Вінницький національний технічний університет
Факультет інформаційних технологій та комп’ютерної інженерії
Кафедра комп’ютерних наук

Пояснювальна записка
до магістерської кваліфікаційної роботи

на тему «Інформаційна технологія розпізнавання зображень дорожніх знаків на основі дескрипторів»

Виконав: студент 2 курсу,
групи 1КН-19 м
спеціальності 122 «Комп'ютерні науки»
Василенко М. Ю.

Керівник: канд.техн.наук, доцент
Колесницький О. К.

Рецензент: канд.техн.наук, доцент
Рейда О. М.

Вінниця
2020 рік

ЗАТВЕРДЖУЮ
Завідувач кафедри____КН____
д.т.н., проф.. Яровий А.А.

_____
(підпис)
“____” _____2020 року

ЗАВДАННЯ

на магістерську кваліфікаційну роботу на здобуття кваліфікації магістра зі спеціальності: 122 – «Комп'ютерні науки»

08-22.МКР.003.19.000.ПЗ

Магістранта групи 1КН-19м Василенка Миколи Юрійовича

Тема магістерської кваліфікаційної роботи: «Інформаційна технологія розпізнавання зображень дорожніх знаків на основі дескрипторів»

Вхідні дані: зображення на якому виконується пошук з роздільною здатністю не менше 800 х 600; зображення яке потрібно знайти з роздільною здатністю не менше 300 х 300, використання мови програмування високого рівня C#, та бібліотеки OpenCV.

Короткий зміст частин магістерської кваліфікаційної роботи:

1. Графічна: алгоритм роботи додатку розпізнавання дорожніх знаків, зображення Гауссової різниці, UML-діаграма класів додатку розпізнавання дорожніх знаків, стартове вікно і вхідне зображення, вихідне зображення.
2. Текстова (пояснювальна записка): вступ, аналіз предметної області розпізнавання дорожніх знаків, розробка інформаційної технології розпізнавання дорожніх знаків за допомогою дескрипторів, програмна реалізація інформаційної технології розпізнавання дорожніх знаків за допомогою дескрипторів, економічна частина, висновки, перелік використаних джерел, додатки.

КАЛЕНДАРНИЙ ПЛАН ВИКОНАННЯ МКР

№ етапу	Назва етапу	Термін виконання		Очікувані результати
		початок	кінець	
1	Аналіз сучасного рівня розвитку інформаційних технологій розпізнавання дорожніх знаків. Постановка задач дослідження	1.5.20	10.6.20	Аналітичний огляд літературних джерел, задачі досліджень, розділ 1 ПЗ
2	Розробка методу та інформаційної технології розпізнавання дорожніх знаків	1.9.20	20.9.20	Метод, інформаційна технологія, розділ 2
3	Програмна реалізація розробленої інформаційної технології, тестування та оцінка параметрів	25.9.20	5.10.20	Програмне забезпечення, розділ 3
4	Підготовка економічної частини	6.10.20	9.10.20	розділ 4
5	Апробація та/або впровадження результатів дослідження	15.10.20	30.10.20	тези доповідей/наукова стаття
6	Оформлення пояснювальної записки, графічного матеріалу та презентації	1.9.20	5.11.20	Пояснювальна записка, графічний матеріал, презентація

Консультанти з окремих розділів магістерської кваліфікаційної роботи

1. Науковий керівник _____ канд. техн. наук, доц., доц. кафедри КН
(підпис) наук. ступінь, вчене звання (посада)

"____" _____20___ р. _____О. К. Колесницький_____
ініціали та прізвище

2. Економічна частина _____ канд. ек. наук, доц. кафедри ЕПВМ
(підпис) наук. ступінь, вчене звання (посада)

_____ М. В. Бальзан _____
ініціали та прізвище

"____" _____20___ р.

Дата попереднього захисту роботи "____" _____20___ р.

Рецензент _____ канд. техн. наук, доц., доц. кафедри ПЗ
(підпис) наук. ступінь, вчене звання (посада)

_____ О. М. Рейда _____
ініціали та прізвище

Завдання видав
науковий керівник _____ канд. техн. наук, доц., доц. кафедри КН
(підпис) наук. ступінь, вчене звання (посада)

_____О. К. Колесницький _____
ініціали та прізвище

"____" _____20___ р.

Завдання отримав магістрант _____ _____ М.Ю.Василенко_____
(підпис) ініціали та прізвище

"____" _____20___ р.

РЕФЕРАТ

Магістерська кваліфікаційна робота присвячена розробці інформаційної технології розпізнавання дорожніх знаків за допомогою дескрипторів. Дана технологія забезпечує автоматизоване розпізнавання знаків по шляху руху автомобіля, що дає змогу підвищити безпеку на проїжджій частині. В ході роботи проведено аналіз предметної області розпізнавання зображень, зокрема дорожніх знаків. Розглянуто аналоги для розпізнавання знаків. Удосконалено роботу SIFT дескрипторів таким чином, щоб покращити якість розпізнавання. Розроблено інформаційну технологію розпізнавання дорожніх знаків на основі SIFT дескриптора та реалізовано на мові програмування C# програмний засіб для розпізнавання знаків за допомогою дескрипторів.

ABSTRACT

The master's qualification work is devoted to the development of information technology for recognizing road signs using descriptors. This technology provides automated recognition of signs along the path of the car, which allows to increase safety on the roadway. In the course of the work the analysis of the subject area of image recognition, in particular road signs, was carried out. Analogues for character recognition are considered. Improved the performance of SIFT descriptors in such a way as to improve the quality of recognition. The information technology of recognition of road signs on the basis of the SIFT descriptor is developed and the software for recognition of signs by means of descriptors is realized in programming language C #.

ЗМІСТ

ВСТУП

Актуальність теми досліджень. З розвитком обчислювальної техніки стало можливим вирішити ряд завдань, що виникають в процесі життєдіяльності, полегшити, прискорити, підвищити якість результату. Наприклад, робота різних систем життєзабезпечення, взаємодія людини з комп'ютером, поява роботизованих систем тощо. Тим не менш, відзначимо, що забезпечити задовільний результат у деяких завданнях (розпізнавання об’єктів, зокрема дорожніх знаків, яке може бути під певним кутом до камери) за допомогою стандартних комп’ютерних засобів не вдається. Сучасним корпораціям потрібно якісно і без великих затрат часу розпізнавати людину для виконання певних маніпуляцій, наприклад перевірка рахунку в банку, в охоронних системах на контрольно пропускних пунктах і навіть використовується спецслужбами різних країн для пошуку злочинців. Тому розробка програмного забезпечення для розпізнавання дорожніх знаків є дуже актуальною.

Розпізнавання дорожніх знаків являє собою дуже складну задачу в теоретичному та практичному сенсі, хоча люди справляються з нею швидко і доволі точно. Досить важко створити штучну систему і технічно її реалізувати для того щоб ефективно виконувати даний процес. Під розпізнаванням образів розуміють співставлення властивостей об’єкта зображенню образу об’єкта. Прикладом розпізнавання образів можуть являтися системи розпізнавання обличчя, тексту так і окремих символів, біометричних параметрів людини, штрих-кодів, номерів машин і т.д.

Розпізнавання дорожніх знаків використовується або може бути використано для оптимізації роботи в багатьох сферах. Наприклад в системах автопілотів автомобілів у майбутньому чи для допомоги при їзді водіїв які нещодавно отримали водійське посвідчення.

Але дана галузь почала швидко розвиватись в першу чергу завдяки розробці автоматичних технологій сегментації, великої потужності.

І лише за останні роки вдалося досягти хороших результатів у вирішенні проблеми автоматизованої розпізнавання об’єктів. Причиною цього є скачок у сфері апаратного забезпечення.

Саме тому дуже доцільною є розробка інформаційної технології, яка вирішує задачу розпізнавання дорожніх знаків.

Зв'язок роботи з науковими програмами, планами, темами. Магістерська робота виконана відповідно до напрямку наукових досліджень кафедри комп’ютерних наук Вінницького національного технічного університету 22 К1 «Моделі, методи, технології та пристрої інтелектуальних інформаційних систем управління, економіки, навчання та комунікацій» та плану наукової та навчально-методичної роботи кафедри.

Мета та завдання дослідження. Метою дослідження магістерської кваліфікаційної роботи є підвищення достовірності розпізнавання дорожніх знаків програмними засобами за рахунок застосування дескрипторів. Для досягнення поставленої мети слід розв’язати такі завдання:

- розглянути та проаналізувати існуючі програмні реалізації розв’язання задачі розпізнавання дорожніх знаків;
- запропонувати математичну модель для інформаційної технології розпізнавання дорожніх знаків;
- навести стадії інформаційної технології та на їх основі розробити структуру та алгоритм роботи програмного засобу;
- виконати програмну реалізацію запропонованої інформаційної технології розпізнавання дорожніх знаків;
- провести тестування програмного продукту та виконати аналіз отриманих результатів.

Об’єкт дослідження – процес комп’ютеризованого розпізнавання дорожніх знаків.

Предмет дослідження – інформаційна технологія та програмні засоби розпізнавання дорожніх знаків з використанням дескрипторів та достовірність їх роботи.

Методи дослідження. У роботі використані такі методи наукових досліджень: методи оброблення цифрової інформації, теорія масштабно-інваріантної трансформація ознак для реалізації інформаційної технології розпізнавання дорожніх знаків, методи математичної статистики для обрахунків результатів отриманих за допомогою програмного засобу, програмування на мовах високого рівня.

Наукова новизна одержаних результатів полягає в наступному:

1. Набула подальшого розвитку інформаційна технологія розпізнавання дорожніх знаків, яка відрізняється використанням дескрипторів, що дозволило підвищити достовірність розпізнавання дорожніх знаків.

2. Вдосконалено метод формування дескрипторів, який відрізняється введенням нормалізації дескриптора та відсікання великих значень яскравості по порогу, що дозволило зменшити вплив різних умов освітленості та підвищити достовірність розпізнавання дорожніх знаків.

Практичне значення одержаних результатів:

1. Розроблено алгоритм розпізнавання дорожніх знаків з використанням дескрипторів.

2. Розроблено програмний продукт розпізнавання дорожніх знаків з використанням бібліотеки комп’ютерного зору що використовує дескриптори.

Достовірність теоретичних положень магістерської кваліфікаційної роботи підтверджується строгістю постановки задач, коректним застосуванням математичних методів під час доведення наукових положень, строгим виведенням аналітичних співвідношень, порівнянням результатів з відомими, та збіжністю результатів математичного моделювання з результатами, що отримані під час впровадження розроблених програмних засобів.

Особистий внесок здобувача. Усі результати, наведені у магістерській кваліфікаційній роботі, отримані самостійно.

Апробація результатів роботи. Результати досліджень апробовані на XLIX науково-технічній конференції підрозділів ВНТУ [1].

Публікації. За результатами досліджень опубліковано одні тези доповіді на науково-технічній конференції [1] та подано заявку на реєстрацію авторського права на твір – комп'ютерна програма «Нейромережевий модуль розпізнавання дорожніх знаків».

