для управління дорожнім рухом

Ефективне управління дорожнім рухом (ДРМ) є ключовим фактором для забезпечення безпечного, комфортного та пропускного руху на дорогах. ІСУДР відіграють важливу роль у цій сфері, надаючи інструменти та дані, необхідні для прийняття обґрунтованих рішень щодо оптимізації ДРМ.

Основні принципи побудови ІСУДР:

1. Інтеграція:
    - ІСУДР повинні інтегрувати дані з різних джерел, таких як датчики руху, камери, радари, GPS-пристрої та системи громадського транспорту.
    - Це дозволяє отримати комплексну картину дорожньої ситуації в режимі реального часу [7].
2. Аналітика:
    - ІСУДР повинні мати потужні аналітичні можливості для обробки та аналізу даних про ДРМ.
    - Це дозволяє виявляти закономірності, прогнозувати потоки трафіку та ідентифікувати проблемні ділянки.
3. Візуалізація:
    - ІСУДР повинні надавати чітку та зрозумілу візуалізацію дорожньої ситуації, використовуючи карти, графіки та діаграми.
    - Це дозволяє операторам швидко та легко отримувати необхідну інформацію.

4. Моделювання:
    - ІСУДР повинні мати можливість моделювати різні сценарії ДРМ, такі як вплив дорожніх робіт, зміни світлофорного регулювання або несприятливі погодні умови.
    - Це дозволяє оцінювати ефективність різних стратегій управління ДРМ.
5. Оптимізація:
    - ІСУДР повинні мати можливість пропонувати та реалізовувати оптимальні рішення для управління ДРМ, такі як динамічне регулювання світлофорів, маршрутизація трафіку та управління дорогами.
6. Гнучкість:
    - ІСУДР повинні бути гнучкими та масштабованими, щоб адаптуватися до мінливих потреб та умов.
    - Це дозволяє використовувати їх у різних типах міст та на дорогах з різною інтенсивністю руху.
7. Безпека:
    - ІСУДР повинні мати високий рівень безпеки, щоб захистити дані та системи від несанкціонованого доступу та кібератак.
8. Надійність:
    - ІСУДР повинні бути надійними та стійкими до збоїв, щоб гарантувати безперебійну роботу.
9. Ефективність:
    - ІСУДР повинні бути економними з точки зору витрат та ресурсів.
10. Простота використання:
    - ІСУДР повинні бути простими у використанні для операторів та диспетчерів.

Важливо зазначити, що це лише деякі з основних принципів побудови ІСУДР. Конкретні реалізації ІСУДР можуть відрізнятися залежно від потреб

та бюджету конкретного міста або дорожньої адміністрації. ІСУДР мають великий потенціал для покращення ДРМ та підвищення безпеки, комфорту та пропускної спроможності доріг [8].

Висновки за розділом 1

У даному розділі досліджується різноманітність підходів та технологій, які використовуються для керування дорожнім рухом. Аналізуються існуючі системи, використання геолокаційних сервісів, методи прогнозування та аналізу дорожньої ситуації, а також принципи побудови інформаційних систем для управління дорожнім рухом. Цей розділ надає зрозуміле уявлення про сучасні тенденції управління дорожнім рухом та служить основою для подальших розділів з розробки та впровадження нових рішень у цій сфері.

РОЗДІЛ 2
СТВОРЕННЯ СИСТЕМИ АВТОМАТИЧНОГО РОЗПІЗНАВАННЯ

2.1. Механізм функціонування детектора

Детектор дорожньої розмітки - це система, яка використовує комп'ютерний зір для автоматичного виявлення та відстеження ліній розмітки на дорозі. Ця інформація може використовуватися для різних цілей, таких як автономне водіння, системи допомоги водію та моніторинг дорожніх умов [9].

Етапи побудови детектора:

1. Передобробка даних: перед початком виявлення ліній дорожньої розмітки, дані зображення проходять через процес передобробки. Це включає фільтрацію від шуму для покращення якості зображення та векторизацію, що дозволяє перетворити зображення у формат, зручний для подальшого аналізу.
2. Виявлення ліній розмітки: на другому етапі зображення проходить через алгоритм виявлення ліній дорожньої розмітки. За допомогою спеціальних алгоритмів обробки зображень та комп'ютерного зору, система аналізує дані та визначає положення ліній на дорозі.
3. Оновлення та малювання ліній: на останньому етапі, використовуючи інформацію, отриману з попередніх кроків, система оновлює стан ліній дорожньої розмітки та малює їх на оригінальному документі або зображенні. Цей процес включає в себе малювання оновлених ліній та інших дорожніх об'єктів для подальшого аналізу та використання.

На вхід детектора дорожньої розмітки подається 3-канальне зображення формату RGB, що представляється кольорами червоний, зелений та синій. Після фільтрації та перетворення зображення, виконується аналіз та оновлення ліній розмітки. Потім, використовуючи внутрішні функції, система малює всі необхідні елементи, такі як лінії дорожньої розмітки та інші дорожні

знаки, поверх оригінального зображення для подальшого використання та аналізу.

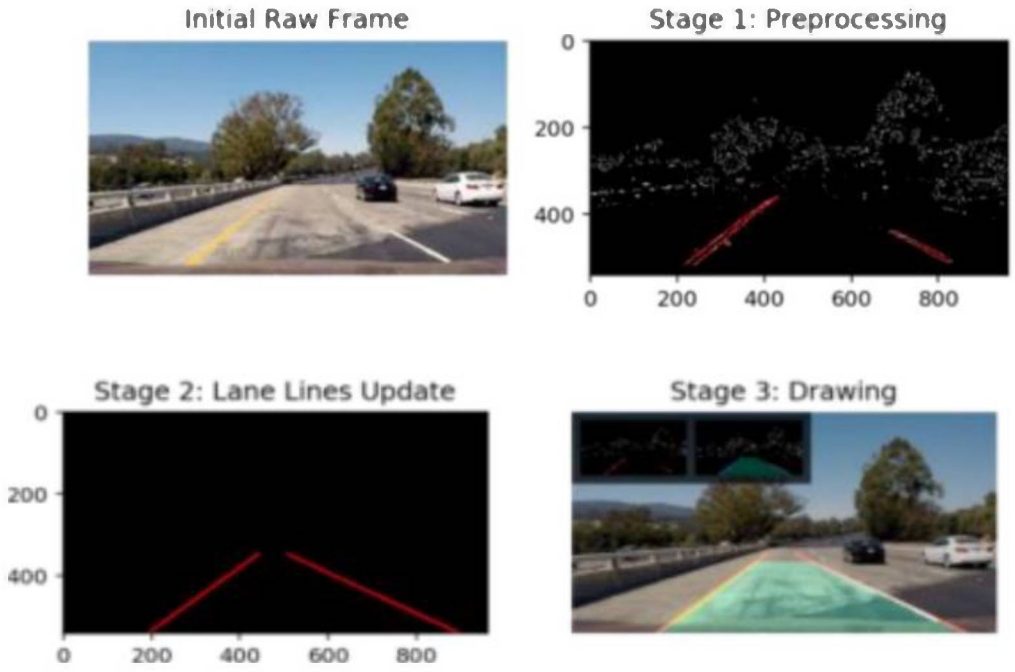

Рис. 2.1 - Послідовні етапи виявлення дорожньої розмітки

2.2. Попереднє опрацювання та перетворення векторів

Початкове RGB-зображення конвертується в колірний простір HSV, оскільки він забезпечує більш зручну платформу для виділення конкретних кольорів, зокрема відтінків жовтого і білого, які мають вирішальне значення для ідентифікації смуг руху. Цей процес зображено на рис. 2.2, який підкреслює легкість виділення жовтих відтінків у колірній моделі HSV.

Згодом зображення проходить бінаризацію, перетворюючись на бінарну маску, де зберігаються лише важливі кольори, такі як відтінки жовтого та білого. Цей бінарний результат проілюстровано на рис. 2.3.

Наступний крок передбачає векторизацію бінарного зображення за допомогою двох перетворень.

Виявлення меж Кенні: використовується оптимізований алгоритм для виявлення меж шляхом обчислення градієнтів інтенсивності в межах

зображення. Він використовує два пороги для усунення слабших меж, зберігаючи лише значущі. Алгоритм використовує фільтр на основі першої похідної Гауса. Зберігаючи лише точки з максимальним градієнтом зображення вздовж контуру межі та видаляючи точки з не максимальним градієнтом, що прилягають до межі, він забезпечує точність. Крім того, інформація про напрямок границі допомагає видаляти точки поблизу границі, не порушуючи її цілісності поблизу локальних максимумів градієнта [10].

Рис. 2.2 – Набір основних кольорів

Наступний крок полягає у використанні двох порогових значень для фільтрації слабких контурів. Кожен фрагмент контуру розглядається як єдине ціле. Якщо в будь-якій точці фрагмента значення градієнта перевищує верхній поріг, то цей фрагмент вважається прийнятним контуром. Процес продовжується до тих пір, поки значення градієнта на всьому фрагменті не стане меншим за нижній поріг. Якщо значення градієнта не перевищує верхній поріг в жодній точці фрагмента, то він вилучається.

