Міністерство освіти і науки України
Криворізький національний університет
Кафедра моделювання та програмного забезпечення

**КВАЛІФІКАЦІЙНА РОБОТА**
**на здобуття ступеня вищої освіти магістра**

за спеціальністю 121 – Інженерія програмного забезпечення

На тему: Дослідження та розробка моделі інтелектуальної системи керування перехрестями

Засвідчую, що в цій кваліфікаційній роботі немає запозичень із праць інших авторів без відповідних посилань.
Студент гр. ІПЗ-23-1м _____ / С. К. Заболотній /

Керівник кваліфікаційної роботи _____ / _____ /

Економіко-організаційна частина _____ / _____ /

Нормоконтроль _____ / _____ /

Завідувач кафедри _____ / А. М. Стрюк /

Кривий Ріг
2024

Криворізький національний університет

Факультет: Інформаційних технологій

Кафедра: Моделювання та програмного забезпечення

Ступінь вищої освіти: магістр

Спеціальність: 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:

Зав. кафедри

_____ А. М. Стрюк

«___» _____ 20__ р.

**ЗАВДАННЯ**

**на кваліфікаційну роботу**

студенту групи ІПЗ-23-1м Заболотньому Станіславу Костянтиновичу

1. Тема: Дослідження та розробка моделі інтелектуальної системи керування перехрестями затверджено наказом по КНУ № __ від «__»_____20__ р.

2. Термін подання студентом закінченої роботи: «__» ____20___ р.

3. Вихідні дані по роботі: Створювана система має відповідати вимогам надійності та безпеки оскільки дорожній рух є середою з високими ризиками і високими наслідками помилок.

4. Зміст пояснювальної записки (перелік питань, що їх треба розробити): виконати аналіз існуючих методів розв’язання задачі, розробити математичні моделі подання знань, дослідити та спроектувати моделі інтелектуальної системи керування перехрестями., виконати аналіз економічної ефективності розроблювального комплексу.

5. Перелік ілюстративного матеріалу: блок-схеми розроблених алгоритмів, схема взаємодії модулів системи, графіки порівняння ефективності розробленої системи з існуючими аналогами.

Календарний план:

| № | Найменування етапів кваліфікаційної роботи | Термін виконання етапів роботи |
|---|---|---|
| 1 | Огляд літератури за тематикою роботи | 01.10.2023 – 15.10.2023 |
| 2 | Проведення аналізу існуючих теоретичних методів дослідження і вирішення проблеми | 16.10.2023 – 31.10.2023 |
| 3 | Проведення критичного порівняльного аналізу існуючих аналогів програмних систем | 01.11.2023– 30.11.2023 |
| 4 | Формулювання актуальності і завдань роботи | 01.12.2023– 15.12.2023 |
| 5 | Оформлення матеріалів першого розділу роботи | 16.12.2023 – 31.12.2023 |
| 6 | Розробка математичних, структурних і функціональних моделей | 01.01.2024– 15.02.2024 |
| 7 | Розробка структур даних, основних класів та алгоритмів програмного комплексу | 16.02.2024– 15.03.2024 |
| 8 | Оформлення матеріалів другого розділу роботи | 16.03.2024 – 31.03.2024 |
| 9 | Розробка баз даних і знань, програмних модулів, бібліотек та інтерфейсу програмного комплексу | 01.04.2024– 15.06.2024 |
| 10 | Оформлення матеріалів третього розділу роботи | 16.06.2024– 30.06.2024 |
| 11 | Дослідження та узагальнення результатів роботи з точки зору наукової та технічної цінності | 01.07.2024 – 31.08.2024 |
| 12 | Оформлення матеріалів четвертого розділу роботи | 01.09.2024 – 15.09.2024 |
| 13 | Аналіз економічної ефективності інновації | 16.09.2024– 31.10.2024 |
| 14 | Оформлення матеріалів п’ятого розділу роботи | 01.11.2024– 10.11.2024 |
| 15 | Остаточне оформлення пояснювальної записки | 11.11.2024 – 20.11.2024 |

Дата видачі завдання: «____» _____ 20___ р.

Студент _____ / С. К. Заболотній /

Керівник роботи _____ / _____ /

# РЕФЕРАТ

ТРАНСПОРТ, ДЕТЕКЦІЯ, УПРАВЛЯННЯ РУХОМ, ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ, КОНТЕКСТ, РОЗПІЗНАВАННЯ, СИСТЕМА

Пояснювальна записка: 87 с., 2 табл., 20 рис., 4 дод., 25 джерел.

Метою кваліфікаційної роботи є розробка та дослідження моделі інтелектуальної системи керування перехрестям.

У теоретичній частині кваліфікаційної роботи проведено аналіз сучасного стану проблеми ефективного керування перехрестями. Розглянуто дослідження з теми, існуючі моделі та рішення. Сформульовано актуальність та завдання дослідження.

У практичній частині кваліфікаційної роботи було визначено необхідні модулі системи, а саме сенсорний модуль, модуль відслідковування об'єктів та модуль керування світлофором, здійснений аналіз технологій та алгоритмів, що можуть бути використані для виконання задачі дослідження. Після проведення аналізу можливих рішень було створено алгоритми основних функцій та спроектовано модулі системи.

У частині дослідження результатів було порівняно ефективність спроектованих сенсорних модулів, що базуються на методі відокремлення фону та моделі виявлення об'єктів, визначено їх переваги та недоліки. Порівняно розроблену систему керування перехрестям з стандартною моделлю світлофору зі статичними фазами у симуляторі дорожнього руху за різних навантажень, визначено що розроблена система є більш ефективною. Було проведено аналіз економічної ефективності інновації, обґрунтовано що розробка і впровадження моделі інтелектуальної системи керування перехрестями є економічно доцільними.

Основні положення кваліфікаційної роботи опубліковано у вигляді тез доповіді на конференції КІСМ-2024.

# ABSTRACT

TRANSPORT, DETECTION, TRAFFIC CONTROL, INFORMATION TECHNOLOGY, CONTEXT, RECOGNITION, SYSTEM

Thesis in 87 p., 2 tables, 20 pics., 4 pp., 25 sources.

The purpose of the qualification work is to develop and research a model of an intelligent intersection management system.

In the theoretical part of the qualification work, an analysis of the current state of the problem of effective control of intersections was carried out. Research on the topic, existing models and solutions are considered. The relevance and objectives of the research are formulated.

In the practical part of the qualification work, the necessary modules of the system were determined, namely the sensor module, the object tracking module and the traffic light control module, the analysis of technologies and algorithms that can be used to fulfill the research task was carried out. After the analysis of possible solutions, the algorithms of the main functions were created and the system modules were designed.

As part of the results study, the effectiveness of the designed sensor modules based on the background separation method and the object detection model was compared, and their advantages and disadvantages were determined. Comparing the developed intersection control system with a standard traffic light model with static phases in a traffic simulator under different loads, it was determined that the developed system is more effective. An analysis of the economic efficiency of the innovation was conducted, and it was substantiated that the development and implementation of a model of an intelligent intersection management system is economically feasible.

The main provisions of the qualification work were published in the form of abstracts of a report at the KISM-2024 conference.

# ЗМІСТ

# ВСТУП

Сучасні міста стикаються з низкою проблем, пов'язаних із зростаючою урбанізацією та збільшенням транспортних потоків. Перехрестя є критичними точками дорожньої інфраструктури, де часто виникають затори, аварії та інші проблеми, що впливають на ефективність транспортної системи та якість життя мешканців. Традиційні методи керування рухом на перехрестях, такі як фіксовані світлофорні цикли, виявляються недостатньо ефективними в умовах змінних та непередбачуваних транспортних потоків. Це зумовлює необхідність розробки нових підходів до керування перехрестями, які б дозволили адаптивно реагувати на поточні умови руху.

Актуальність даного дослідження полягає в необхідності створення інтелектуальних систем керування перехрестями, які базуються на використанні сучасних технологій, таких як штучний інтелект, машинне навчання та Інтернет речей (IoT). Такі системи можуть значно підвищити ефективність транспортних потоків, зменшити час очікування на світлофорах, знизити кількість дорожньо-транспортних пригод та сприяти зменшенню викидів шкідливих речовин в атмосферу. Досвід впровадження інтелектуальних систем у різних містах світу демонструє їх високу ефективність і перспективність.

Наукова проблема, що розглядається в даній роботі, полягає в розробці моделі інтелектуальної системи керування перехрестями, яка б враховувала реальні умови дорожнього руху та могла адаптивно змінювати режими роботи світлофорів для оптимізації транспортних потоків. Це завдання включає аналіз існуючих підходів, розробку алгоритмів керування, створення та впровадження алгоритму пропорційного розподілення тривалості зеленої фази світлофору та тестування розробленої системи в умовах, максимально наближених до реальних.

Проведення даного дослідження є необхідним для подолання існуючих проблем дорожнього руху в містах і сприяння розвитку стійких та розумних

міських інфраструктур. Результати дослідження матимуть важливе значення для транспортних систем великих міст, сприятимуть покращенню якості життя міських мешканців, зменшенню екологічного навантаження та підвищенню безпеки дорожнього руху.

Метою роботи є перевірка гіпотези щодо можливості оптимізації дорожнього руху за допомогою системи управління перехрестями що використовує комп'ютерний зір для визначення наявності та кількості автомобілів та пішоходів та що використовує алгоритм пропорційного розподілення тривалості зеленої фази світлофору для оптимального керування перехрестям.

# 1 АНАЛІТИЧНИЙ ОГЛЯД І ПОСТАНОВКА ЗАВДАННЯ

## 1.1 Актуальність дослідження

Сучасні міста стикаються зі значним збільшенням населення, що призводить до зростання транспортних потоків. За даними ООН, до 2050 року приблизно 68% світового населення проживатиме у містах [1]. Збільшення кількості транспортних засобів створює навантаження на дорожню інфраструктуру, особливо на перехрестях, які є критичними точками транспортної системи. Традиційні системи керування рухом, ґрунтовані на фіксованих часових інтервалах або простих алгоритмах пріоритету, не здатні ефективно справлятися з динамічними змінами в інтенсивності дорожнього руху. Це призводить до неефективного використання дорожньої інфраструктури та збільшення часу простою транспортних засобів.

Затори на дорогах спричиняють значні економічні втрати. Вони призводять до втрати часу, підвищення витрат на паливо, зменшення продуктивності праці та збільшення витрат на обслуговування транспортних засобів. За оцінками, затори коштували економіці Великої Британії 20 мільярдів фунтів у 2004 та прогнозами, що ситуація тільки погіршуватиметься [2]. Аналогічні тенденції спостерігаються і в інших країнах, що робить проблему глобальною.

Транспортні затори призводять до підвищення викидів шкідливих речовин в атмосферу. Автомобілі, які стоять у заторах, споживають більше палива, що збільшує викиди CO2 та інших шкідливих речовин [3]. Розробка ефективних систем керування рухом може зменшити ці негативні наслідки зменшуючи час простою та, через це, зменшуючи час подорожі в цілому.

Перехрестя є місцями підвищеної небезпеки, де відбувається значна кількість дорожньо-транспортних пригод. Неефективне керування перехрестями може призводити до аварій та збільшення кількості жертв на дорогах. Затори, що можуть спричинятися неефективним керуванням

дорожнім рухом, також підвищують шанси агресивної поведінки за кермом, яка пов’язана з підвищеним ризиком ДТП [4] навіть після того, як затор закінчився [5]. Інтелектуальні системи керування, підвищуючи ефективність транспортування і зменшуючи ас заторів, можуть сприяти підвищенню безпеки, зменшуючи ймовірність аварійних ситуацій.

Також інтелектуальні системи керування рухом мають потенціал оптимізувати використання існуючої дорожньої інфраструктури без значних капіталовкладень у будівництво нових доріг. Це особливо актуально в умовах обмежених бюджетів міських адміністрацій. Також розширення доріг у містах може бути неможливим через прилягання дороги до будівель, тож оптимізація ефективності вже існуючої інфраструктури може бути єдиним практичним рішенням.

Сучасні досягнення в галузі штучного інтелекту, машинного навчання та великих даних відкривають нові можливості для розробки інтелектуальних систем керування рухом. Використання датчиків, камер спостереження та інших пристроїв Інтернету речей дозволяє збирати та аналізувати великі обсяги даних у реальному часі, що сприяє прийняттю більш обґрунтованих рішень.

У багатьох країнах світу вже впроваджуються інтелектуальні системи керування рухом, які демонструють позитивні результати. Наприклад, Копенгаген вже випробовує подібне рішення. Місцева влада перепрограмувала свою систему управління дорожнім рухом, щоб віддавати перевагу громадському транспорту в години пік. Вони також тестують динамічне програмування світлофорів, щоб зменшити тривалість простою на перехрестях. У Копенгагені не тільки зменшили забруднення, але й зробили подорожі швидшими – середня швидкість для автомобілів зросла на 4%, а для автобусів – на 9%. Аналіз їх досвіду та адаптація передових практик до місцевих умов може значно покращити транспортну систему в наших містах [6].

Інтелектуальні системи керування перехрестями можуть бути інтегровані з іншими міськими системами, такими як системи громадського транспорту, системи екстреної допомоги та навігаційні системи для водіїв. Це дозволить створити більш ефективну та скоординовану міську інфраструктуру.

Покращення транспортної ситуації в містах сприятиме підвищенню якості життя мешканців. Зменшення заторів підвищить продуктивність праці та знизить рівень стресу, пов'язаного з дорожнім рухом.

Підсумовуючи, дослідження та розробка моделі інтелектуальної системи керування перехрестями є актуальною темою з багатьох причин, включаючи економічні, екологічні та соціальні аспекти. Використання сучасних технологій та методів дозволить значно покращити ефективність транспортної системи, підвищити безпеку дорожнього руху та зменшити негативний вплив на навколишнє середовище.

### 1.2 Об'єкт та предмет дослідження

Об'єктом дослідження є процес розробки та дослідження моделі інтелектуальної системи керування перехрестями (ІСКП), яка здатна динамічно адаптуватися до мінливих умов дорожнього руху, щоб оптимізувати пропускну здатність, безпеку та екологічність транспортних систем.

Дослідження включає:

- аналіз існуючих систем керування дорожнім рухом та їхніх обмежень;
- вивчення принципів роботи інтелектуальних систем керування перехрестями;
- розробку моделі ІСКП, яка враховує такі фактори, як:
    - Інтенсивність трафіку в різних напрямках;
    - Наявність пішоходів;
    - Час очікування для різних видів транспорту.

- розробку алгоритмів прийняття рішень для ІСКП, які дозволяють динамічно регулювати час циклу світлофора та пріоритети проїзду;
- проведення моделювання та симуляції роботи ІСКП в різних сценаріях дорожнього руху;
- оцінку ефективності ІСКП за такими показниками, як:
  - пропускна здатність;
  - час очікування для різних видів транспорту;
  - кількість аварій;
  - рівень викидів забруднюючих речовин.
- розробку рекомендацій щодо впровадження ІСКП в реальних транспортних системах.

Результатом дослідження буде модель інтелектуальної системи керування перехрестями, яка здатна значно покращити роботу транспортних систем, а також рекомендації щодо її впровадження в реальних умовах.

### 1.3 Задачі дослідження

Було сформовано наступні задачі та підзадачі дослідження:

1. аналіз існуючих систем керування дорожнім рухом:
   - вивчити принципи роботи традиційних систем, заснованих на фіксованих часових сигналах;
   - оцінити їхні недоліки та обмеження в динамічних умовах;
   - ознайомитися з існуючими розробками інтелектуальних систем керування перехрестями, оцінити їхні можливості та обмеження.
2. вивчення принципів роботи інтелектуальних систем керування перехрестями:
   - дослідити методи збору та обробки даних про дорожній рух в режимі реального часу;
   - вивчити алгоритми прийняття рішень, які використовуються для динамічного регулювання світлофора та пріоритетів проїзду;

- ознайомитися з методами оптимізації дорожнього руху, що використовуються в цих системах.

3. розробка моделі інтелектуальної системи керування перехрестями:
   - розробити методи збору та обробки даних про інтенсивність трафіку в різних напрямках;
   - створити модель, яка динамічно прогнозує зміни інтенсивності трафіку;
   - розробити методи виявлення та відстеження пішоходів та велосипедистів на перехресті;
   - створити модель, яка враховує їхні потреби при прийнятті рішень про регулювання;
   - розробити методи оцінки часу очікування для різних видів транспорту;
   - створити модель, яка мінімізує загальний час очікування;
4. розробка алгоритмів прийняття рішень для ІСКП:
   - створити алгоритми, які динамічно адаптуються до мінливих умов дорожнього руху;
   - використовувати методи оптимізації для пошуку оптимальних значень часу циклу світлофора та пріоритетів проїзду;
   - забезпечити стійкість алгоритмів до непередбачуваних змін в дорожньому русі.
5. моделювання та симуляція роботи ІСКП:
   - провести симуляцію роботи ІСКП для різних сценаріїв (пікові години, несприятливі погодні умови, ДТП);
   - оцінити ефективність ІСКП за такими показниками, як пропускна здатність, час очікування, кількість аварій, рівень викидів забруднюючих речовин.
6. розробка рекомендацій щодо впровадження ІСКП:

- сформулювати рекомендації щодо технічних та організаційних аспектів впровадження ІСКП;
- оцінити економічну доцільність впровадження ІСКП;
- розробити план впровадження ІСКП.

### 1.4 Методи дослідження

Під час дослідження використовуються аналітичні методи, а саме літературний огляд, вивчення існуючої літератури з теми інтелектуальних систем керування перехрестями, включаючи принципи роботи, алгоритми прийняття рішень, методи оптимізації та результати досліджень, аналіз існуючих даних про дорожній рух, таких як інтенсивність трафіку, час очікування, кількість аварій, викиди забруднюючих речовин, для визначення факторів, які впливають на пропускну здатність та безпеку дорожнього руху, створення математичних моделей дорожнього руху для оцінки ефективності різних алгоритмів керування світлофором, експериментальні методи, а саме симуляція - проведення комп'ютерного моделювання роботи інтелектуальної системи керування перехрестями в різних сценаріях дорожнього руху, методи дослідження систем, а саме розбиття інтелектуальної системи керування перехрестями на її основні компоненти та вивчення їх функцій та взаємодії, вивчення того, як дані про дорожній рух збираються, обробляються та використовуються інтелектуальною системою керування перехрестями для прийняття рішень, вивчення того, як інтелектуальна система керування перехрестями реагує на різні дорожні ситуації та як її поведінка впливає на дорожній рух. Також використовуються методи машинного навчання для визначення та інтенсивності трафіку, часу очікування та інших факторів, які впливають на дорожній рух та для оптимізації алгоритмів керування світлофором, щоб максимізувати пропускну здатність та безпеку дорожнього руху.

# 2 ВІДОМОСТІ ПРО ПРЕДМЕТ (ОБ’ЄКТ) ДОСЛІДЖЕННЯ

## 2.1 Інформація про предметну галузь та її особливості

Керування дорожнім рухом пройшло довгий шлях від простих сигналів та людських регулювальників до складних систем, що використовують штучний інтелект.

Перші зачатки КДР з'явилися ще в Стародавньому Римі, де використовувалися людські регулювальники для контролю руху на жвавих перехрестях. У Середньовіччя для регулювання руху використовувалися дзвони, прапори та інші візуальні сигнали.

Більш сучасні методи керування дорожнім рухом з'явилися у 19-ому сторіччі: у 1868 році біля будівлі парламенту в Лондоні була встановлена перша система керування дорожнім рухом, яка мала на меті замінити поліцейський контроль за рухом транспорту.

Протягом перших двох десятиліть 20-го століття в Сполучених Штатах використовувалися так звані "семафорні" світлофори, подібні до лондонського. Кожен штат мав власну конструкцію пристрою, а в багатьох випадках ним керував дорожній офіцер, який свистком попереджав про зміну сигналу.

У кінці 1930х років було розроблено світлофори, що реагували на сигнали від автомобілів та пішоходів – водії могли посигналити, під'їжджаючи до світлофора з червоним сигналом щоб почати перемикання сигналу. Аналогічна система для пішоходів використовувала кнопку.

У 60-х роках 20 сторіччя для керування сигналом світлофора стали використовувати комп'ютерні системи.

У 1990-ті до світлофорів було додано таймер зворотного відліку, щоб допомогти водіям та пішоходам визначити, чи встигнуть вони перетнути дорогу до зміни кольору сигналу [7].

Зараз зароджуються системи що можуть використовувати підключені до Інтернету транспортні засоби для впливу на сигнал світлофору Це може значно покращити швидкість, час та ефективність на перехрестях.

Керування дорожнім рухом наразі є складною галуззю, що об'єднує декілька напрямків науки і техніки, включаючи транспортну інженерію, інформаційні технології, математику, фізику та штучний інтелект.

Головною метою інтелектуальних систем керування перехрестями (ІСКП) є оптимізація руху транспортних засобів через перехрестя, що сприяє зменшенню заторів, підвищенню безпеки руху і зниженню негативного впливу на навколишнє середовище. Це досягається за рахунок удосконалення роботи світлофорів, надання пріоритету громадському транспорту та екстреним службам, а також забезпечення безпеки пішоходів і велосипедистів. ІСКП базуються на використанні сучасних технологій і методів для збору, обробки і аналізу даних про транспортні потоки. Важливу роль відіграють різноманітні датчики руху, такі як камери, радари, лідари та індуктивні петлі, які збирають дані про кількість транспортних засобів, їх швидкість та напрямок руху. Зібрані дані обробляються за допомогою технологій великих даних, що дозволяє отримувати детальну картину транспортної ситуації в режимі реального часу.

Для аналізу і прийняття рішень в ІСКП широко використовуються методи штучного інтелекту, зокрема машинне навчання та глибоке навчання. Ці методи дозволяють прогнозувати транспортні потоки, оптимізувати фази світлофорів і адаптувати систему до змінних умов руху. Використання технологій зв'язку V2X (Vehicle-to-Everything) забезпечує обмін інформацією між транспортними засобами та інфраструктурою, що покращує координацію і ефективність системи.