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ РОЗПІЗНАВАННЯ ДОРОЖНІХ ЗНАКІВ

1.1 Аналіз відомих методів розпізнавання дорожніх знаків

У даній кваліфікаційній роботі вирішується задача розпізнавання дорожніх знаків. Актуальність завдання обумовлена тим, що з кінця XX століття індивідуальний транспорт за рахунок покращення засобів виготовлення став широко розповсюджуватися по відносно доступним цінам і навіть тоді більшість автомобілів змогли розвивати швидкості більше ста кілометрів на годину [2, 3]. Дані фактори у свою чергу призвели до таких проблем як перенасичення автотранспортних магістралей у великих містах, збільшення кількості дорожньо-транспортних пригод, та загальній напруженості для водія. В таких обставина не кожен водій зможе швидко орієнтуватися, слідкувати за ситуацією на дорозі та велику кількість знаків, а отже ймовірність дорожньо-транспортної пригоди збільшується, тому задача розпізнавання дорожніх знаків стає все більш актуальною в наш час [4].

На сьогоднішній момент розповсюдженні наступні методи розпізнавання.

Оптичне розпізнавання образів. Застосовується метод перебору виду об'єкта під різними кутами, масштабами, зміщеннями. Для букв потрібно перебирати шрифт, властивості шрифту.

Другий підхід - знайти контур об'єкта і досліджувати його властивості (зв'язність, наявність кутів і т. Д.)

Ще один підхід - використовувати штучні нейронні мережі. Цей метод вимагає або великої кількості прикладів завдання розпізнавання (з правильними відповідями), або спеціальної структури нейронної мережі, яка враховує специфіку даного завдання.

Процес розпізнавання дорожніх знаків можна умовно поділити на кілька етапів:

1. Сегментація – це процес поділу цифрового зображення на декілька сегментів. Мета сегментації полягає в спрощенні і зміні представлення зображення, щоб його було простіше і легше аналізувати [5]. Сегментація зображень зазвичай використовується для того, щоб виділити об'єкти і кордони (лінії, криві, і т. д.) на зображеннях. Більш точно, сегментація зображень – це процес присвоєння таких міток кожному пікселю зображення, що пікселі з однаковими мітками мають загальні візуальні характеристики;

2. Виділення на зображені простих геометричних фігур, таких як коло, квадрат чи трикутник;

3. Подальше уточнююче розпізнавання зображень на простих фігурах.

Проаналізувавши схожі дослідницькі роботи було зроблено висновок, що основною проблемою, яка впливає на якість колоризації, на даний момент є незадовільні погодні умови, нічний або вечірній час та забрудненість знаків. Для вирішення даних проблеми доцільно використати некласичну кольорову модель [6].

1.2 Аналітичний огляд аналогів

Однією з основних причин дорожньо-транспортних пригод з тяжкими наслідками є перевищення швидкості. Система розпізнавання дорожніх знаків покликана попереджати водіїв про необхідність дотримання швидкісного режиму. Дана система визначає дорожні знаки обмеження швидкості при їх проїзді і нагадує водієві поточну максимальну дозволену швидкість, якщо він рухається швидше.

Систему розпізнавання дорожніх знаків (Traffic Sign Recognition, TSR) мають в своєму активі багато відомих автовиробників - Audi, BMW, Ford, Mercedes-Benz, Opel, Volkswagen (Рисунок 1.1). Система

розпізнавання дорожніх знаків на автомобілях Opel входить до складу системи Opel Eye (разом з системою Lane Departure Warning) [7]. Система Opel Eye відзначена в числі кращих розробок в області автомобільної безпеки 2010 року. Mercedes-Benz назвав свою систему Speed Limit Assist (система контролю обмеження швидкості), Volvo - Road Sigh Information, RSI (система інформування про дорожні знаки) [8].

Рисунок 1.1 – Приклад роботи системи розпізнавання знаків

Застосовувані на автомобілях системи розпізнавання дорожніх знаків мають типову конструкцію, яка включає відеокамеру, блок управління і засіб виведення інформації.

Відеокамера розташовується на вітровому склі за дзеркалом заднього виду. Камера знімає простір перед автомобілем в зоні розташування дорожніх знаків (праворуч і зверху по ходу руху) і передає зображення в електронний блок керування. Відеокамера також використовується іншими системами активної безпеки - системою виявлення пішоходів, системою допомоги руху по смузі.

Електронний блок управління реалізує наступний алгоритм роботи:

– розпізнавання форми дорожнього знака (кругла форма);
– розпізнавання кольору знака (червоний колір на білому);
– розпізнавання написи (величина швидкості);

- розпізнавання інформаційної таблички (вид транспорту, час дії, зона дії);
- аналіз фактичної швидкості автомобіля;
- порівняння швидкості автомобіля з максимально допустимою швидкістю;
- візуальне і звукове попередження водія при відхиленні

Зображення у вигляді знаку обмеження швидкості виводиться на дисплей комбінації приладів або дисплей інформаційної системи і залишається видимим, поки обмеження не закінчиться або буде змінено. На автомобілях, обладнаних інформаційним дисплеєм, зображення виводиться на лобове скло.

У ряді конструкцій система розпізнавання дорожніх знаків взаємодіє з навігаційною системою і використовує відомості про знаки обмеження швидкості з навігаційних карт. Навіть якщо символ не буде визначено відеокамерою, інформація про нього буде виведена на панель приладів.

Система здатна розпізнавати обмеження швидкості, що діють для певного виду транспорту (по знакам додаткової інформації - табличок), а також знаки скасування обмеження швидкості. Система Opel Eye пішла далі - вона розпізнає поряд зі знаками обмеження швидкості, знаки, що забороняють обгін.

Система розпізнавання дорожніх знаків другого покоління інформує водія про різні дорожні знаки. Крім знаків обмеження швидкості, заборони обгону, окремих знаків додаткової інформації, система розпізнає такі знаки:

- Проїзд без зупинки заборонено;
- в'їзд заборонений;
- головна дорога (кінець головної дороги);
- перевага зустрічного руху (перевага перед зустрічним рухом);
- поступіться дорогою;
- кінець зони всіх обмежень;

– початок (кінець) населеного пункту;
– початок (кінець) автомагістралі;
– житлова зона.

Перераховані знаки на дисплеї не відображаються. Інформація про розпізнаних знаках узгоджується з даними навігаційної системи, поточними параметрами руху автомобіля. В результаті система інформує водія про поточну дорожню ситуацію і сприяє безпечному руху.

Також доволі відомим є програмне забезпечення під назвою «Roadly dashcam & speed camera». Даний програмний продукт розповсюджується безкоштовно в магазині «GooglePlay».

Рисунок 1.2 – Приклад роботи додатку «Roadly dashcam & speed camera»

Дана програма має наступні недоліки: смартфон потребує постійного підключення до джерела живлення, для коректної роботи потребує сучасне програмне забезпечення. Також при використанні даного застосунку смартфон потрібно додатково охолоджувати, оскільки при роботі камера перегрівається і може вийти із ладу. Неможливість установити даний програмний продукт на відео реєстратор, перелік знаків які вдається розпізнати є невеликим.

В загальних рисах даний продукт не може задовільнити вимог пересічного користувача, про що також говорять відгуки. В кінці можна сказати, що програма «Roadly dashcam & speed camera» не рекомендується до встановлення та використання.

Наступним варіантом який потрібно розглянути виступає відео реєстратор «PAPAGO! P2Pro NEW». До його переваг слід віднести високу роздільну здатність, автоматичний початок роботи після початку руху автомобіля, широкий кут огляду, Вбудований GPS модуль, можливість визначити швидкість руху автомобіля та можливість розпізнання майже всіх дорожніх знаків.

Але навіть у даного пристрою також є свої недоліки. Це невелика роздільна здатність дисплею, неможливість працювати у темну пору доби і те що при температурі нижче нуля градусів пристрій може не ввімкнутися або ж вийти із ладу.

Хоча маючи навіть такі недоліки «PAPAGO! P2Pro NEW» все одно залишився достойним пристроєм для відео реєстрації та допомоги у водінні якби не те що він випускається невеликими партіями, тобто фактична відсутність на ринку.

Рисунок 1.3 – Відео реєстратор «PAPAGO! P2Pro NEW»

1.3 Постановка задачі

У ході виконання магістерської кваліфікаційної роботи необхідно вивчити предметну область процесу розпізнавання. Далі необхідно

ознайомитися з існуючим станом справ в області розпізнавання, вивчити чи є аналоги і зробити висновок про необхідність розробки нового продукту, необхідно вибрати засоби розробки: операційну систему, мову і середовище програмування.

Отже, потрібно розробити інформаційну технологію розпізнавання дорожніх знаків. Функціонально розроблений програмний додаток має дозволити користувачеві завантажити зображення в додаток, відобразити його на екрані і виділити на зображені дорожні знаки, у вигляді прямокутних фігур на зображенні. Для фігури програма повинна запропонувати знак, який найбільш імовірний для обраного сегмента зображення. Якщо ж даний розпізнаний знак виявиться хибним, повинна бути можливість повторного сканування зображення і можливість встановлення правильного знаку самостійно.

1.4 Висновок

В даному розділі проаналізовано предметну область, зазначено актуальність дослідження, визначено основну проблему при розпізнаванні дорожніх знаків, що полягає в факторах впливу навколишнього середовища, ускладнення дорожнього руху, та збільшення швидкостей водіння. Виконано аналітичний огляд аналогів вирішення технічної проблеми. Аналіз показав, що не існує програмної реалізації, яка водночас дозволяє проводити швидко, якісно та дешево розпізнавання дорожніх знаків. Одним із найкращих в цьому плані можна вважати програмне забезпечення Opel Eye, яке дозволяє розпізнати практично всі групи дорожніх знаків. Якість розпізнавання також задовільна, але дана система застосовується на автомобілях бізнес класу, які далеко не кожному по-кишені. Враховуючи недоліки існуючих програмних засобів розпізнавання виконано постановку задачі, рішення якої призведе до підвищення точності розпізнавання зображень.

2 РОЗРОБКА ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ РОЗПІЗНАВАННЯ ДОРОЖНІХ ЗНАКІВ

2.1 Обґрунтування використання методу обробки зображення в задачі розпізнавання дорожніх знаків

Описавши широко відомі методи в попередньому розділі вибір впав н метод розпізнавання дорожніх знаків за допомогою дескрипторів.

На даний момент широко використовуються два основні методи розпізнавання зображень, які зарекомендували свою високу достовірність: нейронні мережі які здійснюють пошук по зображенні за допомогою кольорової палітри та системи на основі дескрипторів [9].

При розв’язанні задач розпізнавання об’єктів та предметів використовуються моделі RGB та HSV.

З RGB можна швидко працювати, оскільки вхідні данні з камери подаються саме в цьому форматі. Проте, дана модель, чутлива до деяких факторів, які зовсім не потрібно враховувати, наприклад, зміну освітлення. Якщо освітленість суттєво змінюється − тоді потрібно використовувати кольорову модель HSV. Алгоритм її реалізації потребує більше часу, оскільки спочатку треба перевести зображення з RGB в HSV, проте перевагою є те що в даній кольоровій палітрі можна не враховувати освітленість, тобто компоненту V. Для створення образу об’єкта потрібні його зображення та бітова маска, яка показує які точки зображення йому належать. По цим даним визначаються інші потрібні відомості про об’єкт: площа, гістограма, тощо.

При розпізнаванні спочатку проводиться пошук об’єкта найшвидшими методами. При цьому якщо вони не дають однозначного результату, або він є незадовільним (не достатньо точним) проводиться перевірка “за зростаючою” від швидких алгоритмів до більш точних. І так доти, поки не буде отримана потрібна достовірність. При цьому система збирає статистику, і визначає, чи коректно працює алгоритм для даного середовища, чи потрібно його

використовувати при наступних пошуках, і при кожному повторюванні циклу, якщо потрібно змінює набір ”активних” алгоритмів. Якщо результат, наприклад, метода розпізнавання об’єкта за площею (який є швидкодіючим) збігається із загальним результатом отриманим за допомогою сукупності методів − можна використовувати тільки цей один метод поки не зміниться навколишнє оточення або сам об’єкт [10].