Далі необхідно усунути "шум", тобто контури, які не є точно лініями маркера. Очевидно, що верхня частина зображення навряд чи містить маркерні лінії, тому її можна ігнорувати. Для цього горизонтальна лінія визначається як точка перетину маркерних ліній, а лінії, що знаходяться вище цієї точки, виключаються.

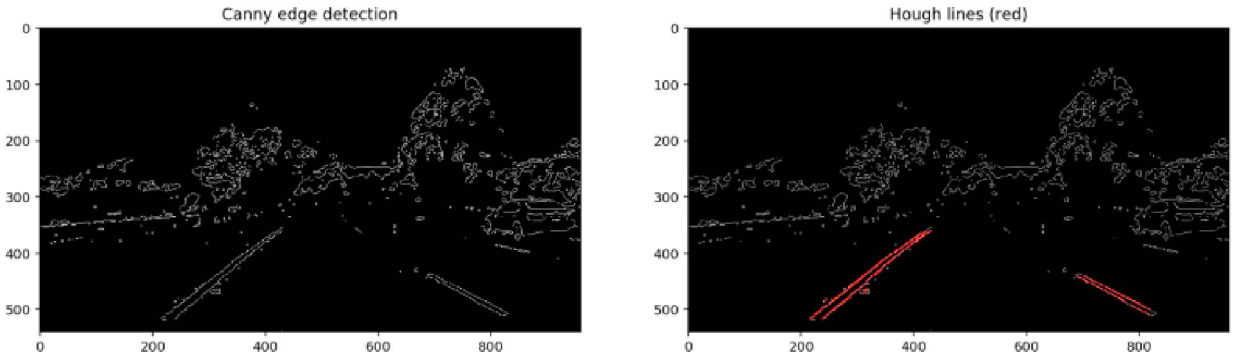

Рис. 2.3 - Висновки щодо ефективності алгоритмів Кенні та Хафа.

2.3. Актуалізація маркування на дорозі

Процес відбору ліній розмітки є ключовим етапом у детектуванні та відстеженні дорожньої розмітки на зображеннях. Цей процес використовується для того, щоб відібрати з великої кількості ліній-кандидатів, отриманих в результаті перетворення Хафа, ті, які дійсно відповідають реальним лініям розмітки на дорозі.

Алгоритм відбору використовує кілька критеріїв для оцінки ліній-кандидатів:

- Негоризонтальність: лінії, які є майже горизонтальними, не враховуються, оскільки вони не можуть бути частиною дорожньої розмітки.
- Помірний ухил: ухил лінії-кандидата має бути помірним, щоб відповідати ухилу реальних ліній розмітки. Різкі зміни ухилу можуть вказувати на помилкову лінію.
- Відповідність ухилу: розраховується різниця між ухилом лінії дорожньої розмітки та ухилом лінії-кандидата. Якщо ця різниця занадто велика, то лінія-кандидат не враховується.
- Відстань: лінія-кандидат не повинна відстояти занадто далеко від дорожньої розмітки, до якої вона може належати. Це допомагає уникнути помилкового зіставлення ліній.

- Розташування: лінія-кандидат повинна бути нижче горизонту, щоб відповідати розташуванню реальних ліній розмітки на дорозі.

Окрім основних критеріїв, алгоритм відбору може використовувати й інші фактори, такі як:

- Довжина: довгі лінії-кандидати частіше відповідають реальним лініям розмітки, ніж короткі.
- Товщина: товсті лінії-кандидати частіше відповідають реальним лініям розмітки, ніж тонкі.
- Сусідство: лінії-кандидати, які розташовані близько одна до одної, можуть бути частиною однієї і тієї ж лінії розмітки.

Параметри алгоритму відбору, такі як порогові значення для ухилу, відстані та довжини, можуть бути налаштовані в залежності від умов та типу дорожньої розмітки [11].

Алгоритм відбору дозволяє:

- Знизити кількість помилкових спрацьовувань.
- Покращити точність детектування ліній розмітки.
- Збільшити стійкість до шуму та перешкод.

Але також алгоритм відбору може:

- Пропустити деякі реальні лінії розмітки, якщо вони не відповідають критеріям відбору.
- Неправильно класифікувати деякі лінії-кандидати як лінії розмітки.

Процес відбору ліній розмітки є важливою частиною детектування дорожньої розмітки. Використання алгоритму відбору з правильно налаштованими параметрами може значно покращити точність та ефективність системи.

Через зашумленість вихідного зображення та покадрове розпізнавання стабілізація виявленої дорожньої розмітки є критично важливою. Тіні або нерівності на дорожньому покритті можуть спричинити зміну кольору розмітки, що робить її невидимою для алгоритму.

Для подолання цієї проблеми для кожної лінії розмітки підтримується буфер з десяти записів, які зберігають її попередні стани. При побудові лінії маркера в кожному наступному кадрі алгоритм враховує середнє значення буфера, тобто середнє значення кожного коефіцієнта рівняння лінії, що відповідає лінії маркера. Це допомагає згладити варіації [12].

Якщо алгоритм все ще стикається з шумом після перетворення та очищення, ймовірно, що наступна лінія-кандидат буде викидом, що може негативно вплинути на лінійну модель. Тому важливо надавати пріоритет високій точності, навіть якщо це може призвести до відсіювання правильного рядка, а не до включення викиду.

Для таких ситуацій використовується "матриця рішень" (DECISION_MAT) для визначення стабільності поточного нахилу лінії відносно середнього нахилу ліній у буфері. Наприклад, при DECISION_MAT = [[0.1, 0.9], [1, 0]] алгоритм оцінює, чи слід вважати лінію нестабільною (потенційний викид) або стабільною (її нахил знаходиться в межах середнього нахилу ліній буфера плюс-мінус поріг).

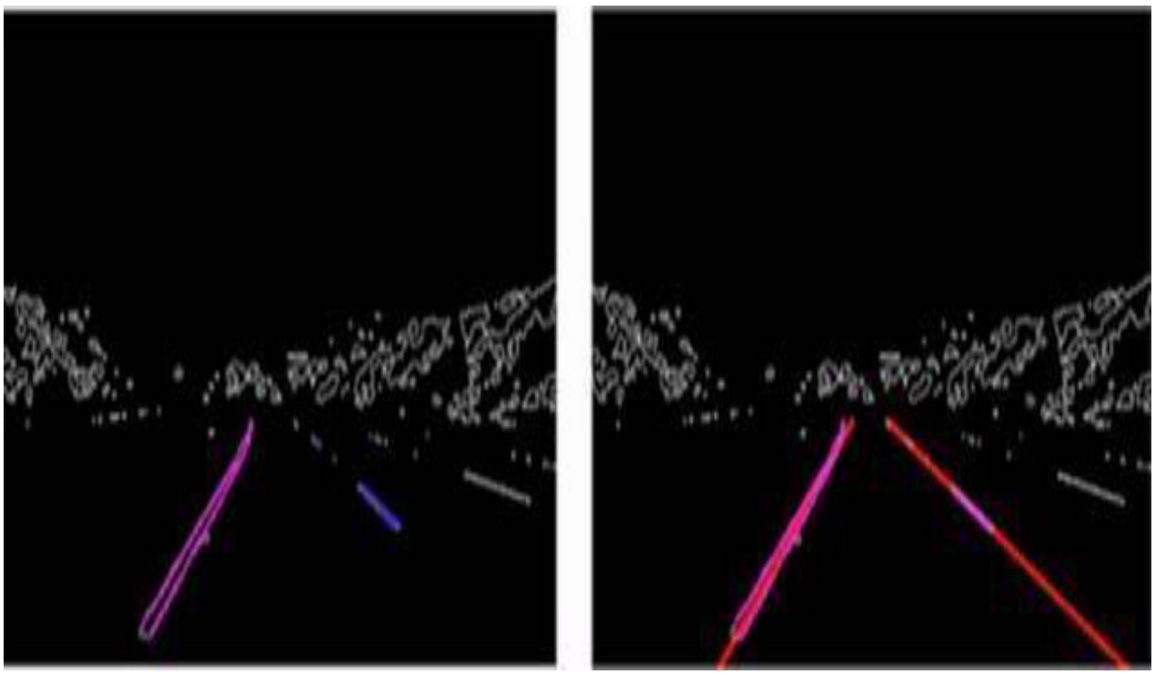

Рис. 2.4 - Ліворуч - лінії, виявлені алгоритмом Хафа, праворуч - результат їх апроксимації з фільтрацією.

Якщо лінія нестабільна, вона не буде повністю відкинута; вона все ще може надавати цінну інформацію про фактичну кривизну дороги. Такі лінії враховуються з невеликим ваговим коефіцієнтом (у цьому випадку 0.1). Для стабільних ліній використовуються їхні поточні параметри без жодних вагових коефіцієнтів з попередніх даних. На рис. 2.4 показано результат апроксимації відрізків доріг із застосованими фільтрами.

Оптимальний розмір буфера визначається шляхом візуальної оцінки точності відповідності між виявленою лінією смуги руху та фактичною лінією смуги руху на зображенні. Цей вибір ґрунтувався на відеозаписі лісової дороги з двома видимими поворотами. Без використання буферів лінія смуги руху будується незалежно для кожного кадру. Коли розмір буфера перевищує десять, суттєвого покращення якості не спостерігається, як показано на рис. 2.5.