Для моделювання та оптимізації роботи перехресть застосовуються різноманітні математичні та комп'ютерні моделі. Симуляційні моделі дозволяють імітувати транспортні потоки і оцінювати ефективність різних стратегій керування рухом. Математичні моделі використовуються для

аналізу транспортних систем і розробки оптимальних алгоритмів керування світлофорами.

Таким чином, дослідження та розробка моделі інтелектуальної системи керування перехрестями вимагає інтеграції багатьох різних технологій і методів. Це міждисциплінарна область, що постійно розвивається і вдосконалюється, сприяючи підвищенню ефективності транспортних систем і якості життя в містах.

Галузь керування дорожнім рухом має низку особливостей, а саме:

1. Комплексність. КДР охоплює широкий спектр задач, від проектування та будівництва доріг до регулювання руху транспорту, пішоходів та велосипедистів. Ця складність потребує глибоких знань в різних галузях, таких як інженерія, транспорт, безпека дорожнього руху, комп'ютерні науки та економіка;
2. Динамічність. Дорожній рух постійно змінюється, залежно від часу доби, дня тижня, пори року, погодних умов та інших факторів. Системи КДР повинні бути динамічними та адаптивними, щоб ефективно реагувати на ці зміни та забезпечувати безпечний та плинний рух;
3. Взаємодія з людьми. КДР має безпосередній вплив на життя людей, які користуються дорогами. Важливо, щоб системи КДР були зручними, зрозумілими та безпечними для всіх користувачів, включаючи водіїв, пішоходів, велосипедистів та людей з обмеженими можливостями;
4. Вплив на навколишнє середовище. Транспорт є одним з основних джерел забруднення повітря та шуму. Системи КДР повинні бути розроблені таким чином, щоб мінімізувати негативний вплив на навколишнє середовище та сприяти сталому розвитку;
5. Швидкий розвиток технологій. Галузь КДР постійно розвивається з появою нових технологій, таких як датчики, камери, штучний інтелект та підключені транспортні засоби. Важливо, щоб фахівці з

КДР були в курсі останніх досягнень та могли використовувати ці технології для покращення систем КДР;

6. Міждисциплінарна співпраця. Ефективне КДР потребує співпраці між фахівцями з різних галузей, таких як інженери, транспортні планувальники, фахівці з безпеки дорожнього руху, програмісти та представники влади.

### 2.2 Аналіз досліджень з теми

Дослідники визначають наступні джерела заторів у містах:

- недостатня пропускна здатність дороги;
- дорожні роботи;
- погана погода;
- дорожні роботи;
- рідкісні події;
- неоптимальний час сигналу [8].

Дослідження що фокусуються на покращенні ситуації з неоптимальним часом сигналу зосереджуються як на різних методах виявлення автомобілів та пішоходів так і на безпосередньо обробці даних отриманих від систем виявлення для визначення оптимального часу керуючого сигналу світлофора.

Для модуля виявлення автомобілів у дослідженнях використовувалися такі типи сенсорів як індуктивний контур, п'єзоелектричні сенсори, детектори у інфрачервоному спектрі, акустичні сенсори, мікрохвильові радари та обробка даних з відеокамер [9]. Кожен тип сенсору має свої переваги та недоліки.

Індуктивні контури мають високу точність у визначенні наявності транспортних засобів на перехресті. Вони можуть точно виявляти автомобілі незалежно від їх швидкості та типу, також вони менш чутливі до змін погодних умов, таких як дощ, сніг чи туман, що забезпечує стабільність роботи системи. Проте, процес встановлення індуктивних контурів вимагає розрізання дорожнього полотна, що є трудомістким та дорогим. Крім того,

ремонт та обслуговування також потребують значних витрат. Індуктивні контури можуть бути пошкоджені під час ремонтних робіт на дорозі або через інтенсивне навантаження від важких транспортних засобів, що може призвести до необхідності частих ремонтів.

П'єзоелектричні сенсори швидко реагують на зміни трафіку, що дозволяє оперативно передавати інформацію для керування дорожнім рухом. Також сенсори цього типу можуть точно визначати навантаження, що дозволяє контролювати перевантажені транспортні засоби і забезпечувати безпеку дорожнього руху. Але встановлення п'єзоелектричних сенсорів вимагає врізання в дорожнє полотно, що є дорогим та трудомістким процесом, інтенсивний рух важких транспортних засобів може пошкодити сенсори, що знижує їх ефективність, а також малий вихід напруги з п'єзоелектричних матеріалів викликає необхідність використання підсилювачів.

Інфрачервоні детектори працюють незалежно від рівня освітленості, що дозволяє їм функціонувати ефективно як вдень, так і вночі, можуть визначати розміри і форму об'єктів, що дозволяє класифікувати різні типи транспортних засобів (легкові автомобілі, вантажівки, мотоцикли). Проте вони є чутливими до погодних умов: інфрачервоні детектори можуть зазнавати труднощів під час сильного дощу, снігу або туману, що може знижувати точність виявлення транспортних засобів.

Акустичні детектори не залежать від погодних умов, таких як дощ, сніг чи туман, що забезпечує стабільність їх роботи в різних умовах, можуть охоплювати більшу зону, ніж деякі інші типи сенсорів, що дозволяє виявляти транспортні засоби на декількох смугах руху одночасно. Проте на точність підрахунку транспортних засобів може вплинути низька температура [8].

Мікрохвильові радари можуть точно визначати наявність і швидкість транспортних засобів, забезпечуючи високу точність виявлення та моніторингу трафіку, ефективно працюють в будь-яких погодних умовах, включаючи дощ, сніг, туман і темряву, що забезпечує стабільну роботу

системи, мають тривалий термін служби та високу надійність, що робить їх економічно ефективними в довгостроковій перспективі. Недоліками є обмежена точність виявлення у складних умовах: у густонаселених міських районах або при високій щільності трафіку радари можуть мати труднощі з точним виявленням окремих транспортних засобів, а також можуть зазнавати інтерференції від інших електронних пристроїв, що може знизити точність виявлення. Радари потребують кваліфікованого персоналу для їх налаштування, калібрування та обслуговування, що може збільшити витрати на експлуатацію системи.

Відеокамери можуть надавати високоточні дані про транспортні засоби, включаючи їх розміри, типи, швидкість та напрямок руху, охоплювати велику територію, дозволяючи моніторити декілька смуг руху і перехресть одночасно. Дані з відеокамер можуть бути інтегровані з іншими системами безпеки та моніторингу, такими як системи розпізнавання номерних знаків та облич. Також відеокамери легко налаштовуються і масштабуються, що дозволяє швидко адаптувати систему до змін у дорожній інфраструктурі та вимогах. Проте вони також мають велику кількість недоліків. Системи відеоспостереження можуть бути дорогими у встановленні та вимагати значних витрат на обслуговування, включаючи регулярне очищення та калібрування камер. Якість зображень з відеокамер може знижуватися в умовах поганої видимості, таких як туман, дощ або сніг, а також в умовах недостатнього або надлишкового освітлення. Обробка відео в реальному часі потребує значних обчислювальних ресурсів, що може вимагати інвестицій у потужне апаратне забезпечення та програмне забезпечення для аналізу відеоданих.

Також дані про автомобілі можуть подаватися до системи керування перехрестям безпосередньо з транспортних засобів використовуючи широкий спектр технологій без провідного зв'язку, проте, незалежно від типу такої системи, всі вони мають значний недолік, а саме необхідність наявності

передаючого пристрою на транспортному засобі в той час як значна кількість автомобілів може бути їм не оснащена.

Іншим підходом до зменшення часу очікування на світлофорі є системи що покладаються на комунікацію між транспортними засобами замість впливу на сигнал світлофору. Системи V2V дозволяють автомобілям обмінюватися даними про своє місцезнаходження, швидкість, напрямок руху та інші параметри, що дозволяє координаувати свою швидкість і дистанцію, щоб забезпечити плавний потік трафіку і зменшити затори. Дослідження [10] використовує алгоритми Circuit Patrol і Greedy Patrol для покращення оцінки матриці дорожнього руху на основі чого можуть будуватися більш оптимальні маршрути що оминають затори. У [11] досліджувалися Спеціальні Автомобільні Мережі, або VANETs. VANET дає змогу всім учасникам дорожнього руху (наприклад, транспортним засобам, світлофорам або придорожнім пристроям) обмінюватися інформацією та координувати свою поведінку, що є змішаним підходом між V2V та системами керування транспортною інфраструктурою. На симуляція дослідження показало зменшення часу подорожі навіть при невеликій розповсюдженості технології – 1 із 10 транспортних засобів комунікував з іншими.

Перейдемо до досліджень що фокусуються саме на алгоритмах управління перехрестям. Існує три типи керування часом керуючого сигналу: попередньо оптимізований контроль, адаптивне керування або реагування на трафік і напівактивне керування [12].

При попередньо оптимізованому контролі час сигналу є заздалегідь визначеним і оптимізується в режимі офлайн базуючись на історичних даних про завантаженість певного світлофора. Цей метод є найпростішим як з боку необхідності наявності додаткового обладнання обладнання, так як для нього не потрібні детектори, так і з боку програмної реалізації. Проте попередньо оптимізований контроль не може адаптивно реагувати на незаплановані зміни в інтенсивності дорожнього руху і є неефективним на перехрестях з

невеликою кількістю автомобілів через те що у такому випадку прибуття транспорту є наближеним до випадкового.

Адаптивне керування з'явилося після появи сенсорів для виявлення авто. Ідея адаптивного керування полягає в тому, що керуючий алгоритм змінює тривалість керуючих сигналів відповідно до отриманої з сенсорів інформації про наявність та об'єм трафіку. Більшість сучасних досліджень ведуться саме у напрямі адаптивного керування, оскільки потенційне зменшення часу очікування є більшим ніж у попередньо оптимізованих систем.

При напівактивному керуванні сенсор застосовується тільки для керування певною групою світлофорів [13].

Прикладом дослідження більш ефективного алгоритму для попередньо оптимізованого контролю є дослідження [14], де було порівняно алгоритм Maximum Intersection Utilization зі стандартним.

Алгоритм Maximum Intersection Utilization:

1. визначте дороги на перехресті в послідовності від 0 до 3 за годинниковою стрілкою;
2. почніть із ініціалізації всіх доріг до червоного стану;
3. припустимо, що будь-яка 1 дорога є початком, цикл продовжується після 4-го етапу;
4. на кожному етапі регулюються 2 дороги (один навпроти одного);
5. транспортні засоби на кожній смузі чекають своєї черги 3 цикли (як у звичайному русі), тоді як ті, що повертають ліворуч, чекають 2 цикли кожен;
6. якщо зіткнення не стосується алгоритму, можна надати вільний ліворуч для всіх етапів;
7. алгоритм працює за круговим графіком, надаючи однакові часові проміжки для кожної дороги;
8. продовжуйте роботу, доки користувач не видасть команду паузи або зупинки.

Середній час очікування для стандартної моделі дорівнював 26.7 циклів у симуляції, тоді як час очікування для запропонованого алгоритму становив 22.6, тобто покращення на 16%.

Розглянуті далі дослідження стосуються активних методів керування.

У дослідженні [15] було запропоновано модель без циклів, що переоцінює ситуацію на кожній фазі та обирає наступну фазу на основі спостережуваних параметрів системи. Алгоритм використовує матрицю конфліктів, яка описує всі можливі випадки конфліктних рухів і керує створенням фаз.

Дослідження [16] використовує сенсори для визначення довжини черги для кожної сторони перехрестя, за допомогою статистичних даних підраховує кількість зелених циклів, необхідних для того, щоб вся черга очистилася, і підвищує тривалість зеленого сигналу для сторони, що потребує найбільшої кількості циклів.

У [17] пропонується рішення усунення заторів шляхом синхронізації сигналів світлофорів, забезпечуючи сталий потік транспортних засобів. Це забезпечується комунікацією між світлофорами про навантаження на дорозі та відповідне адаптування тривалості сигналу під ситуацію.

Дослідження [18] пропонує модель контролера що базується на нечіткій логіці. Нечітка логіка є підходом до обробки інформації, який дозволяє враховувати невизначеність і нечіткість, характерні для багатьох реальних систем. Вона відрізняється від класичної булевої логіки, яка оперує двома станами: істинним (1) і хибним (0). Нечітка логіка, натомість, працює з величинами, які можуть мати значення між 0 і 1, що дозволяє більш точно відображати реальні умови та процеси. Основною концепцією нечіткої логіки є нечіткі множини. Кожен елемент у такій множині характеризується ступенем належності, що показує, наскільки цей елемент відповідає певній властивості. Наприклад, у множині "високі люди" кожна людина матиме певний ступінь належності, який показує, наскільки вона висока. Це значення може варіюватися від 0 до 1, де 0 означає, що людина зовсім не є високою, а

1 - що вона повністю відповідає поняттю "висока". Модель експертної системи, що базується на нечіткій логіці, отримує чисельні дані, конвертує їх у нечіткі значення, користуючись базою правил приймає рішення щодо планування довжини керуючого сигналу світлофора та виводить їх. У симуляціях запропонована модель демонструє кращі результати ніж контролер із фіксованим часом або контролер, що приводиться в дію транспортним засобом.

У дослідженні [19] також використовується модель нечіткої логіки але у взаємодії з генетичним алгоритмом. Генетичні алгоритми – це методи оптимізації та пошуку, які імітують природні еволюційні процеси, такі як селекція, схрещування та мутація. Вони належать до класу еволюційних алгоритмів і використовуються для знаходження наближених рішень складних задач, де традиційні методи можуть бути неефективними. Генетичні алгоритми базуються на ідеях Чарльза Дарвіна про природний відбір і "виживання найсильніших". Основна ідея полягає в тому, щоб використовувати популяцію можливих рішень (індивідів), які еволюціонують з покоління в покоління, поступово покращуючи свої характеристики для досягнення оптимального або близького до оптимального результату. У цьому випадку генетичний алгоритм створює таблицю правил для нечіткої логіки, а також корегує функції приналежності для фазифікації та дефазифікації.

Дослідження [20] використовує глибоку нейронну мережу у поєднанні з навчанням з підкріпленням. Глибокі нейронні – це тип штучних нейронних мереж, що складаються з багатьох шарів обчислювальних вузлів, або нейронів. Вони належать до галузі глибокого навчання, яка є підмножиною машинного навчання. Глибокі нейронні мережі здатні виявляти складні патерни в даних і виконувати широкий спектр завдань, включаючи розпізнавання образів, обробку природної мови, машинний переклад, автономне керування і багато іншого. Основою глибоких нейронних мереж є концепція шарів. Вхідний шар отримує дані, а кожен наступний шар, званий

прихованим, обробляє виходи попереднього шару за допомогою математичних операцій, таких як зважування входів і застосування активаційних функцій. Кожен нейрон в прихованому шарі обчислює зважену суму своїх входів, яку потім пропускає через активаційну функцію. Ця функція додає нелінійність до моделі, що дозволяє мережі навчатися і представляти складні взаємозв'язки в даних. Останній шар називається вихідним шаром і він виробляє кінцевий результат мережі. Кількість нейронів у цьому шарі залежить від конкретного завдання (наприклад, один нейрон для бінарної класифікації або кілька нейронів для багатокласової класифікації). У випадку з задачею керування світлофором виходом вихідного шару буде тривалість наступної фази керуючого сигналу. Навчання з підкріпленням є типом машинного навчання, в якому агент навчається приймати рішення, взаємодіючи з середовищем, щоб максимізувати накопичену винагороду. Цей підхід базується на пробах і помилках, де агент експериментує з різними діями, отримуючи позитивні або негативні підкріплення, і використовує цю інформацію для покращення своєї стратегії. Основна ідея навчання з підкріпленням полягає в тому, що агент знаходиться в певному стані, виконує дію і переходить у новий стан, отримуючи при цьому винагороду. Мета агента - знайти політику, яка максимізує його очікувану сумарну винагороду в довгостроковій перспективі. Політика визначає, яку дію агент повинен виконати в кожному можливому стані. При задачі оптимізації руху автомобілів агент буде отримувати винагороду при зменшені середнього часу очікування. У дослідженні, використовуючи ці методики, вдалося знизити час очікування приблизно на 50% після 500 епох (циклів) навчання моделі.

# 3 МОДЕЛЮВАННЯ ТА ПРОЕКТУВАННЯ СИСТЕМИ

## 3.1 Визначення вимог до системи

Функціональні вимоги до інтелектуальної системи керування перехрестями:

1. базові функції:

- регулювання руху: система повинна автоматично регулювати рух транспорту, пішоходів та велосипедистів на перехресті;
- збір даних: система повинна збирати дані про дорожній рух, такі як інтенсивність трафіку, час очікування, швидкість руху та інші фактори;
- аналіз даних: система повинна аналізувати дані про дорожній рух, щоб виявляти закономірності та прогнозувати потоки транспорту;
- прийняття рішень: система повинна приймати рішення про те, як регулювати рух на перехресті на основі аналізу даних;
- адаптивність: система повинна бути адаптивною до мінливих умов дорожнього руху;
- оптимізація: система повинна оптимізувати рух на перехресті, щоб зменшити затори, час очікування та викиди забруднюючих речовин;
- віддалений доступ: система повинна мати можливість керуватися та моніторитися віддалено службами що мають доступ до керування світлофорами.

2. додаткові функції:

- інформація для користувачів: система повинна надавати інформацію для користувачів, таким як час очікування та рекомендовані маршрути;
- зв'язок з іншими системами: система повинна мати можливість зв'язуватися з іншими системами, такими як GPS-навігатори та системи управління громадським транспортом;

– підтримка аварійних ситуацій: система повинна мати можливість підтримувати роботу в аварійних ситуаціях, таких як вимкнення електроенергії або дорожні аварії.

Нефункціональні вимоги до інтелектуальної системи керування перехрестями (ІСКП):

1. продуктивність:

– швидкість обробки даних: система повинна швидко обробляти дані про дорожній рух, щоб забезпечити своєчасне та ефективне регулювання;
– час відгуку: система повинна мати швидкий час відгуку на зміни дорожнього руху;
– масштабованість: система повинна бути масштабованою, щоб її можна було використовувати на перехрестях з різною інтенсивністю руху.

2. надійність:

– доступність: система повинна бути доступною 24/7 з мінімальним простоєм;
– стійкість до збоїв: система повинна бути стійкою до збоїв обладнання та програмного забезпечення;
– відновлення після збоїв: система повинна швидко відновлюватися після збоїв.

3. безпека:

– захист даних: система повинна захищати дані про дорожній рух від несанкціонованого доступу, модифікації або знищення;
– кібербезпека: система повинна бути захищеною від кібератак;
– фізична безпека: система повинна бути захищена від фізичного пошкодження.

4. підтримка:

– документація: система повинна мати чітку та вичерпну документацію;

– навчання: повинні бути доступні навчальні матеріали та програми для операторів та користувачів;
– технічна підтримка: повинна бути доступна технічна підтримка для вирішення проблем та усунення несправностей.

5. відповідність регуляторним нормам:
– оскільки система взаємодіє у регульованому оточені з чіткими правилами, що, при їх порушенні, можуть становити загрозу життю та здоров'ю користувачів, необхідно щоб система чітко дотримувалася встановлених норм керування дорожнім рухом.

## 3.2 Проектування сенсорного модулю системи на основі відокремлення фону

Як вже було зазначено в розділі аналізу досліджень з теми, у системах керування перехрестями можуть використовуватись такі типи сенсорів як індуктивний контур, п'єзоелектричні сенсори, детектори у інфрачервоному спектрі, акустичні сенсори, мікрохвильові радари та обробка даних з відеокамер. У даному досліджені була використана обробка даних з відеокамер, оскільки зразки вхідних даних для такої системи містяться у вільному доступі у Інтернеті.

Задачею сенсорного модулю системи є підрахунок транспортних засобів що знаходяться у зоні дії сенсору (у даному випадку – у кадрі камери відео спостереження). Оскільки камера на перехресті є нерухомою, а отже вид дороги та її оточення залишаються незмінними від кадру до кадру, підрахувати транспортні засоби або пішоходів можна знайшовши різницю поточного кадру з порожньою дорогою.

Знайти різницю поточного кадру з порожньою дорогою можна, представивши кадри як двомірний масив, ітерацією по пікселям та порівнянням відповідних пікселів. У разі, якщо різниця у RGB значенні пікселю перевищує деяке порогове значення (порогове значення є необхідним для того, щоб нівелювати вплив можливих незначних змін у

освітленні дороги, тощо) зберігати координати цього пікселя або додавати його до окремого масиву. Алгоритм функції знаходження різниці між кадрами зображено на рисунку 3.1.

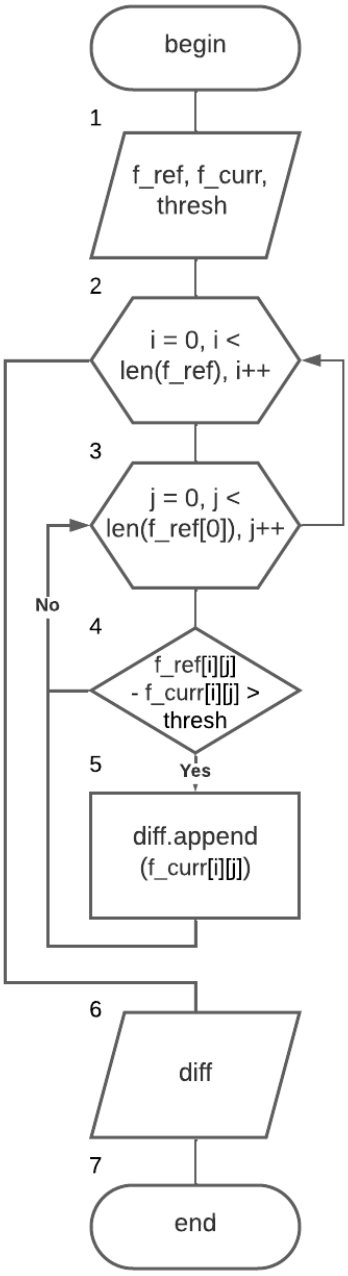


Рисунок 3.1 – Алгоритм функції знаходження різниці між кадрами

f_ref – порожній кадр, f_curr – поточний кадр, thresh – порогове значення, diff – масив, що зберігає пікселі що були присутні лише на поточному кадрі.