Тому можна сказати, що даний метод пошуку є незадовільним для поставленої задачі, оскільки в даному методі використовуються нейронні мережі для яких необхідні високі апаратні потужності та висока чутливість на такі зовнішні фактори як освітленість та роздільна здатність зображення.

Наступним методом розпізнавання буде розпізнавання за допомогою дескрипторів.

Для будь-якого об'єкта на зображенні можливо отримати точки інтересу на об'єкті, щоб створити «опис ознак» об'єкта. Ці ознаки, отримані із зображення, можна використовувати для ідентифікації об'єкта при знаходженні його на іншому тестовому зображенні, що містить багато інших об'єктів. Для надійного розпізнавання важливо, щоб отримані ознаки на початковому зображенні можна було розрізнити навіть при зміні масштабу зображення, освітлення і наявності шуму. Такі точки, зазвичай, знаходяться на висококонтрастних ділянках зображення, наприклад на межах об'єкта [11].

Іншою важливою характеристикою цих ознак є те, що відносні позиції між ними не повинні змінюватися від одного зображення до іншого. Наприклад, якщо як ознаки було обрано чотири кути дверей, вони будуть порівнюватися незалежно від позиції дверей; але якщо разом з тим використати точки на дверній рамі, розпізнавання не спрацює, коли двері відкриті чи закриті. Так само, ознаки що знаходяться на рухомих чи гнучких об'єктах, зазвичай не працюватимуть, якщо відбуватимуться будь-які зміни у їх внутрішній геометрії при порівнянні двох зображень. Однак, на практиці дескрипторні алгоритми визначають і використовують значно більшу

кількість ознак на зображенні, що зменшує внесок похибок, спричинених локальними змінами, до середньої похибки загальних порівнянь ознак.

Дескриптор може достовірно ідентифікувати об'єкти навіть серед шуму або часткового перекриття, тому що дескриптор ознак, що використовується в дескрипторах, є інваріантним до лінійного масштабування, змін просторової орієнтації, освітлення, і частково до афінного перетворення.

Отже для вирішення задачі розпізнавання дорожніх знаків буде використовуватися метод на основі дескриптора. Наступним кроком буде вибір дескриптора який найкраще підходить для вирішення задачі розпізнавання дорожніх знаків.

2.2 Вибір типу дескриптора

Результатом роботи детекторів є безліч особливих точок, для яких необхідно побудувати математичний опис. У цьому розділі робиться огляд деяких існуючих дескрипторів.

Вхідними даними дескриптора є зображення і набір особливих точок, виділених на заданому зображенні. Виходом дескриптора є безліч векторів ознак для вихідного набору особливих точок. Необхідно відзначити, що якісь дескриптори вирішують одночасно два завдання - пошук особливих точок і побудова опису цих точок [12].

Ознаки (описи) будуються на підставі інформації про інтенсивність, колір і текстуру особливої точки. Але особливі точки можуть представлятися кутами, ребрами або навіть контуром об'єкта, тому, як правило, обчислення виконуються для деякої околиці. В ідеалі хороші ознаки повинні володіти рядом властивостей:

Повторюваність. На зображеннях одного і того ж об'єкта або сцени, зроблених з різних точок зору і при різних умовах освітленості.

Локальність. Ознаки повинні бути максимально локальними, щоб знизити ймовірність перекриттів.

Репрезентативність. Кількість ознак повинно бути достатнім, щоб розумне число ознак детектувати навіть на невеликому зображенні об'єкта.

Точність. Ознаки повинні бути точно продетектовані по відношенню до масштабу і форми об'єкта [13].

Ефективність. Для додатків реального часу критично, щоб процедура обчислення ознак не вимагала значних обчислювальних витрат.

Дескриптор SURF (Speeded up Robust Features) відноситься до числа тих дескрипторів, які одночасно виконують пошук особливих точок і будують їх опис, інваріантний до зміни масштабу і обертання. Крім того, сам пошук ключових точок має інваріантністю в тому сенсі, що повернений об'єкт сцени має той же набір особливих точок, що і зразок.

Визначення особливих точок на зображенні виконується на підставі матриці Гессе (FAST-Hessian detector). Використання гессіан забезпечує інваріантність щодо перетворення типу "поворот", але не інваріантність щодо зміни масштабу. Тому SURF застосовує фільтри різного масштабу для обчислення гессіан. Вихідне зображення задається матрицею інтенсивностей [13].

Детермінант матриці Гессе досягає екстремуму в точках максимальної зміни градієнта яскравості. Тому SURF пробігається фільтром з гауссовим ядром по всьому зображенню і знаходить точки, в яких досягається максимальне значення детермінанта матриці Гессе. Відзначимо, що такий прохід виділяє як темні плями на білому тлі, так і світлі плями на темному тлі.

Далі для кожної знайденої особливої точки обчислюється орієнтація - переважний напрямок перепаду яскравості. Поняття орієнтації близьке до поняття напрямку градієнта, але для визначення орієнтації особливої точки застосовується фільтр Хаара[14].

На підставі наявної інформації виконується побудова дескрипторів для кожної особливої точки.

Навколо точки будується квадратна околиця розміром 20s, де s - масштаб, на якому отримано максимальне значення детермінанта матриці Гесса.

Отримана квадратна область розбивається на блоки, в результаті область буде розбита на 4 х 4 регіони.

Для кожного блоку обчислюються більш прості ознаки. Як наслідок, виходить вектор, що містить 4 компоненти: 2 - це сумарний градієнт по квадранту, 2 - сума модулів точкових градієнтів.

Дескриптор формується в результаті склеювання зважених описів градієнта для 16 квадрантів навколо особливої точки. Елементи дескриптора зважуються на коефіцієнти Гауссового ядра. Ваги необхідні для більшої стійкості до шумів у віддалених точках.

Додатково до дескриптора заноситься слід матриці Гессе. Ці компоненти необхідні, щоб розрізняти темні і світлі плями. Для світлих точок на темному тлі слід негативний, для темних точок на світлому фоні - позитивний.

Слід зазначити що хоча SURF використовується для пошуку об'єктів. Проте, дескриптор ніяк не використовує інформацію про об'єкти. SURF розглядає зображення як єдине ціле і виділяє особливості всього зображення, тому він погано працює з об'єктами простої форми. Отже даний дескриптор не підходить для вирішення задачі розпізнавання дорожніх знаків.

Наступним дескриптором який розглядається буде SIFT.

Для формування дескриптора SIFT (Scale Invariant Feature Transform) спочатку обчислюються значення магнітуди і орієнтації градієнта в кожному пікселі, що належить околиці особливої точки розміром 16 х 16 пікселів. Магнітуди градієнтів при цьому враховуються з вагами, пропорційними значенню функції щільності нормального розподілу з математичним очікуванням в розглянутій особливій точці і стандартним відхиленням, рівним половині ширини околиці (ваги Гауссового розподілу використовуються для того, щоб зменшити вплив на підсумковий дескриптор градієнтів, обчислених в пікселях, які перебувають далі від особливої точки)[15].

У кожному квадраті розміром 4 х 4 пікселя обчислюється гістограма орієнтованих градієнтів шляхом додавання зваженого значення магнітуди градієнта до одного з 8 комірок гістограми. Щоб зменшити різні "граничні" ефекти, пов'язані з віднесенням схожих градієнтів до різних квадратах (що може виникнути внаслідок невеликого зсуву розташування особливої точки) використовується білінійна інтерполяція: значення магнітуди кожного градієнта додається не тільки в гістограму, відповідну квадрату, до якого даний піксель відноситься, але і до гістограми, відповідним сусіднім квадратах. При цьому значення магнітуди додається з вагою, пропорційною відстані від пікселя, в якому обчислений даний градієнт, до центру відповідного квадрата. Всі обчислені гістограми об'єднуються в один вектор, розміром, рівним 128 = 8 (число бінов)х 4 х 4 (число квадратів)[16].

Отриманий дескриптор перетворюється, щоб зменшити можливі ефекти від зміни освітленості. Зміна контрасту зображення (значення інтенсивності кожного пікселя множиться на деяку константу) призводить до такого ж зміни в значеннях магнітуд градієнтів. Тому очевидно, що даний ефект може бути знівельовано шляхом нормалізації дескриптора таким чином, щоб його довжина стала дорівнює одиниці. Зміни яскравості зображення (до значення інтенсивності кожного пікселя додається деяка константа) не впливають на значення магнітуд градієнтів. Таким чином, SIFT-дескриптор є інваріантним по відношенню до змін освітленості. Однак можуть виникати і нелінійні зміни в освітленості внаслідок, наприклад, різної орієнтації джерела світла по відношенню до поверхонь тривимірного об'єкту. Дані ефекти можуть викликати велика зміна в ставленні магнітуд деяких градієнтів (при цьому мають незначний вплив на орієнтацію вектора градієнта). Щоб уникнути цього, використовують відсікання по деякому порогу (за результатами експериментів показано, що оптимальним є значення 0.2), яке застосовують до компонентів нормалізованого дескриптора. Після застосування порога дескриптор знову нормалізується. Таким чином, зменшується значення

великих магнітуд градієнтів і збільшується значення розподілу орієнтацій даних градієнтів в околиці особливої точки.

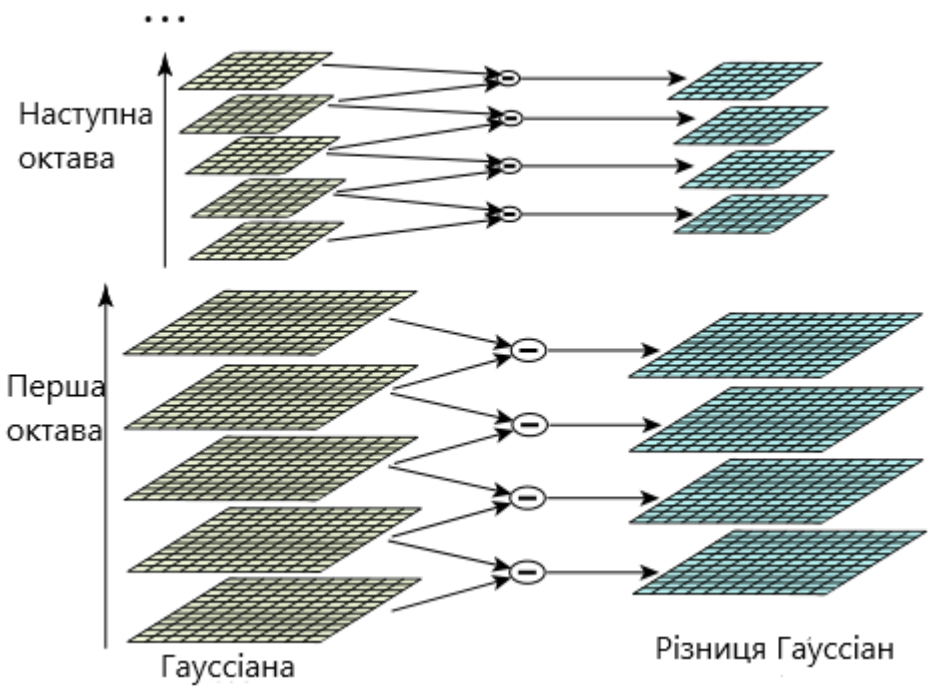

Рисунок 2.1 – Різниця Гауссіан у SIFT дескрипторах

Отже SIFT дескриптор задовольняє вимоги для виконання задачі розпізнавання дорожніх знаків.