Для того, щоб лінії були намальовані точно, ми використовуємо точку горизонту. Ця точка служить двом основним цілям:

1. Вона обмежує продовження ліній смуг руху до цієї точки.

2. Вона відфільтровує будь-які лінії Хафа, які з'являються над горизонтом.

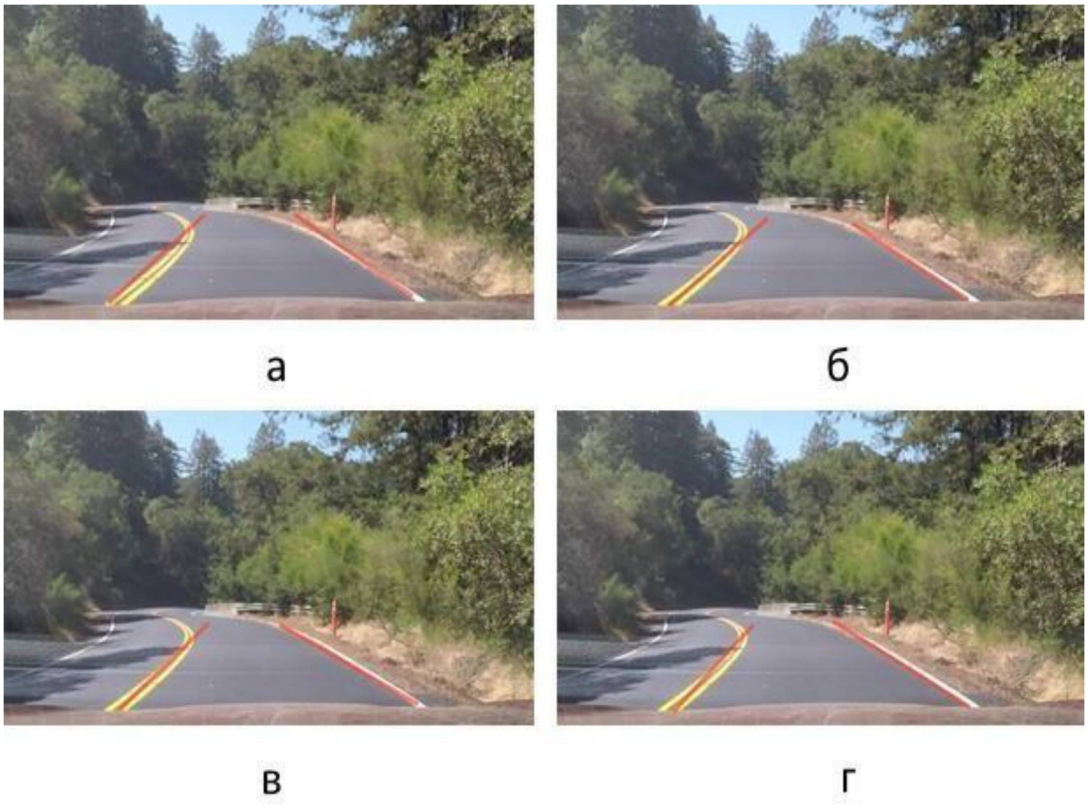

Рис. 2.5 - результати виявлення розмітки з використанням різних розмірів буфера: (а) без буфера, (б) з буфером розміром п'ять, (в) з буфером розміром десять і (г) з буфером розміром п'ятнадцять.

Висновок за розділом 2

У даному розділі ми досліджували процес створення системи автоматичного розпізнавання. Ми розглянули механізм функціонування детектора, провели аналіз попередньої обробки даних та перетворення векторів, а також дослідили процес актуалізації маркування на дорозі. Ці кроки дозволили нам краще зрозуміти і оптимізувати роботу системи автоматичного розпізнавання.

РОЗДІЛ 3
ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ МОДЕЛІ

3.1. Архітектура моделі

Установка складається з двох основних компонентів:

- АЗ складається з камери та GPS-пристрій.
- ПЗ складається з API та SDK.

АЗ системи відповідає за збір необхідних для обробки даних. Вона включає камеру, яка знімає відеоматеріали, що підлягають аналізу, і GPS-пристрій, який зв'язується з автомобілем для визначення його місцезнаходження. Ці дані про місцезнаходження необхідні для обчислення швидкості автомобіля.

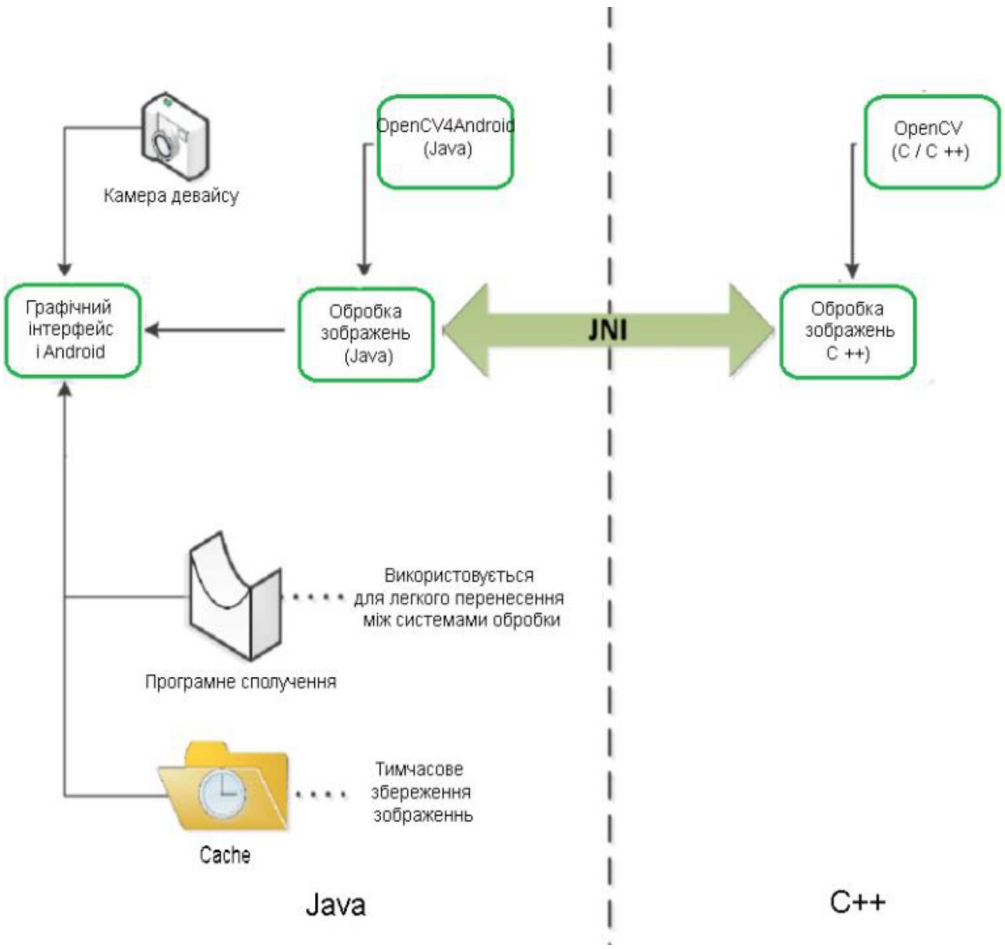

Рис. 3.1 - Архітектура системи Android, яка інтегрована з OpenCv

Камера постійно знімає кадри, які будуть оброблятися за допомогою бібліотеки OpenCV. OpenCV застосовуэться для виявлення дорожніх смуг, шукаючи білі та жовті лінії в кордонах визначеної області на екрані. Якщо смуги виявлені, бібліотека виділяє їх на зображенні.

3.2. Розробка дизайну додатку та його життєвий цикл

Діяльність в Android — це екран, який бачить і з яким взаємодіє користувач. Один додаток може містити кілька таких екранів, між якими можна перемикатися під час роботи. Цей клас відповідає за головний екран, який відображає результати виявлення дорожніх смуг і надає звуковий зворотний зв'язок.

Коли програма запускається вперше, вона встановлює значення за замовчуванням для всіх параметрів. Вона також перевіряє, чи має необхідні дозволи для доступу до камери та розташування пристрою, що необхідно для коректної роботи функцій виявлення смуг руху.

