Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Форостіна Ігоря Олександровича*
(ПІБ)

академічної групи *122-18-1*
(шифр)

спеціальності *122 Комп'ютерні науки*
(код і назва спеціальності)

освітньої програми *Комп'ютерні науки*
(назва освітньої програми)

на тему: *Розробка інтелектуальної системи допомоги водіям при екстрених ситуаціях на дорозі*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	*доц. Удовик І.М.*			
розділів:				
спеціальний	*доц. Удовик І.М.*			
економічний	*доц. Касьяненко Л.В.*			
Рецензент				
Нормоконтролер	*доц. Гуліна І.Г.*			

Дніпро
2022

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

_____ І.М. Удовик
(підпис) (прізвище, ініціали)

« » _____ 2022 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *122-18-1* (група) *Форостіна Ігоря Олександровича* (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка інтелектуальної системи допомоги водіям при екстрених ситуаціях на дорозі*

затверджена наказом ректора НТУ «ДП» від *18.06.2022 р.* № *268-с*

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів проєктно-технологічної практики та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі. Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***13.05.2022 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки***	***27.05.2022 р.***

Завдання видав _____ *доц. Удовик І.М.*
(підпис) (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ *Форостін І.О.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *14.01.2022 р.*

Термін подання кваліфікаційної роботи до ЕК: *13.06.2022 р.*

РЕФЕРАТ

Пояснювальна записка: 66 с., 12 рис., 3 дод., 21 джерело.

Об'єкт розробки: система допомоги водіям при екстрених ситуаціях на дорозі.

Мета кваліфікаційної роботи: створення додатку що допоможе водіям реагувати на екстрені події на дорозі, у разі виникнення аварійної ситуації або з'їзду зі смуги руху за рахунок використання технологій комп'ютерного зору.

У вступі розглядається аналіз та сучасний стан проблеми, конкретизується мета кваліфікаційної роботи та галузь її застосування, наведено обґрунтування актуальності теми та уточнюється постановка завдання.

У першому розділі проаналізовано предметну галузь, визначено актуальність завдання та призначення розробки, сформульовано постановку завдання, зазначено вимоги до програмної реалізації, технологій та програмних засобів.

У другому розділі проаналізовані наявні рішення, обрано платформи для розробки, виконано проектування і розробка програми, описана робота програми, алгоритм і структура її функціювання, а також виклик та завантаження програми, визначено вхідні і вихідні дані, охарактеризовано склад параметрів технічних засобів.

В економічному розділі визначено трудомісткість розробленої інформаційної системи, проведений підрахунок вартості роботи по створенню програми та розраховано час на його створення.

Практичне значення полягає у створенні програмного додатка, що надає можливість підвищити безпеку на дорозі та знизити навантаження на водія транспортного засобу у випадку позаштатної ситуації.

Актуальність даного програмного продукту визначається високою кількістю автомобільних транспортних засобів на дорогах, що у свою чергу спричиняє велику кількість дорожньо-транспортних пригод із людськими жертвами; наявні світові системи зазвичай є вбудованими і доступні лише на сучасних автомобілях преміального сегменту, створення портативної і доступної системи дозволить значно покращити стан безпеки на дорогах.

Список ключових слів: АВТОМОБІЛЬ, ІНТЕЛЕКТУАЛЬНА СИСТЕМА, КОМП'ЮТЕРНИЙ ЗІР, ІНТЕРФЕЙС, ДОДАТОК, СИСТЕМА ПІДВИЩЕННЯ БЕЗПЕКИ.

ABSTRACT

Explanatory note: 66 p., 12 figs., 3 apps., 21 sources.

Object of development: the system of helping drivers in emergencies on the road.

The purpose of the qualification work: to create an application that will help drivers to react to emergency situation on the road in case of an accident or exiting the lane through the use of computer vision technology.

The introduction examines the current problem statement, specifies the purpose of the qualification work and the area of its application, justifies the relevance of the topic and specifies the problem statement.

In the first section the analysis of the subject area are carried out, the relevance of the task and the dedication of the development are determined, task statement is created, the software and hardware requirements of the product are specified, technologies and tools for development are chosen.

In the second section the existing solutions are analyzed, the platform for development is chosen, design created and development of the product finished, the algorithm, structure and architecture solutions in the system are described, the input and output data defined, characteristics of the technical resources usedand ways of running a program and features of user interaction described, between serve and client parts of product are mentioned.

The economic section defines the complexity of the information system, calculates the cost of work on creating the application and defines the time for its creation.

The practical significance is creation a product which provides an opportunity to improve safety on the road and decrease the preassure to the driver in case of emergency situation.

The relevance of this software product is determined by the high number of vehicles on the road that results the high number of road accidents with human loses; modern systems that improve safety on the road are as a rule built-in and available only modern premium cars, creating a portable and affordable system will allow to significantly improve safety on the road.

Keywords: AUTOMOBILE (VEHICLE/CAR), INTELLIGENT SYSTEM, COMPUTER VISION, INTERFACE, APPLICATION, SYSTEM OF SAFETY IMPROVING.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

СУБД - Система управління базами даних;

БД - база даних;

ПЗ - програмне забезпечення;

IDE - Integrated development environment;

ОС - операційна система;

TP - true positive;

TN - true negative;

AR – доповнена реальність (англ. augmented reality);

ШІ -штучний інтелект.

ВСТУП

Завдання даної кваліфікаційної роботи та об’єкт його діяльності безпосередньо пов’язані з напрямом підготовки та відповідає узагальненій тематиці кваліфікаційних робіт і переліку зазначених виробничих функцій, типових задач діяльності, умінню та компетенціям, якими повинні володіти бакалаври напряму 122 «Комп’ютерні науки».

У наш час широкого розповсюдження набувають персональні засоби пересування, зокрема автомобілі. Це дозволяє значно покращити рівень комфорту та забезпечити більшу свободу пересування. Автомобілі дозволяють швидко та ефективно доставляти людей та грузи, а також переміщуватися як на невеликі відстані, так і здійснювати подорожі на тисячі кілометрів. Все більше людей користується автомобілями для пересування по місту та за його межами, як у комерційних, так і у персональних цілях. Однак, одночасно із появою автомобілів, з’явилася і небезпека, яку вони несли. Перша автомобільна аварія відбулася у світі у 1891 році, а перший смертельний випадок стався уже у 1896, незважаючи на те, що швидкість зіткнення становила лише 6.4 км/год. Теоретично, найбільш безпечним автомобіль є при нульовій швидкості, тобто той, що не рухається, це, звісно, неприйнятно у сучасному світі і таке використання транспортного засобу не має ефективності. Враховуючи всі фактори ризику, робота над збільшенням безпеки автомобілів велася з початку 20 століття. Як результат, наразі ми маємо пасивні системи безпеки (поглинаючий удари корпус, паски безпеки), активні (подушки безпеки, електронні системи стабілізації ABS, ESP), а також деякі сучасні системи дозволяють уникнути зіткнення з іншими автомобілями та об’єктами на дорозі.

Метою кваліфікаційної роботи є вивчення можливості удосконалення систем активної безпеки, які включають у себе сучасні технології такі як комп’ютерний зір. На мою думку, ця проблема є актуальною, адже за даними експертів, щороку внаслідок ДТП у світі гине майже 1 мільйон 300 тисяч

людей, інвалідами стають близько 50 мільйонів. В Україні, згідно зі статистикою, — понад 3,6 тисячі летальних та майже 30 тисяч випадків інвалідності. Розробка ефективного програмного продукту, доступного широкій масі людей, який міг би моніторити дорожню обстановку у режимі реального часу та попереджати водія у разі екстреної ситуації значно б посприяла збереженню великої кількості людських життів.

Дана робота знайомить з методами обробки відеоряду та розпізнаванням об’єктів у ньому (комп’ютерний зір).

Основними вимогами до системи будуть: доступні для розуміння інтерфейс, надійна та стабільна робота системи, точність розрахунків.

Результатом виконання даної роботи буде програма, яка буде підвищувати безпеку на дорогах а також покращить обізнаність водія у дорожній обстановці навколо нього.

Саме тому завданням кваліфікаційної роботи було обрано “Розробка інтелектуальної системи допомоги водіям при екстрених ситуаціях на дорозі”. Завдання даної кваліфікаційної роботи та об’єкт його діяльності безпосередньо пов’язані з напрямом підготовки, відповідає узагальненій тематиці кваліфікаційних робіт та переліку зазначених навичок та компетенцій, якими повинен володіти фахівець напряму “Комп’ютерні науки”.

РОЗДІЛ І
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Загальні відомості з предметної галузі

Системи допомоги водієві - це системи, метою запровадження яких є підвищення рівня безпеки на дорозі. На даний момент існує декілька таких систем, кожна з яких розроблюється і встановлюється виробниками автомобілів на авто власної марки, наприклад, такі запатентовані системи є у Audi, BMW, Volkswagen тощо.

Існують й сторонні рішення, наприклад, ADAS. ADAS — це система допомоги водію на основі машинного зору. Ціль системи ADAS — підвищення безпеки руху шляхом інформування водія та привертання його уваги. Як мінімум, це попередження звуковим сигналом або вібрацією водія про ймовірний або ризик, що виник та який вимагає уваги.

Перші подібні системи з’явилися близько 50 років тому і мали значно спрощений функціонал, порівняно із сучасними такими системами. Серед систем, які можна зустріти на сучасних авто популярними є системи, що використовують дальноміри (лідари) для попередження водія про наближення до перешкоди, камери для відслідковування дорожнього полотна та об’єктів на ньому або ж комбінація вище перелічених засобів.

За своєю суттю ці системи можуть відрізнятися за кількістю датчиків, за максимальною швидкістю, на якій можлива коректна робота та за діапазоном перед авто, у якому вони можуть працювати[1].

ADAS розрізняють за п'ятьма рівнями: від нульового коли керує повністю водій до п'ятого, який теоретично зможе повністю виконувати керування, тобто безпілотний автомобіль – це екстремальний випадок ADAS.

Еволюцію таких систем ми можемо спостерігати з 1965 року, коли вперше почали використовувати систему круїз-контролю, яка дозволяла автомобілю утримувати задану швидкість без участі водія, надалі із розвитком технологій з’явився адаптивний круїз-контроль, який завдяки датчикам

дозволяє уникати зіткнення із автомобілем, що їде попереду у режимі автоматичної їзди. Наразі ми можемо спостерігати більш досконалі системи, які за допомогою групи датчиків та камер із використанням технологій комп'ютерного зору можуть частково виконувати керування транспортним засобом, яскравий приклад – автомобілі Tesla.

Для подібної системи надзвичайно важлива швидкість роботи та дальність, з якої вона може розпізнати загрозу. Для прикладу, через збільшення часу реакції та гальмівного шляху вірогідність отримати травми несумісні з життям на швидкості 65 км/год дорівнює 70% у той час, як рухаючись зі швидкістю 50 км/год шанс вижити становить 80%.

Далі хотілося би звернути увагу на конкретні існуючі рішення від різних виробників та їхні особливості:

- Адаптивний круїз-контроль (ACC) та асистент контролю дистанції попереду автомобіля Front Assist автоматично корегує швидкість відповідно до швидкості автомобіля, який рухається попереду й тримає дистанцію, задану водієм. Асистент контролю дистанції попереду з автоматичною системою аварійного гальмування в умовах міського циклу, що входить до його складу, допомагає в критичних ситуаціях зменшити шлях гальмування.
- Асистент утримання смуги руху (Lane Assist). У разі випадкового покидання смуги руху скеровує автомобіль назад на смугу й інформує водія за допомогою звукового сигналу та повідомлення на екрані.
- Попередження про лобове зіткнення: Audi pre sense front. Система безпеки Audi pre sense front доповнює роботу системи попередження зіткнення і автоматичну функцію гальмування системи Audi pre sense city щодо транспортних засобів, які їдуть попереду, на всьому діапазоні швидкості від 0 до 250 км/год. Система використовує два радари і камеру та має завдання або уникнути лобових зіткнень, чи принаймні зменшити тяжкість їх

наслідків на більш високих швидкостях. У ситуації, яка може бути розцінена як небезпечна, система спонукає водія натиснути на гальма згідно з комплексною концепцією попереджень - візуальними та акустичними сигналами, а також гальмівними поштовхами.