2.3 Метод та математична модель розпізнавання дорожніх знаків за допомогою дескрипторів

Першою операцією відбувається пошук екстремумів в зображенні.

Ми починаємо з виявлення точок інтересу, які називаються ключовими точками в структурі SIFT. Зображення згорнуто за допомогою фільтрів Гаусса в різних масштабах, а потім знімається різниця послідовних зображень з гаусовим розмиванням. Потім ключові точки приймаються як максимуми / мінімуми різниці гауссіанів, які відбуваються в декількох масштабах. Зокрема, образ DoG D(x,y,q) [17].

$$D(x,y,q) = L(x,y,k_i,q) - L(x,y,k_j,q) \quad (2.1)$$

Де L(x,y,kq) це згортка вихідного зображення, I(x,y) з розмиттям по Гауссу G(x,y,kq) в маштабі kq, тобто L(x,y,kq) = G(x,y,kq) * I(x,y).

Отже, зображення DoG між шкалами $k_i q$ та $k_j q$ це просто різниця розмитих по Гаусові зображень в масштабах $k_i q$ та $k_j q$. Для виявлення екстремумів в масштабному просторі в алгоритмі SIFT зображення спочатку згортається за допомогою розмиття по Гауса в різних масштабах. Згорнуті зображення згруповані по октаві (октава відповідає подвоєння значення q), а значення k_i вибирається так, щоб ми отримували фіксовану кількість згорнутих зображень на октаву. Потім зображення різниці Гаусса беруться з сусідніх зображень з розмиванням по Гауса на октаву.

Після отримання зображень DoG ключові точки ідентифікуються як локальні мінімуми / максимуми зображень DoG за масштабами. Це робиться шляхом порівняння кожного пікселя в зображеннях DoG з його вісьмома сусідами в тому ж масштабі і дев'ятьма відповідними сусідніми пікселями в кожному з сусідніх масштабів. Якщо значення пікселя є максимальним або мінімальним серед всіх порівнюваних пікселів, воно вибирається в якості ключової точки-кандидата.

Виявлення екстремумів в масштабному просторі створює занадто багато кандидатів в ключові точки, деякі з яких нестабільні. Наступним кроком в алгоритмі є виконання детального підбору найближчих даних для точного визначення місця розташування, масштабу і співвідношення основних кривизни. Ця інформація дозволяє відкидати точки, які мають низький контраст (і тому чутливі до шуму) або погано локалізовані по краю [18].

Рисунок 2.2 – Фільтрування ключових точок алгоритмом SIFT

По-перше, для кожної ключової точки-кандидата використовується інтерполяція найближчих даних для точного визначення її положення. Початковий підхід полягав у тому, щоб просто розмістити кожну ключову точку в місці і масштабі ключовий точки-кандидата. Новий підхід обчислює інтерпольоване положення екстремуму, що істотно покращує узгодження і стабільність. [2] Інтерполяція виконується з використанням квадратичного розкладання Тейлора функції різниці масштабів Гаусса, D(x,y,q) з ключовою точкою-кандидатом в якості джерела. Це розкладання Тейлора визначається наступним чином:

$$D(\mathbf{x}) = D + \frac{\partial D}{\partial \mathbf{x}}^T \mathbf{x} + \frac{1}{2}\mathbf{x}^T \frac{\partial^2 D}{\partial \mathbf{x}^2}\mathbf{x} \quad (2.2),$$

де D і його похідні оцінюються в ключовій точці кандидата і $X = (x,y,q)^T$ це зміщення від цієї точки. Розташування екстремуму, визначається взяттям похідної від цієї функції по екстремумі і встановивши його на нуль. Якщо зсув екстремуми більше ніж 0,5 В будь-якому вимірі, то це ознака того, що екстремум знаходиться ближче до іншої ключовій точці кандидата. В цьому випадку ключова точка змінюється, і замість неї виконується інтерполяція. В іншому випадку зміщення додається до його ключовій точці-кандидату, щоб отримати інтерпольованого оцінку положення екстремуму. Подібне субпіксельне визначення місць розташування екстремумів простору масштабу виконується в реалізації в реальному часі на основі гібридних пірамід, розроблених Ліндеберга і його співробітниками.

Функція DoG матиме сильні відгуки по краях, навіть якщо передбачувана ключова точка не стійка до невеликого шуму. Отже, щоб підвищити стабільність, нам потрібно усунути ключові точки, які мають погано певні місця розташування, але мають високі характеристики краю.

Для погано певних піків в функції DoG основна кривизна по краю буде набагато більше, ніж основна кривизна по ньому. Виявлення цих основних викривлень зводиться до вирішення для власних від другого порядку матриці Гесса:

$$\mathbf{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix} \tag{2.3}$$

На цьому етапі кожній ключовій точці призначається одна або кілька орієнтацій на основі локальних напрямків градієнта зображення. Це ключовий крок у досягненні інваріантності до обертання, оскільки дескриптор ключовий точки може бути представлений щодо цієї орієнтації і, отже, домогтися інваріантності до обертання зображення.

По-перше, згладжені по Гауса зображення L(x,y,q) в масштабі ключовий точки q береться таким чином, щоб всі обчислення виконувалися масштабно-

інваріантним чином. Для зразка зображення L(x,y) в масштабі q, величина градієнта, m(x,y), і орієнтація, O(x,y) попередньо обчислюються з використанням різниці пікселів:

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2}$$
$$\theta(x,y) = \operatorname{atan2}(L(x,y+1) - L(x,y-1), L(x+1,y) - L(x-1,y)) \quad (2.4)$$

Обчислення величини і напрямки градієнта виконуються для кожного пікселя в сусідній області навколо ключової точки в зображенні з розмитим по Гауса чином L. Формується гістограма орієнтації з 36 осередками, кожна клітинка покриває 10 градусів. Кожна вибірка в сусідньому вікні, що додається в клітинку гістограми, зважується за величиною градієнта і за допомогою зваженого по Гауса кругового вікна з q що в 1,5 рази більше масштабу ключовий точки. Піки на цій гістограмі відповідають домінуючим орієнтаціям. Після заповнення гістограми ключовій точці призначаються орієнтації, відповідні найвищим піку і локальним піків, які знаходяться в межах 80% від найвищих піків. У разі призначення кількох орієнтацій створюється додаткова характерна точка з тим же розташуванням і масштабом, що і вихідна характерна точка для кожної додаткової орієнтації.

2.4 Структура інформаційної технології розпізнавання зображень дорожніх знаків за допомогою дескрипторів

Структура інформаційної технології розпізнавання зображень дорожніх знаків за допомогою дескрипторів, детально описана у попередніх підпунктах, зображена на рис. 2.3.

Для роботи інформаційної технології розпізнавання зображень дорожніх знаків за допомогою дескрипторів вхідною є інформація у вигляді зображення (png, jpeg або gif формату). Потім здійснюється процес виявлення екстркмумів

у просторі. Після цього розпочнеться процес локалізації ключових точок, з їх подальшею фільтрацією та створенням вектора цих точок. Дані операції будуть проводитися з використанням SIFT дескрипторів. Далі відбувається процес розпізнавання з показом результатів роботи програмного продукту.

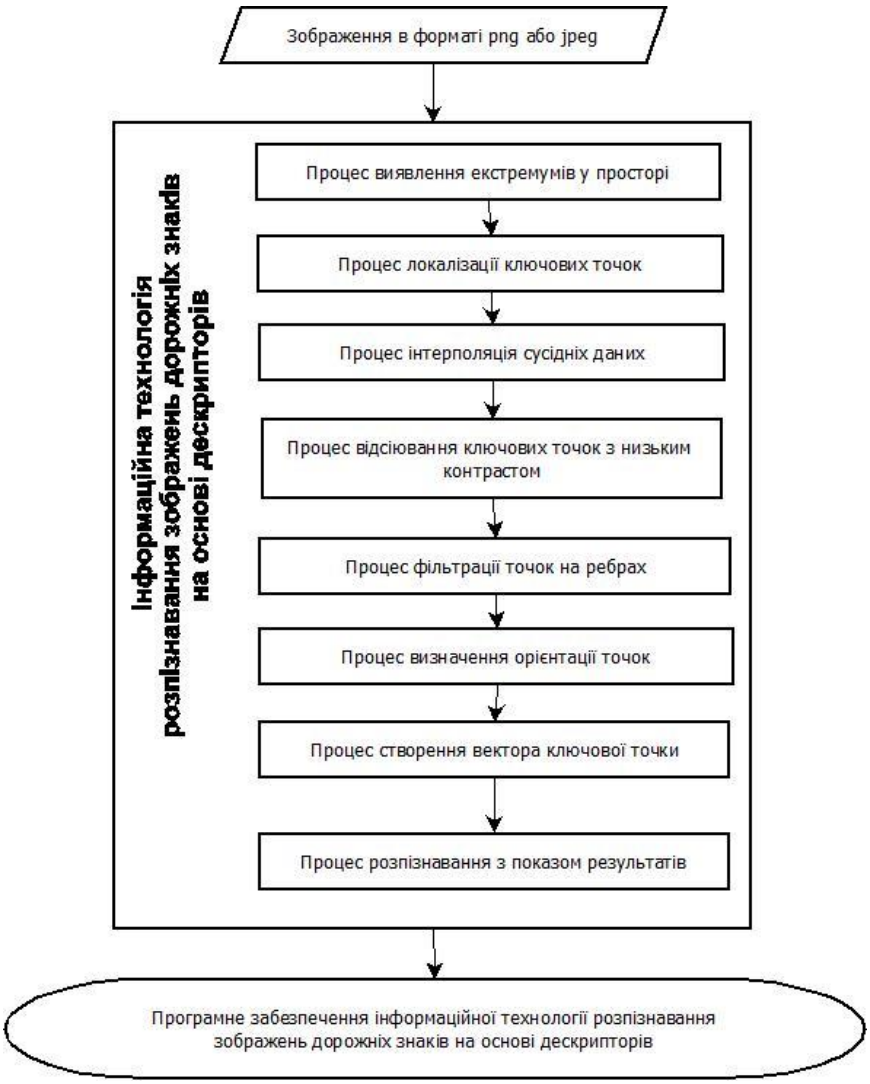

Рисунок 2.3 - Структура інформаційної технології розпізнавання зображень дорожніх знаків за допомогою дескрипторів

Наступним кроком буде розробка структурної схеми роботи інформаційної технології розпізнавання зображень дорожніх знаків за допомогою дескрипторів, дана блок схема зображена на рисунку 2.4.

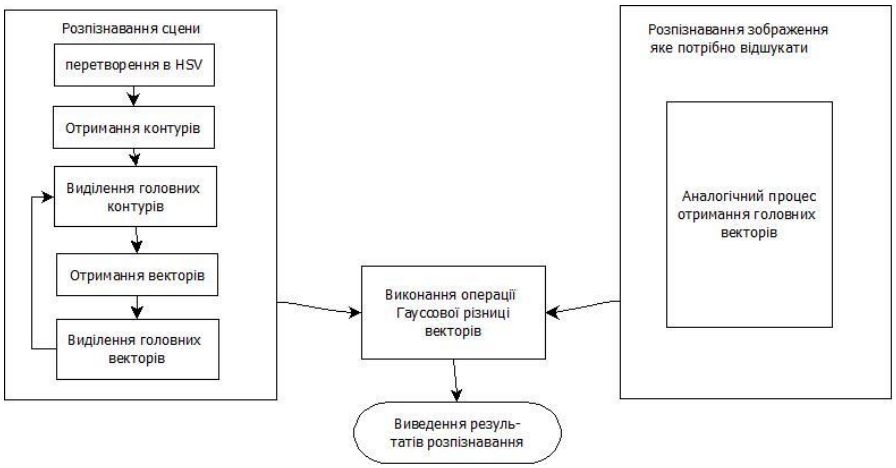

Рисунок 2.4 – Структурна схема алгоритму роботи інформаційної технології розпізнавання зображень дорожніх знаків за допомогою дескрипторів.