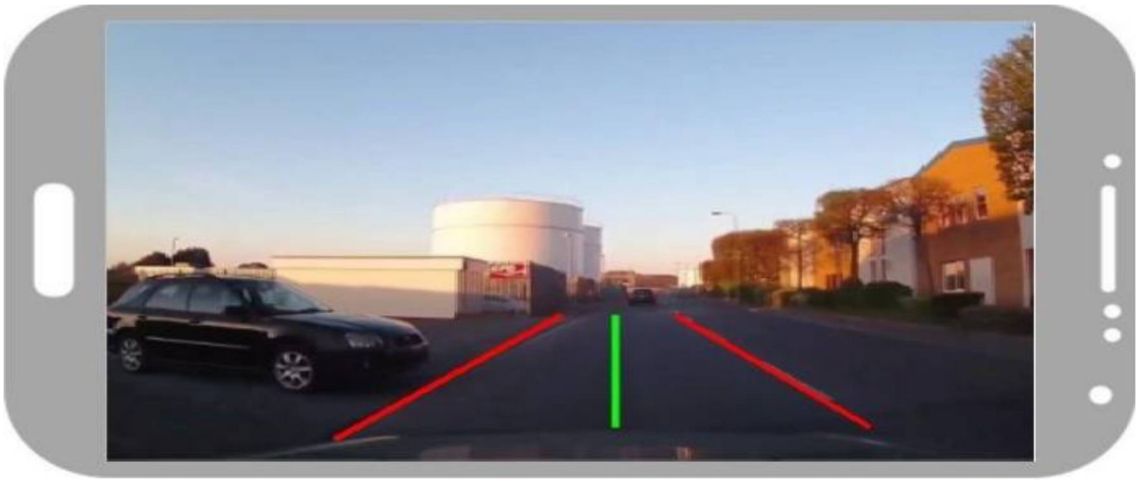

Рис. 3.2 – Дизайн додатку

Активність у середовищі Android має певний життєвий цикл. На початку система створює екземпляр активності, який проходить кілька етапів, перш ніж бути знищеним для звільнення ресурсів. Активність може перебувати в одному з трьох основних станів:

1. Активна (active або running): У цьому стані активність знаходиться на передньому плані екрана, і користувач може безпосередньо взаємодіяти з нею.
2. Призупинена (paused): Активність більше не має фокусу, але залишається видимою. Це означає, що інша активність частково її перекриває. У цьому стані активність може бути знищена системою, якщо пристрій відчуває нестачу пам'яті.
3. Зупинена (stopped): Активність повністю закрита іншою активністю і більше не видима користувачу. Якщо система потребує більше пам'яті для інших процесів, зупинена активність може бути знищена.

Коли система знищує активність, і її потрібно знову показати на екрані, вона повинна бути повністю перезапущена. При цьому активність має відновити свій попередній стан, щоб користувач міг продовжити роботу з того місця, де він зупинився.

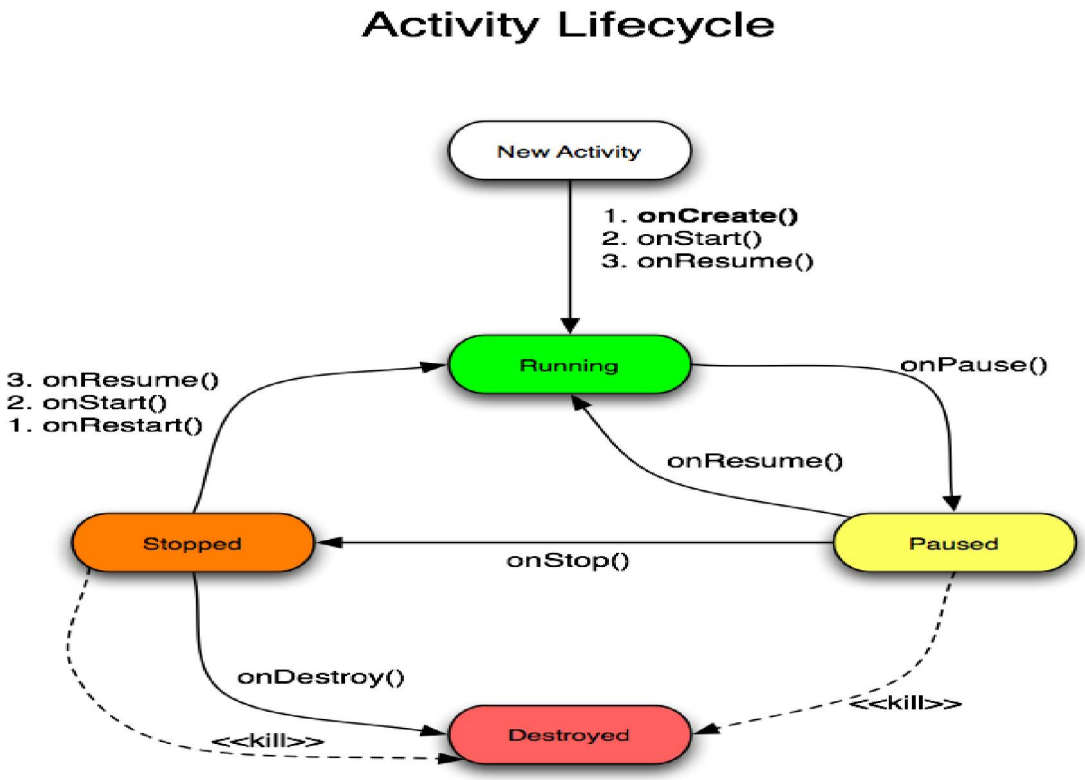

Рис. 3.3 – Життєвий цикл

3.3 Розпізнавання дорожньої розмітки

На зображенні 3.4 видно жовту лінію. У моделі кольорів RGB червоний і зелений компоненти утворюють цей відтінок, тому на зображенні 3.5 ця лінія відображається як біла. Це означає, що в цій області було виявлено значну кількість червоного кольору. Діаграма на зображенні 3.6 показує, що як жовтий, так і білий мають високе значення в просторі кольорів HSV, що робить їх ідеальними для виявлення дорожніх смуг. Оскільки всі маркування дорожньої смуги відносяться до одного з цих двох кольорів, це полегшує їх виявлення за допомогою алгоритмів обробки зображень.

Рис. 3.4 – Тест без обробки

Рис. 3.5 - Результат, що виникає після сортування за кольором

Після того, як зображення оброблено для вилучення двох конкретних кольорів, створюється маска, яка являє собою нове зображення, де всі пікселі відображаються або як чистий чорний, або як чистий білий колір. Після цього моменту застосовується певний код, який перевіряє, чи значення пікселів у створених кадрах знаходяться в певних межах, і тільки ці пікселі будуть відображені на масці. Цей процес допомагає зменшити вплив інших об'єктів, які не є дорожніми смугами, на кінцевий результат.

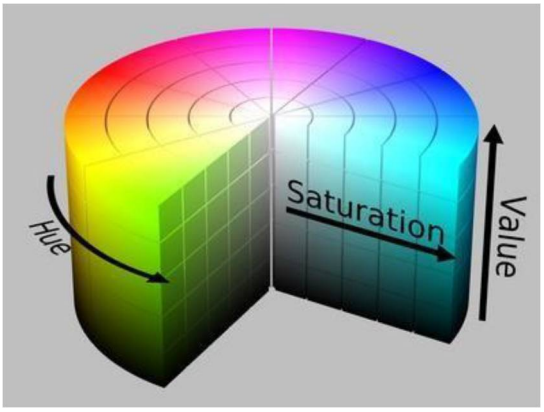

Рис. 3.6 - Графічне зображення спектра кольорів

На рис. 3.7 додаток виконує завдання з пошуку та аналізу ліній у певній області, яка може включати лінії на дорозі. Після знаходження цих ліній вони перевіряються на відповідність певному діапазону кутів нахилу. Це пояснюється тим, що лінії руху зазвичай виглядають так, що здається, ніби вони збігаються до однієї точки на горизонті, з точки зору водія. Встановлення такого обмеження допомагає зменшити кількість помилкових результатів виявлення ліній та забезпечує більш точні результати аналізу.

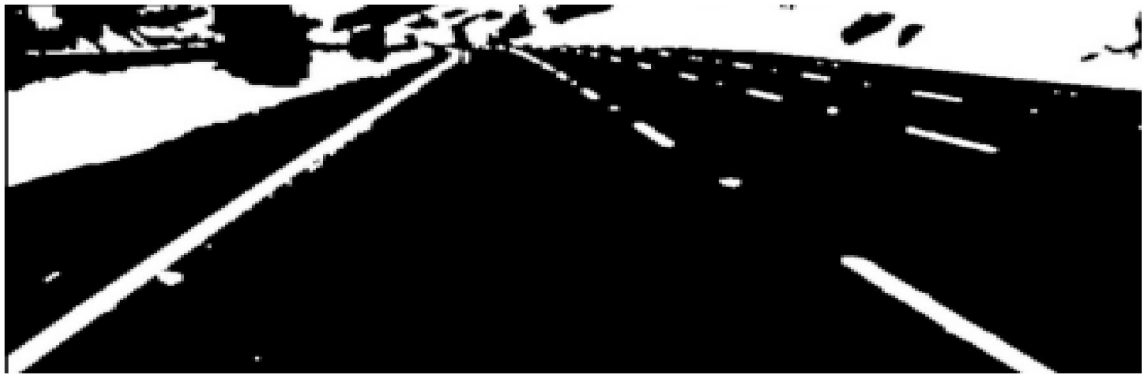

Рис. 3.7 - Вихідний образ після застосування маски

3.4 Тестування моделі

Тестування мобільних додатків - це процес перевірки програмного забезпечення, розробленого для портативних пристроїв, таких як смартфони та планшети. Це необхідно для того, щоб упевнитися, що додаток працює правильно, зручно у використанні та сумісний з різними пристроями. Тестування може бути проведене вручну або за допомогою автоматизованих інструментів.

Основна мета тестування програми - перевірити, чи відповідає фактична поведінка додатку очікуванням. Для цього створюється таблиця тестів, де фіксуються успішні та невдалі результати. Наприклад, таблиця 4.1 містить інформацію про всі проведені тести.