Як ми можемо побачити, системи, що інтегровані в автомобіль, дозволяють частково виконувати керування та здійснювати маневри, реалізувати аналогічний функціонал із портативною системою, що встановлюється окремо, буде важко як з технічного, так і з правового боку адже дискусії щодо масового використання безпілотних автомобілів або таких, що можуть бути частково керованими у безпілотному режимі у світі усе ще ведуться. Однак, неможливість такої системи бути використаної для автономного керування транспортним засобом нівелюється її вартістю, доступністю а також сумісністю з великою кількістю транспортних засобів, на яких не передбачена штатна аналогічна система.

З огляду на сучасні можливості обчислювальної техніки, основний акцент буде зроблено на використанні камери відеоспостереження, яка у режимі реального часу буде слідкувати за дорогою, визначати смугу руху, а також перешкоди на дорозі з використанням технологій комп’ютерного зору, опціонально, можливе також встановлення датчиків наближення, проте, як наслідок невеликої дальності їхньої точної роботи, вони будуть ефективними лише на невеликих відстанях (у межах декількох метрів) та на малих швидкостях.

Комп'ютерний зір - теорія та технологія створення машин або механізмів, які можуть проводити виявлення, стеження за об’єктами та їх визначення.

Як наукова дисципліна, комп'ютерний зір належить до теорії та технології створення штучних або рукотворних систем, які отримують інформацію у візуальному вигляді. Відеодані можуть бути представлені у

вигляді багатьох форм, таких як відеоряд, зображення з різних камер або тривимірними даними з спеціалізованого сканера.

Безсумнівно, один із найзначніших проривів, пов'язаних із комп'ютерним зором та ШІ, стався у сфері відеоспостереження, яке є важливою частиною фізичної безпеки. Інтелектуальне відеоспостереження, вже багато в чому перевершило можливості людини, навіть професійно навченої. Комп'ютер не втрачає уважності, не втомлюється та не відволікається – людський фактор не погіршує його роботу. Системи безпеки та відеоспостереження, оснащені алгоритмами комп'ютерного зору дозволяють здійснювати безперервний моніторинг обстановки всередині та ззовні приміщень, проводити інспекцію різних об'єктів без безпосереднього відвідування їх, розпізнавання та порівняння осіб людей, проведення ситуаційного моніторингу обстановки у громадських місцях та на транспортних вузлах.

За даними з патрульної служби, 7 основних причин ДТП це:

- Свідоме порушення ПДР;
- Недотримання дистанції та інтервалу;
- Водіння у нетверезому стані;
- Виїзд на смугу зустрічного руху;
- Перевищення швидкості;
- Порушення правил обгону та випередження;
- Розсіювання уваги.

Система, обладнана камерами з можливостями комп’ютерного зору допоможе значно зменшити вплив принаймні трьох з семи компонентів, а саме: недотримання дистанції, виїзд на смугу протилежного руху, неуважність.

1.2. Призначення розробки та галузь застосування

Інформаційна система, що виконана для кваліфікаційної роботи, має назву «Розробка інтелектуальної системи допомоги водіям при екстрених ситуаціях на дорозі».

Основні терміни та ключові слова:

Python - високорівнева мова програмування загального призначення з динамічної строгою типізацією і автоматичним управлінням пам'яттю, орієнтований на підвищення продуктивності розробника, читання коду і його якості, а також на забезпечення переносимості написаних на ньому програм. Мова є повністю об'єктно-орієнтованим - все є об'єктами.

OpenCV - це open source бібліотека комп'ютерного зору, яка призначена для аналізу, класифікації та обробки зображень. Широко використовується в таких мовах як C, C++, Python та Java.

NumPy — бібліотека з відкритим кодом для мови програмування Python. Можливості: підтримка багатовимірних масивів; підтримка високорівневих математичних функцій, призначених до роботи з багатовимірними масивами.

Розроблений продукт може використовуватися на легкових та вантажних транспортних засобах незалежно від їхнього призначення, його головна мета – підвищити безпеку пересування. На мою думку, даний продукт при його масовому розповсюдженню міг би значно посприяти покращенню статистики ДТП на дорогах.

Призначення розробки - надати можливість водіям отримати інструмент, який буде інформувати їх про небезпеку та можливі аварійні ситуації. Моя комп’ютерна система дозволить водію оперативно отримувати сповіщення про з’їзд зі смуги руху або небезпечне наближення до сторонніх об’єктів.

1.3. Підстави для розробки

Відповідно до освітньої програми, згідно навчального плану та графіків навчального процесу, в кінці навчання студент виконує кваліфікаційну роботу.

Тема роботи узгоджується з керівником проекту, випускаючою кафедрою, та затверджується з наказом ректора.

Таким чином підставами для розробки (виконанням кваліфікаційної роботи) є:

- освітня програма спеціальності 122 “Комп’ютерні науки”;
- навчальний план та графік навчального процесу;
- наказ ректора Національного технічного університету “Дніпровська політехніка” № 268-с від 18.06.2022р;
- завдання на кваліфікаційну роботу на тему “ Розробка інтелектуальної системи допомоги водіям при екстрених ситуаціях на дорозі ”.

1.4. Постановка завдання

Метою проекту є розробити додаток для збирання даних з камери з метою виділення об’єктів на них, а також сповіщеннями у випадку виникнення позаштатних ситуацій.

Даний продукт дозволить водіям отримувати більше інформації про дорожню обстановку у режимі реального часу, що має на меті підвищення безпеки пересування транспортних засобів.

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Кінцевий продукт повинен дотримуватися наступних функціональних вимог:

– інтуїтивно зрозумілий інтерфейс користувача;
– дані повинні виводитися користувачу на екран;
– трекінг дорожнього полотна у режимі реального часу;
– налагоджена система сповіщень про наближення потенційно небезпечних об’єктів.

1.5.2. Вимоги до інформаційної безпеки

Додаток має запускатися локально задля того аби інформація надходила у режимі реального часу, також це унеможливить тимчасову несправність у випадку якщо б обчислення були хмарними і мала місце нестача інтернет-з’єднання або несправності на сервері.

Безпека має бути досягнута також шляхом приховання вразливих даних методом інкапсуляції.

Оскільки система має бути використана під час руху транспортного засобу, живлення її електронних компонентів, у тому числі камери відеоспостереження має здійснюватись безпосередньо з бортової системи живлення автомобіля для безперервної роботи.

1.5.3. Вимоги до складу та параметрів технічних засобів

Для підтримки стабільної роботи системи, сервер має знаходитися локально і відповідати таким технічним вимогам:

– операційна система: Windows 10, Windows Server 2016 або Windows Server 2019;
– оперативної пам'яті 1 гігабайт;
– процесор x64 з тактовою частотою 1,4 ГГц;
– 6 гігабайт вільного місця на диску;
– встановлений Python 3.

– наявність дискретного відеоадаптора

1.5.4. Вимоги до інформаційної та програмної сумісності

Основною мовою програмування був обраний Python3[2]. Також ця програмна мова була використана для побудови скриптового допоміжного файлу.

Задля реалізації машинного зору використана стороння бібліотека cv2[3]. Робота з даними організована за рахунок використання бібліотеки NumPy.

РОЗДІЛ 2
ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

2.1. Функціональне призначення програми

Результатом даної кваліфікаційної роботи має бути застосунок, який у режимі реального часу за допомогою камери стежить за дорожньою обстановкою.

Кінцевим результатом роботи програми повинне бути виведене на екран зображення з камери із виділеними на ньому полосою для руху, а також об’єктами, що наближаються.

Основне призначення додатку:

- підвищення уваги водія на дорозі;
- відслідковування об’єктів, що наближаються та можуть створювати небезпеку;
- виділення графічно полоси для руху.

Для досягнення поставленої задачі додаток повинен уміти оброблювати великі об’єми користувацьких відеоданних та видавати результат у графічному вигляді на екран у режимі реального часу.

2.2. Опис застосованих математичних методів

В даній кваліфікаційній роботі для виділення та відслідковування об’єктів використовується технологія комп’ютерного зору – її головною метою є розрізнення окремих частин об’єктів для їхньої ідентифікації.

Перш за все необхідно зрозуміти задачу, яка ставиться перед програмою, що використовує технологію комп’ютерного зору. Задачі комп’ютерного зору передбачають певний рівень невизначеності. Це є основною причиною використання теорії нечіткої логіки.

Вважається, що можна досягти значного покращення завдань комп'ютерного зору, якщо ви збільшите кількість інформації, яку можете

обробляти у відповідний період часу. Але хоча людина не вміє обробляти величезні потоки даних в реальному часі, вона виконує завдання розпізнавання дуже і дуже успішно. Можливість вибору правильного рівня деталізації– це те, що дозволяє людині розпізнавати об’єкти, звернувши увагу на основні деталі і при цьому не відвернути увагу від основних характеристик. Теорія нечіткої логіки дозволяє варіювати цей рівень узагальнення шляхом зміни кількості лінгвістичних змінних в системі та варіювання типу функцій належності нечітких множин[4].

Комп’ютерні системи зору повинні вміти представляти наявну невизначеність і очікуванні ефекти невизначеності для правильної інтерпретації отриманих результатів[5].

Невизначеність часто розглядається як результат якогось випадкового процесу, однак у невизначеності комп’ютерного зору це також відбувається з інших причин, зокрема: проекція зображень у менший простір, зміна освітлення, вибірка просторових параметрів або часових (у випадку відео) координатів, невідома або низька якість зображення, неточні обчислення, перетин частин розпізнаних об'єктів, тобто нездатність чітко сформулювати атрибути для визначення об'єктів. Теорія нечіткої логіки ідеально підходить для вирішення проблем невизначеності такого роду. Нечітка логіка полегшує перенесення накопиченого досвіду у сфері комп'ютерних наук та машинного бачення за допомогою простого і зрозумілого правила. Особа, яка виконує класифікацію зображення у реальному житті, працює не з чисто числовими характеристиками об'єктів, які зображуються, але й класифікує предмет на основі бази даних правил, які отримані з досвідом і можуть бути легко сформульовані. Це велика перевага таких систем, оскільки потенціал для їх удосконалення практично невичерпний і завжди можна сформулювати правила більш суворо або доповнити їх новими, уточнюючими правилами[6].

Основна концепція нечіткої логіки — нечітка множина[7]. Нечітка множина - це множина, елементи якої певною мірою належать до неї, на основі на відміну від традиційних наборів, де є елементи, що або належать до

множини, або ні. Нечітка множина — це пара: (U, m), де U — множина, і m: u→[0,1] - функція належності або характеристична функція множини. Нечіткі набори дозволяють виконувати більшість операцій, які можна виконувати зі звичайними множинами, такими як додавання, перетин і об'єднання.

Припускаючи, що елемент належить до нечіткої множини у певній мірі, судження в нечіткій логіці також вірні в певній мірі, на відміну від традиційної логіки, де твердження може бути істинним або хибним.

Розглянемо конкретні математичні методи, що використані у даній роботі і дозволяють виокремлювати необхідні об’єкти на відео.