Після знаходження різниці кадрів навіть при коректно підібраному пороговому значенні частина пікселів автомобілів не буде відділеною на кадр різниці і тому, якщо застосувати алгоритм пошуку контуру до необробленого кадру різниці, для кожного авто буде знайдено декілька контурів, отже після підрахунку кількості контурів їх кількість буде значно більшою за реальну кількість авто на початковому кадрі. Для злиття груп пікселів що відповідають одному авто можна застосувати алгоритм розмиття по Гаусу, після чого бінаризувати результуюче зображення для спрощення знаходження контурів.

Алгоритм розмиття по Гаусу є однією з основних технік в обробці зображень [21], що використовується для згладжування зображень і зменшення шуму. Ідея цього алгоритму полягає в тому, щоб кожен піксель зображення замінити зваженим середнім значенням його сусідів, де ваги визначаються Гауссовою функцією розподілу.

Щоб реалізувати розмиття по Гаусу, спочатку потрібно створити Гауссове ядро. Це ядро представляє собою двовимірну матрицю, розмір якої зазвичай є непарним числом (наприклад, 3x3, 5x5, 7x7). Розмір ядра визначає, скільки сусідніх пікселів буде враховано під час розрахунку нового значення для кожного пікселя. Значення елементів в ядрі розраховуються за допомогою Гауссової функції, яка має форму дзвону і визначається стандартним відхиленням (σ). Чим більше значення σ, тим сильніше розмиття.

Формула для знаходження значення в ядрі Гауса (3.1):

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \quad (3.1)$$

де $x$ і $y$ - координати пікселя в ядрі;

$\sigma$ - стандартне відхилення.

Для обробки пікселів, які знаходяться на краю зображення, використовуються різні методи, оскільки ядро частково виходить за межі зображення. Найбільш поширеними методами є заповнення меж нулями, дублювання крайніх пікселів або "обгортання" зображення, де пікселі з одного краю зображення з'єднуються з протилежним краєм.

Алгоритм створення ядра функції Гауса зображено на рисунках 3.2-3.3

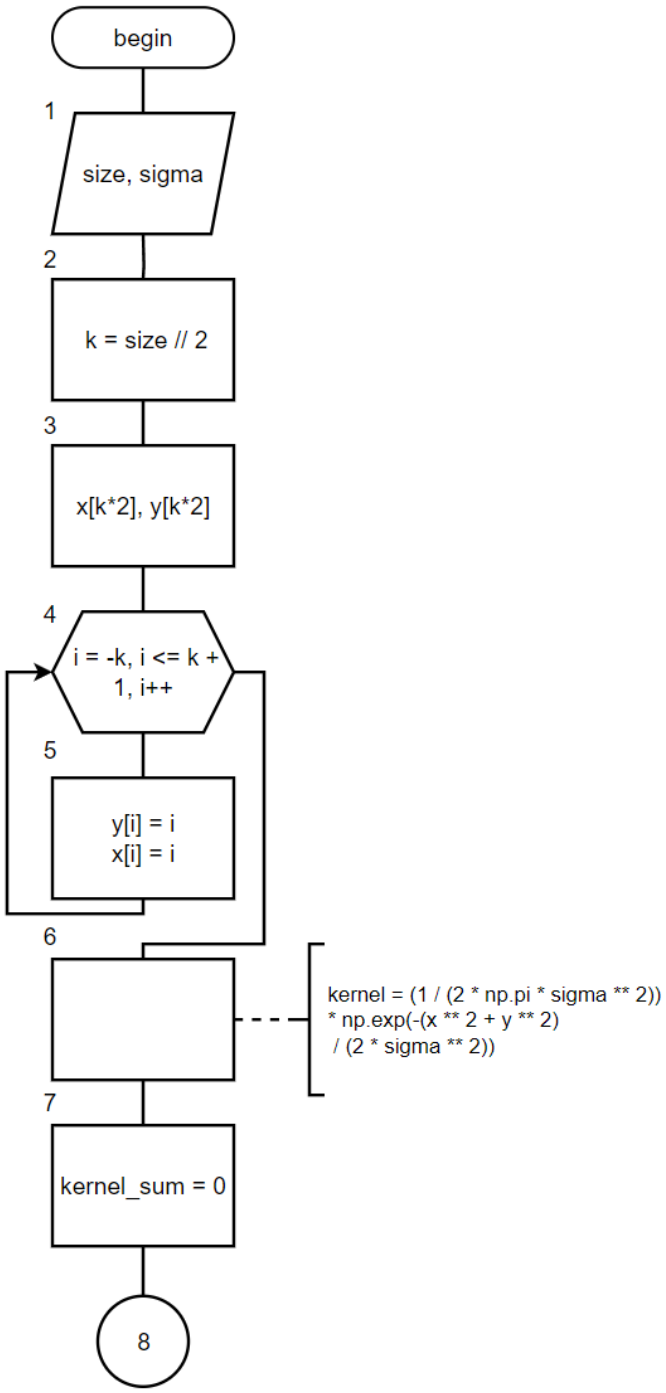


Рисунок 3.2 – Алгортим створення ядра для функції Гауса, частина 1

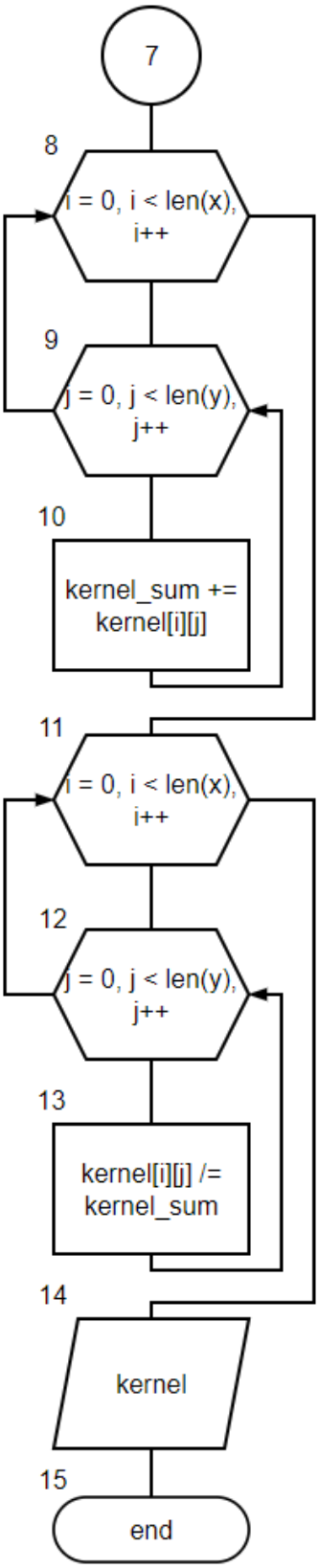


Рисунок 3.3 – Алгортим створення ядра для функції Гауса, частина 2

Після того як ядро згенеровано, алгоритм застосовує його до кожного пікселя зображення. Для цього використовують операцію згортки: ядро

"ковзає" по всьому зображенню, і для кожної позиції обчислюється нове значення пікселя як зважене середнє значення його сусідів, взятих із вихідного зображення. Таким чином, нове значення пікселя є сумою добутків інтенсивностей пікселів із вихідного зображення на відповідні значення в ядрі Гауса. Це дозволяє отримати більш згладжений вигляд зображення, де шуми та різкі переходи згладжуються. Алгоритм конволюції зображено на рисунках 3.4-3.5.

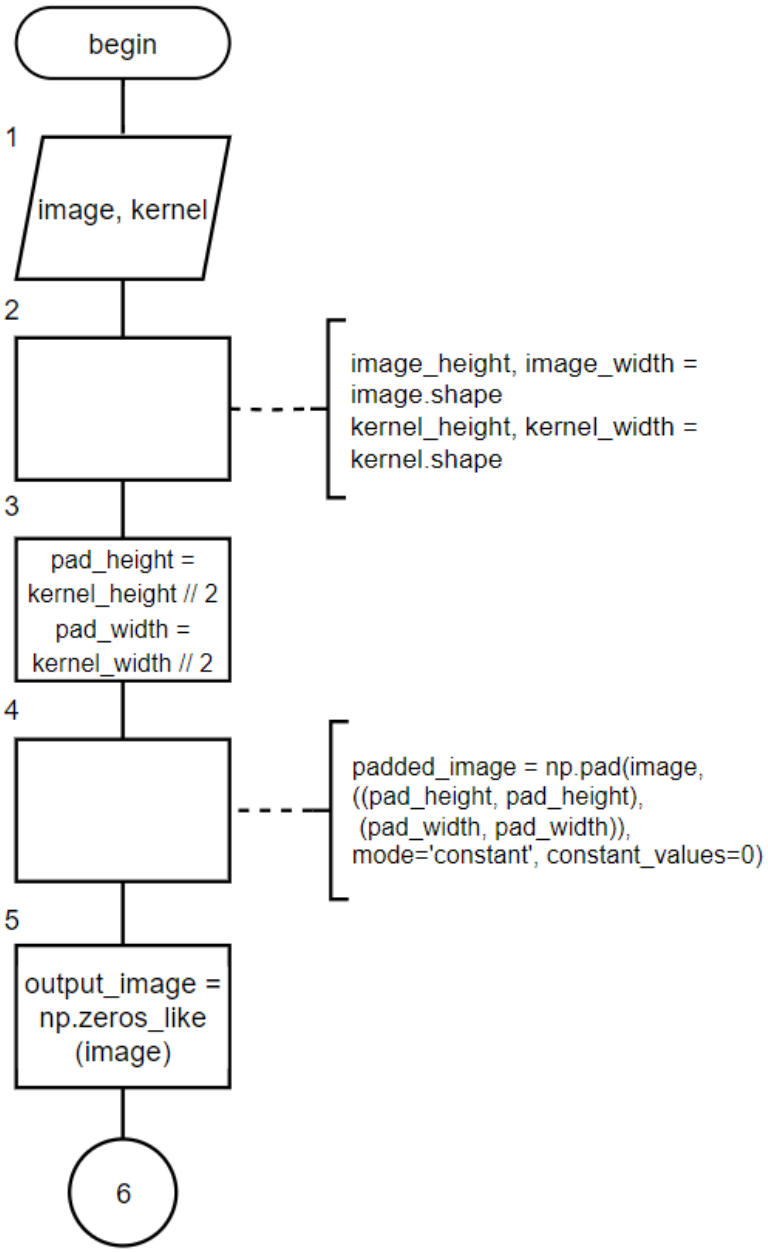


Рисунок 3.4 – Алгоритм конволюції, частина 1

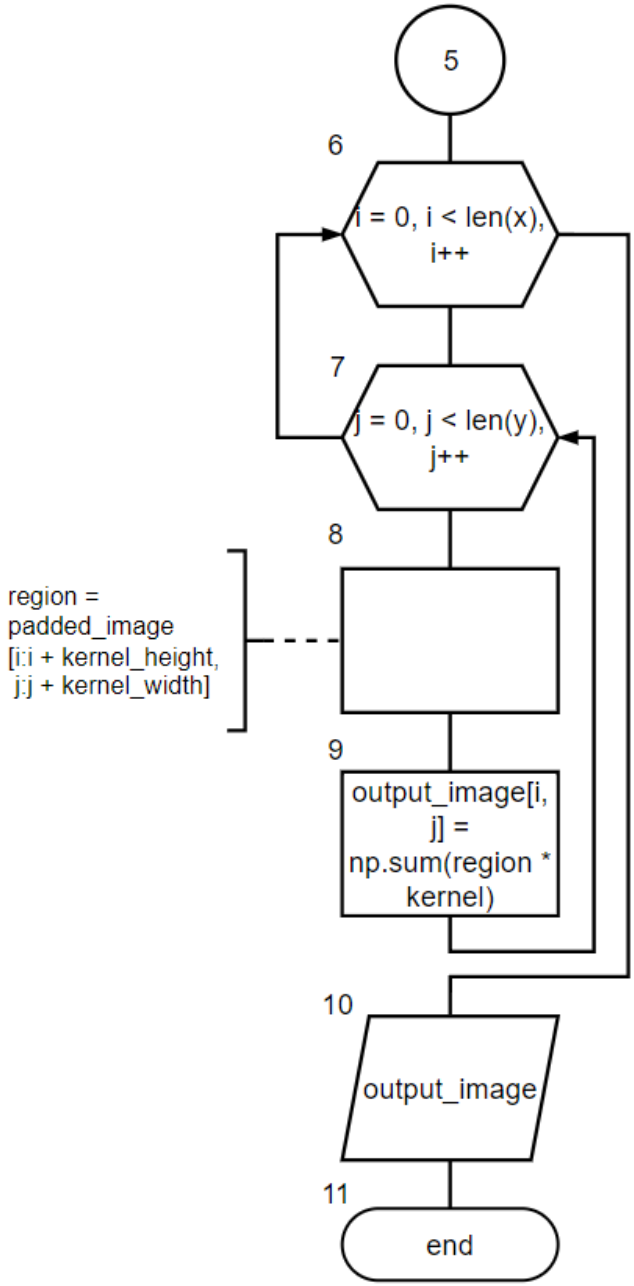


Рисунок 3.5 – Алгоритм конволюції, частина 2

Результатом роботи алгоритму є зображення на якому пікселі що належать одному автомобілю з'єднані в пляму. Для подальшого знаходження контурів авто необхідно провести бінарізацію зображення – всі пікселі, що мають прозорість вище певного порогового значення отримують нове значення 1, всі що мають значення нижче порогового отримують значення 0.

Алгоритм бінарізації зображено на рисунку 3.6.

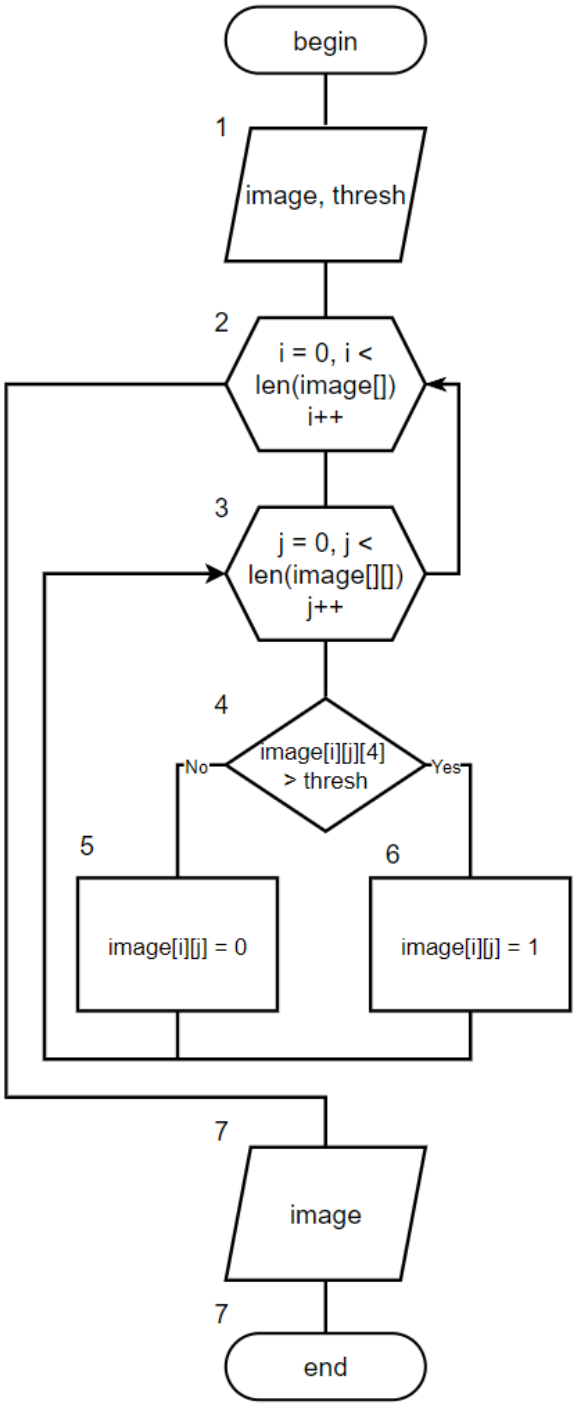


Рисунок 3.6 – Алгоритм бінарізації зображення

Після бінарізації зображення все ще деякі частини одного автомобілю можуть бути відокремленими або два або більша авто, що знаходяться на початковому кадрі можуть злитися в одну пляму. Для того, щоб підвищити якість плям і позбутися шуму, що міг залишитися, до отриманого у

попередньому кроці бінаризованого зображення можна застосувати морфологічні перетворення.

Морфологічні перетворення – це набір операцій в обробці зображень, які базуються на теорії множин і використовуються для аналізу та обробки бінарних зображень, а також для роботи з відтінковими зображеннями. Ці перетворення застосовують для виділення структурних елементів та об'єктів на зображеннях, покращення контурів, усунення шуму, а також для інших завдань комп'ютерного зору [22].

Основою морфологічних перетворень є поняття структурного елемента. Структурний елемент – це невелика матриця, яка визначає, як буде змінюватися форма пікселів на зображенні. Зазвичай, структурний елемент має невеликий розмір і просту форму, наприклад, квадрат або коло, але він може бути довільним залежно від завдань. Морфологічні операції працюють шляхом переміщення структурного елемента по зображенню і застосування логічних операцій до пікселів, що потрапляють під його дію.

Однією з базових операцій морфології є ерозія, яка "з'їдає" межі об'єктів на зображенні. Ерозія зменшує розміри об'єктів і видаляє маленькі деталі, тому вона використовується для усунення шуму. При застосуванні ерозії піксель залишається на зображенні лише в тому випадку, якщо всі пікселі, які знаходяться під структурним елементом, мають значення "1". Якщо хоча б один піксель не задовольняє цій умові, піксель вважається "0".

Інша ключова операція – дилатація. Вона є протилежністю ерозії та збільшує розміри об'єктів на зображенні, заповнюючи пробіли та розширюючи краї. Дилатація додає пікселі до меж об'єкта. Якщо хоча б один піксель, що потрапляє під структурний елемент, має значення "1", то центральний піксель також стає "1". Це дозволяє збільшити об'єкт і з'єднати розірвані частини.

На основі ерозії та дилатації будуються складніші морфологічні операції, такі як відкриття і закриття. Відкриття поєднує ерозію і дилатацію. Спочатку зображення піддається ерозії, що видаляє шум та дрібні об'єкти, а

потім дилатації, яка відновлює об'єкти, які залишилися. Ця операція дозволяє ефективно видаляти маленькі об'єкти, не впливаючи на великі структури. Закриття, навпаки, спершу виконує дилатацію, а потім ерозію. Воно дозволяє заповнювати прогалини всередині об'єктів і з'єднувати розірвані частини, при цьому згладжуючи їхні контури.

Для виконання поточної задачі підходять операції ерозії та дилатації з різним розміром ядра.

Після проведення морфологічних перетворень необхідно провести пошук контурів. На бінаризованих зображеннях алгоритми знаходження контурів ґрунтуються на пошуку меж між областями з різними значеннями пікселів, зазвичай 0 (чорний) і 1 (білий). У цьому випадку контур визначається як лінія, що розділяє області чорного і білого кольорів. Завдання знаходження контурів на бінаризованому зображенні полягає у виявленні пікселів, які утворюють межу між цими двома областями. Одним з можливих алгоритмів знаходження контурів є алгоритм ланцюгового кодування Фрімана. Цей метод представляє контур у вигляді послідовності кроків або рухів у восьми напрямках, що дозволяє компактно зберігати форму об'єкта. Починаючи з певного пікселя на контурі, алгоритм переходить до сусіднього пікселя відповідно до напрямку і продовжує цей процес до замикання контуру. Таке кодування є зручним для подальшого аналізу і обробки зображень, оскільки воно дозволяє зберегти інформацію про форму об'єкта в стислому вигляді.

Кількість контурів, знайдених на результуючому зображені має відповідати кількості авто на початковому зображенні.

Для коректної роботи системи також необхідно періодично оновлювати еталонний кадр шляхом додавання до нього через визначену кількість кадрів поточного кадру помноженого на певний невеликий ($0.001 < k < 0.01$) коефіцієнт для того, щоб система могла адаптуватися до змін освітлення через денний цикл, або об'єктів що не є частиною ні дорожнього руху ні початкового зображення, наприклад плям на асфальті.

### 3.3 Проектування сенсорного модулю системи на основі моделі виявлення об'єктів

Object detection (виявлення об'єктів) – це завдання комп'ютерного зору, яке полягає у визначенні та локалізації об'єктів певних класів на зображенні або відео. На відміну від задачі класифікації, де система визначає, до якого класу належить зображення, при виявлені об'єктів необхідно знайти не тільки клас об’єкта, але й його місцезнаходження у вигляді прямокутної рамки (bounding box), проте в основі моделі виявлення об'єктів використовується модель-класифікатор, найчастіше це конволюційна мережа.

Конволюційні нейронні мережі є типом глибоких нейронних мереж, які спеціально розроблені для обробки даних з певною структурою, таких як зображення. Основна ідея CNN полягає в автоматичному виявленні ієрархії ознак зображення через застосування конволюційних операцій, що робить їх особливо ефективними для задач комп'ютерного зору, таких як класифікація зображень, виявлення об'єктів, сегментація та інше [23].