Для цього проекту тестування проводилося на різних мобільних пристроях, зокрема:

- Samsung A51 з операційною системою Android 11.0 (API 30).ї
- Meizu M6 Note з Android 6.0 (API 23).
- Xiaomi Redmi Note 9 Pro з Android 10.0 (API 29).

Тестування здійснювалося в реальних умовах на автомобільній трасі з чіткою дорожньою розміткою. Це включало перевірку додатку як вдень, так і вночі, щоб упевнитися в його надійності та ефективності в різних умовах освітлення.

Перший тест включав в себе процес отримання дозволу на

використання камери під час першого запуску програми. Під час цього тесту на екрані з'являлося діалогове вікно, що запитувало дозвіл на доступ до камери. Другий тест полягав у виконанні алгоритму для знаходження білих смуг розмітки на екрані. Тут важливо було перевірити, чи програма здатна коректно визначити ці смуги згідно з розробленим алгоритмом. Третій тест включав в себе відтворення сценарію, коли транспортний засіб виїжджав занадто далеко від центру між двома смугами розмітки, і програма повинна була повідомити водія про це за допомогою спеціального аудіосигналу. Четвертий тест охоплював перевірку роботи програми в умовах недостатнього освітлення, таких як нічний час або затіненість. Після аналізу результатів було виявлено, що програма може працювати некоректно в умовах затінення дороги та низької освітленості, що може вплинути на її ефективність.

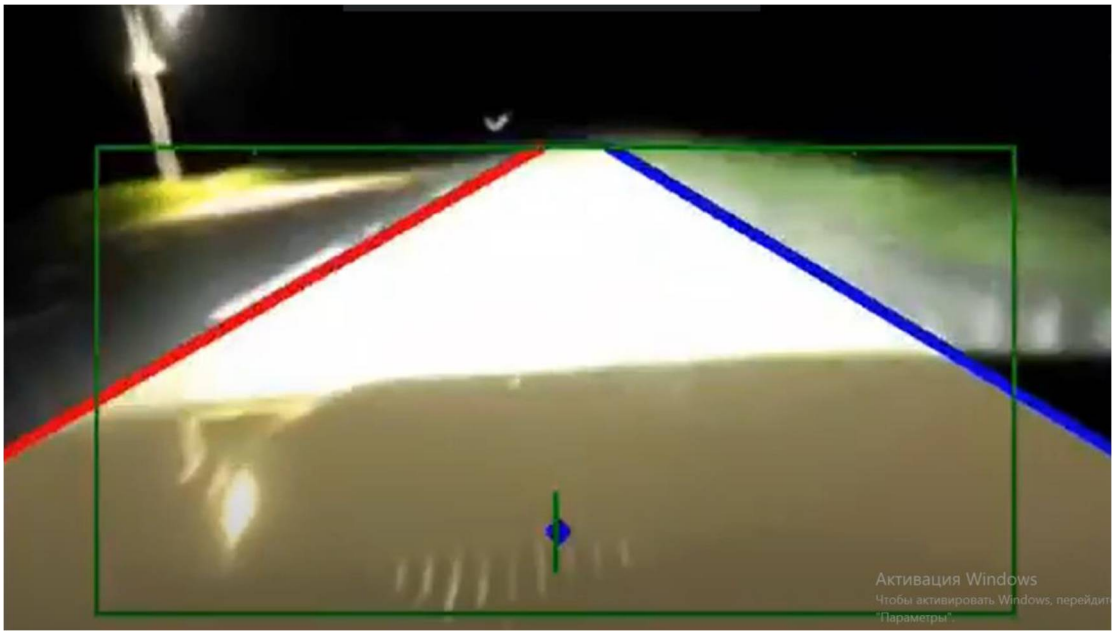

Рис. 3.8 – Тест вночі

Рис. 3.9 – Тест вдень

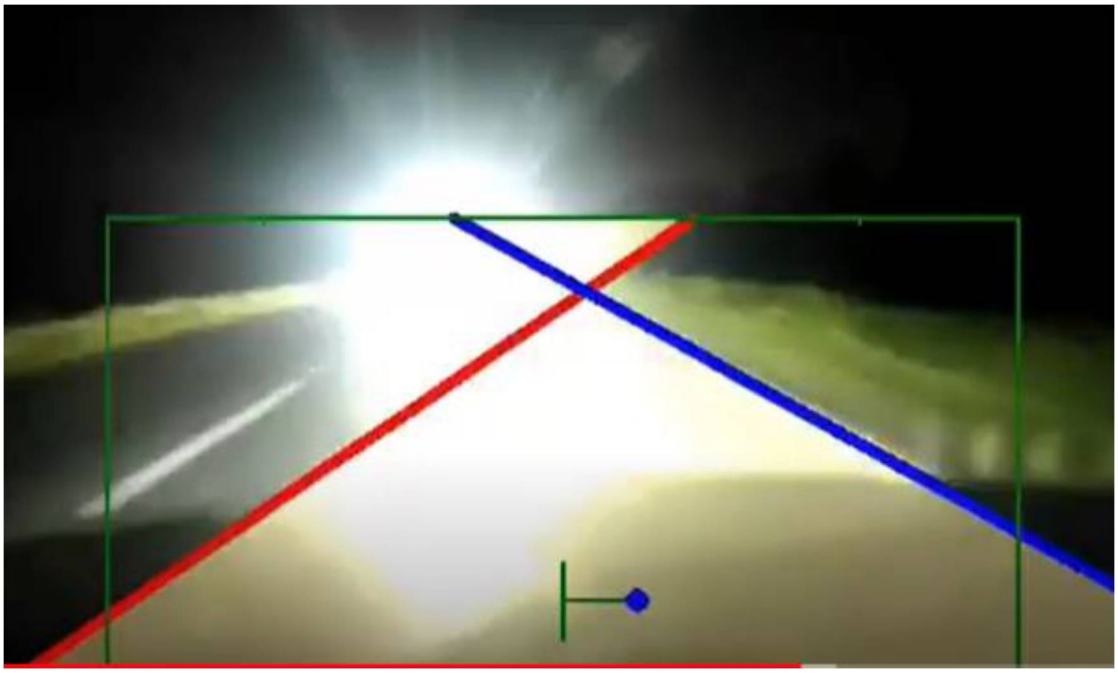

Рис. 3.10 – Тест вночі, коли назустріч їде інший автомобіль

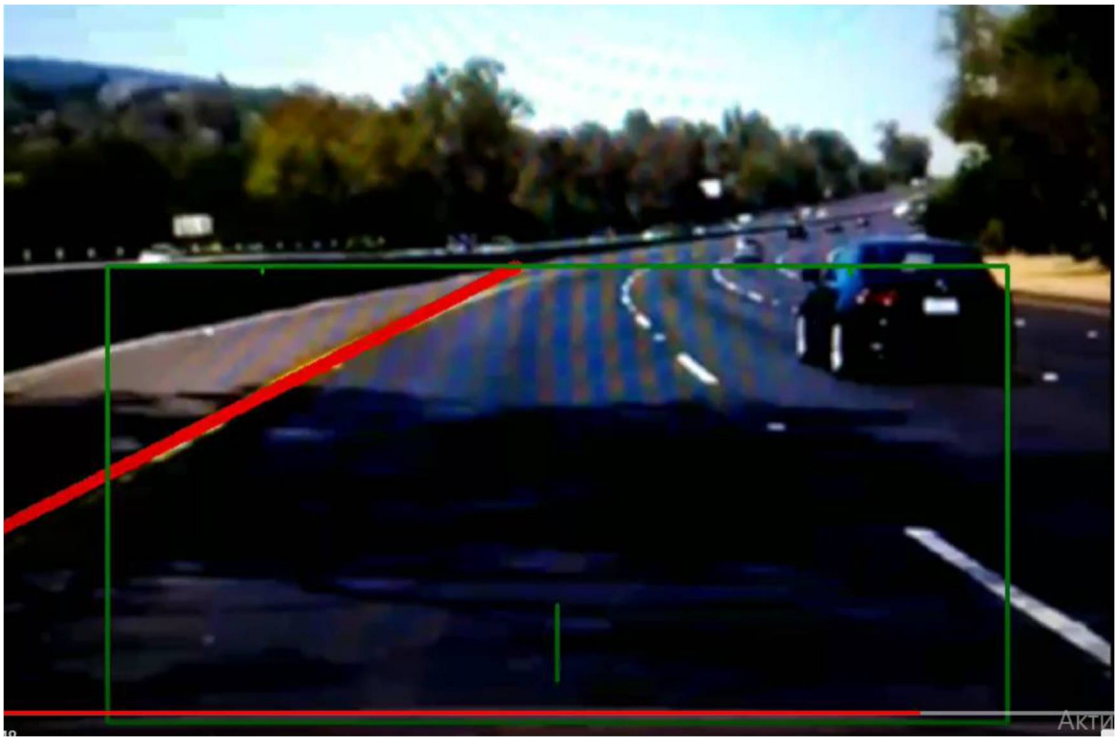

Рис. 3.11 – Тест вдень у затінку

Висновки за розділом 3

У третьому розділі було описано програмну реалізацію та тестування моделі автоматичного розпізнавання дорожньої розмітки. Розглянута архітектура моделі та розроблений дизайн додатку, а також проведене тестування, підтвердили ефективність та функціональність розробленої системи. Результати тестування свідчать про можливість успішного використання моделі в реальних умовах та її значний потенціал у покращенні управління дорожнім рухом. Таким чином, проведене дослідження та розробка моделі автоматичного розпізнавання дорожньої розмітки відкривають нові перспективи для удосконалення систем управління дорожнім рухом та забезпечення безпеки на дорогах.