Оскільки відео складається з кадрів, що змінюють один одного з певною частотою, то усі описані нижче методи будуть примінятися до окремо взятого кадру з відео, на які цей відеоряд буде попередньо розбитий.

Перш за все, нам необхідно виокремити об’єкти, які нас цікавлять, оскільки ми прагнемо виділити межі проїзної частини, то матимемо чіткі прямі лінії розмітки або переходу дороги на узбіччя. Для визначення таких ліній спочатку нам необхідно виділити різкі переходи на нашому зображенні, для цього використаємо метод блюр по Гаусу – таким чином ми приховаємо невеликі дефекти зображення проте значні елементи залишаться видимими[8].

Ефект Gaussian Blur є лінійною операцією і математично являє собою згортку зображення з матрицею фільтра. При цьому кожен піксель в зображенні заміщається на суму прилеглих, взятих з певними ваговими коефіцієнтами[9].

Фільтр називається Gaussian, тому що він будується з функції, відомої як гауссіана, e^(- x^2) , вона продемонстрована на рисунку 2.1.

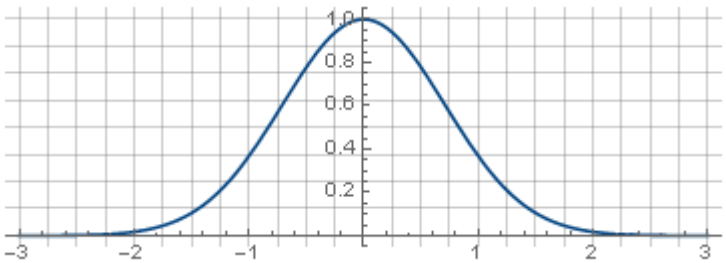

Рис. 2.1. Функція гауссіана у двомірному просторі

Тривимірний же варіант такої функції (Рис. 2.2) отриманий шляхом її обертання навколо осі ординат e^(- (x^2+y^2))

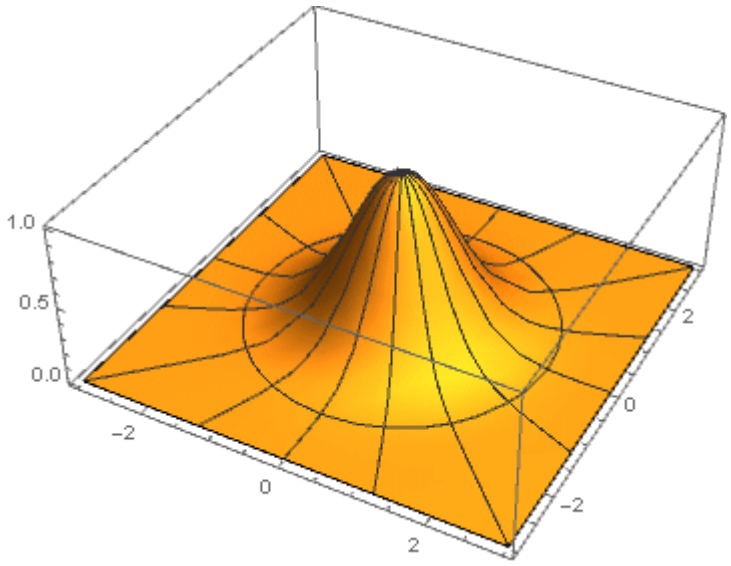

Рис. 2.2. Функція гауссіана у тривимірному просторі

Наступною дією є накладання маски на отримане зображення тобто виділення області, на якій ми шукатимемо цікавлячі нас об’єкти, зроблено це буде за допомогою виділення області трапецевидної форми[10]. Також зображення переводиться у монохромне, це дозволяє підвищити контраст та прибрати зайві об’єкти, які не несуть корисної інформації.

2.3. Опис використаних технологій та мов програмування

Дана комп’ютерна система розроблена за допомогою наступних технологій:

- бібліотека алгоритмів комп’ютерного зору OpenCV;
- мова програмування Python.

OpenCV (Open Source Computer Vision Library) - це бібліотека програмного забезпечення для комп'ютерного зору і машинного навчання з відкритим вихідним кодом. OpenCV була створена для застосунків з

використанням комп'ютерного зору і прискорення використання машинного сприйняття в некомерційних та комерційних програмних продуктах. Будучи продуктом, ліцензованим BSD, OpenCV спрощує для підприємств використання і модифікацію коду[11].

Бібліотека містить значну кількість універсальних алгоритмів, які включають в себе набір як класичних, так і модифікованих алгоритмів комп'ютерного зору і машинного навчання. Ці алгоритми можна використовувати для виявлення і розпізнавання осіб, ідентифікації об'єктів, класифікації дій в відео, відстеження рухів камери, відслідковування рухомих об'єктів, створення 3D-моделей об'єктів та 3D-множин точок зі стереокамер, «зклеювання» зображень для отримання вигляду всієї сцени з високою чіткістю, пошуку подібних зображень, прибирання ефекту червоних очей з зображень, зроблених за допомогою спалаху фотокамери, відстеження рухів очей, розпізнавання пейзажів та встановлення маркерів для накладення на пейзаж доповненої реальності тощо. Спільнота користувачів OpenCV налічує понад десятки тисяч осіб, а оціночна кількість завантажень перевищує 18 мільйонів. Бібліотека широко використовується компаніями, науковцями, дослідницькими групами та державними органами різних країн.

Поряд з відомими компаніями, такими як Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota, які використовують бібліотеку, існує безліч розвиваючихся стартапів, таких як Applied Minds, VideoSurf і Zeitera, що широко застосовують можливості OpenCV у своїх проектах. Розгорнуті на даний момент програми з використанням технологій OpenCV покривають широкий спектр застосувань: об'єднання вуличних зображень, виявлення проникань у відеоспостереження в Ізраїлі, моніторинг шахтного обладнання в Китаї, допомога навчання роботів навігації та оперуванням предметами, виявлення нещасних випадків з утопленнями в басейнах, перевірка злітно-посадкових смуг на предмет сторонніх предметів в Туреччині, перевірка

етикеток на продуктах на фабриках по всьому світу, швидке та точне розпізнавання осіб.

Бібліотека має інтерфейси для роботи з такими технологіями як C++, Python, Java і MATLAB, підтримує платформи Windows, Linux, Android і Mac OS. OpenCV в основному орієнтується на додатки для роботи з комп'ютерним зором у реальному часі і використовує переваги таких інструкцій, як MMX і SSE, коли вони доступні. Прямо зараз активно розробляються повнофункціональні інтерфейси CUDA і OpenCL. Існує понад півтисячі алгоритмів і приблизно в 10 разів більше функцій, які складають та підтримують ці алгоритми. OpenCV початково була написана на C++, має шаблонний інтерфейс, який легко працює з контейнерами STL.

Python - високорівнева мова програмування загального призначення з динамічною строгою типізацією і автоматизованим керуванням ресурсами (оперативною пам'яттю), орієнтована на підвищення продуктивності процесу розробки, читання коду і його якості, а також на забезпечення універсальності написаних на ній програм. Мова є об'єктно-орієнтованою - усе в ній є об'єктами. Незвичайною особливістю мови є виділення блоків коду відступами з використанням пробілів замість звичних дужок та лапок. Синтаксис ядра мови мінімалістичний, як наслідок при роботі нечасто виникає необхідність звертатися до документації, до того ж Python пропонує вбудовану функцію help() для уточнення необхідної інформації не відволікаючись від роботи. Сама ж мова є інтерпретованою і використовується в тому числі для написання скриптів. Недоліками даної мови є нерідко більш низька швидкість роботи і більш високе споживання пам'яті написаних на ній програм в порівнянні з аналогічним кодом, написаним на компільованих мовах низького рівня, таких як Сі або C ++, проте великим плюсом є високорівневість – тобто можливість спрощено і більш стисло вирішувати типові задачі.

Python є мультипарадигменою мовою програмування, що підтримує імперативний, процедурний, структурний, об'єктно-орієнтований підходи, метапрограмування і функціональне програмування. Завдання узагальненого

програмування вирішуються за рахунок динамічної типізації[12]. Аспектно-орієнтоване програмування частково підтримується через декоратори, більш повноцінна підтримка забезпечується додатковими бібліотеками (фреймворками). Такі методики як контрактне і логічне програмування можна реалізувати за допомогою бібліотек або додатків. Основні архітектурні риси - динамічна типізація, автоматичне керування пам'яттю, повна інтроспекція, механізм обробки виключень, підтримка багатопоточних обчислень з глобальним блокуванням інтерпретатора (GIL), високорівневі структури даних. Підтримується розбиття програм на модулі, які, в свою чергу, можуть об'єднуватися в пакети. Мова програмування Python була створена у 1991 році голландцем Гвідо ван Россумом.

Python характеризується простим для розуміння синтаксисом. Читати код на ньому легше, ніж на інших мовах програмування, так як в Python мало використовуються такі допоміжні синтаксичні елементи як дужки, крапки з комою. З іншого боку, правила мови змушують програмістів робити відступи для позначення вкладених конструкцій. Зрозуміло, що добре оформлений текст з малою кількістю відволікаючих елементів читати і розуміти легше аніж текст нагромаджений спецсимволами[13].

Python - це повноцінна багато в чому універсальна мова програмування, що використовується в різних сферах. Основна, але не єдина, підтримувана їм парадигма, - об'єктно-орієнтоване програмування.

Варто також згадати, що інтерпретатори Python поширюється вільно на підставі ліцензії подібної GNU General Public License.

У дипломному проекті я використовую стандартну бібліотеку Python оскільки вона надає широкі можливості для навчання та роботи. Стандартна бібліотека Python дуже обширна і пропонує широкий спектр можливостей. Бібліотека містить вбудовані модулі (написані на C), які забезпечують доступ до системних функцій, таких як введення/ виведення данних, запис та читання файлів, які в іншому випадку були б недоступні програмістам на Python, а також модулі, написані на Python, які надають стандартизовані рішення для

багатьох проблем, що виникають в повсякденному програмуванні. Деякі з цих модулів явно розроблені для заохочення і підвищення універсальності програм на Python шляхом абстрагування від специфіки платформи в платформно-нейтральні API[14].

Установники Python для платформи Windows зазвичай включають в себе всю стандартну бібліотеку і часто також включають безліч додаткових компонентів. Для Unix-подібних операційних систем Python зазвичай надається у вигляді набору пакетів, тому може знадобитися використовувати засоби упаковки, що поставляються з операційною системою, для отримання усіх необхідних додаткових компонентів.

У своїх кваліфікаційній роботі я також використовую Pytnon бібліотеку numpy.

NumPy-це проект з відкритим кодом, спрямований на забезпечення чисельних обчислень за допомогою Python. Він був створений в 2005 році на основі попередніх напрацювань - бібліотек Numeric і Numarray. NumPy був і залишається програмним забезпеченням з повністю відкритим кодом, безкоштовним для всіх і випущеним на умовах модифікованої ліцензії BSD.

NumPy розробляється у відкритому доступі на GitHub, завдяки консенсусу NumPy і більш широкого наукового співтовариства Python.

2.4. Опис структури системи та алгоритмів її функціювання

Для проектування та розробки системи, що буде використовувати комп'ютерний зір у режимі реального часу необхідно:

- отримати вхідне зображення;
- розбити відеоряд на окремі статичні зображення;
- на кожному зображенні обрати область обробки та виділити на ній необхідні компоненти;
- позначити межі отриманих компонентів/виділити їх кольором.

У навчальних цілях задля ефективної розробки та тестування у якості вхідного зображення використовувався записаний відеоряд, однак система дозволяє отримувати вхідний відеоряд безпосередньо з камери пристрою.