Конволюційні мережі складаються з кількох типів шарів: конволюційних шарів, шарів підвибірки (pooling) і повнозв'язних шарів (fully connected layers). Конволюційний шар є основним компонентом CNN, в якому застосовується фільтр (ядро) для перетворення вхідних даних. Фільтр "ковзає" по всьому зображенню, застосовуючи операцію згортки, яка дозволяє виділити локальні патерни, такі як контури, грані, кути, текстури тощо. В результаті створюються карти ознак (feature maps), що є виходом конволюційного шару і представляють різні аспекти вихідного зображення.

Шар підвибірки або пулингу зменшує розмірність карт ознак, зберігаючи найбільш важливу інформацію. Пулинг часто застосовується після конволюційного шару і дозволяє зменшити обчислювальні витрати і знизити ризик перенавчання. Найпоширеніший тип пулингу – це Max Pooling, який обирає максимальне значення в кожному регіоні, що дозволяє зберегти яскраво виражені ознаки та зменшити розмірність.

Після декількох шарів конволюції та пулингу, конволюційна нейронна мережа часто включає повнозв'язні шари, які відповідають за класифікацію та інші завдання. Ці шари мають кожен нейрон, з’єднаний з усіма нейронами попереднього шару, що дозволяє отримувати комплексні зв'язки між ознаками і приймати рішення на основі їх комбінацій. Наприкінці мережі зазвичай використовується шар softmax або сигмоїдальна функція активації для отримання ймовірностей належності до класів.

Навчання конволюційних нейронних мереж базується на методі зворотного поширення помилки (backpropagation) в поєднанні з оптимізаційними алгоритмами, такими як градієнтний спуск. Під час навчання мережа автоматично підбирає ваги фільтрів у конволюційних шарах і ваги у повнозв'язних шарах таким чином, щоб мінімізувати похибку на навчальних даних. Здатність конволюційних мереж автоматично виділяти та оптимізувати ознаки зображення на основі навчання робить їх потужним інструментом для розпізнавання об'єктів, класифікації та інших задач комп'ютерного зору.

Конволюційні нейронні мережі відзначаються властивістю інваріантності до зсувів та масштабів об'єктів на зображенні завдяки своїй структурі та використанню шарів пулингу. Вони здатні захоплювати локальні патерни незалежно від їх розташування на зображенні, що робить їх надзвичайно ефективними для задач, де об'єкти можуть перебувати в різних позиціях і масштабах.

Сьогодні CNN є однією з найуспішніших архітектур глибокого навчання і широко застосовується в різних галузях, включаючи медицину для аналізу медичних знімків, автономні транспортні засоби для розпізнавання доріг та об'єктів, а також в багатьох інших додатках, де необхідно обробляти візуальні дані. Завдяки своїй універсальності, простоті та високій точності, конволюційні нейронні мережі стали основою багатьох сучасних систем комп'ютерного зору.

Існує багато моделей і алгоритмів для виявлення об'єктів, які можна розділити на два основні підходи: двоетапні моделі (two-stage) і одноетапні моделі (one-stage).

Двоетапні моделі, такі як R-CNN (Region-based Convolutional Neural Networks) і її вдосконалені версії Fast R-CNN та Faster R-CNN, працюють у два етапи. Перший етап передбачає генерацію можливих регіонів (пропозицій регіонів), які можуть містити об'єкти. Ці регіони визначаються за допомогою методів, таких як Selective Search або регіонних пропозицій (Region Proposal Network, RPN), які використовуються у Faster R-CNN. На другому етапі для кожного регіону пропозиції застосовується нейронна мережа для класифікації і точного визначення меж об'єкта. Цей підхід забезпечує високу точність, але є досить повільним через необхідність обробки кожного регіону окремо.

Одноетапні моделі, такі як YOLO (You Only Look Once), SSD (Single Shot MultiBox Detector) і RetinaNet, працюють швидше, оскільки вони не генерують окремі регіони пропозицій, а безпосередньо передбачають класи і межі об'єктів на основі всього зображення за один прохід через нейронну мережу. YOLO розділяє зображення на сітку і для кожної комірки передбачає рамки, які можуть містити об’єкти. SSD використовує кілька масштабів рамок і різні рівні ознак, що дозволяє краще обробляти об'єкти різних розмірів. RetinaNet, зі свого боку, вводить концепцію Focal Loss, що допомагає справлятися з проблемою незбалансованості між фоновими і об'єктними прикладами.

Більш сучасні моделі, такі як EfficientDet, CenterNet, та DETR (DEtection TRansformer), пропонують покращення за рахунок нових архітектур і механізмів. EfficientDet оптимізує як точність, так і швидкість, використовуючи комбіновані блоки і різні рівні ознак. CenterNet визначає центри об'єктів і будує рамки на основі цих центрів, що спрощує процес детектування. DETR використовує трансформерні моделі і механізм уваги

(attention), що дозволяє уникнути складних процедур постобробки, таких як non-maximum suppression (NMS).

Кожна модель об'єктного детектування має свої переваги і недоліки, залежно від завдання. Наприклад, Faster R-CNN часто використовується для високоточної обробки, коли швидкість не є критичною. YOLO і SSD більше підходять для реального часу, коли потрібна швидкість обробки. Сучасні моделі, такі як DETR, пропонують нові можливості для поліпшення результатів і спрощення архітектури, що робить їх актуальними для багатьох новітніх застосувань у комп'ютерному баченні.

Для використання у сенсорному модулі системи було обрано архітектуру YOLO, яка є швидшою за аналоги, через необхідність працювати з даними в реальному часі.

Архітектура YOLO (You Only Look Once) є однією з найбільш відомих моделей для виявлення об'єктів в реальному часі. Її основна концепція полягає в тому, щоб перетворити завдання детектування об'єктів на задачу регресії, яка передбачає одночасне визначення меж (bounding boxes) об'єктів і їх класифікацію за один прохід через нейронну мережу [24]. Цей підхід дозволяє досягти дуже високої швидкості обробки зображень, що робить YOLO ідеальним для застосувань, де важлива продуктивність в реальному часі, таких як відеоспостереження, автономні транспортні засоби, та інші задачі комп'ютерного зору.

Архітектура YOLO працює шляхом розбиття вхідного зображення на сітку розміром $S \times S$. Кожна комірка сітки відповідає за виявлення об'єктів, центральна частина яких знаходиться в цій комірці. Мережа передбачає певну кількість межевих рамок (bounding boxes) і ймовірностей класів для кожної комірки. Для кожної межевої рамки визначається координати центру, ширина, висота та оцінка впевненості, що вказує, наскільки ймовірно, що рамка дійсно містить об'єкт. Класи об'єктів також прогнозуються для кожної рамки, що дозволяє визначити, який саме об'єкт знаходиться в межах цієї рамки.

Основною частиною архітектури YOLO є нейронна мережа на основі згорток, яка обробляє вхідне зображення і виявляє об'єкти. Базова версія YOLO використовує архітектуру Darknet, яка складається з декількох шарів згортки (convolutional layers) з різними розмірами фільтрів, за якими слідують шари підвибірки (pooling layers) для зменшення розмірності ознак. Завдяки цьому мережа отримує можливість захоплювати локальні патерни на різних рівнях абстракції і поєднувати їх для точного визначення контурів і класів об'єктів.

Однією з ключових переваг YOLO є її підхід до передбачення одночасно всіх об'єктів на зображенні. Замість обчислення ознак для кожного можливого регіону (як це робиться в двоетапних моделях, наприклад, Faster R-CNN), YOLO обробляє все зображення за один прохід, що значно підвищує швидкість. Це досягається завдяки використанню єдиного вихідного тензора, який містить всю необхідну інформацію для межевих рамок і класифікацій. Алгоритм також має вбудовані механізми для оптимізації результатів виявлення об'єктів, такі як non-maximum suppression (NMS). Ця техніка допомагає відфільтровувати накладені межі об'єктів, залишаючи тільки ті рамки, які мають найвищу впевненість для кожного об'єкта. Таким чином, можна зменшити кількість помилкових спрацьовувань і отримати більш точні результати.

Подальші версії YOLO, такі як YOLOv2, YOLOv3, YOLOv4, YOLOv5 і інші, включають різні покращення, спрямовані на підвищення точності, зменшення помилок і збільшення швидкості обробки. Вони вводять нові методи обробки ознак, оптимізаційні стратегії, такі як використання якірних рамок (anchor boxes), різні розміри шарів, багатомасштабне навчання, а також інші інновації, які дозволяють адаптувати модель до більш складних задач.

В цілому, архітектура YOLO є ефективною завдяки своєму підходу до єдиного проходу і простій структурі, що дозволяє досягати значної продуктивності в різних застосуваннях комп'ютерного зору. Це робить

YOLO одним із найпопулярніших виборів для реальних проектів, де важливі швидкість і точність виявлення об'єктів.

Архітектура YOLOv1 складається загалом із 24 згорткових шарів, за якими слідують 2 повністю з’єднані шари. На виході модель видає координати центрів обмежувальних рамок, їх висоту та ширину.

Для застосування моделі у розроблюваній системі її необхідно навчити на зображеннях, що містять автомобілі з різних ракурсів. Підбір навчальних даних є критичним етапом для успішного створення моделі виявлення об'єктів. Якість і різноманітність даних безпосередньо впливають на здатність моделі коректно виявляти об'єкти у реальних сценаріях. Модель повинна бути навчена на даних, які відображають різноманітність об'єктів у різних умовах, включаючи різні фони, освітлення, кути огляду та масштаби. Це особливо важливо для виявлення об'єктів, таких як автомобілі, де вплив навколишнього середовища та різноманітність об'єктів можуть бути значними.

Початковий крок у підборі даних – це визначення області застосування. Для задачі виявлення об'єктів важливо мати зображення, на яких чітко видно об'єкти, які необхідно виявити, з позначеними межами і відповідними класами об'єктів. Це називається анотуванням даних. Анотації повинні бути точними та добре визначеними, оскільки вони служать орієнтирами для моделі під час навчання. Дані повинні включати різні класи об'єктів, їх розміри, кількість об'єктів на зображенні, різні умови освітлення і фонові сцени. Це допомагає моделі навчитися ідентифікувати об'єкти в різноманітних ситуаціях.

Важливо враховувати баланс між класами у навчальних даних. Якщо дані незбалансовані (наприклад, значно більше зображень одного класу, ніж іншого), модель може навчитися віддавати перевагу більш представленому класу, що може призвести до значного погіршення якості виявлення менш представлених класів. Для вирішення цієї проблеми можна застосовувати методи збалансування даних, такі як надсемплінг рідкісних класів,

підсемплінг частих класів або генерування нових даних за допомогою технік аугментації.

Загальні вимоги до набору навчальних даних включають високу якість зображень, наявність різноманітних кутів огляду та відстаней до об'єктів, різні типи фону, включення зображень з шумами і артефактами, а також забезпечення різноманітності в розмірі та формі об'єктів. Крім того, корисно мати різні сценарії, в яких зображення можуть містити як декілька об'єктів одного класу, так і декілька об'єктів різних класів, щоб навчити модель добре розрізняти об'єкти в складних сценах.

Коли мова йде про виявлення автомобілів вибір навчальних даних має кілька особливостей. Важливо включати зображення автомобілів у різних умовах: вдень і вночі, при різних погодних умовах, таких як дощ, сніг, туман; на різних типах доріг (міські вулиці, швидкісні шосе, сільські дороги). Вибір даних також повинен враховувати різні типи автомобілів – легкові, вантажні, автобуси, мотоцикли тощо.

Для тренування обраної моделі використовувалися кадри з камер фіксації дорожнього руху що знаходяться у вільному доступі. Після отримання набору нерозмічених фото для того, щоб модель розуміла, що є цільовим об'єктом а що ні, необхідно провести анотування зображень. Анотування зображень – це процес маркування візуальних даних, таких як зображення чи відео, для навчання моделей машинного навчання, особливо у задачах комп'ютерного зору. Цей процес передбачає визначення і позначення різних об'єктів, ознак або областей на зображеннях, щоб надати машиним моделям інформацію про те, що вони бачать. Анотації є критично важливими для навчання моделей глибокого навчання для різних завдань, таких як виявлення об'єктів, класифікація, сегментація, розпізнавання образів, трекінг об'єктів та інші.

Процес анотування зображень може бути різним залежно від задачі, яка вирішується. Наприклад, для задачі класифікації зображень необхідно позначити весь кадр певним класом, тоді як для виявлення об'єктів потрібно

виділити кожен об'єкт прямокутною рамкою (bounding box) і прив'язати до нього мітку класу. Для задач сегментації анотація може бути більш детальною – необхідно позначити кожен піксель зображення, який належить певному об'єкту або фону.

Існують кілька основних методів анотування зображень:

- bounding boxes (Прямокутні рамки). Цей метод широко використовується для задач виявлення об'єктів. Анотування полягає у оточенні кожного об'єкту на зображенні прямокутною рамкою. Наприклад, якщо задача полягає у виявленні автомобілів на дорозі, кожен автомобіль повинен бути помічений прямокутною рамкою з відповідною міткою "автомобіль". Цей метод добре підходить для об'єктів, що мають регулярну форму, але може бути менш точним для об'єктів з нерегулярною формою.
- полігони. Для складніших форм, коли прямокутні рамки не забезпечують достатньої точності, використовують полігони. Анотування полігонами дозволяє точно визначати контури об'єктів будь-якої форми. Наприклад, для сегментації об'єктів, таких як дерева або люди, полігони дозволяють точніше виділити межі цих об'єктів. Це особливо важливо в медичній діагностиці, де потрібно точно виділити області інтересу на медичних зображеннях.
- лінії та точки. Цей метод використовується для задач, таких як визначення доріг, трекінг ліній або анотація скелетів людей (pose estimation). Лінії та точки допомагають точно визначити важливі риси об'єкта, такі як перетини або ключові точки.
- сегментація на рівні пікселів. Цей метод використовується для задач сегментації зображень і вимагає анотування кожного пікселя зображення, який належить певному класу. Це найточніший метод анотування і широко використовується у медичних, автомобільних і робототехнічних задачах. У таких випадках, як автономне водіння,

сегментація на рівні пікселів дозволяє точно визначити межі дороги, пішоходів, транспортних засобів, тротуарів тощо.

– ключові точки. Анотування ключових точок включає розміщення маркерів на важливих місцях об'єктів. Наприклад, для задач розпізнавання обличчя важливо визначити ключові точки на очах, носі, роті та інших частинах обличчя. Це дозволяє навчити моделі розпізнавати обличчя навіть за умов зміни виразу, кута огляду або освітлення.

Анотування зображень може бути виконано вручну або автоматизовано за допомогою спеціалізованих інструментів і платформ, таких як Labelbox, VGG Image Annotator, CVAT (Computer Vision Annotation Tool), LabelImg та інші. Ручне анотування вимагає залучення кваліфікованих фахівців і може бути дуже трудомістким, особливо для великих наборів даних. Водночас автоматизовані інструменти можуть використовувати напівконтрольовані або повністю автоматизовані методи для анотування великих обсягів зображень. При автоматизації процесу часто використовуються попередньо навчені моделі або методи активного навчання, що допомагає прискорити процес і зменшити витрати на анотування.

Якість анотацій є надзвичайно важливою для ефективного навчання моделей комп'ютерного зору. Недбало виконані або неточні анотації можуть призвести до зниження продуктивності моделей і їхньої нездатності коректно розпізнавати об'єкти в реальних сценаріях. Тому в процесі анотування важливо враховувати різні методи перевірки якості, такі як подвійне маркування, рев'ю експертами, використання метрик якості анотацій і проведення регулярних тестів для покращення точності.

При виконанні роботи для розмічення даних було використано платформу Roboflow. Приклад розмічених та нерозмічених даних зображено на рисунку 3.7.

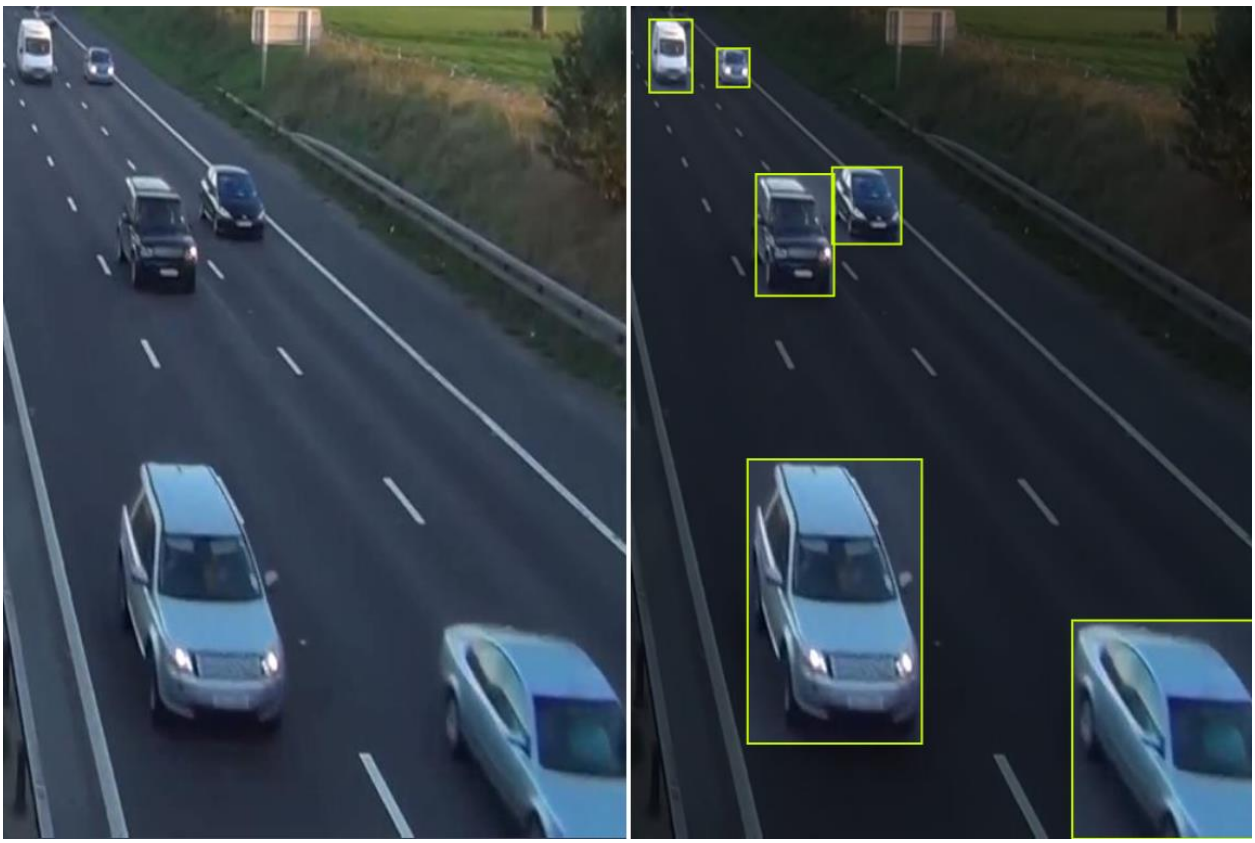

Рисунок 3.7 – Дані перед (зліва) та після (справа) створення анотацій

Також важливо забезпечити аугментацію даних, яка допоможе розширити навчальний набір і підвищити стійкість моделі до різних варіацій. Методи аугментації можуть включати повороти, зміну масштабу, горизонтальні і вертикальні відображення, зміну яскравості, контрасту, додавання шуму і розмиття. Це дозволяє моделі бути більш адаптивною до нових даних, які вона може зустріти у реальних застосуваннях.

Для побудови ефективної моделі виявлення автомобілів необхідно також мати розділені набори даних на навчальні, валідаційні та тестові. Це забезпечує правильне налаштування параметрів моделі та її перевірку на незалежних даних, що важливо для оцінки її загальної ефективності.

Після підготовки навчального набору даних можна перейти до навчання моделі. Першим кроком є ініціалізація ваг та підготовка до навчання. На цьому етапі модель ініціалізується випадковими вагами або вагами, попередньо навченими на іншому великому наборі даних (наприклад, ImageNet). Попереднє навчання дозволяє використовувати попередньо

вивчені ознаки, що значно прискорює навчання та покращує збіжність моделі.

Процес навчання складається з багаторазового проходження (ітерацій) навчального набору даних для оновлення ваг моделі. Навчання YOLO відбувається за допомогою алгоритму зворотного поширення помилки (backpropagation) і градієнтного спуску для мінімізації функції втрат.

Функція втрат у YOLO складається з кількох компонентів:

- втрата координат: використовується для покарання різниці між передбаченими межами (bounding boxes) та фактичними координатами меж об'єктів.
- втрата впевненості (confidence loss): використовується для покарання різниці між передбаченою ймовірністю наявності об'єкта в межах рамки і фактичною наявністю.
- втрата класу (classification loss): використовується для покарання різниці між передбаченим і фактичним класом об'єкта в межах рамки.

На кожній ітерації модель обчислює прогнози на основі вхідного зображення, порівнює їх з фактичними анотаціями і обчислює втрати. Потім вона оновлює свої ваги, щоб мінімізувати втрати за допомогою оптимізатора, такого як SGD (Stochastic Gradient Descent) або Adam.

Після кожної епохи або певної кількості ітерацій проводиться оцінка на валідаційному наборі даних. Це допомагає перевірити, як модель узагальнює дані, відмінні від тих, на яких вона навчалася. Метою є уникнути перенавчання (overfitting), коли модель добре працює на навчальних даних, але погано на нових даних. Для цього використовуються такі метрики, як точність (precision), повнота (recall), F1-міра, і mean Average Precision (mAP). Якщо результати на валідаційному наборі погіршуються, можливо, потрібно налаштувати гіперпараметри або використовувати методи регуляризації.

Після завершення навчання модель тестується на незалежному тестовому наборі даних. Це дає можливість оцінити остаточну

продуктивність моделі та її здатність до узагальнення. На цьому етапі можуть бути виміряні такі метрики, як mAP (mean Average Precision), Precision, Recall, IoU (Intersection over Union) для різних класів об'єктів.

Після завершення навчання та оцінки модель може бути збережена у вигляді файлу ваг та конфігурації, що дозволяє подальше її використання.