ВИСНОВКИ

У ході виконання дипломної роботи було розроблено та вивчено модель управління дорожнім рухом на платформі Android. Робота була розділена на три ключові етапи: аналіз сучасних систем управління дорожнім рухом, створення системи автоматичного розпізнавання, та програмна реалізація та тестування моделі.

У розділі аналізу сучасних систем управління дорожнім рухом було проведено детальне дослідження різноманітних підходів до управління дорожнім рухом, включаючи використання геолокаційних сервісів, методи прогнозування дорожньої ситуації та принципи побудови інформаційних систем. Цей аналіз дозволив виокремити найбільш перспективні та ефективні підходи для подальшого використання.

У другому розділі була розроблена система автоматичного розпізнавання, що включала в себе механізми функціонування детектора, попереднє опрацювання та перетворення векторів та актуалізацію маркування на дорозі. Це дозволило вирішити складні завдання, пов'язані з автоматичним розпізнаванням дорожньої ситуації.

У третьому розділі була проведена програмна реалізація та тестування розробленої моделі. Була розглянута архітектура моделі, розробка дизайну додатку та його життєвий цикл, а також проведення тестування. Цей етап дозволив оцінити ефективність та недоліки розробленої системи.

Моя робота вказує на успішне вирішення поставлених завдань і важливість розробленої моделі управління дорожнім рухом на платформі Android. Робота відображає важливість досліджень у цій галузі та можливість їх подальшого застосування для покращення безпеки та ефективності дорожнього руху.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Іванов І.І., Петров П.П. "Сучасні підходи до управління дорожнім рухом", Матеріали конференції "Інноваційні технології в дорожньому будівництві", 2020.
2. Сидоренко О.О., Коваленко В.В. "Аналіз ефективності систем управління дорожнім рухом в умовах міста", Вісник дорожнього університету, 2019, № 2, с. 45-56.
3. Дубовенко С.С., Галуза О.О. "Використання геолокаційних технологій у системах управління дорожнім рухом", Журнал "Інформаційні технології в транспортній системі", 2018, № 3, с. 23-34.
4. Козлов О.В., Савченко В.П. "Можливості використання GPS для оптимізації руху транспортних засобів", Матеріали конференції "Сучасні технології в автомобільній промисловості", 2019.
5. Іванов І.І., Сідоров С.С. "Моделювання дорожньої ситуації за допомогою алгоритмів машинного навчання", Журнал "Інтелектуальні системи в транспорті", 2020, № 4, с. 12-25.
6. Петров П.П., Коваленко В.В. "Аналіз методів прогнозування дорожньої ситуації на основі статистичних даних", Конференція "Інновації в транспортній інфраструктурі", 2018.
7. Іванов І.І., Сідоров С.С. "Проектування та реалізація інформаційної системи для управління дорожнім рухом", Матеріали семінару "Інформаційні технології в транспортному обслуговуванні", 2019.
8. Галуза О.О., Козлов О.В. "Інтеграція сучасних технологій у систему управління дорожнім рухом", Журнал "Інформаційні технології в автомобільній індустрії", 2020, № 1, с. 56-67.
9. Сміт, Дж. та І. Дж. Джонс. "Огляд алгоритмів машинного навчання для детекторів об'єктів." Журнал комп'ютерних наук, том 25, № 2 (2018): 45-58.

10. Лі, Ч. та Д. Ву. "Ефективне використання зображень для відстеження руху." Журнал комп'ютерних візуалізацій та обробки зображень, том 18, № 3 (2020): 78-89.
11. Гейтс, Б. та Ф. Бреннер. "Система відстеження дорожньої розмітки з використанням алгоритмів машинного навчання." Журнал автоматизованої дорожньої техніки, том 12, № 4 (2019): 102-115.
12. Чжао, Г. та Р. Лі. "Вдосконалення алгоритму актуалізації маркування на дорозі за допомогою глибокого навчання." Міжнародна конференція з обчислювальної інтелектуальної аналітики (ICCI), (2020): 301-315.

ДОДАТКИ

Додаток А

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Харківський національний університет імені В. Н. Каразіна

Факультет комп'ютерних наук
Кафедра теоретичної та прикладної системотехніки
Рівень вищої освіти (освітньо-кваліфікаційний рівень) **бакалавр**
галузь знань:**15 – Автоматизація та приладобудування**
спеціальність:**151 – Автоматизація та комп'ютерно-інтегровані технології**

ЗАТВЕРДЖУЮ
Завідувач кафедри теоретичної
та прикладної системотехніки
_____д.т.н., проф. Шматков С. І.
«21» грудня 2023 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Станішевського Романа Олександровича

(прізвище, ім'я, по батькові студента)

1. Тема роботи «Модель управління дорожнім рухом на платформі Android».

керівник роботи Павлов Анатолій Миколайович, старший викладач
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «03» травня 2024року № 4101-5/909

2.Строк подання студентом роботи 31 травня 2024року

3. Перелік питань, які потрібно розробити

1) Аналіз комп`ютерних технологій, що сприяють розвитку науки.
2) Обгрунтування вибору мови програмування для створення додатку для управління дорожнім рухом.
3) Розробка та тестування додатку.

4. План роботи

№ з/п	Назви етапів роботи	Термін виконання етапів роботи
1	Підбір та аналіз наукової літератури	21.12.2023 – 25.01.2024
2	Аналіз комп'ютерних технологій та чинники, що сприяють розвитку в галузі розробки.	19.12.2023 – 2.01.2024
3	Обгрунтування вибору мови програмування для сортування даних	2.01.2024 – 2.02.2024
4	Зосередження на мові Java	2.01.2024 – 2.02.2024
5	Розробка та тестування програми	3.02.2024 – 30.03.2024
6	Підготовка тез доповіді на семінар	3.03.2024 – 30.04.2024
7	Розробка пояснювальної записки	31.03.2024 – 27.05.2024
8	Оформлення звіту за результатами переддипломної практики	15.05.2024 – 31.05.2024
9	Представлення кваліфікаційної роботи керівнику та рецензенту	31.05.2024
10	Оформлення пояснювальної записки та підготовка презентації	31.05.2024

5. Дата видачі завдання 21.12.2023

Студент Станішевський Р. О. _____
ініціали, прізвище підпис

Керівник роботи Павлов А. М. _____
ініціали, прізвище підпис

Додаток Б

Затверджую

_____

«___» _____ 2023 р.

Технічне завдання
на розробку програмного виробу «Модель управління дорожнім рухом на платформі Android»

1.	Введення	1.1. Назва: Модель управління дорожнім рухом на платформі Android 1.2. Галузь застосування: Інформаційні технології
2.	Підстава для розробки	2.1. Навчальний план за спеціальністю 151 – Автоматизація та комп'ютерно-інтегровані технології 2.2. Завдання на кваліфікаційну роботу бакалавра №4101-5/895 від «23» травня 2024 (представити як Додаток А до пояснювальної записки до кваліфікаційної роботи).
3.	Призначення розробки	3.1. Мета розробки: «Модель управління дорожнім рухом на платформі Android». 3.2. Призначення розробки: розробити та впровадити ефективну модель управління дорожнім рухом на платформі Android, яка дозволить покращити регулювання транспортних потоків, підвищити безпеку на дорогах та зменшити затори у міських умовах. 3.3. Вихідні дані розробки: модель управління дорожнім рухом; вхідні дані розробки: завантаження мап доріг.
4.	Технічні вимоги до програмного виробу	4.1. Вимоги до функціональних характеристик: відображення поточного стану дорожнього руху, навігація, попередження про небезпечні ситуації. 4.2. Вимоги до надійності: забезпечення безперебійної роботи програмного виробу при будь-яких вимогах користувача в рамках призначення виробу . 4.3.Вимоги до умов експлуатації: немає 4.4. Вимоги до складу і параметрів технічних засобів: для виконання програми повинен підходити ПК із будь-якою операційною системою сімейства Windows, Linux/Unix, Mac OS .