Нижче перераховані деякі рекомендації, які допоможуть домогтися ефективної роботи додатку:

- для забезпечення достатньої точності необхідно використовувати відеоряд високої чіткості (1280:720 пікселів);
- створюйте обмеження для області обробки зображення;
- фінальний вигляд буде містити два зображення накладені одне на одне для модифікації вихідного відео.

Перш за все необхідно інсталювати бібліотеки OpenCV та Numpy у систему для того щоб працювати із ними, для цього виконаємо послідовно команди (Рис. 2.3):

- pip install opencv-python
- pip install numpy

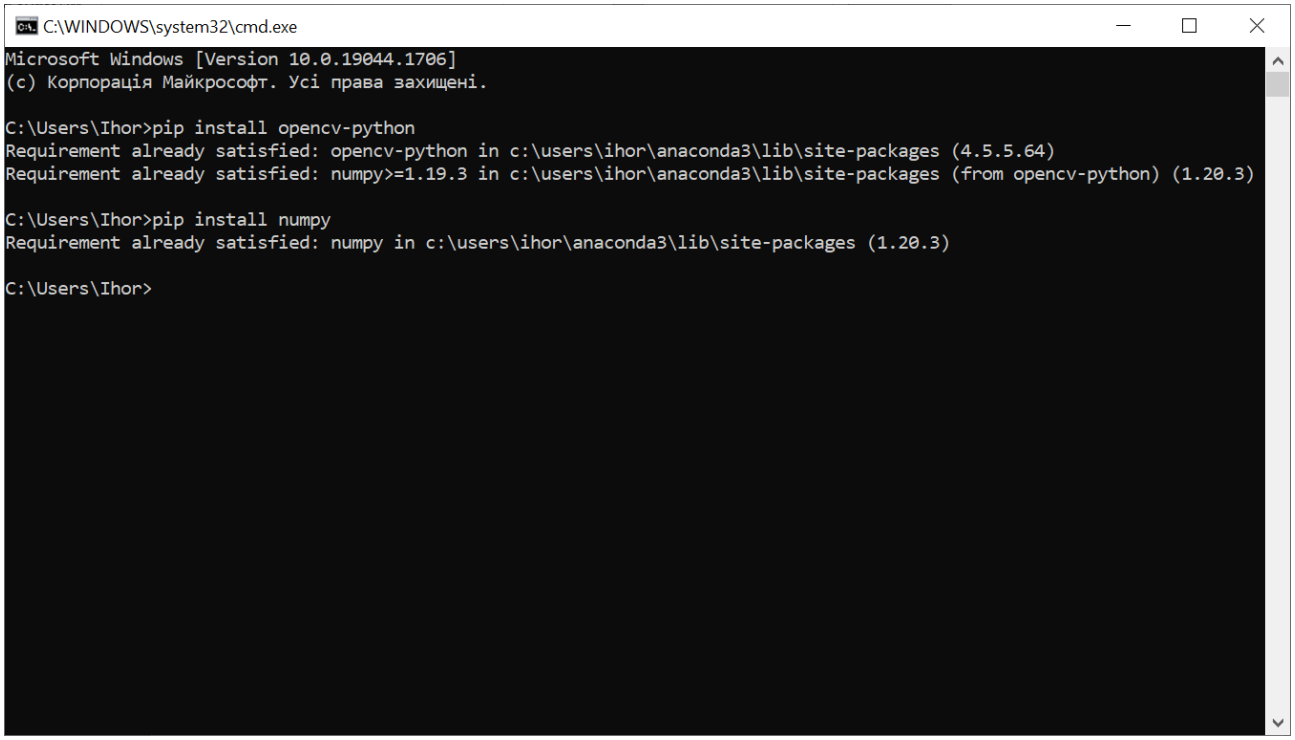

Рис. 2.3. Підготовка середовища для роботи

В даній кваліфікаційній роботі фігурують наступні методи:

Бібліотека OpenCV:

- VideoCapture() - відкриває відеофайл або послідовність файлів зображень, пристрій захоплення, тобто відеокамеру Або IP-відеопотік для захоплення відео;
- Threshold() - функція застосовує граничне значення фіксованого рівня до багатоканального масиву. Ця функція зазвичай використовується для отримання дворівневого зображення з зображення у відтінках сірого або для видалення шуму, тобто для фільтрації пікселів із занадто маленькими або занадто великими значеннями. Існує кілька типів порогового значення, підтримуваних функцією. Вони визначаються параметром типу[15]. У даній роботя ця функція дозволяє відокремити шуми та сторонні об'єкти від автомобілів та ліній розмітки на дорозі;
- namedWindow() - функція namedWindow створює вікно, яке можна використовувати в якості плейсхолдера для зображень і відео. Створені вікна називаються за вказаними іменами. Якщо вікно з таким же ім'ям вже існує, функція не виконує жодних дій. У данному проекті ця функція є допоміжною;
- resizeWindow() – функція дозволяє змінити масштаб вікна, не змінюючи при цьому дозвіл відео, цез роблено з метою зручності та наглядності;
- HoughLinesP() - Знаходить сегменти ліній в двійковому зображенні за допомогою імовірнісного перетворення Хафа[16]. Функція реалізує ймовірнісний алгоритм перетворення Хафа для виявлення ліній. Функція використовується для виділення меж дорожнього полотна;
- addWeighted() - Функція addWeighted обчислює зважену суму двох масивів. У разі багатоканальних масивів кожен канал обробляється незалежно. Фактично, функція виконує накладання двох зображень з метою створення цілісної картинки;
- imshow() - функція imshow відображає зображення у вказаному вікні. Вихідне зображення масштабується так, щоб воно відповідало розміру

вікна. Функція відображає кінцевий відеоряд, який і містить елементи доповненої реальності;

- waitKey() - функція waitKey очікує вводу з клавіатури нескінченно або протягом зазначених мілісекунд затримки, коли це значення позитивне. В данному проекті дана функція дозволяє реалізувати затримку між відтворенням кадрів щоб забезпечити відтворення відео у нормальній швидкості;
- destroyAllWindows() - функція destroyAllWindows знищує всі відкриті вікна з відеорядом. Дозволяє перервати виведення відеоряду на екран;
- cvtColor() - функція перетворює вхідне зображення з одного колірного простору в інший. У разі перетворення в колірний простір RGB, порядок каналів повинен бути вказаний явно (RGB або BGR). Колірний формат за замовчуванням в бібліотеці OpenCV часто називають RGB, але насправді це BGR адже байти міняються місцями. Таким чином, перший байт в стандартному 24-бітному кольоровому зображенні буде 8-бітним синім компонентом, другий байт буде зеленим, а третій байт буде червоним. Завдяки цій функції з повнокольорного зображення ми отримуємо монохромне (COLOR_BGR2BGRA), із яким у подальшому можна працювати;
- GaussianBlur() - функція приміняє до вхідного зображення з зазначене перетворення по Гаусу. Підтримується одночасна фільтрація. Функція дозволяє прибрати з поля зору невеликі сторонні об'єкти, спростити задачу пошуку меж дорожного полотна;
- Canny() - функція знаходить ребра на вхідному зображенні і позначає їх на межах об'єктів за допомогою алгоритму Canny. Найменше значення між пороговими значенням використовується для з'єднання ребер. Найбільше значення використовується для знаходження початкових сегментів сильних ребер[17]. Дана функція дозволяє на підготовленому зображені виділити межі проїздної частини;

- line() – дана функція малює відрізок лінії між двома точками на зображенні. Лінія обмежується кордонами зображення. Для ліній без згладжування з цілочисельними координатами використовується 8-зв'язний або 4-зв'язний алгоритм Брезенхема. Товсті лінії малюються з закругленими кінцями. Згладжені лінії малюються з використанням гаусової фільтрації[18]. Функція необхідна для нанесення ліній біля країв проїзної частини;
- fillPoly() – функція заповнює область, обмежену кількома багатокутними контурами. Функція може заповнювати складні області, наприклад, області з отворами, контури з самопересеченіями, деякі їх частини і так далі, слугує для привертання уваги до дорожнього полотна;
- bitwise_and() - обчислює побітове з'єднання двох масивів, обчислює побітове з'єднання для кожного елемента двох масивів або масиву і скаляра. Функція виконує об’єднання двох зображень;
- VideoWriter() – функція для створення відеоряду з кадрів, використовується для збереження відео із накладеними ефектами доповненої реальності у навчальних цілях;
- createBackgroundSubtractorMOG2() – функція віднімання фону є поширеним і широко використовуваним методом для створення маски переднього плану (а саме двійкового зображення, що містить пікселі, що належать рухомим об'єктам в сцені) з використанням статичних камер, тобто в даному випадку функція допомагає вирізнити рухомі об’єкти на тлі пейзажу;
- findContours() – функція витягує контури з двійкового зображення за допомогою алгоритму Suzuki 85. Контури є корисним інструментом для аналізу форми, виявлення і розпізнавання об'єктів[19]. Функція у даному проекті дозволяє знайти межі об’єктів, що рухаються та виділити їх;
- contourArea() - функція обчислює площу контуру. Площа обчислюється за формулою Гріна;

– boundingRect() - функція обчислює і повертає найменший верхній правий обмежуючий прямокутник для зазначеного набору точок для зображення у відтінках сірого.

Бібліотека NumPy:

– array() – створює масив елементів;
– copy() – повертає копію переданого об’єкта;
– polyfit() - функція NumPy. polyfit () допомагає знайти відповідність полінома найменших квадратів. Це означає знаходження найкращої підходящої кривої для заданого набору точок шляхом мінімізації суми квадратів. Він приймає 3 різних вхідних даних від користувача, а саме X, Y і ступінь полінома;
– average() – обчислює зважене середнє по зазначеній осі;
– zeros_like() - повертає масив нулів тієї ж форми і типу, що і даний масив.

У процесі обробки відео спочатку розбивається на окремі кадри і надалі ми оперуватимемо окремими кадрами. Робота програми продовжується до тих пір поки у відео залишаються кадрі тобто відео не завершене, а також робота програми може бути перервана натисканням клавіші “q”.

Для кожного кадру спочатку перевіряється область, у якій існує небезпека появи небезпечних об'єктів і у разі, якщо такі об'єкти будуть помічені, вони будуть виділені зеленим прямокутником з метою привертання уваги. Задля мінімізації появи хибних спрацювань, втілено декілька інструментів, наприклад, область обмежена фізично, тобто об'єкти з обочин чи географічний рельєф не будуть впливати на результати роботи; для визначеної області створюється маска, яка допомагає виділяти лише найбільш контрастні об'єкти, такі як інші транспортні засоби, а, наприклад, тіні від них не розпізнаються як об'єкти, що мають бути виділені; також присутня фільтрація за розміром, тобто елементи дорожньої розмітки, латки тощо не будуть виділяться для привертання уваги.

На наступному етапі необхідно виділити смугу для руху, для цього кадр спочатку необхідно розмити за допомогою використання алгоритму Гаусса, перевести зображення у монохромний вид та виділити на отриманому зображенні контрастні зони – виділяються контури об’єктів. Після цього на зображення накладається маска, яка обмежує зону інтересу – тобто ту зону, в якій буде проводитися пошук ліній дорожньої розмітки.

Останнім етапом є виділення ліній, які й будуть слугувати обмеженнями для смуги руху. Для цього необхідно з усієї множини отриманих ліній контурів отримати середньозважені – вони й будуть відображати межі дороги. Також задля більшої наглядності додається заливання простору дорожнього полотна. Тепер два зображення накладаються (вхідне та з нанесеними лініями) й ми отримаємо фінальний результат. Паралельно із отриманих кадрів з елементами доповненої реальності збирається відео, яке зберігається і може бути переглянуте згодом.