### 3.4 Проектування модулю відслідковування автомобілів

Відслідковування об'єктів у реальному часі – це завдання, яке передбачає постійне спостереження за об'єктами у відеопотоці, що надходить із камер. Головна мета полягає в тому, щоб визначити положення об'єкта та його переміщення в кожному кадрі відео. Це завдання є складним через необхідність одночасної обробки великого обсягу даних у реальному часі, що вимагає високої швидкості та точності алгоритмів.

Щоб реалізувати відслідковування, спочатку об'єкт потрібно розпізнати. Для задачі розпізнавання можуть бути використані сенсорні модулі, розроблені у попередніх пунктах. Після того, як об'єкт розпізнано, система повинна зберігати його положення й прогнозувати його наступне місцезнаходження. Це дозволяє "стежити" за об'єктом навіть тоді, коли він тимчасово зникає з поля зору камери, наприклад, через перешкоди або часткове перекриття іншими об'єктами.

Однією з головних труднощів є обробка динамічних змін у середовищі. Наприклад, об'єкти можуть змінювати форму, розміри або навіть колір через зміну освітлення або кутів огляду. Алгоритми відслідковування повинні бути стійкими до таких змін і адаптуватися до них, щоб уникнути втрати об'єкта або неправильного його відслідковування.

Важливу роль відіграє також швидкість. Для досягнення відстеження в реальному часі необхідно, щоб алгоритми могли обробляти відеопотік з частотою, що відповідає кількості кадрів, яку може надати камера. Зазвичай це 30 кадрів за секунду або більше, тому ефективність алгоритмів має велике значення.

При створені модуля відслідковування було використано алгоритм, оснований на вимірюванні дистанції між центрами меж об'єктів між сусідніми кадрами – адже два найближчих центра меж об'єктів на сусідніх кадрах з високою вірогідністю належать одному і тому ж об'єкту. Для вимірювання дистанції між центрами використовується формула Евклідової відстані (3.2).

$$\sqrt{\sum_{i=1}^{n}(p_i - q_i)^2}, \tag{3.2}$$

де $n$ – вимірність простору, у якому знаходяться точки;

$p_i$ – поточна координата першої точки;

$q_i$ – поточна координата другої точки;

Центри об'єктів знаходяться за формулами (3.3)

$$cx = x + (w / 2) \tag{3.3}$$
$$cy = y + (h / 2)$$

де x, y – координати верхнього лівого кута рамки навколо об'єкту;

h – висота рамки;

w – ширина рамки.

Координати рамок навколо об'єктів, їх ширина та висота отримуються за допомогою сенсорних модулів, розглянутих у попередніх пунктах.

Після вимірювання дистанції між всіма центрами, центрам, що мають найменшу відносну дистанцію між собою, призначається однаковий унікальний ідентифікатор.

Відслідковування об'єктів необхідне для того, щоб мати можливість визначити, скільки об'єктів, у випадку поточного застосування – автомобілів, перетнули певну лінію, інформація про що є необхідною для роботи розробленого алгоритму активного керування світлофором. Визначити, чи перетнув певний автомобіль визначену лінію можна за допомогою

порівняння знаків відповіді формули (3.3) для попередньої та поточної позиції центру об'єкту – у разі, якщо знак відповіді формули відрізняється для попередньої та поточної позицій, точки центрів знаходяться по різні сторони лінії, отже – об'єкт перетнув лінію.

$$(x_1 - x_2) * (y_p - y_1) - (y_1 - y_2) * (x_p - x_1), \quad (3.4)$$

де $x_1$, $y_1$ – координати першої точки лінії;

$x_2$, $y_2$ – координати другої точки лінії;

$x_p$, $y_p$ – координати центру рамки.

При фіксації перетину об'єктом лінії лічильник перетинів збільшується на 1.

### 3.5 Проектування модулю керування світлофором

Логіка модулю активного керування світлофором залежить від поточної фази світлофору.

На початку нової фази, якщо наступна фаза буде для одного з напрямків зеленою, проводиться обхід смуг, для яких сигнал світлофору стане зеленим, від сенсорного модуля системи отримується кількість автомобілів, в що на даний момент знаходяться на цих смугах, для кожної лінії знаходиться добуток кількості авто на смузі на середній час перетину світлофору одним авто для цієї смуги, тривалість зеленої фази визначається найбільшим отриманим значенням, або мінімальним у разі якщо воно менше за мінімальну дозволену тривалість зеленого сигналу, або максимальним якщо воно більше за максимальну дозволену тривалість зеленого сигналу. Прогнозовані значення тривалості перетину світлофору всіма автомобілями на лінії зберігаються для подальшого оновлення середнього часу перетину.

Під час зеленої фази за допомогою розробленого модуля відслідковування автомобілів підраховується кількість перетинів автомобілями світлофору. Якщо кількість зафіксованих перетинів для смуги

збігається з кількістю автомобілів, що знаходилися на смузі на початок зеленого сигналу, проводиться оновлення середнього часу перетину світлофору однім авто для цієї смуги шляхом порівняння реального часу перетину з прогнозованим значенням часу перетину. У разі, якщо час перетину був меншим або більшим за прогнозований, середній час перетину оновлюється згідно формули (3.5)

$$t_{new} = t_{avg} * (1 – ((1 – t_{real} / n_{veh}) * w)), \quad (3.5)$$

де $t_{new}$ – оновлений середній час перетину для одного авто;

$t_{avg}$ – поточний середній час перетину для одного авто;

$t_{real}$ – реальний час перетину для всіх авто;

$n_{veh}$ – кількість авто, що перетнули дорогу;

$w$ – коефіцієнт впливу нового часу.

У формулі використовується коефіцієнт впливу для того, щоб аномальні події, наприклад водій, що не помітив вчасно зміну сигналу світлофору, не могли занадто сильно вплинути на прогнозований час.

При закінченні зеленої фази, якщо кількість авто, що покинули смугу, не досягла кількості авто, що стояли на смузі на початку фази та прогнозований час перетину для смуги не перевищував максимальний, прогнозований середній час перетину світлофору для смуги оновлюється згідно формули (3.6)

$$t_{new} = t_{avg} * (1 – ((1 – t_{ph} / (n_{begin} - n_{end})) * w)), \quad (3.6)$$

де $t_{new}$ – оновлений середній час перетину для одного авто;

$t_{avg}$ – поточний середній час перетину для одного авто;

$t_{ph}$ – тривалість поточної зеленої фази;

$n_{begin}$ – кількість авто, що стояли на смузі на початку фази;

$n_{end}$ – кількість авто, що залишилася на дорозі;

$w$ – коефіцієнт впливу нового часу.

# 4 ДОСЛІДЖЕННЯ РЕЗУЛЬТАТІВ ТА АНАЛІЗ ЕКОНОМІЧНОЇ ЕФЕКТИВНОСТІ

## 4.1 Розробка методик проведення дослідження

### 4.1.1 Методики дослідження ефективності сенсорних модулів

Дослідження ефективності модулів детекції автомобілів передбачає комплексний підхід, який включає аналіз різних аспектів роботи алгоритмів. Основними критеріями є точність виявлення, швидкість обробки, стабільність в різних умовах і адаптивність до змін середовища.

Для оцінки точності детекції використовуються метрики, які допомагають оцінити, наскільки правильно модуль виявляє автомобілі. Ключовими показниками є правильні спрацювання (True Positives), неправильні спрацювання (False Positives) і пропущені автомобілі (False Negatives). З них можна вивести метрики Precision і Recall, які дозволяють порівнювати здатність модулів виявляти автомобілі без помилок і не пропускати об'єкти. Точність можна досліджувати на різних наборах даних, включаючи складні умови руху, різне освітлення або погодні фактори.

Швидкість обробки є ще одним важливим показником, особливо для систем, що працюють у реальному часі. Для порівняння використовується показник FPS (кількість кадрів за секунду), що показує, наскільки швидко алгоритм обробляє відео.

Для порівняння двох підходів мають використовуються різні набори даних, що включають складні сценарії, де можна вивчити ефективність роботи алгоритмів у змінних умовах.

### 4.1.2 Методики дослідження ефективності модулю керування перехрестям

Порівняти ефективність розробленого модулю керування перехрестям зі стандартним світлофором з попередньо заданими фазами можливо

використовуючи симулятор дорожнього руху або виконуючи тестування в умовах реального руху. Оскільки тестування у реальних умовах не є доступним, єдиним можливим методом дослідження залишається використання симулятора.

У розглянутих дослідженнях для тестування систем управління дорожнім рухом найчастіше використовувався Simulator of Urban Mobility, або SUMO [25], тому було прийняте рішення про його використання для тестування розробленого модулю.

SUMO (Simulation of Urban MObility) – це програмне забезпечення з відкритим кодом, яке призначене для моделювання міської мобільності та транспортних систем. Воно дає змогу досліджувати й аналізувати рух транспортних потоків у масштабах міста або окремих ділянок. Однією з ключових особливостей SUMO є його здатність відображати поведінку окремих транспортних засобів та пішоходів у реальному часі, що дозволяє моделювати складні сценарії, включаючи взаємодію різних учасників руху.

SUMO використовується для аналізу таких аспектів, як оптимізація транспортних мереж, ефективність світлофорного регулювання, вплив дорожніх обмежень, аварії та навіть екологічні фактори, такі як викиди шкідливих речовин. Оскільки це програмне забезпечення з відкритим кодом, воно широко застосовується науковцями, інженерами та урядовими установами для моделювання транспортної інфраструктури, планування нових доріг або адаптації існуючих, а також для розробки інтелектуальних транспортних систем.

SUMO підтримує детальне моделювання міських середовищ, враховуючи дорожню інфраструктуру, динаміку руху транспортних засобів, вплив світлофорів та інших регулюючих пристроїв. Користувачі можуть задавати параметри руху, такі як швидкість, маршрути, інтенсивність потоку, і на основі цього отримувати різні результати – від часу подорожі для окремого автомобіля до тривалості зупинок або споживання палива. SUMO

також легко інтегрується з іншими інструментами та програмами, що робить його ефективним інструментом для моделювання.

### 4.2 Порівняння ефективності сенсорних модулів

Для оцінки ефективності розроблених сенсорних модулів було використано три відео виду на дорогу з різними умовами освітлення – вид вдень, ввечері та вночі.

При оптимальних погодних умовах (гарне освітлення, відсутність осадів) та низькій щільності руху обидва розроблені модулі забезпечують високу (>90%) точність детекції. При підвищені щільності руху модель на основі віднімання фону та у разі неоптимального розташування камери відносно дороги, при якому автомобілі можуть наскладуватись один на одного, алгоритм на основі глибинного навчання з більш високою вірогідністю розрізняє декілька авто що накладаються, тоді як алгоритм на основі віднімання фону, навіть при оптимальному використанню морфологічних перетворень для розділення "плям" автомобілів, може розпізнавати декілька авто що накладаються як одне, тобто точність детекції знижується. Приклад такої відмінності в поведінці моделей при використанні одного і того ж джерела зображено на рисунку 4.1.

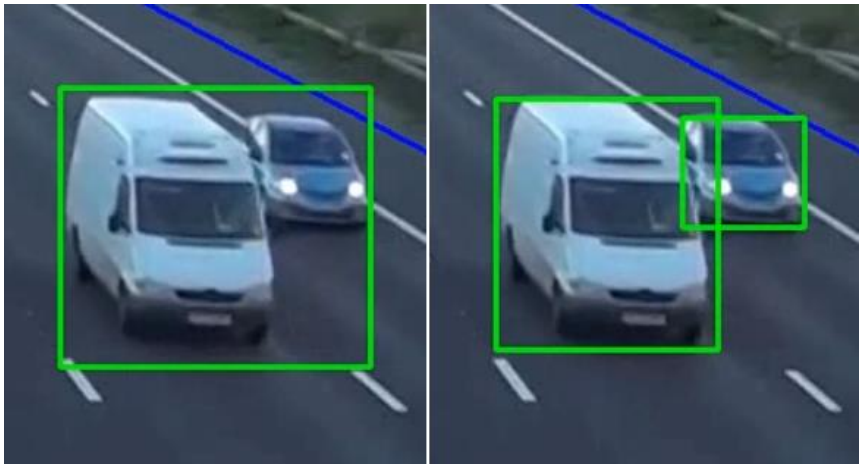

Рисунок 4.1 – Демонстрація різниці у точності детекції авто, що накладаються, між модулем на основі віднімання фону (зліва) та модулем на основі глибинного навчання (справа)

Чутливість до змін середовища є критичним фактором у дослідженні модулів детекції автомобілів. Під час проведення дослідження було виявлено що методи віднімання фону є більш чутливими до змін у зовнішніх умовах, таких як освітлення, погода або рух інших об'єктів, наприклад пішоходів або тіней. Такі зміни можуть викликати помилкові спрацювання або пропуски автомобілів. Особливо сильний негативний вплив на точність розпізнавання моделлю на основі віднімання фону спричиняє погане освітлення дороги, оскільки це знижує контрастність авто відносно дороги та додає світлові плями від фар, що розпізнаються моделлю як авто, підвищуючи хибні спрацювання, як зображено на рисунку 4.2.

Рисунок 4.2 – Демонстрація вразливості алгоритму на основі віднімання фону до умов поганого освітлення

У той час моделі на основі глибинного навчання краще справляються зі складними умовами сцени, оскільки їхня архітектура дозволяє моделювати залежності між пікселями і формами об'єктів, на що менше впливає наявність шумів у вигляді осадів або зміни освітлення. Також віднімання фону є менш гнучким і може викликати помилки, якщо сцена змінюється динамічно (наприклад, у випадку рухомої камери або різких змін у фоні).

Також було підраховано швидкість обробки відео алгоритмами для визначення їх відносної розрахункової ефективності. Вимірювання часу

проводилося використовуючи процессор Ryzen 5600G, розмір оброблюваного відео становив 1150 на 880 пікселів. Швидкість обробки відео модулем що використовує алгоритм на основі віднімання фону становила 46.04 кадри в секунду, швидкість обробки модулем на основі глибинного навчання становила 5.33 кадри в секунду, таким чином алгоритм на основі віднімання фону є в 8.63 рази швидшим.

Підсумовуючи результати дослідження, метод віднімання фону має перевагу у швидкості та простоті, але поступається у точності та адаптивності до складних умов. Модуль на основі об'єктної детекції є потужнішими в складних сценаріях, але вимагає більших обчислювальних ресурсів для роботи.

### 4.3 Порівняння розробленої моделі керування світлофором зі статичною моделлю

Для порівняння розробленої моделі керування світлофором зі статичною моделлю у симуляторі дорожнього руху SUMO було створено дві мапи перехресть, дороги на першій мапі мають дві смуги, по одній в кожному напрямку, дороги на другій мапі мають шість смуг, по три в кожному напрямку.

Перехрестя контролюються статичним світлофором. Статичний світлофор в SUMO задається за допомогою так званих "світлофорних планів" (traffic light programs), де визначаються часові цикли для кожної фази світлофора (зеленого, жовтого, червоного сигналів) на різних напрямках руху. Світлофор працює за фіксованим часом для кожної фази (наприклад, 30 секунд для зеленого сигналу, 5 секунд для жовтого і 25 секунд для червоного). Опис стану світлофора для фази для кожної смуги задається у вигляді набору літер де "G" – зелений, "r" – червоний, "y" – жовтий.

На рисунку 4.3 зображено мапу перехрестя доріг з двома смугами а також програму фаз статичного світлофору.

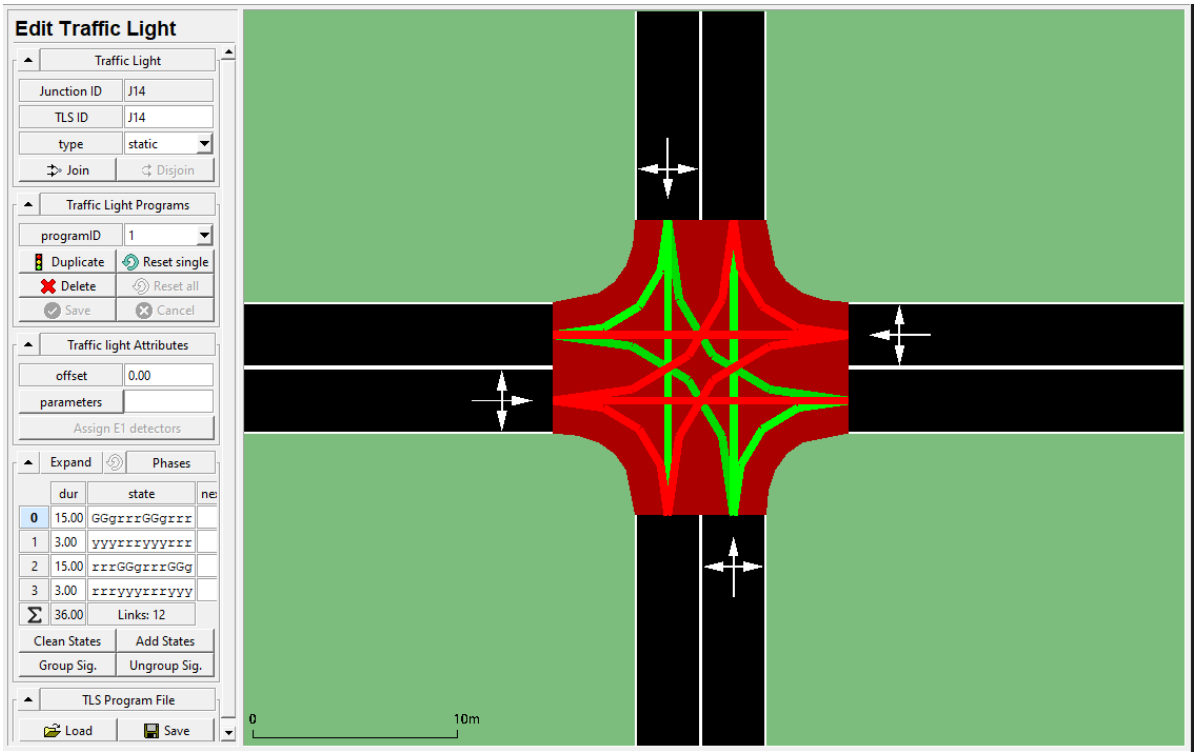


Рисунок 4.3 – Мапа перехрестя доріг з двома смугами

На рисунку 4.3 зображено мапу перехрестя доріг з шістьма смугами а також програму фаз статичного світлофору.

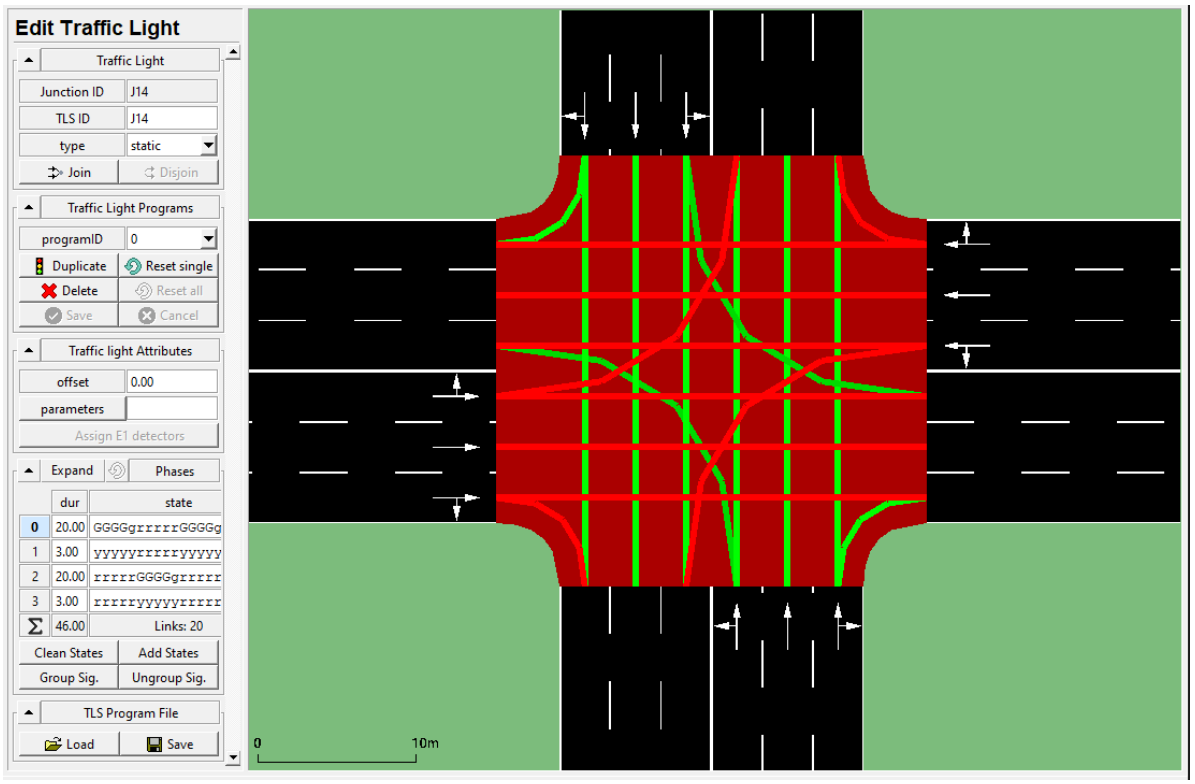


Рисунок 4.4 – Мапа перехрестя доріг з шістьма смугами

Обидва світлофори мають чотири фази – зелена фаза для напрямку північ-південь, перша жовта фаза, зелена фаза для напрямку захід-схід, друга жовта фаза.

В рамках дослідження було проведено заміри середнього часу очікування для статичного світлофору і для розробленого модулю управління світлофором на чотирьох сценаріях навантаження дороги – рівномірно розподіленого по часу та напрямках, рівномірно розподіленого по часу з різницею навантаження між напрямком північ-південь та захід-схід у 1.5 разів, розподіленого по часу відносно функції нормального розподілу з піком в середині часового відрізку та рівномірно розподіленого по напрямкам, а також розподіленого по часу відносно функції нормального розподілу з піком в середині часового відрізку та різницею навантаження між напрямком північ-південь та захід-схід у 1.5 разів. Графік часу відправлення автомобілів згідно нормального розподілу наведено на рисунку 4.5.