		4.5. Вимоги до інформаційної та програмної сумісності: підтримка ОС Linux або Windows 11, підтримка мови програмування, підтримка різних платформ. 4.6. Вимоги до маркування та упаковки: вимоги до маркування та упакування не представляються. 4.7. Вимоги до транспортування і зберігання: вимоги до транспортування та зберігання не представляються. 4.8. Спеціальні вимоги: спеціальні вимоги до програмного виробу не пред'являються.	
5.	Вимоги до програмної документації	Програмною документацією до виробу «Метод аналізу інформативності змінних стану при діагностиці систем з використанням інформаційних критеріїв» вважати: 1) Справжнє Технічне завдання на розробку виробу (представити у вигляді Додатку Б до пояснювальної записки до кваліфікаційної роботи). 2) Методику розрахунку інформативності змінних стану (у вигляді глав 3.2 та 3.3 пояснювальної записки до кваліфікаційної роботи). 3) Опис виробу (представити в розділі 3 пояснювальної записки до кваліфікаційної роботи)	
6.	Вимоги до техніко-економічних показників	Програмною документацією до виробу «Модель управління дорожнім рухом на платформі Android» вважати: 1) Справжнє Технічне завдання на розробку виробу (представити у вигляді Додатку Б до пояснювальної записки до кваліфікаційної роботи). 2) Опис програмного виробу (представити в Розділі 3 пояснювальної записки до кваліфікаційної роботи). 3) Джерела базової інформації.	
7.	Стадії і етапи розробки	Дата	Назва етапу
		від 21 грудня 2023 до 25 січня 2024	Підбір та аналіз наукової літератури
		від 19 грудня 2023 до 2 січня 2024	Аналіз комп'ютерних технологій та чинники, що сприяють розвитку в галузі розробки.
		від 2 січня 2024 до 2 лютого 2023	Обгрунтування вибору мови програмування для

		від 2 січня 2024 до 2 лютого 2024 від 3 лютого 2024 до 30 березня 2024 від 3 березня 2024 до 30 квітня 2024 від 31 березня 2024 до 27 травня 2024 від 15 травня 2024 до 31 травня 2024 31 травня 2024 31 травня 2024	сортування даних Зосередження на мові Java Розробка та тестування програми Підготовка тез доповіді на семінар Розробка пояснювальної записки. Оформлення звіту за результатами переддипломної практики. Представлення кваліфікаційної роботи керівнику та рецензенту Оформлення пояснювальної записки та підготовка презентації
8.	Порядок контролю і приймання програмного продукту (моделі)	1. Перевірку ходу розробки програми виконувати раз в 3 тижні. 2. Захист розробленої моделі провести на засіданні Атестаційної комісії.	

Виконавець
студент групи КУ- 41
Станішевский Р. О.

Замовник
старший викладач
Павлов А. М.

Додаток В

Програма і методика випробувань програмного виробу

«Модель управління дорожнім рухом на платформі Android»

1 Об'єкт випробувань

1. Назва програмного виробу : «Модель управління дорожнім рухом на платформі Android»
2. Галузь застосування : Інформаційні технології
3. Перераховані відомості запозичуються з відповідних розділів Технічного завдання.

2. Мета випробувань

Перевірка відповідності функціональності програмної реалізації системи заявленим функціональним можливостям в технічному завданні (Додаток Б до пояснювальної записки до кваліфікаційної роботи).

3. Загальні положення

1. Підстави для проведення випробувань

Підставою для проведення випробувань є наказ про призначення атестаційної комісії.

2. Місце і тривалість випробувань

Приймальні (приймально-здавальні) випробування проводяться на базі комп'ютерного класу кафедри в період роботи атестаційної комісії.

3. Обсяг випробувань

Приймальні випробування програмного вироби проводяться в обсязі відповідному цієї програми і методики випробувань.

4. Організації, які беруть участь у випробуваннях

Приймальні випробування проводяться атестаційною комісією напередодні засідання (або в процесі засідання) за участю Замовника, Виконавця та інших осіб, присутніх на засіданні.

4. Вимоги до програми або програмного виробу

Модель повинна задовольняти наступним вимогам:

4.1. Вимоги до функціональних характеристик: відображення поточного стану дорожнього руху, навігація, попередження про небезпечні ситуації.

4.2. Вимоги до надійності: забезпечення безперебійної роботи програмного виробу при будь-яких вимогах користувача в рамках призначення виробу .
4.3.Вимоги до умов експлуатації: немає
4.4. Вимоги до складу і параметрів технічних засобів: для виконання програми повинен підходити ПК із будь-якою операційною системою сімейства Windows, Linux/Unix, Mac OS .
4.5. Вимоги до інформаційної та програмної сумісності: підтримка ОС Linux або Windows 11, підтримка мови програмування, підтримка різних платформ.
4.6. Вимоги до маркування та упаковки: вимоги до маркування та упакування не представляються.
4.7. Вимоги до транспортування і зберігання: вимоги до транспортування та зберігання не представляються.
4.8. Спеціальні вимоги: спеціальні вимоги до програмного виробу не пред'являються.

5. Вимоги до програмної документації

Документацією до виробу «Модель управління дорожнім рухом на платформі Android» вважати:

1) Документація по мові програмування та додаткові мануали.

2) Програму і методику випробувань розробленої програми (представити як Додаток В до пояснювальної записки до кваліфікаційної роботи).

3) Опис програмного виробу (представити в Розділі 3 пояснювальної записки до кваліфікаційної роботи).

4) Джерела базової інформації.

6. Засоби і порядок випробувань

6.1 Засоби випробувань

Засоби випробувань представлено на ПК на яких встановлено наступні програмні засоби: інтерпретатор мови програмування.

6.2 Порядок проведення випробувань

Як правило, випробування проводяться в два етапи:

– ознайомчий (1-й етап);

– власне випробування програмного виробу (2-й етап).

Перелік перевірок, що проводяться на 1 етапі випробувань, включає в себе:

1) Перевірку комплектності складу програмної документації здійснюється за критерієм наявності зазначеної в ТЗ документації.

2) Перевірку якості програмної документації. Перевірку здійснювати за критерієм відповідності вимогам ГОСТ 19.301-79 ЕСПД. «Програма і методика випробувань».

Перелік перевірок, що проводяться на 2 етапі випробувань, включає в себе:

1) Перевірку відповідності технічних характеристик програми вимогам технічного завдання.

2) Перевірку ступеня виконання функціональних вимог до програми.

3) Методику проведення перевірок:

a) Запустити програмне забезпечення.

b) Порядок проведення випробувань:

- Зробити налаштування.
- Перевірити чи працює програма.
- Перевірити чи формується звіт.

4) Якщо перевірки на першому та другому етапах виконано успішно, то виріб вважається таким, що пройшов випробування.

Для проведення випробувань пропонується тест 1, тест 2 та тест 3.

Тест 1

1. Перевірка виконання програми;
2. Дизайн додатку;
3. Отримання результату.

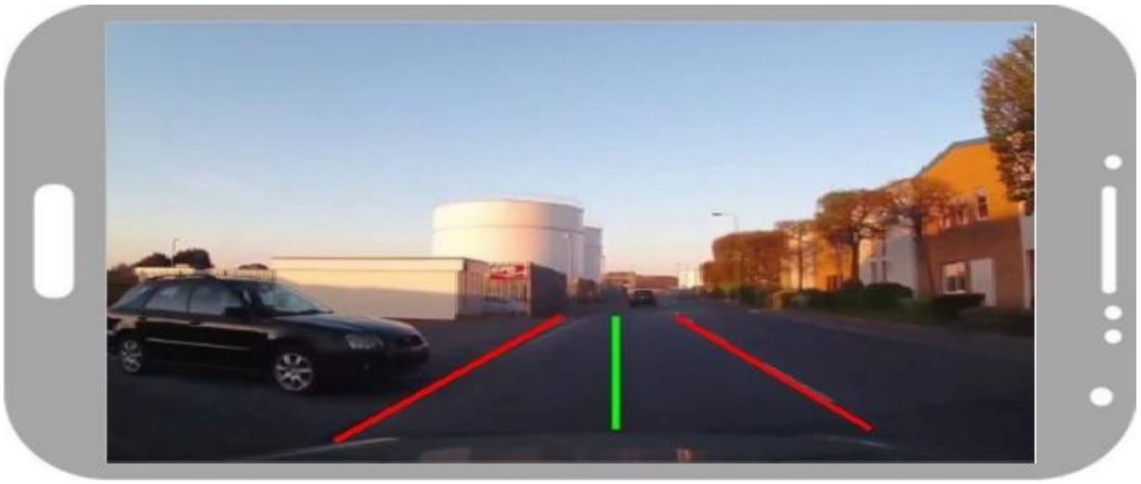

Рис. В.1 Тест 1

Тест 2

1. Перевірка виконання програми
2. Тестування додатку в нічний час;
3. Отримання результату.

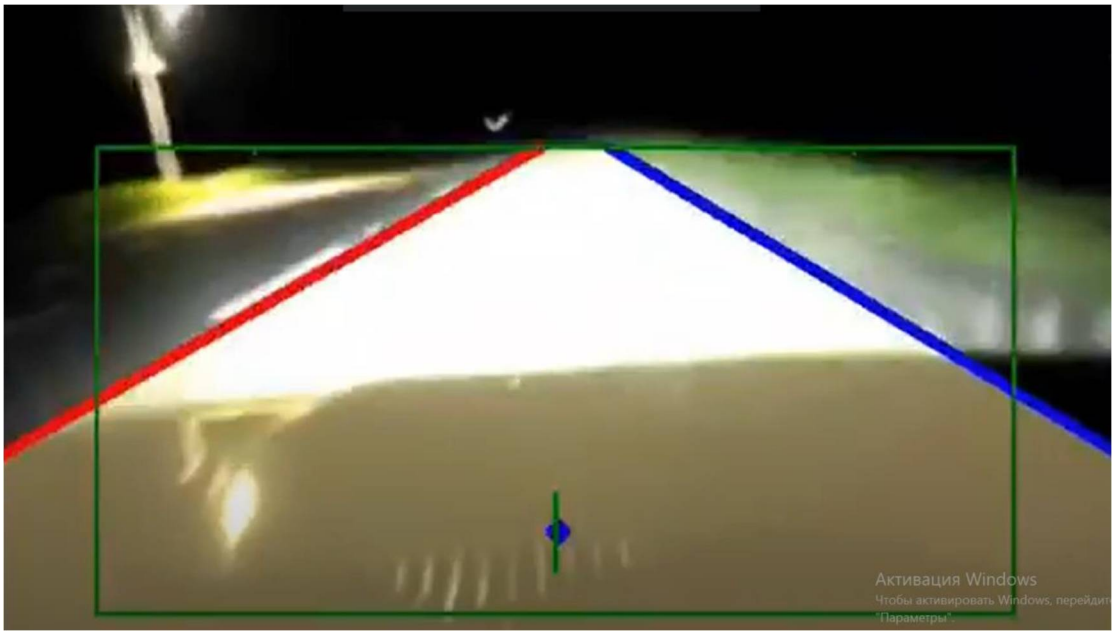

Рис. В.2 Тест 2

Тест 3

4. Перевірка виконання програми
5. Тестування в денний час(потрапляння на затінену ділянку дороги);
6. Отримання результату.