У даній кваліфікаційній роботі застосовані декілька методів обробки візуальних даних: перетворення по Гаусу, перетворення Хафа, алгоритм Canny, алгоритм Брезенхема, алгоритм Suzuki. Про перше перетворення я згадував раніше тому більш детально опишу роботу інших алгоритмів.

Перетворення Хафа - це метод, який може бути використаний для виділення елементів певної форми в зображенні. Оскільки потрібно, щоб бажані об'єкти були задані в деякій параметричній формі, класичне перетворення Хафа найчастіше використовується для виявлення правильних кривих, таких як лінії, кола, еліпси і т.д. Узагальнене перетворення Хафа може бути використано в додатках, де простий аналітичний опис об'єкта неможливий. Незважаючи на обмеження в області застосування, класичне перетворення Хафа зберігає безліч застосувань, оскільки більшість виготовлених деталей і багато анатомічних частин, досліджених на зображеннях, містять межі об'єктів, які можуть бути описані регулярними кривими. Основна перевага методу перетворення Хафа полягає в тому, що він

некритичний до пробілів в описах меж об'єктів і відносно не схильний до впливу шуму зображення.

Метод Хафа особливо корисний для обчислення глобального опису об'єкта, де кількість класів рішень не обов'язково повинна бути відома апріорі, враховуючи локальні вимірювання, які можуть бути зашумленими. Основна ідея, що лежить в основі методу визначення ліній Хафа, полягає в тому, що кожен вхідний вимір вказує на його внесок у глобально узгоджене рішення.

В якості простого прикладу розглянемо поширену проблему підгонки набору лінійних сегментів до набору дискретних точок зображення. На рисунку 2.4 показані деякі можливі рішення цієї проблеми. Тут відсутність апріорних знань про кількість бажаних відрізків лінії (і двозначність щодо того, що являє собою відрізок лінії) роблять цю проблему недостатньо обмеженою.

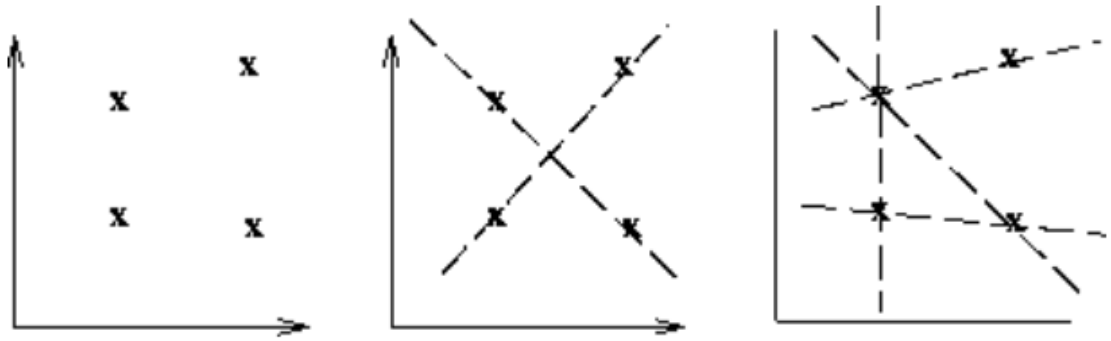

Рис. 2.4.

Алгоритм Кані-це популярний алгоритм виявлення країв. Він був розроблений Джоном Ф. Канні. Перед застосуванням детектора, перетворимо зображення в відтінки сірого, щоб зменшити обчислювальні витрати. Цей етап характерний для багатьох методів обробки зображень. Це багатоступінчастий алгоритм, розглянемо кожен етап більш детально:

– Зниження рівня шуму. Оскільки виявлення країв є чутливим до наявності шуму на зображенні, першим кроком є видалення шуму за допомогою фільтра Гауса розмірністю 5x5;

– Знаходження градієнта інтенсивності зображення. Згладжене зображення фільтрується ядром Собеля [20] як у горизонтальному, так і у вертикальному напрямку, щоб отримати першу похідну в горизонтальному напрямку (Gx) і вертикальному напрямку (Gy). З цих двох зображень ми можемо знайти градієнт краю і напрямок для кожного пікселя, як показано у формулі 2.1;

$$Edge\_Gradient\ (G) = \sqrt{G_x^2 + G_y^2}$$
$$Angle\ (\theta) = \tan^{-1}\left(\frac{G_y}{G_x}\right) \quad (2.1)$$

– Придушення немаксимальних значень. Після визначення величини і напрямку градієнта виконується повне сканування зображення, з метою видалення всіх небажаних пікселів, які можуть не бути частиною краю. Для цього для кожного пікселя перевіряється, чи є піксель локальним максимумом в його околиці в напрямку градієнта. Це продемонстровано на рисунку 2.5, де точка A знаходиться на краю (у вертикальному напрямку). Напрямок градієнта перпендикулярно краю. Точки B і C знаходяться в напрямках градієнта. Таким чином, точка A перевіряється з точками B і C, щоб побачити, чи утворює вона локальний максимум. Якщо це так, то він розглядається для наступного етапу, в іншому випадку він пригнічується (обнуляється);

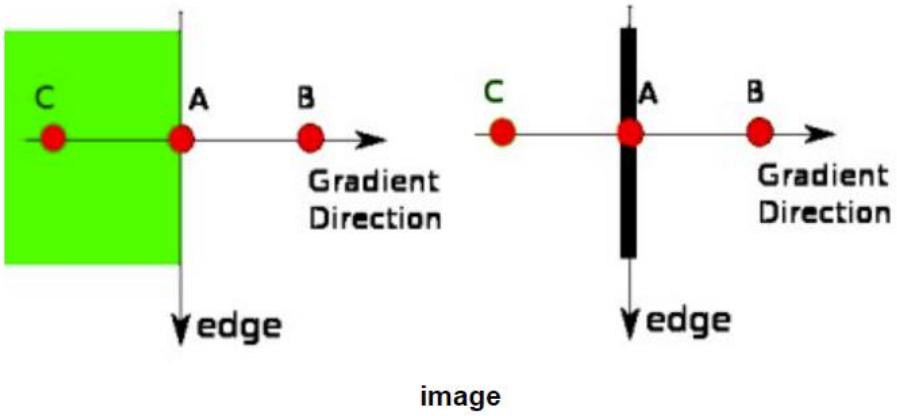

Рис. 2.5.

– Тросування області невизначеності. На цьому етапі вирішується, які ребра дійсно є ребрами, а які ні. Для цього нам потрібні два граничних значення, minVal і maxVal. Будь-які ребра з градієнтом інтенсивності більше maxVal обов'язково будуть ребрами, а ті, що нижче minVal, обов'язково не будуть ребрами, тому відкидаються. Ті, що знаходяться між цими двома пороговими значеннями, класифікуються як або ребра або не ребра в залежності від їх зв'язності. Якщо вони з'єднані з пікселями "впевненого краю", тобто тими, що знаходяться вище maxVal, вони вважаються частиною ребер. В іншому випадку вони також відкидаються. Цей механізм продемонстровано на рисунку 2.6.

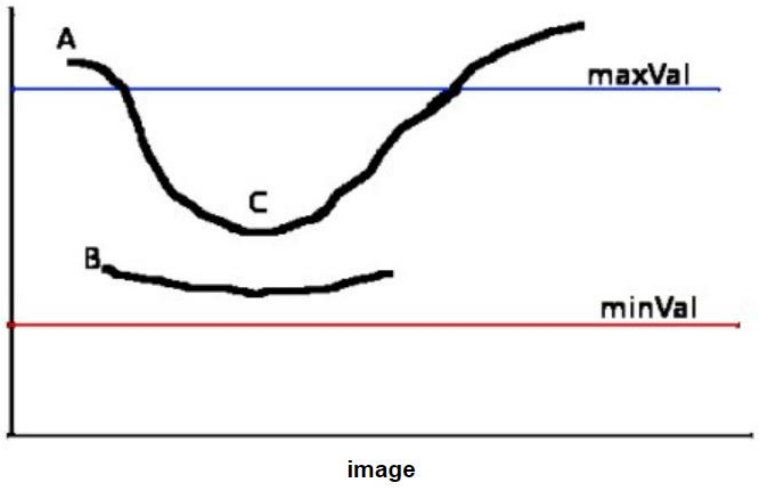

Рис. 2.6.

Алгоритм Брезенхема передбачає, що при обмеженому розширенні точка наближення лінії може перебувати або на самій лінії, або зліва від неї, або справа. Відстань на якій фактично перебуває встановлена точка від шуканої лінії, - помилка при малюванні лінії. Переходячи до кожної наступної точки, помилка буде відображати, яка точка буде більш наближена до кривої, а яка менше.

Принцип алгоритму Брезенхема полягає в тому, щоб з кожною ітерацією рухатися на одну точку по тій осі проекція на яку більше, а по іншій осі зміщення на один піксель відбувається лише тоді, коли лінія відхилилася від поточної осі більш ніж на півпікселя як це показано на рисунку 2.7.

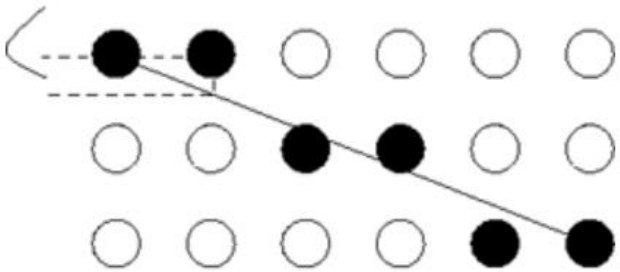

Рис. 2.7.

Алгоритм Судзукі використовується для пошуку меж об’єктів на зображенні[21]. Це реалізація алгоритму виявлення контурів, запропонованого в статті "Топологічний структурний аналіз оцифрованих двійкових зображень за допомогою технології Border Following " Suzuki and Abe 1985. Функція вимагає двійкового вхідного зображення з нульовим заповненням. Вихідними даними є координати контуру, ієрархія контурів і відповідні типи кордонів. Передбачається, що зображення складаються з чотирьох частин.

2.5. Обґрунтування та організація вхідних та вихідних даних програми

В даній кваліфікаційній роботі оброблюються данні відеоряду, вхідні дані отримуються за дапомогою використання методу VideoCapture(), який у тренувальних цілях приймає відео, завантажене локально, проте для прикладного використання достатньо замінити значення на VideoCapture(0) і у якості вхідних данних слугуватиме потокове відео з камери пристрою.

Оптимальним є відео із розширенням 1280:720 оскільки воно дозволяє зберегти високу чіткість при цьому не надто сильно навантажуючи систему, оскільки дана програма вибаглива до апаратних ресурсів. Якщо виконати приблизні розрахунки складності алгоритму для відео з розширенням 1280:720 та відео із рознирення 1920:1080, то ми побачимо різницю у кількості пікселів, які необхідно буде обробити більш ніж у два рази.

У якості вихідних даних виступають зображення, які почергово виводяться на екран і створюють відеоряд із елементами доповненої реальності, а також до вихідних даних відноситься відеофайл, який автоматично записується та фіксує усе, що відбувається на екрані (вхідний відеоряд з елементами доповненої реальності).

2.6. Опис розробленої системи

Для роботи з програмою, необхідно запустити додаток – Python є скриптовою мовою програмування тому достатньо відкрити файл main.py і додаток почне працювати. Після чого програма буде запущена. Важливо звернути увагу на характеристики обчислювальної машини, на якій необхідно запустити застосунок, якщо мінімальні технічні характеристики обладнання на якому було запущено даний додаток не будуть відповідати, користувач не зможе коректно працювати з додатком. Бажане використання персональних комп’ютерів, обладнаних дискретним відеоадаптором для забезпечення стабільної коректної роботи.