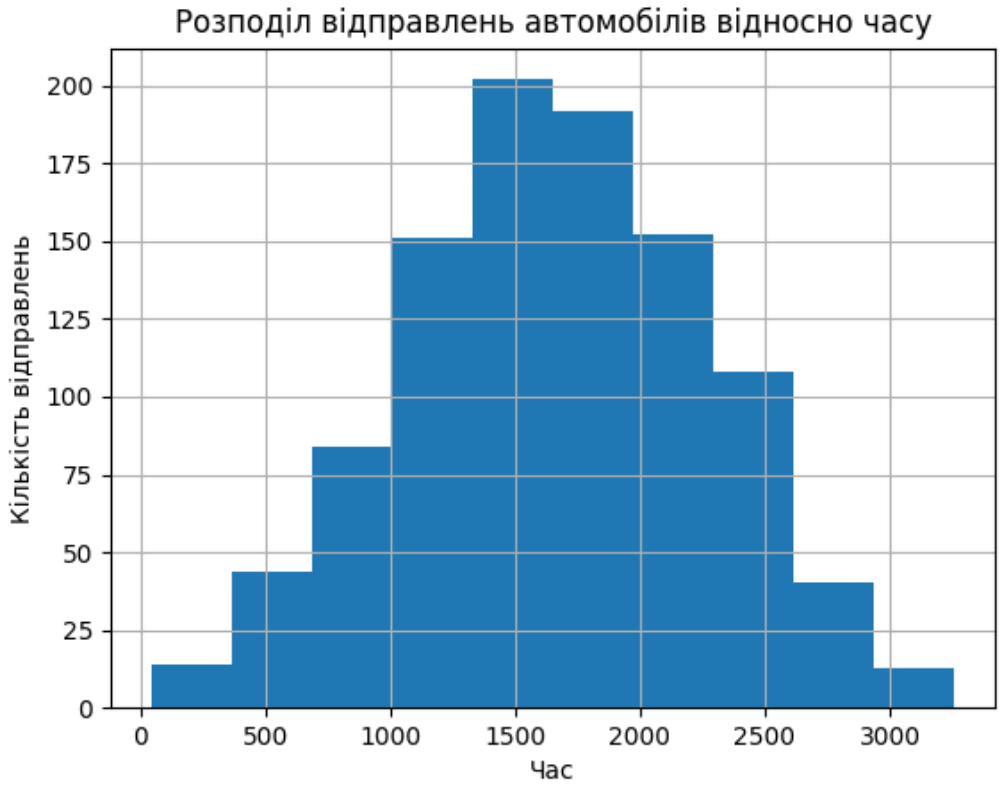


Рисунок 4.5 – Графік кількості відправлень авто згідно нормального розподілу

Тривалість симуляції у кожному сценарію дорівнювала 3600 секундам. Кількість відправлень авто та крок її підвищення для заміру середнього часу очікування при різному навантаженні дороги залежало від експерименту. Для кожного експерименту також попередньо визначалася оптимальна тривалість сигналу статичного світлофору для порівняння розробленої моделі з найкращим випадком статичного світлофору.

Далі наведено графіки залежності середнього часу очікування від кількості відправлень автомобілів на годину.

На рисунку 4.6 наведено графік залежності середнього часу очікування від кількості відправлень автомобілів на годину для сценарію 1 – мапи з дорогами з двома смугами зі рівномірним навантаженням відносно часу та рівномірним навантаженням відносно напрямків руху. Тривалість зелених фаз статичного світлофору підвищувалася з 11 до 15 секунд з кроком в 1 секунду при кожному підвищені загальної кількості відправлень на 200.

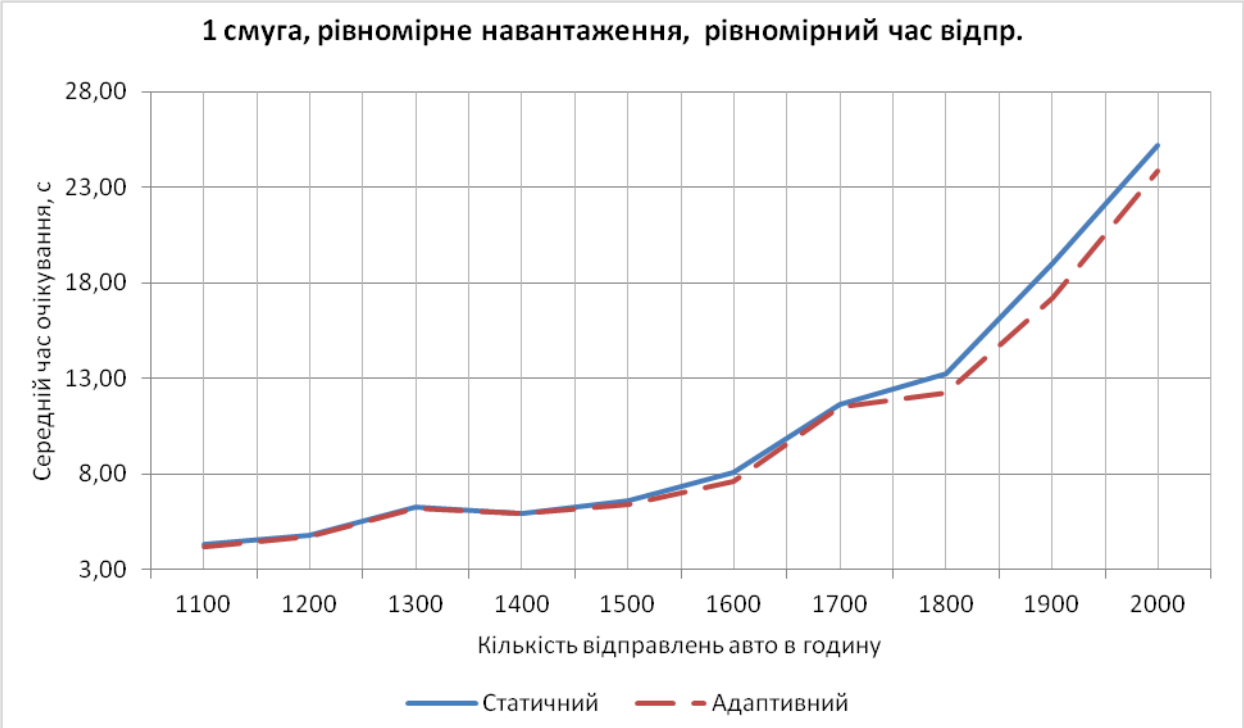


Рисунок 4.6 – Графік залежності середнього часу очікування від кількості відправлень авто в годину для сценарію 1

За результатами експерименту було визначено що для сценарію 1 ефективність розробленого алгоритму управління світлофором є вищою за стандартний світлофор зі статичними фазами в середньому на 3.9% з підвищенням ефективності, як в абсолютних так і в відносних значеннях, при збільшені загальної кількості відправлень.

На рисунку 4.7 наведено графік залежності середнього часу очікування від кількості відправлень автомобілів на годину для сценарію 2 – мапи з дорогами з двома смугами зі рівномірним навантаженням відносно часу та відношенням навантаження на напрямку південь-північ відносно напрямку захід-схід 1 до 1.5. Тривалість зелених фаз статичного світлофору підвищувалася з 11 до 15 секунд з кроком в 1 секунду при кожному підвищені загальної кількості відправлень на 100.

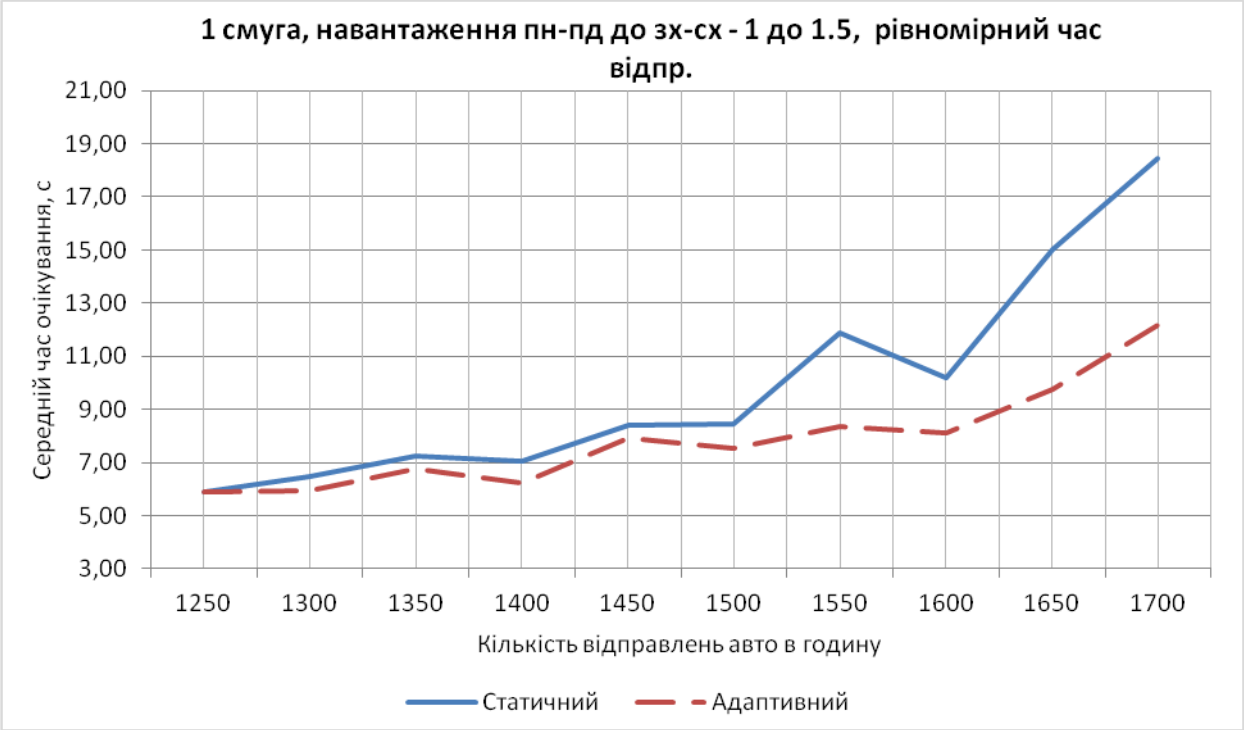


Рисунок 4.7 – Графік залежності середнього часу очікування від кількості відправлень авто в годину для сценарію 2

За результатами експерименту було визначено що для сценарію 2 ефективність розробленого алгоритму управління світлофором є вищою за

стандартний світлофор зі статичними фазами в середньому на 22.1% з підвищенням ефективності, як в абсолютних так і в відносних значеннях, при збільшені загальної кількості відправлень.

На рисунку 4.8 наведено графік залежності середнього часу очікування від кількості відправлень автомобілів на годину для сценарію 3 – мапи з дорогами з двома смугами зі навантаженням відносно часу згідно нормального розподілу та рівномірним навантаженням відносно напрямків руху. Тривалість зелених фаз статичного світлофору підвищувалася з 11 до 15 секунд з кроком в 1 секунду при кожному підвищені загальної кількості відправлень на 100.

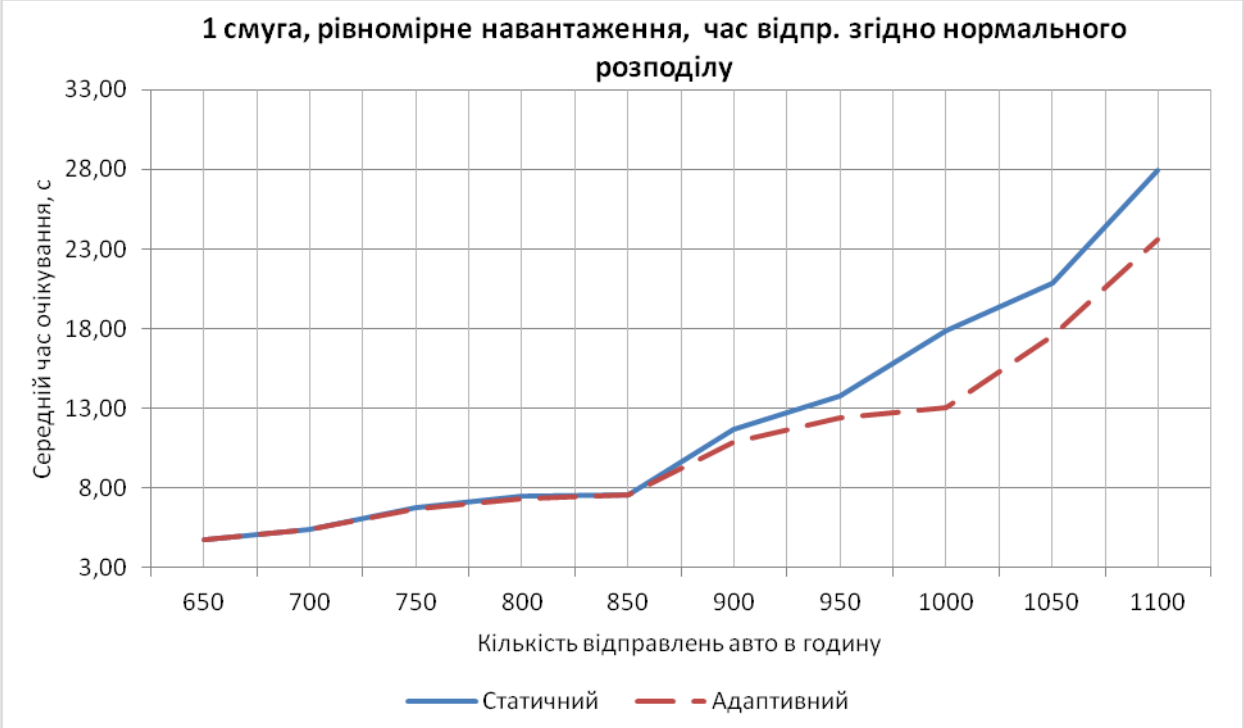


Рисунок 4.8 – Графік залежності середнього часу очікування від кількості відправлень авто в годину для сценарію 3

За результатами експерименту було визначено що для сценарію 3 ефективність розробленого алгоритму управління світлофором є вищою за стандартний світлофор зі статичними фазами в середньому на 9.5% з

підвищенням ефективності, як в абсолютних так і в відносних значеннях, при збільшені загальної кількості відправлень.

На рисунку 4.9 наведено графік залежності середнього часу очікування від кількості відправлень автомобілів на годину для сценарію 4 – мапи з дорогами з двома смугами зі навантаженням відносно часу згідно нормального розподілу та відношенням навантаження на напрямку південь-північ відносно напрямку захід-схід 1 до 1.5. Тривалість зелених фаз статичного світлофору підвищувалася з 11 до 15 секунд з кроком в 1 секунду при кожному підвищені загальної кількості відправлень на 100.

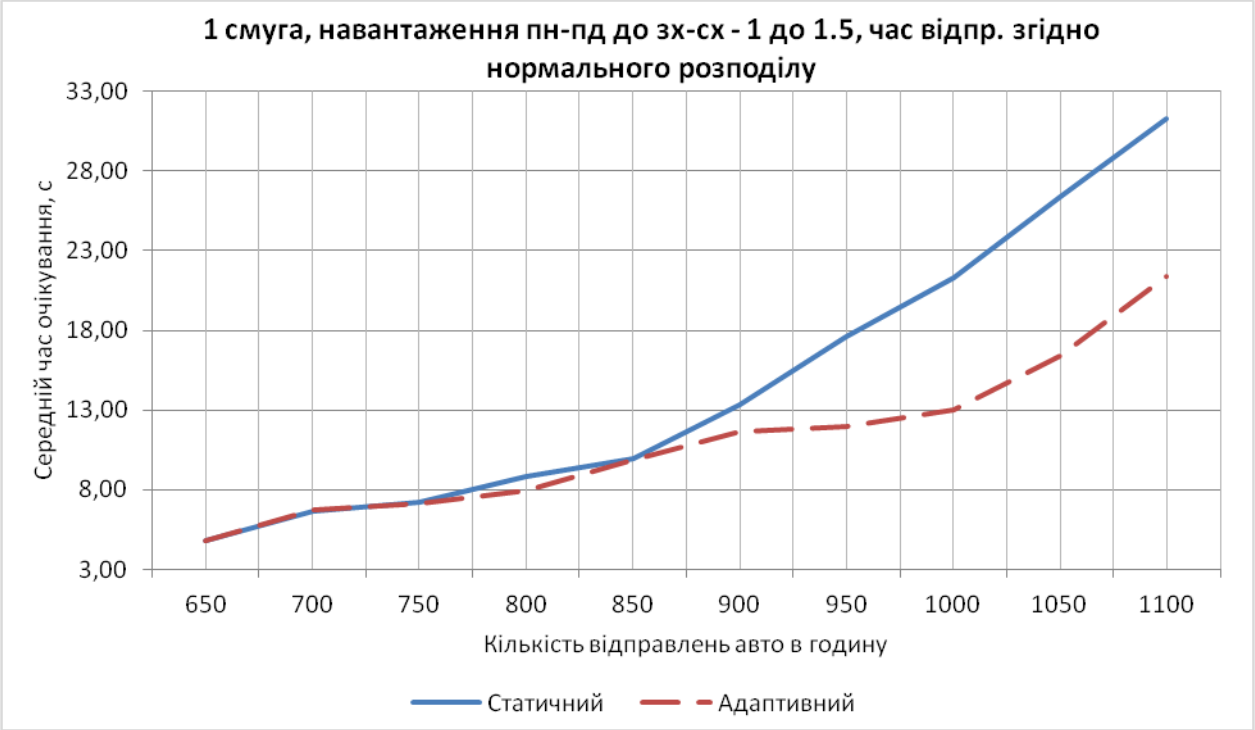


Рисунок 4.9 – Графік залежності середнього часу очікування від кількості відправлень авто в годину для сценарію 4

За результатами експерименту було визначено що для сценарію 4 ефективність розробленого алгоритму управління світлофором є вищою за стандартний світлофор зі статичними фазами в середньому на 24.5% з підвищенням ефективності, як в абсолютних так і в відносних значеннях, при збільшені загальної кількості відправлень.

На рисунку 4.10 наведено графік залежності середнього часу очікування від кількості відправлень автомобілів на годину для сценарію 5 – мапи з дорогами з шістьма смугами зі рівномірним навантаженням відносно часу та рівномірним навантаженням відносно напрямків руху. Тривалість зелених фаз статичного світлофору підвищувалася з 16 до 20 секунд з кроком в 1 секунду при кожному підвищені загальної кількості відправлень на 500.

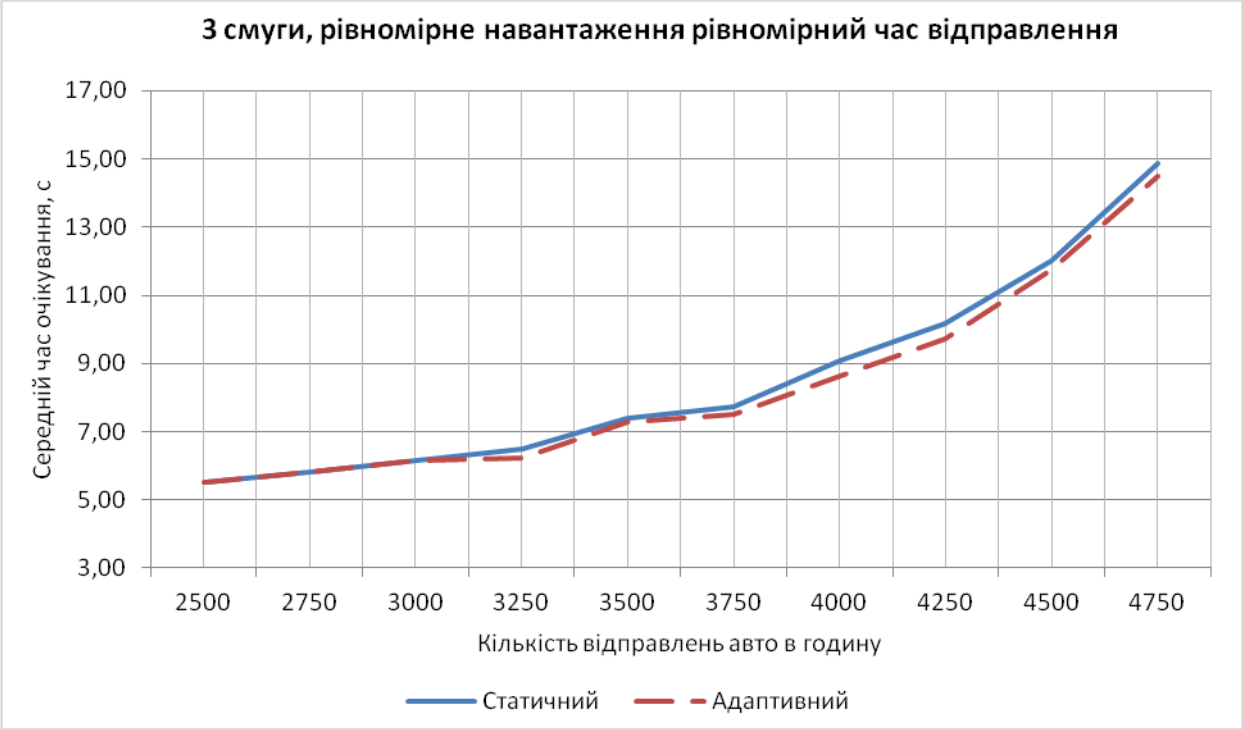


Рисунок 4.10 – Графік залежності середнього часу очікування від кількості відправлень авто в годину для сценарію 5

За результатами експерименту було визначено що для сценарію 1 ефективність розробленого алгоритму управління світлофором є вищою за стандартний світлофор зі статичними фазами в середньому на 2.3% з підвищенням ефективності, як в абсолютних так і в відносних значеннях, при збільшені загальної кількості відправлень.