Таким чином, розроблена структура інформаційної технології розпізнавання зображень дорожніх знаків за допомогою дескрипторів може бути використана для розробки подальшого програмного засобу.

2.5 Висновок

В даному розділі було розглянуто Найбільш популярні методи, які використовуються для розв’язання задач розпізнавання зображень, зокрема розглядались нейронні мережі які використовують кольорову палітру та дескриптори точок. Нейронні мережі показали свою високу вразливість під час впливу різної освітленості, неточності під час роботи з зображеннями низької роздільної здатності та показали свою відносно повільну швидкодію в порівнянні з дексрипторами точок. Наступним кроком було вибір дескриптора з таких варіантів як SURF та SIFT, де SIFT дескриптор виявився більш точнішим. Наступним кроком було проведення опису математичної моделі SIFT дескриптора та наведено алгоритм функціонування системи розпізнавання.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ РОЗПІЗНАВАННЯ ДОРОЖНІХ ЗНАКІВ

3.1 Вибір середовища розробки

Серед всіх IDE, с якими працюють програмісти, найбільш популярною залишається інтегроване середовище розробки Visual Studio. Її ефективність доведена тим, що продукт залишається лідером на ринку вже більше двадцяти років.

Шлях Visual Studio починається в Microsoft, де в 1997 році вирішили взяти всі існуючі засоби розробки і упакувати в один продукт [19]. Тоді ще не йшлося про створення кроссплатформенних додатків, високопродуктивних програм для Android і інших речей, які стали буденністю. Вони лише хотіли придумати програму для розробників, яка полегшила б взаємодію з такими мовами, як C ++ та інші. До моменту виходу програми, розробники змушені були задовольнятися редактором, який називався Developer Studio: його ви могли бачити на уроках інформатики, де змушували вирішувати завдання на Fortran. До речі, свого часу Developer вважався не редактором, а повноцінним середовищем розробки. Сьогодні він менш функціональний, ніж остання версія Notepad ++.

Початок багатосерійної розробки було покладено ще тоді: Windows завжди прагнули уявити додатки в декількох варіаціях, щоб створити видимість широкого модельного ряду (за фактом, всі дистрибутиви відрізняються незначно). Вони випустили версію для особистого використання - Professional, і Enterprise - для великих компаній. По суті, різниця була лише в деякій оптимізації групової розробки. Далі було прийнято рішення випускати версії в порядку вдосконалення, а функціонал надавати в залежності від вартості пакета. Найцікавіше, що вони вирішили застосувати кілька складний підхід до найменування. Наприклад «PROFESSIONAL» і «professional» - це різні пакети для розробки.

Сьогодні оплачений пакет не грає настільки важливу роль, набагато більш вагомими є встановлені фреймворки, бібліотеки та інші інструменти. Наприклад, вельми перспективною сферою є розробка мобільних додатків для Android і iOS. І, якщо сама Visual Studio не має багато точок дотику з андроїдом, то фреймворки Monodroid і Monotouch вирішують цю проблему. Вони створені для того, щоб розробляти мобільні додатки засобами C #, який є абсолютно неспецифічним для Android. Проте, установка віртуальної машини і фреймворка вирішує цю проблему.

Важливим моментом для тих розробників, що створюють додатки під Windows є те, що середовище розробки постійно підключена до товариства, яке може допомогти в проблемних питаннях. Microsoft Developer Network - це структурний відділ компанії Microsoft, який повністю відповідальний за підтримку розробників. Їхня допомога можуть отримати не тільки користувачі Visual Studio, але і інші програмісти, які придбають відповідну підписку [20].

На ресурсі від Microsoft можна знайти не тільки поради від великого штату професіоналів, але і такі незамінні речі, як документацію до продуктів, базу знань для ефективного використання додатків Windows і інших продуктів компанії Microsoft. Завдяки тісній взаємодії не тільки співробітників, але і спільноти користувачів, Visual Studio можна назвати однією з найбільш підтримуваних програм. Єдиний недолік - Microsoft Developer Network підтримує тільки ті мови і утиліти, які розроблені компанією Microsoft. Звучить логічно, але для серйозного програміста цього, звичайно ж, мало. Плюсом для локального розробника можна назвати те, що з 2008 року працює російськомовна служба MSDN. Але за стільки років вона лише наблизилася до оригіналу по наповненості. За образом і подобою MSDN була створена підтримка Google, де стек технологій значно ширше [21].

Робота з IDE. Цим терміном прийнято називати комплект програмного забезпечення, який покликаний максимально оптимізувати процес веб-розробки, створення програмного забезпечення або будь-який інший вид програмування. Безліч текстових редакторів для коду включають в себе

настільки широкий асортимент функцій, що так і хочеться назвати їх IDE. Однак існує певний набір характеристик, які складають IDE. І скільки б «фішок» не містив Sublime Text 3, він не стане в один ряд з Visual Studio. До переваг Visual Studio можна причислити наступні функції:

- редактор коду. Це не просто «текстовик» на зразок блокнота. Він повинен містити інструменти для роботи з мовами. Наприклад, щоб засіб розробки для C ++ можна було назвати редактором коду, програма повинна підсвічувати синтаксис, помилки, типи даних та інше [22];
- компілятор для компільованих мов або інтерпретатор для інтерпретованих. Залежно від типу мови, потрібно засіб для роботи з ним. У Visual Studio є обидва елементи. Але існують IDE для конкретного PL, наприклад, Ninja. Він містить лише компілятор, який здійснює деяку інтерпретацію;
- автоматизація збірки;
- дебагер, або відладчик. Він допомагає детально вивчити код і знайти в ньому помилки. Без подібного кошти писати великі мобільні або веб-додатки неможливо. Ризик помилки при швидкісного друку символів дуже великий.

Якщо всі перераховані інструменти присутні в програмі, значить її можна назвати інтегрованим середовищем розробки. Але сучасні розробники цієї ніші ніколи не задовольняються стандартним набором засобів. У Visual Studio присутній інтегрована система управління версіями, оглядач класів та інші [23].

Якщо ви вже зважилися на використання Visual Studio, значить необхідно пройти той крок, який ви проігнорували при установці Windows - створити обліковий запис Microsoft. До речі, вона може відкрити для вас більше дверей, ніж здається, особливо що стосується веб-розробки: у компанії навіть є свій аналог мови JavaScript. Але для вас, як для розробника на Visual Studio, це означає, що в подальшому середовище буде персоналізована,

незалежно від того, з якого пристрою ви заходите. Тим більше, це означає доступ до Azure [24].

Якщо ви раніше створювали програми за допомогою інших засобів розробки, то ви можете продовжити, не вельми перевчаючись під Visual Studio. Так, є деякі труднощі для пристосування середовища під створення мобільних додатків, але і це легше, ніж робота з більшістю додатків Windows. Що стосується веб-розробки, то рівних тут немає: спеціалізовані розширення постійно виходять і оновлюються.

Але існують спеціальні засоби, не використовувати які - справжнє блюзнірство. Тим більше, що ви вже оплатили покупку пакета. Серед них:

- IntelliSense. Інструмент, який викрадає серця розробників і змушує їх придбати Visual Studio. Він не тільки вміє на льоту виправляти помилки розробника, пропонуючи ряд варіантів, але, в окремих випадках, може і згенерувати фрагменти коду. Тут немає нічого фантастичного або проявів штучного інтелекту вищого порядку. Насправді, це просто документація різних мов, вбудована в програму. Знаходячи подібні елементи, IS підпирає фрагмент або виправлення. Особливо цінується в розробці мобільних додатків і програмного забезпечення, так як такий код зазвичай довгий і містить елементи різних мов [25];
- CodeLens. Як видно з назви, це утиліта, яка дозволяє знаходити помилки в коді. По суті, це коректор, який вбудований прямо в Visual Studio.
- Хвиляста лінія. З нею то ви точно знайомі - стара «фішка» Microsoft для визначення помилок. При наведенні і натисканні можна побачити подробиці помилки і способи її усунення.

Інші засоби, на кшталт швидкого запуску і рефакторінга доступні і в інших програмах для розробки додатків. Але ці - справжнє дітище Microsoft.

Вище ми вже встигли трохи описати ті характеристики, які роблять Visual Studio програмою вибору. Основний висновок, який можна зробити -

вона на голову вище за функціональністю і потужності. Але ж і Vim потужна система, а її вибирають тільки деякі веб-розробники. Що ж робить Visual Studio найпопулярнішою IDE?

Сервер для середовища. Більшість програмістів, які використовують IDE для веб-розробки або створення мобільних веб-додатків знають, що для забезпечення бекенд потрібен віртуальний сервер, який буде обробляти запити і відповіді. За допомогою VS і вбудованого сервера ASP.NET можна не тільки працювати елементами, але і запускати сайт прямо з середовища. Цей підхід значно випереджає аналогічні хмарні сервіси: на стадії розробки ніхто не зможе отримати доступ до продукту в офлайні.

Універсальність мов. Деякі IDE підтримують частина мов, деякі більшість, деякі - тільки один. Microsoft розробили засіб розробки, яке підтримує переважаюче число PL, якщо не брати до уваги екзотичні. Але підтримкою не закінчується універсальність, особливо це стосується веб-розробки: в Visual Studio можна створювати сторінки на різних мовах, а після помістити їх все в одному додатку. Зручно, особливо для спільної розробки.

Менше коду - більше результату. Працюючи з іншими засобами для розробки, ви пишіть багато речей, які в VS додаються завдяки інтелектуальній системі.

Код читається з перших рядків. Буває, дивишся на готове мобільний додаток і розумієш - розроблено за допомогою Visual Studio. Живій людині важко постійно дотримуватися відступи і інтервали. Інтелектуальна система середовища розробки вміє підлаштовувати код під необхідний формат і синтаксис. Таким чином, він стає більш читабельним і доступним для редагування.

Крім MSDN, який постійно готовий дати пораду розробнику, VS підтримує такий інструмент, як Team System. Це платформа для спільної роботи над проектами, тестування і налагодження. Важливим аспектом є і зворотний зв'язок з творцями Visual Studio і операторами підтримки. Користувач VS завжди може відправити лист з побажаннями для програми.

Так як розробники з Visual Studio є пріоритетними користувачами, їх запити дійсно розглядаються.

Але попри все у VS як у будь якого програмного продукту окрім переваг є і ряд недостатків. Хоч вони і не є критичними про їх варто описати.

По-перше, вона і правда масивна. Кожен розробник, який програмує більш, ніж на одній машині, знає, чим загрожує низька продуктивність комп'ютера з встановленим VS. Ви не те що не відчуєте переваг IDE, а навіть згадайте швидкий і простий Sublime Text. Благо, на сучасних машинах VS працює добре. Так що цей аспект залежить від обставин, в яких доводиться працювати.