2.6.1. Використані технічні засоби

При розробці та тестуванні системи була використана клієнтська персональна ЕОМ з наступними мінімальними характеристиками:

- процесор AMD Ryzen(tm) 5 4500U 2.30 GHz;
- монітор;
- 8000Мб ОЗП;
- 200Мб вільного місяця для тренувальних та тестових відеоданих та самого додатку;
- клавіатура;
- комп’ютерна миша.

2.6.2. Використані програмні засоби

Для своєї інформаційної системи я обрав середовище Visual Studio Code.

Visual Studio Code (рис. 2.8) - це дистрибутив сховища code-OSS з налаштуваннями, специфічними для Microsoft, випущений під традиційною Ліцензією продукту Microsoft.

Visual Studio Code поєднує в собі простоту редактора коду з тим, що потрібно розробникам для їх основного циклу редагування-збірки-налагодження, тобто фактично частково виконує функції інтегрованого середовиза розробки. Він забезпечує всебічну підтримку редагування коду, навігації, а також полегшену відладки коду, обширні опції завдяки великій уількості доступних плагінів та полегшену інтеграцію з існуючими інструментами.

Visual Studio Code щомісяця оновлюється - додаються нові функції та поліпшується виправленнями помилок. Ви можете завантажити його для різних платформ, таких як Windows, macOS та Linux на веб-сайті Visual Studio

Code, а щоб отримувати останні версії кожен день, достатньо встановити збірку Insiders.

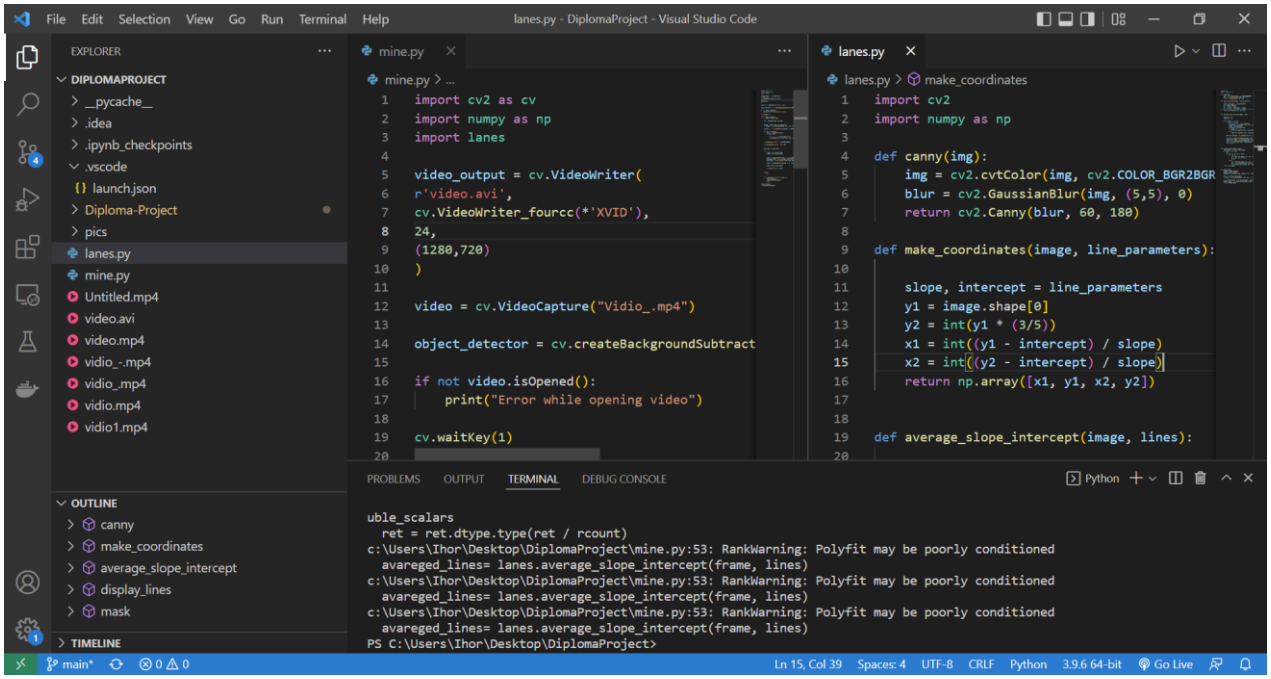

Рис. 2.8 Сторінка розробленого застосунку у Visual Studio Code

Застосунок побудований на Python версії 3.7 , VS Code є зручним рішенням щоб працювати з цією, а також із іншими версіями цієї мови програмування.

Існують також спеціалізовані IDE (PyCharm), проте вони використовують більше системних ресурсів або Jupyter Notebook, проте він має меньший функціонал для дебагінгу а також запуск застосунку дещо ускладнений у порівнянні із звичайним скриптом, саме тому я зупинив свій вибір на VS Code.

Основні функції програмного середовища Visual Studio Code:

- підсвічування вихідного коду з урахуванням синтаксису мови програмування (мова визначається автоматично по розширенню файлу);
- автозавершення слів;
- простий менеджер проектів;
- підтримка плагінів;
- підтримка великої кількості кодувань;

– гнучкий та інтуїтивний інтерфейс;
– інтеграція з GIT;
– інтеграція з Docker.

2.6.3. Виклик та завантаження програми

Для роботи з програмою, досить запустити застосунок. Для цього необхідно запустити скриптовий файл main.py, відкриється вікно з відео і почнеться обробка відеоряду у режимі реального часу.

2.6.4. Опис інтерфейсу користувача

Інтерфейс застосунку мінімалістичний оскільки він передбачає, що водій транспортного засобу під час руху зможе слідкувати за тим, що відбувається на екрані, нічого не має відволікати на себе увагу.

Після запуску додатку з’явиться вікно із відеорядом (рис. 2.9). У якості відео буде виступати або тестовий файл або зображення буде передаватися напряму з камери.

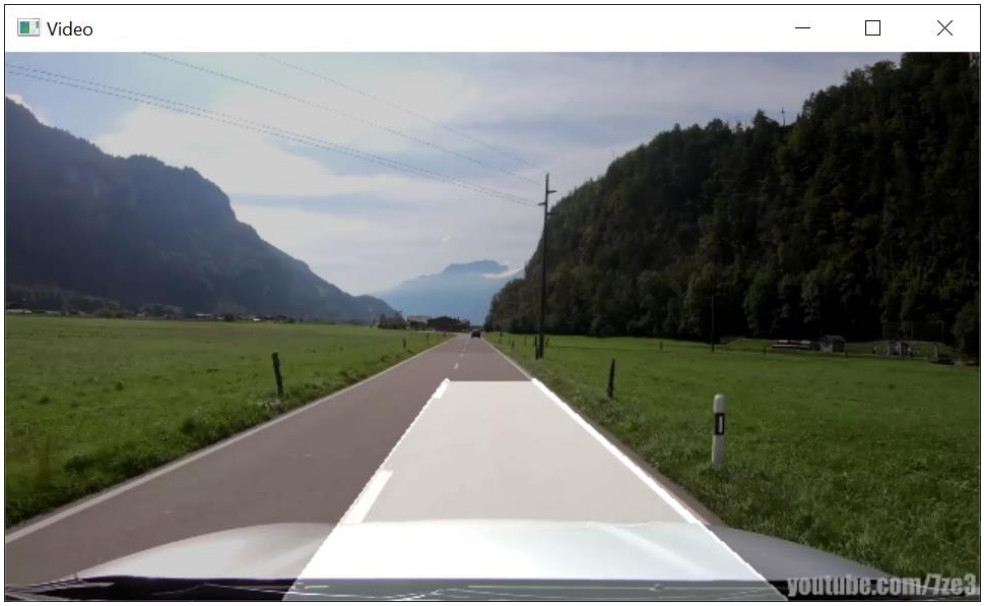

Рис. 2.9 Інтерфейс застосунку

Як можна побачити, у програмі виділяється смуга дороги, якою їде транспортний засіб (рис. 2.9). Якщо на дорозі з’являться об’єкти, що наближаються та можуть створювати потенційну небезпеку і тому потребують більшої концентрації уваги від водія, вони будуть виділені зеленим прямокутником, як показано на рисунку 2.10.

Рис. 2.10 Реакція програми на наближення автомобіля

Після завершення роботи програми, формується файл у форматі «.avi», який містить вхідний відеоряд із накладеними на нього елементами доповненої реальності. Файл формується у спільній директорії програмного засобу.

Під час обробки, зображення проходить декілька етапів перед тим, як формується остаточний результат. На рисунку 2.11 Можна побачити роботу функції Canny, а на рисунку 2.12 продемонстроване подальше накладання маски на зображебння із цим фільтром.

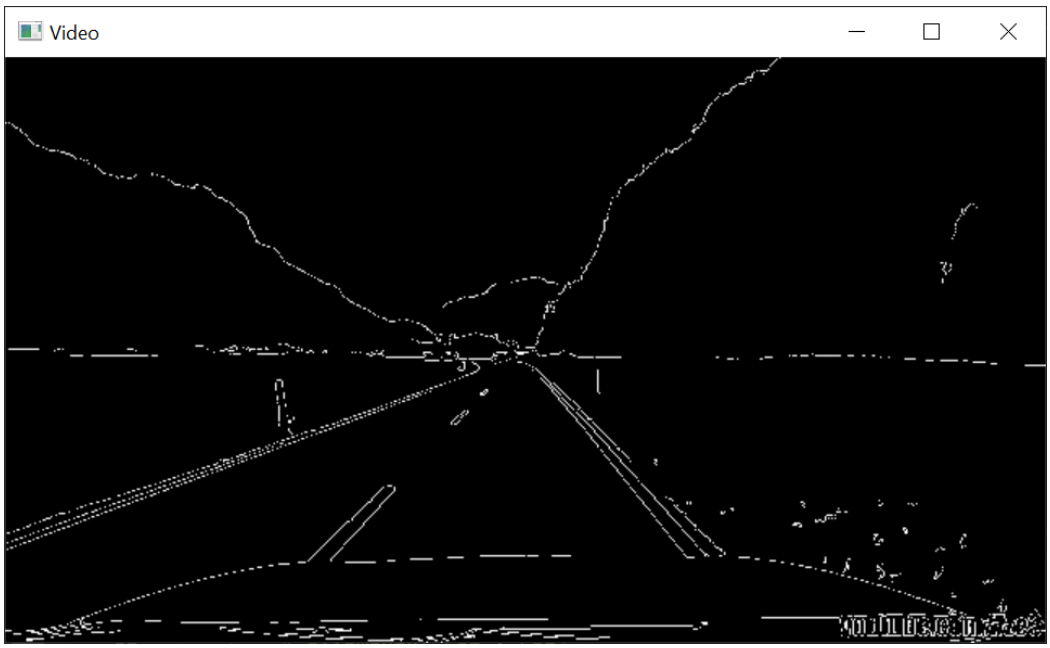

Рис. 2.11. Перетворення Canny

Рис. 2.12 Накладання маски

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Розрахунок трудомісткості та вартості розробки програмного продукту

Початкові дані:

1. передбачуване число операторів програми - 145;
2. коефіцієнт складності програми – 1,4;
3. коефіцієнт корекції програми в ході її розробки – 0,09;
4. годинна заробітна аналітика бази даних – 160 грн/год;

Середня година зарплати розробника на мові програмування Python була вирахувана виходячи з даних «Української спільноти програмістів (DOU)». Середньоукраїнська заробітна плата розробника, який володіє мовою програмування Pyhton з досвідом роботи близько року дорівнює 950 американських доларів на місяць. При курсі валют НБУ на початок червня 2022 року один американський долар дорівнює 29,80 грн, тому середня зарплата в гривнях дорівнює 28310 грн. При восьмигодинному робочому дні (176 годин у місяць в середньому) середня зарплата за годину буде становити 160 грн.

5. коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі – 1,2;
6. коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності – 0,8;
7. вартість машино-години ЕОМ – 15 грн/год

Нормування праці в процесі створення ПЗ істотно ускладнено в силу творчого характеру праці програміста. Тому трудомісткість розробки ПЗ може бути розрахована на основі системи моделей з різною точністю оцінки.

Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = t_o + t_u + t_a + t_n + t_{отл} + t_д, \text{ людино-годин,} \qquad (3.1)$$

де t_o – витрати праці на підготовку й опис поставленої задачі (приймається 50);

t_u– витрати праці на дослідження алгоритму рішення задачі;

t_a– витрати праці на розробку блок-схеми алгоритму;

t_n– витрати праці на програмування по готовій блок-схемі;

$t_{отл}$– витрати праці на налагодження програми на ЕОМ;

t_o– витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у ПЗ, яке розробляється.

Умовне число операторів (підпрограм):

$$Q=q\cdot C\cdot(1+p), \text{ де} \qquad (3.2)$$

q– передбачуване число операторів;

C– коефіцієнт складності програми;

p– коефіцієнт кореляції програми в ході її розробки.

$$Q= 145\cdot 1{,}4\cdot(1+0{,}08)=221;$$

Витрати праці на вивчення опису задачі t_u визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{Q \bullet B}{(75 \ldots 85) \bullet K}, \text{ людино-годин}, \qquad (3.3)$$

де B – коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі;

K – коефіцієнт кваліфікації програміста, обумовлений стажем роботи з даної спеціальності;

$$t_u= \frac{221\cdot 1{,}2}{75\cdot 0{,}8}= 4{,}42 \text{ людино-годин}$$

Витрати праці на розробку алгоритму рішення задачі:

$$t_п = \frac{Q}{(20\ldots25)\cdot k}$$, людино-годин. (3.4)

де Q – умовне число операторів програми;

k – коефіцієнт кваліфікації програміста.

Підставивши відповідні значення в формулу (3.4), людино-годин:

$t_а = \frac{221}{20 \cdot 0{,}8}$ = 13,8 людино-годин.

Витрати на складання програми по готовій блок-схемі:

$$t_п = \frac{Q}{(20\ldots25)\cdot k}$$, людино-годин. (3.5)

$t_а = \frac{221}{20 \cdot 0{,}8}$ = 11,05 людино-годин.

Витрати праці на налагодження програми на ЕОМ:

- за умови автономного налагодження одного завдання:

$$t_{отп} = \frac{Q}{(4..5)\cdot k}$$, людино-годин. (3.6)

$t_п = \frac{221}{5 \cdot 0{,}8} = 55{,}25$ людино-годин,

- за умови комплексного налагодження завдання:

$t_{отл}^{к} = 1{,}2 \cdot t_{отл}$; (3.7)

$t_{отл}^{к} = 1{,}2 \cdot 55{,}25 = 66{,}3$ людино-години

Витрати праці на підготовку документації:

$t_д = t_{др} + t_{до}$; (3.8)

де $t_{др}$ – трудомісткість підготовки матеріалів і рукопису

$t_д = \frac{Q}{(15\ldots20)\cdot K}$; (3.9)

$t_д = \frac{221}{18 \cdot 0{,}8} = 15{,}34$ людино-годин.

$t_{до}$– трудомісткість редагування, печатки й оформлення документації

$t_{до} = 0{,}75 \cdot t_{др}$; (3.10)

$t_{до} = 0{,}75 \cdot 15{,}34 = 11{,}5$ людино-годин.

$t_д = 15{,}34 + 11{,}5 = 26{,}85$ людино-годин.

Отримаємо трудомісткість розробки програмного забезпечення:

t =50+4,42+13,8+11,05+66,3+26,85=172,42, людино-години.

У результаті ми розрахували, що в загальній складності необхідно 172,42 людино-годин для розробки даного програмного забезпечення.

3.2. Розрахунок витрат на створення програми

Витрати на створення ПЗ Кпо включають витрати на заробітну плату виконавця програми Зз/п і витрат машинного часу, необхідного на налагодження програми на ЕОМ.

$$K_{ПО} = З_{ЗП} + З_{МВ}, \text{ грн,} \quad (3.11)$$

де $З_{ЗП}$ – заробітна плата виконавців, яка визначається за формулою:

$$З_{ЗП} = t \cdot C_{ПР}, \text{ грн,} \quad (3.12)$$

де t – загальна трудомісткість, людино-годин;

$C_{ПР}$– середня годинна заробітна плата програміста, грн/година

$З_{ЗП} = 172{,}42 \cdot 160 = 27\,587{,}20$ грн.

$З_{МВ}$– Вартість машинного часу, необхідного для налагодження програми на ЕОМ:

$$З_{МВ} = t_{отл} \cdot C_{М}, \text{ грн,} \quad (3.13)$$

де $t_{отл}$ – трудомісткість налагодження програми на ЕОМ, год.

$C_{МЧ}$– вартість машино-години ЕОМ, грн/год.

$З_{МВ} = 172{,}42 \cdot 15 = 2586{,}3$ грн.

$К_{ПО} = 27\,587{,}20 + 2586{,}3 = 30\,173{,}5$ грн.

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k \cdot F_p}, \text{ мес.} \qquad (3.14)$$

де B_k- число виконавців;

F_p– місячний фонд робочого часу (при 40 годинному робочому тижні Fp=176 годин).

$T = \frac{172,42}{1 \cdot 176} = 1$ міс.

Висновки. На розробку даного програмного забезпечення піде 172,42 людино-годин. Тобто, ймовірна очікувана тривалість розробки складатиме 1 місяць при стандартному 40-годинному робочому тижні і 176-годинному робочому місяці. Очікувані витрати на створення програмного забезпечення складатимуть 30 173,5 грн.

ВИСНОВКИ

В даній кваліфікаційній роботі був розроблений додаток для допомоги водіям при екстрених ситуаціях на дорозі.

Дане програмне забезпечення призначене для відслідковування в режимі реального часу дорожньої обстановки ті її змін. Додаток надасть можливість відслідковувати та позначати на зображені з камери смугу для руху а також об’єкти, що наближаються та можуть створювати небезпеку для руху. Даний програмний продукт має на меті підвищення концентрації уваги водія, а також привернення його уваги до ситуацій, які вимагають від нього негайної реакції у режимі реального часу.

На практиці передбачається, що подібний має позитивно повпливати на безпеку під час руху вцілому, а також зменшити кількість ДТП, які відбуваються через зниження концентрації уваги водія.

Під час виконання даного кваліфікаційної роботи були виконані наступні задачі:

- проаналізовано предметну область задачі, що розв’язується;
- з’ясовані вимоги та підстави для розробки даної інформаційної системи;
- обрано раціональну структуру і параметри програми;
- описана платформа, яка дозволить реалізувати функціонал застосунку;
- створено інтерфейс для аналізу вхідних відеоданих;
- розроблено рекомендації щодо використання програми.

Програмний продукт створено за допомогою мови Python у середовищі Visual Studio Code. Механізм аналізу даних реалізовано за допомогою стандартної бібліотеки мови програмування Python, функціонал машинного зору додатку був створений за допомогою Python-бібліотек OpenCV та NumPy.

Також у кваліфікаційній роботі було визначено трудомісткість розробленого програмного продукту (172,42 люд-год), проведений підрахунок вартості роботи по створенню програми (30 173,5) грн. та розраховано час на його створення (1 міс).

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Englisch Ausgabe. Applied Deep Learning and Computer Vision for Self-Driving Cars: Build autonomous vehicles using deep neural networks and behavior-cloning techniques. Packt Publishing, 2020. – 32 p.
2. Imran Ahmad. 40 Algorithms Every Programmer Should Know: Hone your problem-solving skills by learning different algorithms and their implementation in Python, 1st Edition. - Packt Publishing, 2020. - 84p.
3. Englisch Ausgabe. OpenCV with Python: A basic approach. Independently published, 2020. – 17 p.
4. Michael Walker. Python Data Cleaning Cookbook: Modern techniques and Python tools to detect and remove dirty data and extract key insights, 1st Edition. - Packt Publishing, 2020. - 136p.
5. Shamshad Ansari. Building Computer Vision Applications Using Artificial Neural Networks: With Step-by-Step Examples in OpenCV and TensorFlow with Python. 1st Eddition – Apress. 174 p.
6. Felix Zumstein. Python for Excel: A Modern Environment for Automation and Data Analysis 1st Edition. - O'Reilly Media, 2021. - 265p.
7. Hafsa Asad, Vishwesh Ravi Shrimali, Nikhil Singh. The Computer Vision Workshop: Develop the skills you need to use computer vision algorithms in your own artificial intelligence projects. Packt Publishing, 2020. – 56 p.
8. David Mertz. Cleaning Data for Effective Data Science: Doing the other 80% of the work with Python, R, and command-line tools, 1st Edition. - Packt Publishing, 2021. - 498p.
9. Richard Szeliski. Computer Vision: Algorithms and Applications (Texts in Computer Science). Springer, 2011. - 532 p.
10. Richard Hartley. Multiple View Geometry in Computer Vision, 2nd Edition. Cambridge University Press, 2004. – 67 p.
11. David Forsyth. Computer Vision: A Modern Approach, 2nd Edition. Pearson, 2011 – 12 p.
12. Jesus Sanos. Python Programming. Springer, 2012. – 46 p.

13. Al Sweigart. Automate the Boring Stuff with Python, 2nd Edition, -No Starch Press, 2021. – 621 p.

14. Jaime Buelta. Python Automation Cookbook: 75 Python automation ideas for web scraping, data wrangling, and processing Excel, reports, emails, and more, 2nd Edition. - Packt Publishing, 2020. - 280p.

15. Loginom «Логістична регресія і ROC-аналіз - математичний апарат». [Електронний ресурс]. Режим доступу: https://loginom.ru/blog/logistic-regression-roc-auc

16. «Перетворення Хафа». [Електронний ресурс]. Режим доступу: https://homepages.inf.ed.ac.uk/rbf/HIPR2/hough.htm

17. OpenCV «Canny Edge Detection». [Електронний ресурс]. Режим доступу: https://docs.opencv.org/3.4/da/d22/tutorial_py_canny.html

18. Jim Frost. Regression Analysis: An Intuitive Guide for Using and Interpreting Linear Models, 1st Edition. - Statistics By Jim Publishing, 2020. -148 p.

19. Jim Frost. Introduction to Statistics: An Intuitive Guide for Analyzing Data and Unlocking Discoveries, 1st Edition. - Statistics By Jim Publishing, 2020. -176 p.

20. Stephen Klosterman. Data Science Projects with Python: A case study approach to successful data science projects using Python, pandas, and scikit-learn, 1st Edition. - Packt Publishing, 2019. - 71p.

21. Hyatt Saleh. The Machine Learning Workshop: Get ready to develop your own high-performance machine learning algorithms with scikit-learn, 1st Edition. - Packt Publishing, 2020. - 209p.