На рисунку 4.11 наведено графік залежності середнього часу очікування від кількості відправлень автомобілів на годину для сценарію 6 – мапи з дорогами з шістьма смугами з рівномірним навантаженням відносно

часу та відношенням навантаження на напрямку південь-північ відносно напрямку захід-схід 1 до 1.5. Тривалість зелених фаз статичного світлофору підвищувалася з 16 до 20 секунд з кроком в 1 секунду при кожному підвищені загальної кількості відправлень на 500.

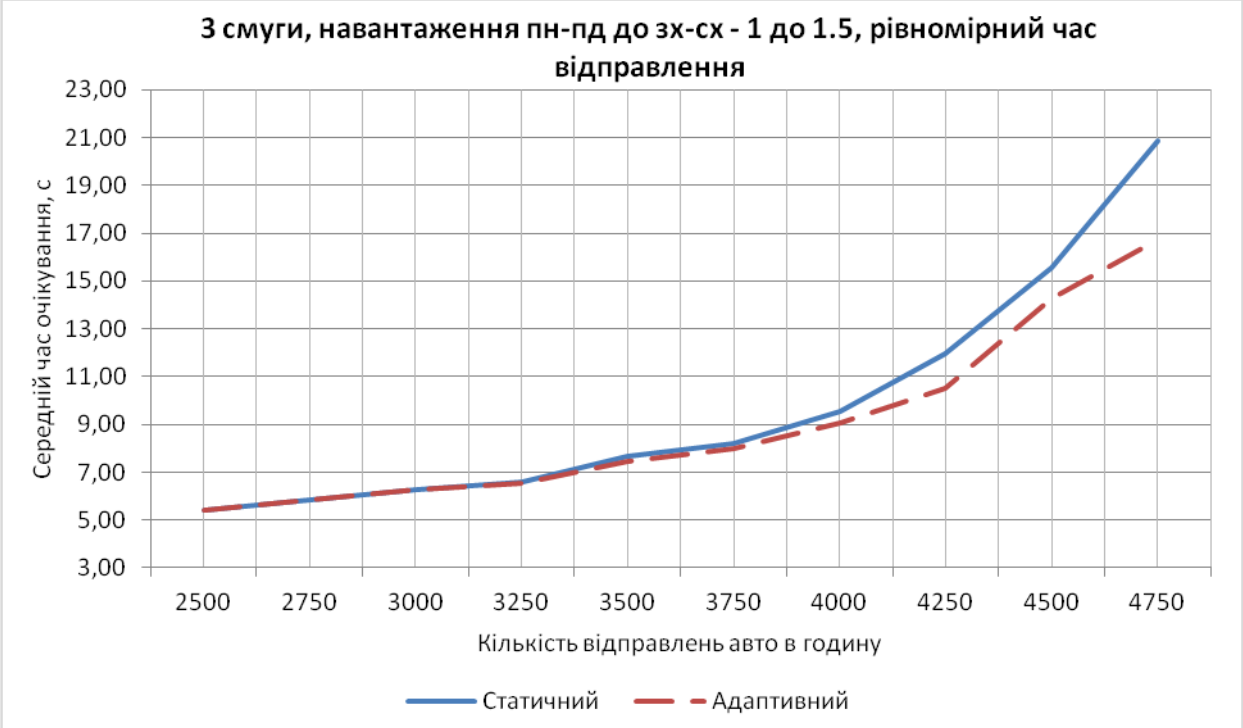


Рисунок 4.11 – Графік залежності середнього часу очікування від кількості відправлень авто в годину для сценарію 6

За результатами експерименту було визначено що для сценарію 6 ефективність розробленого алгоритму управління світлофором є вищою за стандартний світлофор зі статичними фазами в середньому на 5.9% з підвищенням ефективності, як в абсолютних так і в відносних значеннях, при збільшені загальної кількості відправлень.

На рисунку 4.12 наведено графік залежності середнього часу очікування від кількості відправлень автомобілів на годину для сценарію 7 – мапи з дорогами з шістьма смугами зі навантаженням відносно часу згідно нормального розподілу та рівномірним навантаженням відносно напрямків руху. Тривалість зелених фаз статичного світлофору підвищувалася з 10 до

20 секунд з кроком в 1 секунду при кожному підвищені загальної кількості відправлень на 200.

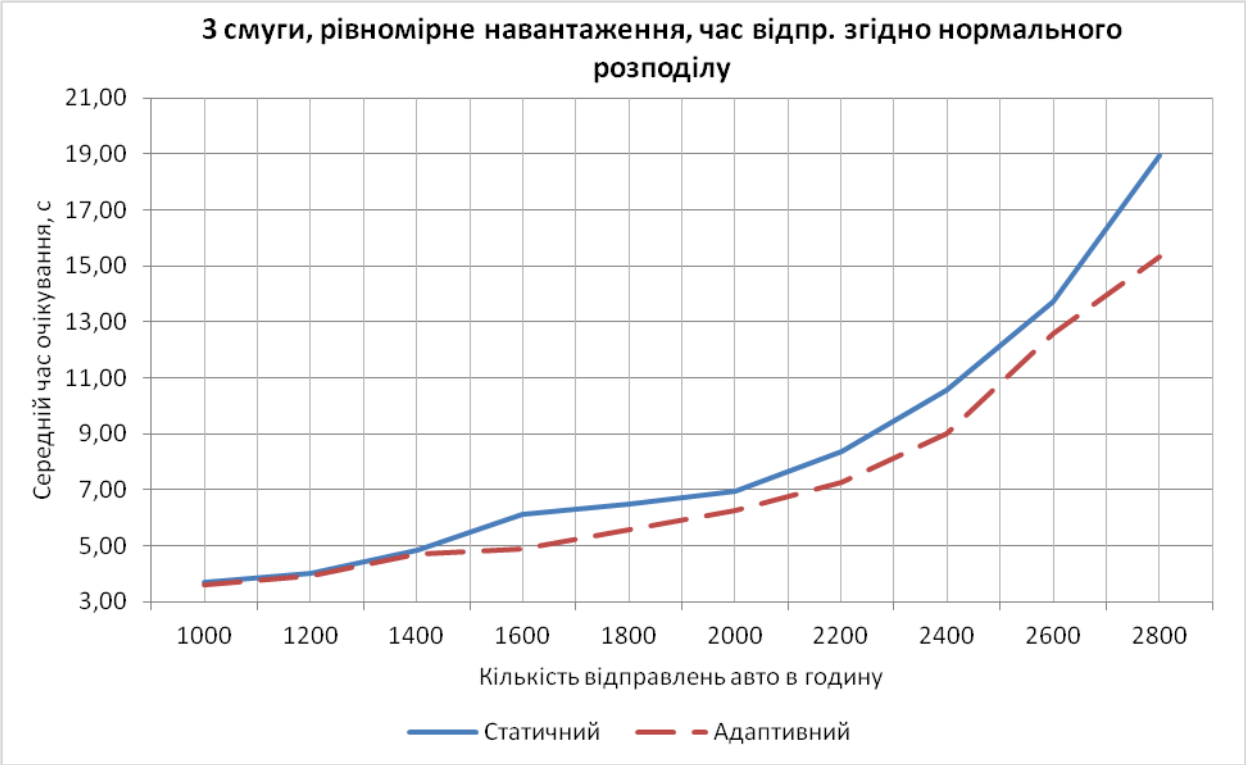


Рисунок 4.12 – Графік залежності середнього часу очікування від кількості відправлень авто в годину для сценарію 7

За результатами експерименту було визначено що для сценарію 7 ефективність розробленого алгоритму управління світлофором є вищою за стандартний світлофор зі статичними фазами в середньому на 12.6% з підвищенням ефективності, як в абсолютних так і в відносних значеннях, при збільшені загальної кількості відправлень.

На рисунку 4.13 наведено графік залежності середнього часу очікування від кількості відправлень автомобілів на годину для сценарію 4 – мапи з дорогами з шістьма смугами зі навантаженням відносно часу згідно нормального розподілу та відношенням навантаження на напрямку південь-північ відносно напрямку захід-схід 1 до 1.5. Тривалість зелених фаз статичного світлофору підвищувалася з 10 до 20 секунд з кроком в 1 секунду при кожному підвищені загальної кількості відправлень на 200.

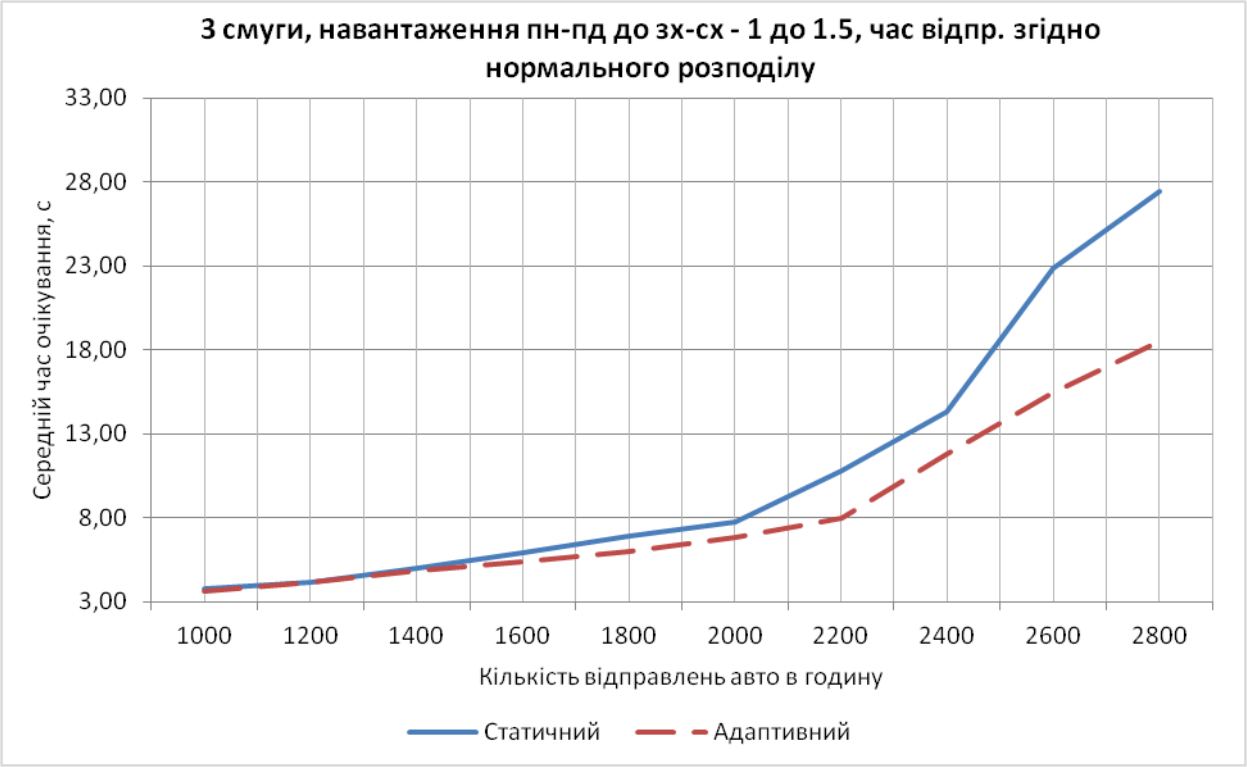


Рисунок 4.13 – Графік залежності середнього часу очікування від кількості відправлень авто в годину для сценарію 8

За результатами експерименту було визначено що для сценарію 8 ефективність розробленого алгоритму управління світлофором є вищою за стандартний світлофор зі статичними фазами в середньому на 20% з підвищенням ефективності, як в абсолютних так і в відносних значеннях, при збільшені загальної кількості відправлень.

Підсумовуючи результати дослідження, розроблений модуль керування світлофором є більш ефективним за алгоритм керування світлофором зі статичними фазами у всіх сценаріях роботи з підвищенням відносної ефективності при збільшені завантаженості дороги.

### 4.4 Аналіз економічної ефективності

Згідно із завданням визначимо собівартість розробленої системи інтелектуального управління перехрестям.

Для виконання розрахунку були використані початкові дані, представлені в таблиці 1.

Таблиця 1 – Початкові дані для визначення собівартості

| Найменування початкових даних | Показник | Джерело отримання |
|---|---|---|
| Трудомісткість складання програми, днів | 60 | Фактичні витрати часу на розробку програмного продукту (3 місяці по 20 робочих днів) |
| Місячна ставка укладача програми, грн | 16000,00 | Дані дипломного проекту |
| Кількість годин в місяці, год | 160 | Кількість робочих днів – 20 |
| Додаткова зарплата (%) | 12 | Дані дипломного проекту |
| Відрахування до соціальних фондів (%) | 15 | Дані дипломного проекту |
| Загальновиробничі витрати (%) | 100 | Дані дипломного проекту |
| ПДВ (податок на додану вартість) (%) | 20 | Дані дипломного проекту |

Робимо розрахунок на 1 місяць (20 днів):

Стаття 1. Комплектуючі вироби – 1 компакт-диск:

$$З_к = \sum Ц_к * n_к, \quad (1.1)$$

де $З_к$ – витрати на комплектуючі вироби, грн.;

$Ц_к$ – ціна за 1 одиницю комплектуючих виробів, грн.;

$n_к$ – кількість комплектуючих виробів по кожному типорозмірі, шт.

Визначимо витрати на комплектуючі вироби:

$$З_к = \sum 15 * 1 = 15{,}00 \text{ грн.} \quad (1.2)$$

Витрати на комплектуючі виробі склали 15,00 грн.

Стаття 2. Витрати на електроенергію розраховуємо за формулою 1.3:

$$B_E = P_E \sum Wi * t_{шт\,i}, \quad (1.3)$$

де $P_E$ – ціна за 1 кВт-год, грн.;

$W_i$ – середня потужність, що споживається.

Визначимо витрати на електроенергію за формулою 1.4:

$$B_E = 4.32 * \sum 0{,}400 * 160 = 246{,}48 \text{ грн} \quad (1.4)$$

Витрати на електроенергію дорівнюють 246,48 грн.

Стаття 3. Основна заробітна плата визначається за формулою 1.5:

$$З_{осн} = l_{год} * T_{год}, \quad (1.5)$$

де $l_{год}$ – годинна тарифна ставка оператора, грн.;

$T_{год}$ – кількість годин у місяці, приймається 160 год. – вихідні дані.

Визначаємо годинну тарифну ставку укладача ПЗ:

$$l_{год} = L_{міс} / n_t , \quad (1.5)$$

де *Lміс* – місячна ставка оператора, грн;

$n_t$ – кількість годин у місяці

Годинна тарифна ставка укладача ПЗ дорівнює 100 грн.

Визначимо основну заробітну плату за формулою 1.6:

$$З_{осн} = 100 * 160 = 16000{,}00 \text{ грн.} \quad (1.6)$$

Основна заробітна плата дорівнює 16000,00 грн.

Стаття 4. Додаткова заробітна плата визначається за формулою 1.7:

$$З_{дод} = \frac{З_{осн} * Д\%}{100}, \quad (1.7)$$

де *Здод* – додаткова заробітна плата, грн.;

*Д%* – відсоток додаткової заробітної плати, приймається 10% – вихідні дані.

Визначимо додаткову заробітну плату за формулою 1.8:

$$З_{дод} = \frac{16000*12}{100} = 1920{,}00 \text{ грн.} \quad (1.8)$$

Додаткова заробітна плата дорівнює 1920,00 грн.

Стаття 5. Відрахування в соціальні фонди знайдемо за формулою 1.9:

$$З_{соц} = \frac{(З_{осн}+З_{доп})*С\%}{100}, \quad (1.9)$$

де $З_{соц}$ –відрахування в соціальні фонди, грн.;

*С%* – відсоток відрахувань у соціальні фонди, приймається 44 – вихідні дані.

Визначимо відрахування в соціальні фонди за формулою 1.10:

$$З_{соц} = \frac{(16000+1920)*15}{100} = 2688{,}00 \text{ грн.} \quad (1.10)$$

Відрахування в соціальні фонди дорівнює 2688,00 грн;

Стаття 6. Загальновиробничі витрати визначаються за формулою 1.11:

$$З_{заг} = \frac{З_{осн}*Н_1\%}{100}, \quad (1.11)$$

де $З_{заг}$ – загальновиробничі витрати, грн.;

$Н_1\%$ – відсоток загальновиробничих витрат, приймається 100% – вихідні дані.

Визначимо загальновиробничі витрати за формулою 1.12:

$$З_{заг} = \frac{16000*100}{100} = 16000{,}00 \text{ грн.} \quad (1.12)$$

де $З_{заг}$ – загальновиробничі витрати, грн.

Загальновиробничі витрати дорівнюють 16000,00 грн.

Визначимо виробничу собівартість склавши всі попередні розрахунки (формула 1.13)

$$S_{nn} = 246{,}48 + 15{,}00 + 16000{,}00 + 1920{,}00 + 2688{,}00 + 16000{,}00 = 36869{,}48 \text{ грн.} \quad (1.13)$$

де *Snn* – виробнича собівартість, грн.

Виробнича собівартість дорівнює 36869,48 грн.

В таблиці 1.2 наведено планову калькуляцію виробничої собівартості виробничої собівартості, ціни підприємства й ціни для замовника на виконання розробки програми.

Таблиця 1.2 – Планова калькуляція

| Статті калькуляції | Сума, грн. |
|---|---|
| Стаття 1 Комплектуючи вироби | 15,00 |
| Стаття 2. Витрати на електроенергію: | 246,48 |
| Стаття 3 Основна заробітна плата | 16000,00 |
| Стаття 4 Додаткова заробітна плата | 2688,00 |
| Стаття 5 Відрахування в соціальні фонди | 1584,00 |
| Стаття 6 Загальновиробничі витрати | 16000,00 |
| Виробнича собівартість, 1 місяць | 36869,48 |
| Собівартість ПЗ | 110608,44 |

Робота по розробці ПЗ проводилась протягом 3 місяців, таким чином розрахунок показав, що собівартість розробленої системи складає 110608,44 грн.

Далі необхідно провести розрахунок інноваційного економічного ефекту від впровадження розробленої системи.

Для того, щоб визначити економічний ефект від впровадження більш ефективної системи управління перехрестями необхідно визначити ціну за одиницю часу простою автомобілю в заторі. У дослідженні, підготовленому для Європейського інвестиційного банку [26], авторами було визначено середню вартість часу, проведеного в дорозі. В дослідженні не міститься даних для автомобілів в Україні, але за результатами дослідження було визначено, що вартість часу в дорозі в автомобілі корелює з ВВП на душу населення за паритетом купівельної спроможності і підвищення ВВП на 100% відповідає збільшенню вартості часу в дорозі в автомобілі на 70%-85%. Найближчою до України країною, для якої в дослідженні є необхідні дані є Албанія, одже для знаходження приблизної вартості часу в дорозі для України буде використано дані цієї країни з використанням коефіцієнту. Згідно з даними Міжнародного банку [27], ВВП на духу населення за ПКС для України становить 18,007.50 доларів США, для Албанії – 21,395.30 доларів США. В дослідженні [26] середня вартість години в дорозі для Албанії була визначена в 6.88 Євро, таким чином приблизну вартість години в дорозі в Україні можна визначити за формулою 1.14.

$$P\backslash h_{ua} = P\backslash h_{al} * (1 - ((GDP_{al} / (GDP_{al} - GDP_{ua})) * 0{,}8)), \quad (1.14)$$

де $P\backslash h_{ua}$ – Вартість години в дорозі в Україні;

$P\backslash h_{al}$ – Вартість години в дорозі в Албанії;

$GDP_{al}$ – ВВП на душу населення Албанії;

$GDP_{ua}$ – ВВП на душу населення України.

$$P\backslash h_{ua} = 6{,}88 * (1 - (((21{,}395.30 - 18{,}007.50) / 21{,}395.30) * 0{,}8)) = 6{,}01$$

Вартість години в дорозі в Україні дорівнює 267,55 грн, вартість секунди – 0,074 грн. Найбільш наближені до реальних умов руху симуляції з проведених – симуляції з частотою відправлення авто згідно нормального розподілення. В таких симуляціях економія часу в порівнянні зі стандартною моделлю в середньому становила 1,27 секунд.

Розрахувати середню економію для заданої кількості авто в день можна за формулою 1.15.

$$P = N_v * P_s * t \quad (1.15)$$

де P – економія або прибуток в день;

$N_v$ – кількість авто в день;

$P_s$ – розрахована вартість однієї секунди часу;

t – розрахована середня економія часу для одного автомобіля.

Розрахуємо прибуток при встановлені системи на одному перехресті з навантаженням 20000 авто в день.

P = 20000 * 0,074 * 1,27 = 1480

Тепер розрахуємо термін окупності за формулою 1.16.

$$T = C / П \quad (1.16)$$

де T – термін окупності проекту, місяців,

C – собівартість проекту, грн.,

П – прибуток, який повертається за день, грн.

Термін окупності проекту становить 110608,44/1480=74,73 днів.

Затрачені кошти на розробку системи окупляться за 2.5 місяців, крім того при використанні системи на більшій кількості перехресть економічний ефект і швидкість окупності системи підвищується, отже можна говорити про економічну доцільність використання такої системи.

# ВИСНОВКИ

У даній роботі було проведено дослідження та проектування моделі інтелектуальної системи керування перехрестями, що використовує сучасні методи машинного навчання для оптимізації сигналів світлофору.

У першому розділі було обґрунтовано актуальність дослідження після ознайомлення з даними про причини та наслідки заторів у містах, а також їх вплив на економічну та екологічну ситуацію, якість життя учасників дорожнього руху та людей, що мешкають біля доріг, що стикаються з заторами. Було визначено предмет та об'єкт дослідження, встановлено його задачі та методи, що використовуються при його проведені.