Друга причина, по якій Visual Studio ще не стоїть на всіх в світі комп'ютерах - це ціна. Середні і малі компанії не завжди можуть дозволити собі корпоративну підписку, що вже говорити про незалежні розробників, вони віддадуть перевагу безкоштовні аналоги. Звичайно, якість коштує своїх грошей, але старт від 500 $ відлякує новачків. А професіонал готовий платити тільки за те, з чим він звик працювати. Але оскільки робота буде проводитися в дистрибутиві Common, а він розповсюджується безкоштовно.

Ну і третім, істотним недоліком є обмеженість платформ, здатних взаємодіяти з Visual Studio. Сьогодні це тільки Windows і Mac. Лінуксоїди залишилися без передової IDE, хоч і користувачі Linux-осей посідають третє місце відразу за OS X і Windows. Якщо говорити про СНГ сегментт, то їх чи не більшість. Не виключено, що портування ще попереду: просто це дуже схоже на тактику Microsoft.

Проте, Visual Studio запускають на всіх системах, застосовуючи різні Wine-подібні технології - емулятори. Скільки б програміст не хвалив Vim або Emac, він не може не оцінити зручність роботи з такою системою, як Visual Studio. Оскільки розробка буде проходити в VS мовою реалізації було вибрано C# [26].

3.2 Обґрунтування вибору мови програмування

C++ – мова програмування високого рівня з підтримкою кількох парадигм програмування: об'єктно-орієнтованої, узагальненої та процедурної.

Недоліки мови C++:

- наявність безліч можливостей, що порушують принципи типобезпеки приводить до того, що в C++ програми може легко закрастися важковловима помилка. Замість контролю з боку компілятора розробники вимушені дотримуватися вельми нетривіальних правил кодування. По суті, ці правила обмежують C++ рамками якоїсь безпечнішої підмови. Більшість проблем типобезпеки C++ успадкована від C, але важливу роль в цьому питанні грає і відмова автора мови від ідеї використовувати автоматичне управління пам'яттю (наприклад, збірку сміття). Так візитною карткою C++ стали вразливості типу «переповнювання буфера»;
- погана підтримка модульності. Підключення інтерфейсу зовнішнього модуля через препроцесорну вставку заголовного файлу (#include) серйозно уповільнює компіляцію, при підключенні великої кількості модулів. Для усунення цього недоліку, багато компіляторів реалізують механізм прекомпіляції заголовних файлів;
- недостача інформації про типи даних під час компіляції;
- мова C++ є складною для вивчення і для компіляції;
- деякі перетворення типів неінтуїтивні. Зокрема, операція над беззнаковим і знаковим числами видає беззнаковий результат;
- препроцесор C++ (успадкований від C) дуже примітивний. Це приводить з одного боку до того, що з його допомогою не можна (або важко) здійснювати деякі завдання метапрограмування, а з

іншого, в наслідок своєї примітивності, він часто приводить до помилок і вимагає багато дій з обходу потенційних проблем;

– хоча декларується, що C++ мультипарадигмена мова, реально в мові відсутня підтримка функціонального програмування [27].

Мова Java має такі особливості, яких немає в мові C++:

– Java є типобезпечною мовою. Типобезпека гарантує відсутність у програмах помилок, які важко знайти і які пов'язані з неправильною інтерпретацією пам'яті комп'ютера. Це робить процес розробки надійнішим і передбачуваним, а отже швидшим. Так само це дозволяє залучати до розробки програмістів, що мають меншу кваліфікацію і мати великі групи розробників;
– Java-код компілюється спочатку не в машинний код, а в певний проміжний код, який надалі інтерпретується або компілюється, тоді як багато C++ компіляторів орієнтовані на компіляцію в машинний код заданої платформи;
– у мові Java є чіткі певні стандарти на введення-виведення, графіку, геометрію, діалог, доступ до баз даних і інших типових застосувань. Завдяки цим особливостям, застосунки на Java мають значно кращу кросплатформність, ніж C++, і часто, будучи написані для певного комп'ютера і операційної системи, працюють під іншими системами без змін. Програмісти, що пишуть на мові Java, не залежать від пакунків, нав'язаних розробниками компіляторів на дане конкретне середовище, що різко спрощує портування програм;
– у мові Java реалізовано повноцінне збирання сміття, якого немає в C++. Немає в C++ і засобів перевірки правильності вказівників. З іншого боку, C++ володіє набором засобів (конструктори і деструктори, стандартні шаблони, посилання), що дозволяють майже повністю виключити виділення і звільнення пам'яті вручну і небезпечні операції з вказівниками. Проте таке виключення вимагає

певної культури програмування, тоді як в мові Java воно реалізується автоматично;

- мова Java є чисто об'єктно-орієнтованою, тоді як C++ підтримує як об'єктно-орієнтоване, так і процедурне програмування;
- в C++ відсутня повноцінна інформація про типи під час виконання RTTI. Цю можливість можна було б реалізувати в C++, маючи повну інформацію про типи під час компіляції CTTI;
- у C++ є можливість введення призначеного для користувача синтаксису за допомогою #define, що може привести до того, що модулі у великих пакетах програм стають сильно пов'язані один з одним. Це різко знижує надійність пакетів і можливість організації розділених модулів. З іншого боку, C++ надає достатньо засобів (константи, шаблони, вбудовані функції) для того, щоб практично повністю виключити використання #define [28].

Враховуючи дані особливості слід обрати мову Java, так як, використовуючи її, більшість помилок можна виявити в процесі компіляції, а не під час виконання. А також перевагою мови Java є наявність збирання сміття, яке дозволяє запобігти витоку пам’яті.

Тепер порівняємо C# і Java. В багато чому C# схожий на Java. В C# включені мовні властивості Java, такі як: можна наслідувати лише один клас, інтерфейси, збирання сміття, внутрішні класи, близький синтаксис і компіляція в проміжковий формат. Для обох мов компіляція виконується в машинно-незалежний код, який запускається в рамках керованого середовище виконання. Концепція JVM (Java Virtual Machine) дуже подібна підсистемі CLR (Common Language Runtime) мови C#. Мови дуже схожі, але за рахунок віку Java (C# є молодшим, ніж Java) має більшу кількість бібліотек, які полегшують розробку, велика кількість труднощів у розробці вже була розглянута іншими розробниками і вирішена. Тому за рахунок цього для розробки програмного проекту слід обрати мову програмування C#.

Таблиця 3.1 – Порівняння мов програмування C# та Java

Синтаксис	Java	C#
Імпорт статичних імен	дозволяє окремо імпортувати деякі або всі статичні методи і змінні класу і використовувати їх імена без кваліфікації імпортуючий модулі	Починаючи з C # 6.0 це було введено (наприклад (using static System.Math)).
Оператор switch	Аргумент оператора switch повинен ставитися або до целочисленному, або до перелічуваних типу. Починаючи з версії Java 7 в операторі switch стало можливо використовувати рядкові літерали і ця різниця з C # було усунуто	Підтримуються як константні типи, так і строкові. У C # 7 з'явилася підтримка посилальних типів і null. Також можливо задавати додаткові умови для блоку case за допомогою ключового слова when. На відміну від Java, прямого переходу до наступного блоку case немає.
Оператор переходу goto	від використання goto свідомо відмовилися	goto зберігся, його звичайне використання - передача управління на різні мітки case в операторі switch і вихід з вкладеного циклу
Константи	констант як таких немає, замість них використовуються статичні змінні класу з модифікатором	окреме поняття іменованої типизированной константи і ключове слово const
Відключення перевірок	У Java всі динамічні перевірки включаються / вимикаються тільки на рівні пакета	C # містить конструкції checked і unchecked, що дозволяють локально включати і вимикати динамічну перевірку арифметичного переповнення.

Оператори в мові C# - це спеціальні символи, які повідомляють транслятор про те, що ви хочете виконати операцію з деякими операндами.

Об’єктно-орієнтовані оператори використовуються під час роботи з класами.

Таблиця 3.2 – Об’єктно-орієнтовані оператори C#

Оператор	Операція
:	Успадкування класів
Public	Необмежений доступ до класу
Protected	Захищений доступ
Internal	Доступ обмежений збіркою
Private	Доступ можливий з даного класу
Params	Передає параметри
Void	Методи класу
ref	Параметр із посиланням
Out	Параметр виводу
Return	Повертає управління об’єкту який викликає
Static	Оголошення статичного методу
Virtual	Оголошення віртуального методу

Цілочисельні бітові оператори використовуються для цілих числових типів даних - long, int, short, char і byte, визначено додатковий набір операторів, за допомогою яких можна перевіряти і модифікувати стан окремих бітів відповідних значень. Оператори бітової арифметики працюють з кожним бітом як з самостійною величиною.

Оператори відношення (також називають операторами порівняння). Вони визначають відношення одного операнду до іншого. Зокрема, вони визначають рівність та впорядкування операндів.

Оператори швидкої оцінки логічних виразів (short circuit logical operators)

Таблиця 3.3 - Бітова оператори C#

Оператор	Результат
~	побітове унарное заперечення (NOT)
&	побітове І (AND)
|	побітове АБО (OR)
^	побітове виключає АБО (XOR)
>>	зсув вправо
>>>	зрушення управо із заповненням нулями
<<	зсув вліво
&=	побітове І (AND) з привласненням
|=	побітове АБО (OR) з привласненням
^=	побітове виключає АБО (XOR) з привласненням
>> =	зсув вправо з привласненням
>>>=	зрушення управо із заповненням нулями з привласненням
<<=	зсув вліво з привласненням

Таблиця 3.4 – Оператори відношення C#

Оператор	Опис
==	Рівно
!=	Не рівно
>	Більше
<	Менше
>=	Більше рівне
<=	Менше рівне

Існують два цікавих доповнення до набору логічних операторів. Це - альтернативні версії операторів AND і OR, службовці для швидкої оцінки

логічних виразів. Ви знаєте, що якщо перший операнд оператора OR має значення true, то незалежно від значення другого операнда результатом операції буде величина true. Аналогічно в разі оператора AND, якщо перший операнд - false, то значення другого операнда на результат не впливає - він завжди буде дорівнює false. Якщо ви в використовуєте оператори && і || замість звичайних форм & і |, то C# не проводить оцінку правого операнда логічного виразу, якщо відповідь ясна з значення лівого операнда. Загальноприйнятою практикою є використання операторів && і || в переважній більшості випадків оцінки булевих логічних виразів. Версії цих операторів & і | застосовуються тільки в бітової арифметики.

Тернарний оператор if-then-else

Загальна форма оператора if-then-use така:

вираз1? вираз2: вираз3

Як перший операнд - «вираз1» - може бути використано будь-який вираз, результатом якого є значення типу boolean. Якщо результат дорівнює true, то виконується оператор, заданий другим операндом, тобто, «вираз2». Якщо ж перший операнд рівний false, то виконується третій операнд - «вираз3». Другий і третій операнди повинні повертати значення одного типу і не повинні мати тип void.

Спадкування класів – дуже потужна можливість в об'єктно орієнтованому програмуванні. Воно дозволяє створювати похідні класи (класи спадкоємці), взявши за основу всі методи і елементи базового класу (класу батька). Таким чином економиться маса часу на написання і налагодження коду нової програми. Об'єкти похідного класу вільно можуть використовувати все, що створено і налагоджено в базовому класі. При цьому, ми можемо в похідний клас, дописати необхідний код для удосконалення програми: додати нові елементи, методи і т.д.. Базовий клас залишиться недоторканим. Цю тему цілком можливо освоїти новачкам. Необхідно тільки познайомитися з синтаксисом і деякими особливостями.