ДОДАТОК А

КОД ПРОГРАМИ

operations.py

```
import cv2
import numpy


def mark_coordinates(image, line_parameters):
    """
    The function marks the coordinates of the edge points
    of objects - road marking on this case
    """

    side, interception = line_parameters
    y1 = image.shape[0]
    y2 = int(y1 * (3/5))
    x1 = int((y1 - interception) / side)
    x2 = int((y2 - interception) / side)
    res = numpy.array([x1, y1, x2, y2])
    return res

def canny_processing(img):
    """
    The function applies Canny processing to the picture
    so as a result we receive a monochrome image with marked
    of all objects on it
    """

    img = cv2.cvtColor(
```

```
        img,
        cv2.COLOR_BGR2BGRA)

    blur = cv2.GaussianBlur(
        img,
        (5,5),
        0)

    res = cv2.Canny(
        blur,
        60,
        180)

    return res

def average_line(image, lines):
    """
    The function receives the array of lines and
    returnes avareged lines that represent the roadmarking
    """

    l_fit = []
    r_fit = []

    while lines is not None:
        for line in lines:
            x1, y1, x2, y2 = line.reshape(4)

            parameters = numpy.polyfit(
                (x1, x2),
```

```
            (y1, y2),
            1)

        side = parameters[0]
        interception = parameters[1]
        if side < 0:
            l_fit.append((side, interception))
        else:
            r_fit.append((side, interception))

    l_fit_average = numpy.average(
        l_fit,
        axis=0)

    l_line = mark_coordinates(
        image,
        l_fit_average)

    r_fit_average = numpy.average(
        r_fit,
        axis=0)

    r_line = mark_coordinates(
        image,
        r_fit_average)

    res = numpy.array([l_line, r_line])

    return res
```

```
def show_lines(image, lines):
    """
    The function is made for drawing the lines according to the
    roadmarking, also it fills the area between the lines
    wuth solid color
    """

    linear_image = numpy.zeros_like(image)
    if lines is not None:
        i=1
        for x1, y1, x2, y2 in lines:
            if i == 1:

                cv2.line(
                    linear_image,
                    (x1, y1),
                    (x2, y2),
                    (255, 255, 255),
                    8)

                pl = [x1, y1, x2, y2]
                i += 1
            else:

                cv2.line(
                    linear_image,
                    (x1, y1),
                    (x2, y2),
                    (255, 255, 255),
```

```
                8)

            parm = numpy.array(
                [[
                    [pl[0],
                    pl[1]],
                    [pl[2],
                    pl[3]],
                    [x2, y2],
                    [x1, y1]]],
                dtype=numpy.int32)

            cv2.fillPoly(
                linear_image,
                parm,
                (202, 205, 191),
                lineType=8,
                shift = 0,
                offset = None)

    return linear_image


def mask(pict):
    """
    The functions applies mask to the
    input frame and returnes a modified picture
    """
    height = pict.shape[0]
    width= pict.shape[1]
```

```
msk_area = numpy.array(
    [(550, height//1.7),
    (740, height//1.7),
    (850, 600),
    (350, 600)])

msk = numpy.zeros_like(pict)

_, msk = cv2.threshold(
    msk,
    254,
    255,
    cv2.THRESH_BINARY)

cv2.fillPoly(
    msk,
    numpy.array(
        [msk_area],
        dtype=numpy.int64),
    1024)

masked_pict = cv2.bitwise_and(
    pict,
    msk)
return masked_pict
```

mine.py

```
import cv2
import numpy
import operations

def main():
    """
    The main function of the programm
    that provides all actions in it
    """

    video_output = cv2.VideoWriter(
        r'video.avi',
        cv2.VideoWriter_fourcc(*'XVID'),
        24,
        (1280,720)
        # Preparations for writting output video
    )

    video = cv2.VideoCapture("Vidio_.mp4")
    # Capturing the videofile

    car_detector = cv2.createBackgroundSubtractorMOG2(
        history=100,
        varThreshold=95
        # Setting preferrences for car detection function
        )

    if not video.isOpened():
```

```
    print("Error while opening video occurred")
    # Error openning video processing

cv2.waitKey(1)

while video.isOpened():
    _, frame = video.read()
    # Readding frames from video

    roi = frame[
        300:500,
        450:650]
    # Setting the region of interest for detecting cars

    masked = car_detector.apply(roi)
    # Applying the object detection to the region of interest

    _, masked = cv2.threshold(
        masked,
        254,
        255,
        cv2.THRESH_BINARY)
        # Setting properties for threshold function

    road_contours, _ = cv2.findContours(
        masked,
        cv2.RETR_TREE,
        cv2.CHAIN_APPROX_SIMPLE)
        # The function finds the countors on the road
```

```
for cnt in road_contours:
    area = cv2.contourArea(cnt)
    if area > 900:
        # For the zone of a set area we apply a rectangle bounding

        x, y, w, h = cv2.boundingRect(cnt)

        cv2.rectangle(
            roi,
            (x,y),
            (x+w, y+h),
            (0, 255, 0),
            3)



cv2.namedWindow(
    "Drivers assistant",
    cv2.WINDOW_NORMAL)

cv2.resizeWindow(
    "Drivers assistant",
    640,
    360)
    # Set the size of the window

copy_frame= numpy.copy(frame)
# Save a copy of the frame to use it as background

try:
    frame= operations.canny_processing(frame)
```

```
frame = operations.mask(frame)

lines= cv2.HoughLinesP(
    # Getting the lines from the roadmarking
    frame,
    2,
    numpy.pi/180,
    100,
    numpy.array([()]),
    minLineLength=10,
    maxLineGap=2)

avareged_lines= operations.average_line(
    frame,
    lines)

linear_image = operations.show_lines(
    copy_frame,
    avareged_lines)

result= cv2.addWeighted(
    # Combining the two images
    copy_frame,
    0.8,
    linear_image,
    0.5,
    1)

video_output.write(result)
# Writting frames to the result file that will be saved
```

```
            cv2.imshow(
                # Showing the final result
                "Drivers assistant",
                result)

        except:
            pass

        if cv2.waitKey(3) & 0xFF == ord('q'):
            # The block is used to stop displaying video
            video.release()
            cv2.destroyAllWindows()
            break

    video.release()
    # Deleating video fron RAM
    cv2.destroyAllWindows()
    # Closing all windows with video

if __name__ == "__main__":
    # Launching the main function of the programm
    main()
```

launch.json

```
{
   // json file generated by the environment
   // Use IntelliSense to learn about possible attributes.
   // Hover to view descriptions of existing attributes.
   // For more information, visit: https://go.microsoft.com/fwlink/?linkid=830387
   "version": "0.2.0",
   "configurations": [
      {
         "name": "Python: Current File",
         "type": "python",
         "request": "launch",
         "program": "${file}",
         "console": "integratedTerminal",
         "justMyCode": true
      }
   ]
}
```

ДОДАТОК Б

ВІДГУК
керівника економічного розділу
на кваліфікаційну роботу бакалавра
на тему:
«Розробка інтелектуальної системи
допомоги водіям при екстрених ситуаціях на дорозі»
студента групи 122-18-1 Форостіна Ігоря Олександровича

Керівник економічного розділу
доцент каф. ПЕП та ПУ, к.е.н **Л. В. Касьяненко**

ДОДАТОК В

Перелік файлів на диску

Перелік документів на магнітному носії

Ім’я файлу	Опис
Пояснювальні документи	
Кваліфікаційна робота Форостін.docx	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
Кваліфікаційна робота Форостін.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF
Програма	
Форостін.rar	Архів. Містить коди програми і скомпільовану програму
Презентація	
Форостін.ppt	Презентація кваліфікаційної роботи

ВІДГУК

на дипломний проект бакалаврана тему:
«Розробка інтелектуальної системи
допомоги водіям при екстрених ситуаціях на дорозі»
студента групи 122-18-1 Форостіна Ігоря Олександровича

Розроблене в кваліфікаційній роботі програмне забезпечення призначене для створення інформаційної системи, яка спроможна виконувати аналіз відеоряду та на його основі виконувати розпізнавання об’єктів на ньому та виділяти їх візуально.

Актуальність розробленого програмного продукту полягає в створенні такого додатку, що дозволяє підвищити безпеку на дорозі завдяки зосередженню уваги на дорожній обстановці.

Перевага запропонованої програми в тому, що вона обробляє відеоряд і у режимі реального часу виводить на екран зображення з камери з елементами доповненої реальності.

Для вирішення поставлених задач при розробці додатку були здійснені наступні дії:

1. Розробка логічної моделі програми.
2. Проектування та розробка програмного забезпечення.
3. Розробка простого й зрозумілого мінімалістичного інтерфейсу програми.

Додаток розроблений на мові Python у середовищі VS Code, а для роботи з потоковим відео була використана бібліотека OpenCV.

Практична значимість створення даного програмного продукту полягає в можливості змусити водіїв більш оперативно реагувати на зміни у дорожній обстановці аби зменшити ризики, пов’язані із загрозами на дорозі.

Працездатність представленої програми підтверджена налагоджувальними випробуваннями та тестуванням програми.

У економічному розділі визначено трудомісткість розробленого додатку, проведений підрахунок вартості роботи по створенню програми та розраховано час на його створення.

Тема кваліфікаційної роботи безпосередньо пов’язана з об’єктом діяльності бакалавра за напрямом підготовки 122 Комп’ютерні науки.

Оформлення пояснювальної записки до кваліфікаційної роботи викано відповідно до стандартів на програмну документацію.

В кваліфікаційній роботі були виявлені недоліки у оформленні, але вона виконана самостійно та заслуговує оцінки 95 балів «відмінно», а студенту Форостіну Ігорю Олександровичу присвоєння йому кваліфікації бакалавра за спеціальністю «фахівець з розробки та тестування програмного забезпечення».

Керівник кваліфікаційної роботи

доц. каф. ПЗКС, к.т.н. Удовик І.М.

РЕЦЕНЗІЯ
на дипломний проект бакалавра
на тему:
«Розробка інтелектуальної системи
допомоги водіям при екстрених ситуаціях на дорозі»
студента групи 122-18-1 Форостіна Ігоря Олександровича

Кваліфікаційна робота на тему «Розробка інтелектуальної системи допомоги водіям при екстрених ситуаціях на дорозі» виконана в повному обсязі, відповідно до технічного завдання.

Мета кваліфікаційної роботи: створення додатку що допоможе водіям реагувати на екстрені події на дорозі, у разі виникнення аварійної ситуації або з'їзду зі смуги руху за рахунок використання технологій комп'ютерного зору.

У пояснювальній записці розглянуто необхідність створення і сфера застосування розробленої, виконано постановку завдання, опис вхідних і вихідних даних, розроблено логічну структуру програми, розроблено інформаційне забезпечення системи, наведені загальні відомості про додаток та його функціональне призначення, зазначені використовувані технічні засоби, визначені джерела, використані при розробці.

Для реалізації інформаційної системи була використана мова програмування Python.

Вважаю завдання і зміст дипломного проекту відповідним для перевірки ступеня підготовленості Форостіна І.О. за напрямом 122 «Комп'ютерні науки».

Список літератури, наведений в роботі, налічує більше 20 джерел, що свідчить про вміння автора працювати з літературою та іншими джерелами інформації. Якість оформлення дипломного проекту можна визнати задовільною, він супроводжується достатньою кількістю рисунків. В роботі присутні заключні висновки. Працездатність цієї інформаційної системи підтверджується експлуатаційними випробуваннями.

Рівень теоретичної та практичної підготовки автора, логіка і стиль викладу матеріалу в цілому відповідає кваліфікаційним вимогам.

Дипломний проект виконаний самостійно та заслуговує оцінки «відмінно», а студент Форостін Ігор Олександрович присвоєння йому кваліфікації бакалавра за спеціальністю «фахівець з розробки та тестування програмного забезпечення».

Рецензент кваліфікаційної роботи
к.т.н., доцент каф. ПЗКС