У другому розділі було висвітлено інформацію про предмет дослідження, проведено аналіз існуючих досліджень з теми. У дослідженнях з теми використовувалися різні підходи та методи до вирішення проблеми неоптимальності керуючого сигналу світлофора – попередньо оптимізований контроль, адаптивне керування і напівактивне керування а також методи що опираються на комунікацію між транспортними засобами або транспортними засобами та інфраструктурою для уникнення заторів та оптимізацію швидкості руху та позиціонування у транспортному потоці для адаптації до поточної ситуації.

У третьому розділі було визначено функціональні та нефункціональні вимоги до системи, спроектовано сенсорний модуль, модуль відслідковування об'єктів та аналітичний модуль. Для сенсорного модулю було обрано аналіз даних з відеокамери за допомогою визначення різниці поточного кадру від адаптивного зображення порожньої дороги, альтернативний модуль аналізу даних з відеокамери використовує модель виявлення об'єктів. Аналітичний модуль розподіляє час зеленого сигналу світлофору відповідно до завантаженості керованого сегменту дороги.

У четвертому розділі було проведено порівняння ефективності сенсорних модулів та аналіз ефективності аналітичного модулю в порівнянні

з статичним алгоритмом керування світлофором. Визначено що сенсорний модуль що використовує метод віднімання фону має перевагу у швидкості та простоті, але поступається у точності та адаптивності до складних умов, тоді як модуль на основі об'єктної детекції є більш точним в складних сценаріях, але вимагає більших обчислювальних ресурсів для роботи. Після проведення симуляцій дорожнього руху з використанням світлофору зі статичними фазами та розробленим модулем адаптивного керування було визначено, що розроблений модуль керування світлофором є більш ефективним за алгоритм керування світлофором зі статичними фазами у всіх сценаріях роботи з підвищенням відносної ефективності при збільшені завантаженості дороги. Також у цьому розділі було проведено оцінку економічної доцільності впровадження розробленої системи, обґрунтовано, що розробка і впровадження інтелектуальної системи керування перехрестями є економічно доцільним.

# ПЕРЕЛІК ПОСИЛАНЬ


1. United Nations, Department of Economic and Social Affairs, Population Division (2019). World Urbanization Prospects: The 2018 Revision. [Електронний ресурс] – 2019. – Режим доступу до ресурсу: https://population.un.org/wup/Publications/Files/WUP2018-Report.pdf.
2. Goodwin P. The economic costs of road traffic congestion [Електронний ресурс] / Phillip Goodwin. – 2004. – Режим доступу до ресурсу: https://discovery.ucl.ac.uk/id/eprint/1259/1/2004_25.pdf.
3. Chin H. An Impact Evaluation of Traffic Congestion on Ecology [Електронний ресурс] / H. Chin, M. Rahman. // Planning Studies and Practice. – 2011. – №3. – С. 32–44.
4. Emo A. The slow and the furious: Anger, stress and risky passing in simulated traffic congestion / A. Emo, G. Matthews, G. Funke. // Transportation Research Part F: Traffic Psychology and Behaviour. – 2016. – №42. – С. 1–14.
5. Influence of traffic congestion on driver behavior in post-congestion driving [Електронний ресурс] / [G. Li, W. Lai, X. Sui та ін.]. // Accident Analysis & Prevention. – 2020. – №141.
6. Hesham S. Intelligent Traffic Management Systems: A review [Електронний ресурс] / S. Hesham, E. Magda. // Nile Journal of Communication and Computer Science. – 2023. – №5. – С. 42–56.
7. Clark L. Traffic signals: A brief history [Електронний ресурс] / Larry Clark. // Washington State Magazine. – 2019. – Режим доступу до ресурсу: https://magazine.wsu.edu/web-extra/traffic-signals-a-brief-history/
8. Nellore K. A Survey on Urban Traffic Management System Using Wireless Sensor Networks / K. Nellore, G. Hancke. // Sensors. – 2016. – №16.
9. Paulus R. Intelligent traffic light design and control in smart cities: a survey on techniques and methodologies / R. Paulus, A. Agrawal. // International

Journal of Vehicle Information and Communication Systems. – 2020. – №5. – С. 436.

10. Effective Urban Traffic Monitoring by Vehicular Sensor Networks / R.Du, C. Chen, B. Yang, N. Lu. // IEEE Transactions on Vehicular Technology. – 2014. – №64.
11. Reducing traffic jams via VANETs / F. Knorr, D. Baselt, M. Schreckenberg, M. Mauve // IEEE Transactions on Vehicular Technology. – 2012. – №61.
12. Sin Ee Chuo H., Keng Tan M., Chee Hoe Chong A., Chin R. Evolvable traffic signal control for intersection congestion alleviation with enhanced particle swarm optimization. Матеріали IEEE 2nd International Conference on Automatic Control and Intelligent Systems (I2CACIS) : 2017.
13. Fleuren S. Optimizing pre-timed control at isolated intersections / Stijn Fleuren., 2016. – 287 с.
14. Srivastava. J. R., Sudarshan T. S. B. Intelligent Traffic Management with Wireless Sensor Networks. Матеріали IEEE/ACS International Conference on Computer Systems and Applications, AICCSA. 1-4 : 2013.
15. Faye S., Chaudet C., Demeure I. A Distributed Algorithm for Adaptive Traffic Lights Control. Матеріали 15th International IEEE Conference on Intelligent Transportation Systems (ITSC) : 2013.
16. Dynamic Traffic Light Management System Based On Wireless Sensor Networks For The Reduction Of The Red-Light Running Phenomenon / G.Scata, T. Campisi, M. Collotta, G. Pau. // Transport and Telecommunication. – 2014. – №15.
17. Halit E., Hariani P., Alghamdi A. S., Yue Y. Instrumentation for safe vehicular flow in intelligent traffic control systems using wireless networks. Матеріали IEEE Instrumentation and Measurement Technology Conference : 2013.

18. Fahmy M. An Adaptive Traffic Signaling For Roundabout With Four Approach Intersections Based On Fuzzy Logic / Maged M. M. Fahmy. // Journal of Computing and Information Technology. – 2007. – №15. – С. 33–45.
19. Araghi S., Khosravi A., Creighton D. ANFIS Traffic Signal Controller for an Isolated Intersection. Матеріали конференції International Conference on Fuzzy Computation Theory and Applications, 2014. С. 175-180
20. Online Incremental Machine Learning Platform for Big Data-Driven Smart Traffic Management / [D. Nallaperuma, R. Nawaratne, T. Bandaragoda та ін.]. // IEEE Transactions on Intelligent Transportation Systems. – 2019. – №20. – С. 4679–4790.
21. Shapiro L. Computer Vision / L. Shapiro, G. Stockman., 2001. – 580 с.
22. Haralick R. Image Analysis Using Mathematical Morphology / R. Haralick, S. Sternberg, X. Zhuang. // IEEE Transactions On Pattern Analysis And Machine Intelligence. – 1987. – №4. – С. 532–550.
23. A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects / [L. Zewen, L. Fan, Y. Wanjie та ін.]. // IEEE Transactions on Neural Networks and Learning Systems. – 2021. – №99. – С. 1–21.
24. You Only Look Once: Unified, Real-Time Object Detection : матеріали IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 27-30 червня 2016 у Лас Вегас, NV, США, ст. 779-788.
25. SUMO – Simulation of Urban MObility: An Overview : матеріали третьої Міжнаціональної конференції по досягненням у системному моделюванні. 23-29 жовтня 2011 р. – Барселона, Іспанія.
26. European wide meta-analysis of values of travel time / Wardman Mark, 2012. – 57 с.
27. Дані Світового банку щодо ВВП на душу населення по ПКС для України та Албанії [Електронний ресурс] – Режим доступу до ресурсу: https://data.worldbank.org/indicator/NY.GDP.PCAP.PP.CD?locations=UA-AL

# Додаток А

## Код сенсорного модуля на основі віднімання фону

```
import time
import cv2 as opencv
import numpy as np

cap = opencv.VideoCapture('E:\Учёба\!
Диплом\Материалы\Видео\dark 1280X720 cut 1.mp4')
object_detector =
opencv.createBackgroundSubtractorMOG2(history=1000, varThreshold
= 90)

frame_width = int(cap.get(opencv.CAP_PROP_FRAME_WIDTH))
frame_height = int(cap.get(opencv.CAP_PROP_FRAME_HEIGHT))
fps = int(cap.get(opencv.CAP_PROP_FPS))
timestamp = time.strftime("%Y-%m-%d_%H-%M-%S")
out = opencv.VideoWriter(rf'E:\Учёба\!
Диплом\Материалы\Видео\out_video_bgsub_{timestamp}.mp4',

opencv.VideoWriter.fourcc('m','p','4','v'),
                         fps, (frame_width, frame_height))


def click_event(event, x, y, flags, params):
    if event == opencv.EVENT_LBUTTONDOWN:
        print(x, ' ', y)

        font = opencv.FONT_HERSHEY_SIMPLEX
        opencv.putText(frame, str(x) + ',' +
                    str(y), (x, y), font,
                    1, (255, 0, 0), 2)
        opencv.imshow("Frame", frame)


start_time = time.time()
while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        break

    height, width, _ = frame.shape

    #Setting ROI bounds
    roi_bounds = np.array([[220, 222], [356, 225], [1149, 644],
[1146, 877], [332, 878]], np.int32)
    roi_bounds = roi_bounds.reshape((-1, 1, 2))
    x, y, w, h = opencv.boundingRect(roi_bounds)
    roi = frame
    # opencv.polylines(frame, [roi_bounds], isClosed=True,
color=(255, 0, 0), thickness=2)

    #Applying a mask to make an ROI
```

```
    mask = np.zeros(roi.shape[:2], dtype=np.uint8)
    # opencv.fillPoly(mask, pts=[roi_bounds], color=(255, 255,
255))
    bin_roi = object_detector.apply(roi, learningRate = 0.01)
    # bin_roi = opencv.bitwise_and(bin_roi, bin_roi, mask=mask)

    #Image processing and binarization
    bin_roi = opencv.GaussianBlur(bin_roi, (49, 49), 0)
    _, bin_roi = opencv.threshold(bin_roi, 35, 255,
opencv.THRESH_BINARY)
    bin_roi = opencv.dilate(bin_roi, (5, 5), iterations=2)
    bin_roi = opencv.erode(bin_roi,

opencv.getStructuringElement(opencv.MORPH_ELLIPSE, (8, 8)),
                           iterations=1)

    #Detecting contours
    contours, _ = opencv.findContours(bin_roi,
opencv.RETR_EXTERNAL, opencv.CHAIN_APPROX_NONE)
    detections = []
    for cnt in contours:
        area = opencv.contourArea(cnt)
        if area > 150:
            x, y, w, h = opencv.boundingRect(cnt)
            detections.append([x, y, w, h])

    #Drawing rectangles
    for detection in detections:
        x, y, w, h = detection
        opencv.rectangle(roi, (x, y), (x + w, y + h), (0, 255,
0), 2)

    #Visual output
    opencv.putText(frame, f"Amount of vehicles in ROI:
{len(detections)}",
                   (10, 30), opencv.FONT_HERSHEY_PLAIN, 2, (0,
0, 255), 2)
    opencv.imshow("Frame", frame)
    opencv.imshow("Mask", bin_roi)
    opencv.setMouseCallback('Frame', click_event)
    out.write(frame)

    if opencv.waitKey(1) & 0xFF == ord('q'):
        break


end_time = time.time()
elapsed_time = end_time - start_time
print(f"Elapsed time: {elapsed_time:.2f} seconds")

cap.release()
out.release()
opencv.destroyAllWindows()
```

Додаток Б

Код сенсорного модуля на моделі глибинного навчання

```
import time
import cv2 as opencv
import numpy as np
from ultralytics import YOLO
from VehicleTracker import VehicleTracker

model = YOLO(model_path)
cap = opencv.VideoCapture(capture_src)
tracker = VehicleTracker([288, 612], [930, 530])
passed_count = 0

frame_width = int(cap.get(opencv.CAP_PROP_FRAME_WIDTH))
frame_height = int(cap.get(opencv.CAP_PROP_FRAME_HEIGHT))
fps = int(cap.get(opencv.CAP_PROP_FPS))
timestamp = time.strftime("%Y-%m-%d_%H-%M-%S")
out = opencv.VideoWriter(rf'E:\Учёба\!
Диплом\Материалы\Видео\out_video_cnn_{timestamp}.mp4',

opencv.VideoWriter.fourcc('m','p','4','v'),
                         fps, (frame_width, frame_height))

start_time = time.time()
while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        break

    # Setting ROI bounds
    roi_bounds = np.array([[220, 222], [356, 225], [1149, 644],
[1146, 877], [332, 878]], np.int32)
    roi_bounds = roi_bounds.reshape((-1, 1, 2))
    roi = frame
    #Drawing ROI and masking
    opencv.polylines(frame, [roi_bounds], isClosed=True,
color=(255, 0, 0), thickness=2)
    mask = np.zeros_like(roi)
    opencv.fillPoly(mask, pts=[roi_bounds], color=(255, 255,
255))

    # Performing object detection and counting vehicles in ROI
    detections = model.predict(source=opencv.bitwise_and(frame,
mask),
                               save=False, show=False,
verbose=False, conf=0.5)

    opencv.line(frame, [288, 612], [930, 530], color=(0, 0,
255), thickness=2)

    vehicle_count = 0
    bounding_boxes = []
```

```
    for detection in detections:
        boxes = detection.boxes
        for box in boxes:
            x1, y1, x2, y2 = map(int, box.xyxy[0])
            bounding_boxes.append([x1, y1, x2 - x1, y2 - y1])
            opencv.rectangle(frame, (x1, y1), (x2, y2), (0, 255,
0), 2)
            vehicle_count += 1

    # Tracking and counting vehicles that passed the line
    passed_count += tracker.update(bounding_boxes)

    # Display vehicle count on frame
    opencv.putText(frame, f'Amount of vehicles in ROI:
{vehicle_count}',
                   (20, 30), opencv.FONT_HERSHEY_SIMPLEX, 1, (0,
0, 255), 2)
    opencv.putText(frame, f'Vehicles passed: {passed_count}',
                   (20, 60), opencv.FONT_HERSHEY_SIMPLEX, 1, (0,
0, 255), 2)
    opencv.imshow('Vehicle Detection and Counting', frame)
    out.write(frame)

    if opencv.waitKey(1) & 0xFF == ord('q'):
        break

end_time = time.time()
elapsed_time = end_time - start_time
print(f"Elapsed time: {elapsed_time:.2f} seconds")

cap.release()
out.release()

opencv.destroyAllWindows()
```

## Додаток В

## Код модуля відслідковування об'єктів

```
import math
import numpy as np

class VehicleTracker:
    def __init__(self, start_point, end_point):
        self.center_points = {}
        self.start_point = start_point
        self.end_point = end_point
        self.curr_last_id = 0


    def _has_crossed_line(self, prev_pos, curr_pos):
        # Function to check if the car has crossed the line
        prev_side = np.sign((self.end_point[0] -
self.start_point[0])
                            * (prev_pos[1] -
self.start_point[1]) -
                            (self.end_point[1] -
self.start_point[1])
                            * (prev_pos[0] -
self.start_point[0]))

        curr_side = np.sign((self.end_point[0] -
self.start_point[0])
                            * (curr_pos[1] -
self.start_point[1]) -
                            (self.end_point[1] -
self.start_point[1])
                            * (curr_pos[0] -
self.start_point[0]))

        return prev_side != curr_side


    def update(self, objects_rect):
        bounding_boxes = []
        crossings = 0
        for rect in objects_rect:
            x, y, w, h = rect
            cx = x + (w // 2)
            cy = y + (h // 2)

            prev_detected = False
            for curr_id, curr_center in
self.center_points.items():
                dist = math.hypot(cx - curr_center[0], cy -
curr_center[1])

                if dist < h / 3:
```

```
                    if
self._has_crossed_line(self.center_points[curr_id], [cx, cy]):
                        crossings += 1
                    self.center_points[curr_id] = (cx, cy)
                    bounding_boxes.append([x, y, w, h, curr_id])
                    prev_detected = True
                    break

            if prev_detected is False:
                self.center_points[self.curr_last_id] = (cx, cy)
                bounding_boxes.append([x, y, w, h,
self.curr_last_id])
                self.curr_last_id += 1

        new_center_points = {}
        for bounding_box in bounding_boxes:
            curr_center = self.center_points[bounding_box[4]]
            new_center_points[bounding_box[4]] = curr_center

        self.center_points = new_center_points.copy()
        return crossings
```

Додаток Г

Код модуля управління світлофором

```
import time
import traci
import sumolib
from AvgWaitTimeCalc import calculate_avg_wait_time

def get_standing_vehicles(vehicles_on_lane):
    standing_vehicles = []
    for vehID in vehicles_on_lane:
        speed = traci.vehicle.getSpeed(vehID)
        if speed == 0:  # Vehicle is standing
            if traci.vehicle.getDistance(vehID) > 10:
                standing_vehicles.append(vehID)
    return standing_vehicles


# Start the SUMO simulation
timestamp = time.strftime("%Y-%m-%d_%H-%M-%S")
tripinfo_file = rf"E:\Учёба\!
Диплом\Программы\SUMO\tripinfo_{timestamp}.xml"
traci.start(["sumo-gui",
             "-c", path,
             "--tripinfo-output", tripinfo_file, "--quit-on-
end"])

# Traffic light ID
tls_id = "J14"

logic = traci.trafficlight.getAllProgramLogics(tls_id)
phases = logic[0].getPhases()
num_phases = len(phases)

# Controlled edges for the intersection
edges = ["E4", "-E5", "E6", "E3"]

# north_edge = "E4"
# east_edge = "-E5"
# south_edge = "E6"
# west_edge = "E3"

min_green_time = 10
max_green_time = 30
adjust_weight = 1
init_vehicle_p_time = 1

last_adjusted_phase = -1

phase_green_edge_dict = {0: [edges[0], edges[2]],
                         2: [edges[1], edges[3]]}

edge_lane_dict = {}
```

```
lane_time_dict = {}
for edge in edges:
    edge_lane_dict[edge] = []
    for lane_num in range(traci.edge.getLaneNumber(edge)):
        lane_id = edge + "_" + str(lane_num)
        edge_lane_dict[edge].append(lane_id)
        lane_time_dict[lane_id] = init_vehicle_p_time

tracked_vehicles = {}
n_vehicles_on_phase_start_by_lane = {}

step = 0
phase_start_step = 0
while step < 3600:
    traci.simulationStep()
    current_phase = traci.trafficlight.getPhase(tls_id)

    #Start of a new phase
    if current_phase != last_adjusted_phase:
        if current_phase in phase_green_edge_dict: #Green phase
            phase_start_step = step
            green_duration = min_green_time

            #Iterating over lanes and multiplying number of
vehicles currently on them on avg pass time
            for edge in phase_green_edge_dict[current_phase]:
                for lane in edge_lane_dict[edge]:
                    tracked_vehicles[lane] =
get_standing_vehicles(traci.lane.getLastStepVehicleIDs(lane))
                    n_vehicles_on_phase_start_by_lane[lane] =
len(tracked_vehicles[lane])
                    proposed_lane_g_time = (lane_time_dict[lane]
                                            *
n_vehicles_on_phase_start_by_lane[lane])
                    if proposed_lane_g_time > green_duration:
                        green_duration = proposed_lane_g_time

            if green_duration > max_green_time:
                green_duration = max_green_time
            traci.trafficlight.setPhaseDuration(tls_id,
green_duration)
            print(f"Step:{step}, Phase:{current_phase},
Duration:{green_duration}")
            last_adjusted_phase = current_phase
        else: #Yellow phase
            if len(tracked_vehicles) != 0: #Tracked vehicles
dict is not empty - not all vehicles passed through
                # Adjusting theoretical avg vehicle pass time to
real data
                for lane, vehicles in
list(tracked_vehicles.items()):
                    if (n_vehicles_on_phase_start_by_lane[lane]
!= 0 and
```

```
n_vehicles_on_phase_start_by_lane[lane] != len(vehicles)):
                        # raw_multiplier =
(n_vehicles_on_phase_start_by_lane[lane] /
                        #
(n_vehicles_on_phase_start_by_lane[lane] - len(vehicles)))
                        raw_multiplier = ((step -
phase_start_step) /

((n_vehicles_on_phase_start_by_lane[lane] - len(vehicles))
                                                *
lane_time_dict[lane]))
                        lane_time_dict[lane] *= 1 - ((1 -
raw_multiplier) * adjust_weight)
                    del tracked_vehicles[lane]
            n_vehicles_on_phase_start_by_lane = {}
    #Current phase continues
    else:
        if current_phase in phase_green_edge_dict:  # Green
phase
            for lane, vehicles in
list(tracked_vehicles.items()):
                no_tracked_v_on_lane = True
                for vehicle in vehicles:
                    if traci.vehicle.getLaneID(vehicle) == lane:
                        no_tracked_v_on_lane = False
                    else:
                        tracked_vehicles[lane].remove(vehicle)
                if no_tracked_v_on_lane:
                    # Adjusting theoretical avg vehicle pass
time to real data
                    time_passed = step - phase_start_step - 1
                    if n_vehicles_on_phase_start_by_lane[lane]
!= 0 and time_passed > min_green_time:
                        raw_multiplier = ((step -
phase_start_step) /

(n_vehicles_on_phase_start_by_lane[lane] *
lane_time_dict[lane]))
                        lane_time_dict[lane] *= 1 - ((1 -
raw_multiplier) * adjust_weight)
                    # Removing line from tracking
                    del tracked_vehicles[lane]

    step += 1

# Close TraCI connection
traci.close()
average_waiting_time, vehicle_count =
calculate_avg_wait_time(tripinfo_file)
print(f"Average waiting time =
{average_waiting_time}".replace('.', ',') + '\n'
      + f"Vehicle count = {vehicle_count}")
```