Спадкування – це визначення похідного класу, який може звертатися до всіх елементів і методам базового класу за винятком тих, які перебувають у полі private;

Похідний клас ще називають нащадком або подклассом, а базовий – батько або надклас;

Синтаксис визначення похідного класу: class Імя_Похідного_Класса : специфікатор доступу Імя_Базового_Класса { /*Код * / } ;

Похідний клас має доступ до всіх елементів і методам базового класу, а базовий клас може використовувати тільки свої власні елементи і методи.
У похідному класі необхідно явно визначати свої конструктори, деструктори і перевантажені оператори присвоювання через те, що вони не успадковуються від базового класу. Але їх можна викликати явним чином при визначенні конструктора, деструктора або перевантаження оператора присвоєння похідного класу, наприклад таким чином (для конструктора): Конструктор_Проізводного_Класса (/*параметри * /) : Конструктор_Базового_Класса (/*параметри * /) { } .

3.3 Бібліотека комп'ютерного зору EmguCV

Оскільки для вирішення задачі розпізнавання дорожніх знаків використовують технології комп'ютерного зору, було вибрано технологію OpenCV. Але дана технологія не використовується напряму мовою програмування C#, тому компанія Microsoft вирішила розробити додатковий модуль для взаємодії між всі продуктами Microsoft та бібліотекою комп'ютерного зору OpenCV і дали назву цій технології EmguCV [29].

Emgu CV - це кроссплатформенная оболонка .Net для бібліотеки обробки зображень OpenCV. Дозвіл виклику функцій OpenCV з мов, сумісних з .NET. Оболонку можна скомпілювати за допомогою Visual Studio, Xamarin Studio і Unity, вона може працювати в Windows, Linux, Mac OS, iOS і Android.

Emgu повністю написано на C #. Перевага полягає в тому, що він може бути скомпільовано в Mono і, отже, може працювати на будь-якій платформі, яку підтримує Mono, включаючи iOS, Android, Windows Phone, Mac OS X і Linux. Було витрачено багато зусиль, щоб отримати чисту реалізацію C #, оскільки заголовки повинні бути перенесені, в порівнянні з керованою реалізацією C ++, де файли заголовків можуть бути просто включені. Але воно того варте, якщо ви побачите, що Emgu CV працює на Fedora 10. Крім того, ви завжди будете почувати себе комфортно, знаючи, що ваш код кросплатформенний [30].

Після того як було описано основні процеси розробки слід представити на описати діаграму класів.

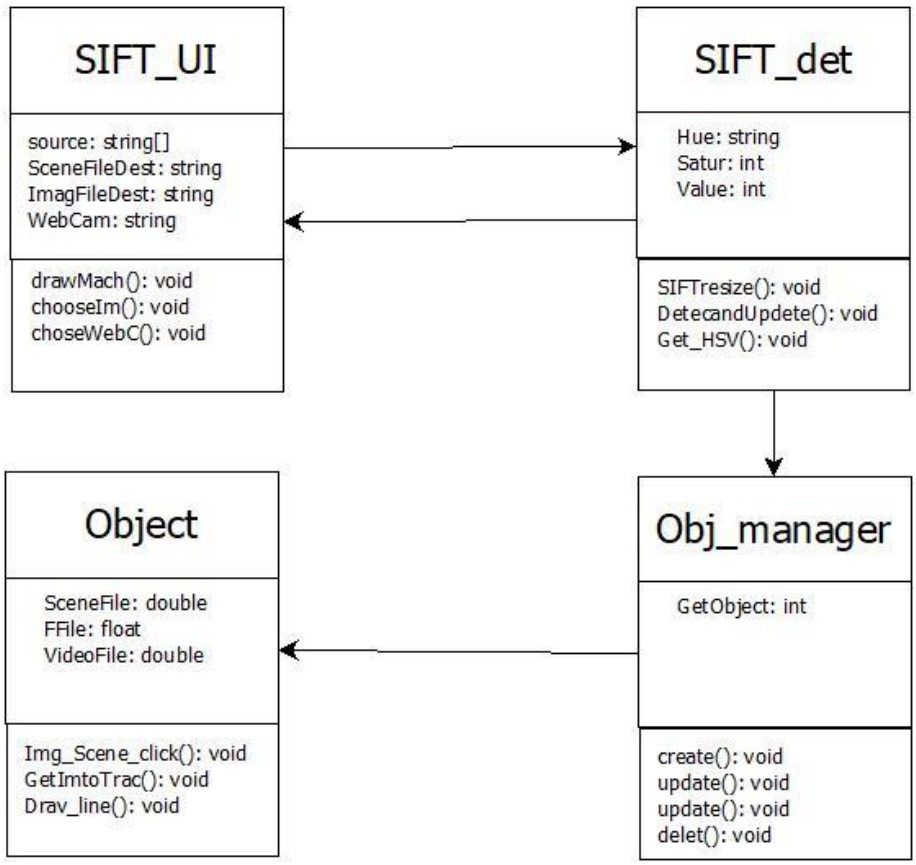

Рисунок3.1 – Діаграма класів інформаційної технології розпізнавання зображень дорожніх знаків за допомогою дескрипторів

Дана діаграма описує основні класи програмного продукту та об’єкти і методи які будуть використовуватися в подяльшому:

1. SIFT_UI;
2. SIFT_det;
3. Obj_manager;
4. Object.

Клас SIFT_UI служить для реалізації графічного інтерфейсу користувача за допомого WindowsForm. Даний клас містить методи які виконують запити на вибір режиму роботи розпізнавання, та безпосередньо відображення результатів роботи.

Клас SIFT_det відповідає за роботу з палітрою зображення, оновляє можливість порівняння нових зображень без необхідності перезапуску програми.

Клас Obj_manager служить для звернення до класу Object для подальшої роботи із зображеннями.

Клас Object зберігає у собі інформацію про файли з якими проводиться розпізнавання та співставлення

3.4 Розробка алгоритму розпізнавання дорожніх знаків

Блок схема алгоритм роботи додатку розпізнавання дорожніх знаків за допомого дескрипторів представлена на рисунку 3.2.

У першому блоці ініціалізуються компоненти бібліотеки комп’ютерного зору OpenCV.

У другому блоці представлено варіант вибору режиму роботи додатку розпізнавання зображень.

У третьому блоці ініціалізуються методи які відповідають за розпізнавання знаків у реальному часі та вмикається веб камера.

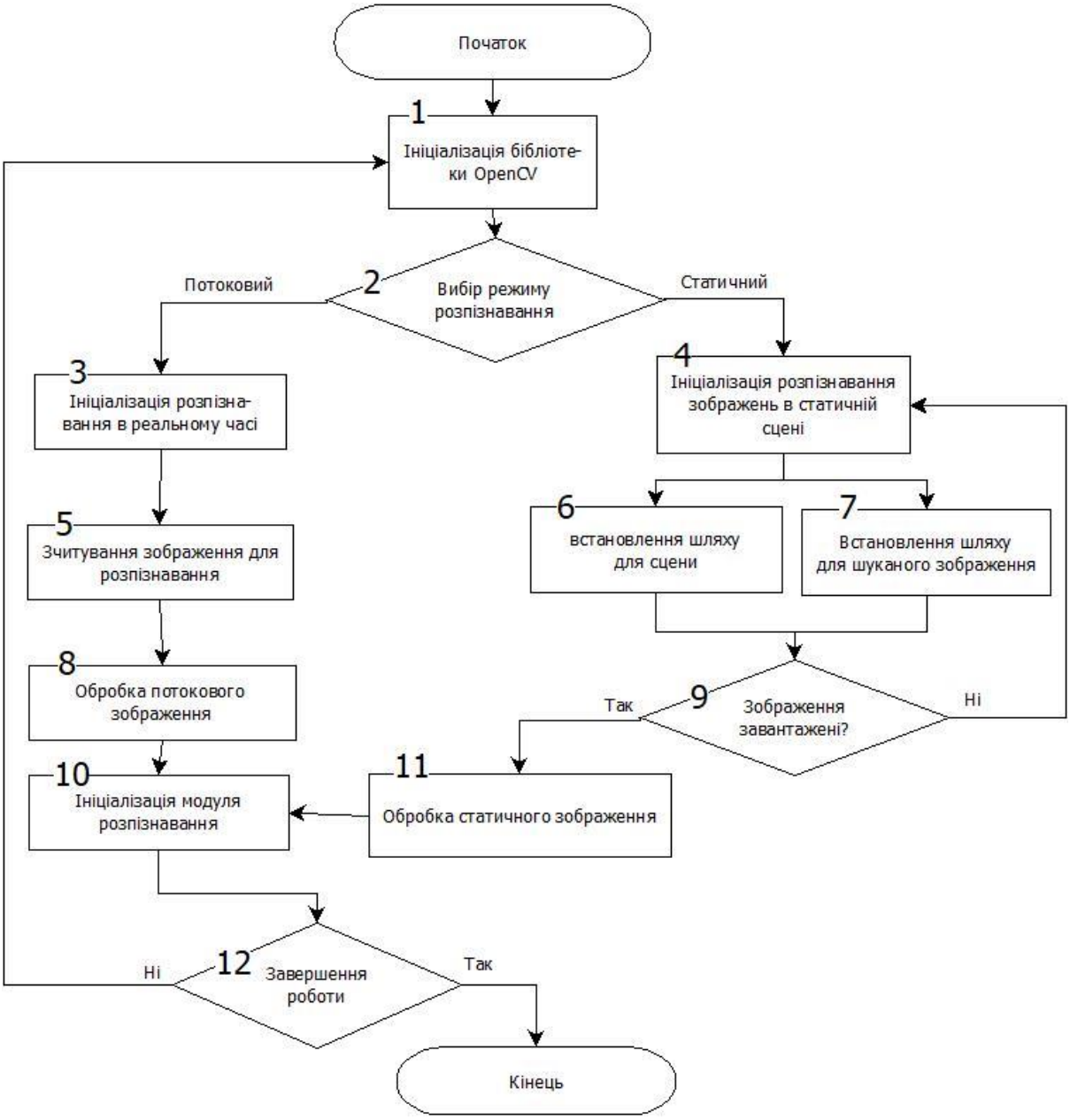

Рисунок 3.2 – Блок схема робота додатку розпізнавання дорожніх знаків

У четвертому блоці розблоковуються сховища в які в подальшому використовуються для виконання розпізнавання.

У п’ятому блоці відбувається знімок за допомогою веб камери, це зображення і буде еталонним про порівнянні.

У шостому блоках шість та сім виконується встановлення шляху програми до сховища із зображеннями для тестування.

У блоках вісім та одинадцять відбувається детектування зображень для різних методів розпізнавання.

У блоці дев’ять відбувається перевірка на те що два зображення завантаженні правильно і можна відобразити.

Блок десять відповідає безпосередньо за операцію розпізнавання за допомогою дескрипторів.

Блок одинадцять дає змогу завершити розпізнавання або ж розпочати його спочатку.

3.5 Тестування розробленого програмного засобу

Розроблена система розпізнавання дорожніх знаків пройшла глибоке тестування, в результаті чого було підтверджено її коректність роботи.

Було проведено близько 100 тестових запусків програмного продукту, і після чого була змога адекватної оцінки її роботи.

Після запуску програмного продукту перед користувачем з’являється робоча область(рисунок 3.1).