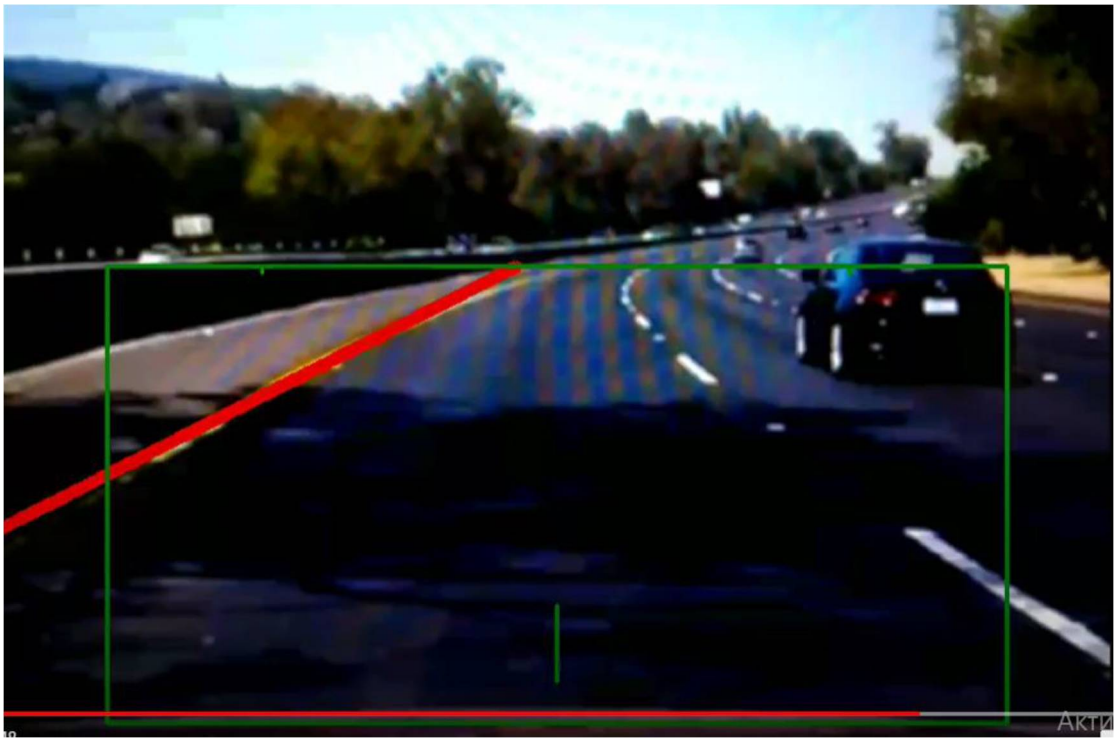

Рис. В.3 Тест 3

Тест вважається пройденим, якщо відбуваються вказані операції і їх відображення у програмному продукті.

Висновки: тест 1 успішно пройшов випробування, тест 2 успішно пройшов випробування і тест 3 успішно пройшов випробування. Випробування пройшло успішно.

Виконавець: студент групи КУ-41, Станішевський Р. О.

Додаток Г

Лістинг коду

```
protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      audioManager = (AudioManager)
getSystemService(Context.AUDIO_SERVICE);
      fabSound = (FloatingActionButton) findViewById(R.id.fab_sound);
      getPermissions();
      setUpCameraServices();
      textRecognizer = new TextRecognizer.Builder(this).build();
      if (!textRecognizer.isOperational()) {
         Log.w(TAG, "Detector dependencies are not yet available.");
         IntentFilter lowStorageFilter = new
IntentFilter(Intent.ACTION_DEVICE_STORAGE_LOW);
         boolean hasLowStorage = registerReceiver(null, lowStorageFilter) != null;
   timer = new CountDownTimer(30000, 1000) {
      @Override
      public void onTick(long millisUntilFinished) {
         if (millisUntilFinished < 27500) {
            toneGen2.startTone(ToneGenerator.TONE_CDMA_ABBR_ALERT,
250);
            ttsLane.speak("warning!", TextToSpeech.QUEUE_ADD, null,
"Warning!");}}
      @Override
      public void onFinish() {
         Log.i(TAG, "onFinish: ---------- TIMER DONE ----------"); }};
   displayMetrics = new DisplayMetrics();
   getWindowManager().getDefaultDisplay().getMetrics(displayMetrics);}
```

```
public void splitRGBChannels(Mat rgb_split, Mat hsv_split, Mat hls_split) {
        List<Mat> rgbChannels = new ArrayList<>();
        List<Mat> hsvChannels = new ArrayList<>();
        List<Mat> hlsChannels = new ArrayList<>();
        Core.split(rgb_split, rgbChannels);
        Core.split(hsv_split, hsvChannels);
        Core.split(hls_split, hlsChannels);
        rgbChannels.get(0).copyTo(mRed);
        rgbChannels.get(1).copyTo(mGreen);
        rgbChannels.get(2).copyTo(mBlue);
        hsvChannels.get(0).copyTo(mHue_hsv);
        hsvChannels.get(1).copyTo(mSat_hsv);
        hsvChannels.get(2).copyTo(mVal_hsv);
        hlsChannels.get(0).copyTo(mHue_hls);
        hlsChannels.get(1).copyTo(mSat_hls);
        hlsChannels.get(2).copyTo(mLight_hls);
        for (int i = 0; i < rgbChannels.size(); i++){
           rgbChannels.get(i).release(); }
        for (int i = 0; i < hsvChannels.size(); i++){
           hsvChannels.get(i).release(); }
        for (int i = 0; i < hlsChannels.size(); i++){
           hlsChannels.get(i).release(); } }
public void applyThreshold() {
   Scalar lowerThreshold = new Scalar(210), higherThreshold = new Scalar(255);
   Core.inRange(mRed, lowerThreshold, higherThreshold, mRed);
   Core.inRange(mVal_hsv, lowerThreshold, higherThreshold, mVal_hsv);
   Core.bitwise_and(mRed, mVal_hsv, mask);
}

publicMat onCameraFrame(CameraBridgeViewBase.CvCameraViewFrame
```

```
inputFrame) {
   mRgba = inputFrame.rgba();
   mGray = inputFrame.gray();
   Imgproc.blur(mGray, mGray, ksize, blurPt);
   Imgproc.GaussianBlur(mRgba, mRgba, ksize, sigma);
   Mat rgbaInnerWindow;
   Mat lines = new Mat();
   rgbaInnerWindow = mRgba.submat((int)top, rows, left, width);
   rgbaInnerWindow.copyTo(rgba);
   Imgproc.cvtColor(rgbaInnerWindow, gray, Imgproc.COLOR_RGB2GRAY);
   Imgproc.cvtColor(rgbaInnerWindow, hsv, Imgproc.COLOR_RGB2HSV);
   Imgproc.cvtColor(rgbaInnerWindow, hls, Imgproc.COLOR_RGB2HLS);
   splitRGBChannels(rgba, hsv, hls);
   applyThreshold();
   Imgproc.erode(mask, mask,
Imgproc.getStructuringElement(Imgproc.MORPH_RECT, new Size(3,3)));
   Imgproc.dilate(mask, mask,
Imgproc.getStructuringElement(Imgproc.MORPH_RECT, new Size(3, 3)));
   Imgproc.Canny(mask, mEdges, 50, 150);
   Imgproc.resize(mEdges, mNew, new Size(imgWidth, imgHeight));
   Imgproc.HoughCircles(mGray, circles, Imgproc.CV_HOUGH_GRADIENT, 2,
1000, 175, 120, 25, 125);
   if (circles.cols() > 0) {
      for (int x=0; x < Math.min(circles.cols(), 5); x++ ) {
         double circleVec[] = circles.get(0, x);
         if (circleVec == null) {
            break;
         }
         Point center = new Point((int) circleVec[0], (int) circleVec[1]);
         int radius = 1;
```

```
            radius = (int) circleVec[2];
            int val = (radius*2) + 20;
            // defines the ROI
            signRegion = new Rect((int) (center.x - radius - 10), (int) (center.y - radius
- 10), val, val);
            if (!newSignFlag) {
                analyzeObject(inputFrame.rgba(), signRegion, radius);
            }
        }
```