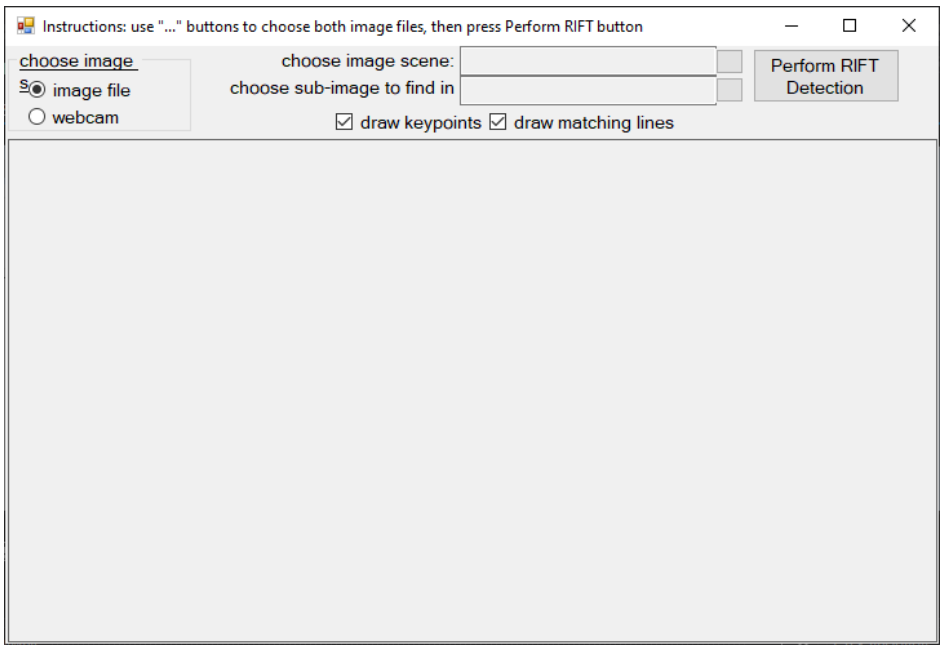

Рисунок 3.1 – Вікно програмного продукту розпізнавання дорожніх знаків

В програмному продукті присутні клавіші «choose image scene», що означає, для початку потрібно вибрати зображення – сцену на якому буде відбуватися пошук.

Клавіша «choose sub-image to find in» відповідає за зображення яке необхідно знайти.

Після їх завантаження вікно програми буде мати наступний вигляд(рисунок 3.2).

Рисунок 3.2 – Завантаження зображень в код програми

Далі виконується клавіша «Perform SIFT detection», тобто виконання виявлення на основі SIFT дескрипторів, що буде мати наступний результат(рисунок 3.3).

Рисунок 3.3 – Активації функції пошуку

Для подальшої перевірки завдання ускладняється і тепер потрібно знайти знак у темну пору доби що і відбувається на рисунку 3.4.

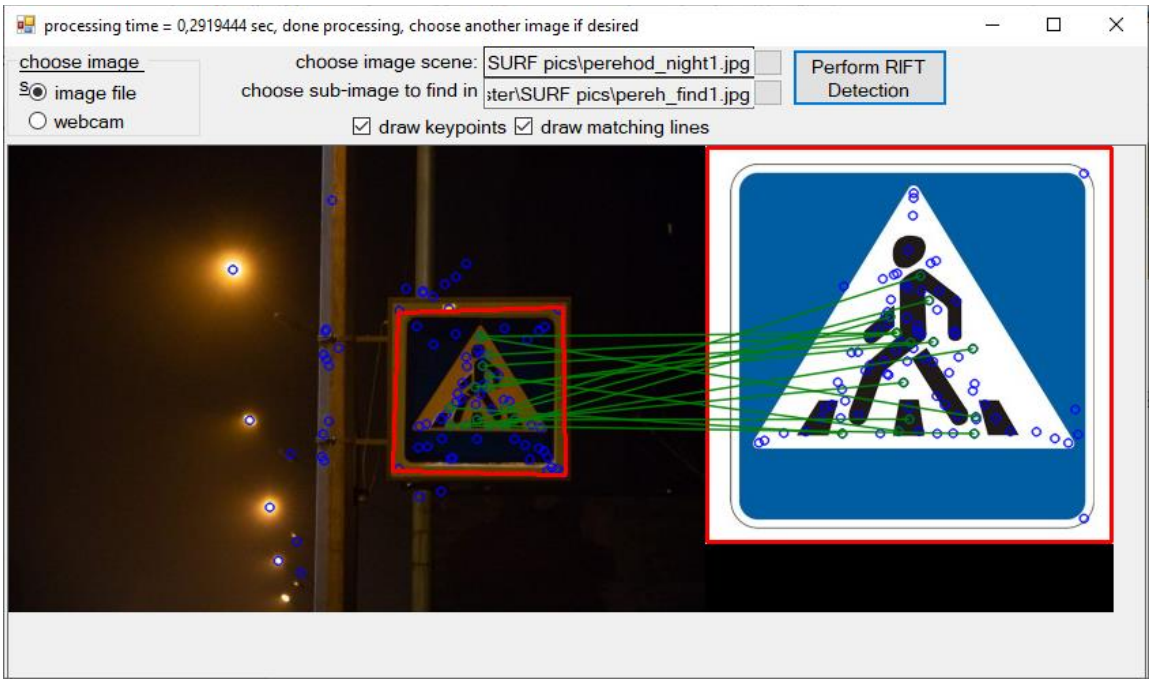

Рисунок 3.4 – пошук зображення у темну пору доби

Далі було проведено перевірку пошуку знаку який знаходиться за перешкодою. Результати даного випробування приведені на зображенні 3.5. На даному зображенні починають виявлятися певні дефекти пошуку, проте програмному продуктові все ж вдається розпізнати шуканий знак дорожнього руху.

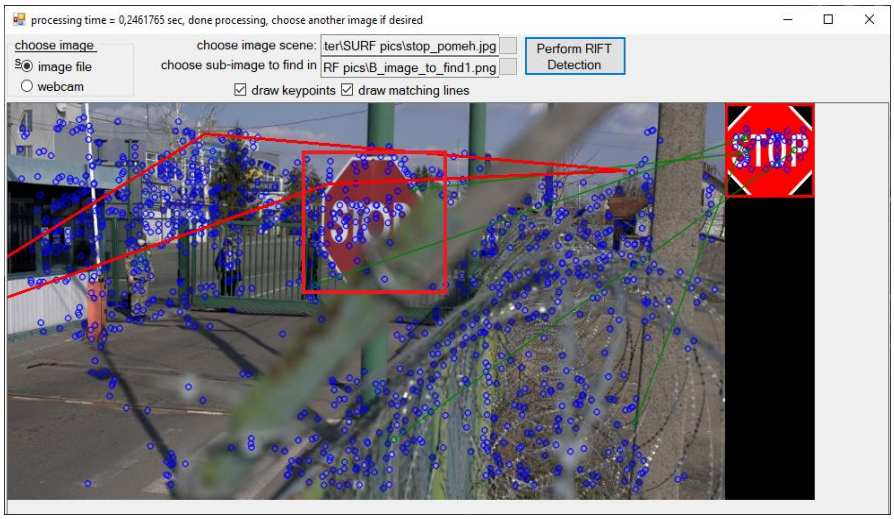

Рисунок 3.5 – пошук дорожнього знаку який знаходиться за завадами

Можна також зазначити, що знак не завжди дивиться строго на водія, тому наступним тестом буде пошук знаку під нахилом, що представлено на рисунку 3.6.

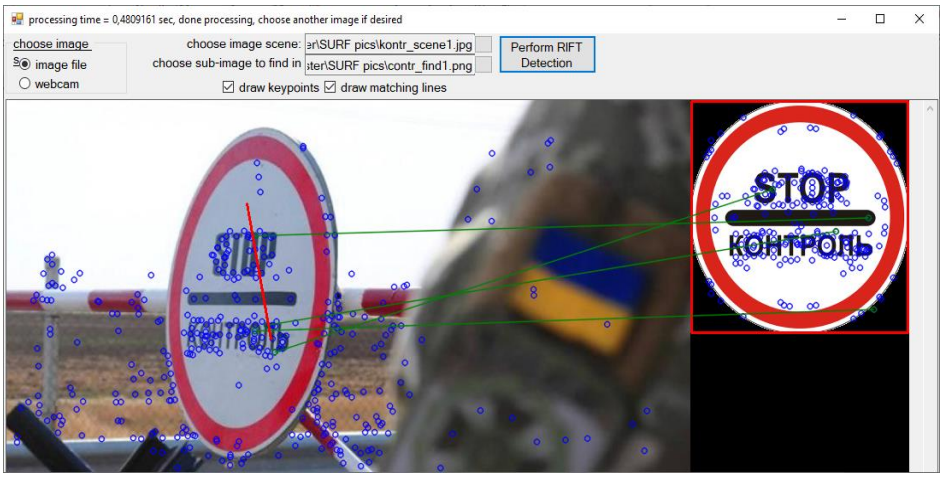

Рисунок 3.6 – Пошук дорожнього знаку розташованого під кутом нахилу до спостерігача

В даному тесті було виявлено те що програмний продукт, має певні проблеми у режимі роботи пошуку дорожнього знаку, проте хоч і частково але знак було знайдено.

Підсумовуючи всі проведені типи тестів можна сказати наступні висновки: програмний продукт знаходить зображення дорожніх знаків у звичайну погоду та нічний час без перешкод, Під час тестування пошуку дорожнього знаку, який був не повністю видимий програма видавала графічні артефакти, але також успішно виконувала поставлені задачі. Проблеми розпочалися під час останнього типу тесту, коли потрібно було розпізнати знак нахилений під певним кутом. При проведені даного тесту програмний продукт виявив задовільний результат. Результати, які були отримані під час тестування програмного продукту, відповідають результатам які очікувалися.

Оцінювання результатів розробки проводилося наступним чином: було відібрано зображення в чотири категорії, а саме денний час, нічний час, зображення з завадами та зображення під нахилом. Далі зображення проходили тестування в програмному продукті розробленому в магістерській кваліфікаційній роботі та в програмі аналогові. Результати даного тестування приведені в таблиці 3.1.

Таблиця 3.1 – Порівняння роботи програмного продукту з аналогом

	Аналог	SIFT розпізнавання
Достовірність розпізнавання в світлу пору доби(%)	94	97
Достовірність розпізнавання в темну пору доби(%)	93	96

Достовірність розпізнавання знаків з перешкодами(%)	79	85
Достовірність розпізнавання знаків під нахилом(%)	60	62
Середня розрахована достовірність розпізнавання	81,5	85

Розроблений програмний додаток працює відповідно до проекту завдання. Розроблений додаток має кращу достовірність розпізнавання, яка зросла в середньому на 3,5% порівняно з аналогом.

3.6 Висновок

У даному розділі розглянуто особливості архітектурної організації обчислень за допомогою дескрипторів і обгрунтовано доцільність їх використання. Здійснено вибір мови програмування, інтегрованого середовища розробки, бібліотеки комп'ютерного зору. В роботі використано мову програмування C# та бібліотеку OpenCV, які є однією із найпоширеніших бібліотек, яка використовується в задачах комп'ютерного зору, а Visual Studio 2019 common є найзручнішим середовищем розробки для даної мови програмування. Розроблено алгоритми пошуку та співставлення зображень. На основі цього було програмно розроблено програмний продукт. Проведене тестування розробленого програмного продукту підтвердило коректність його роботи. Також було проведено порівняльний аналіз створеного продукту з існуючими аналогами по критеріях «достовірність розпізнавання. В підсумку в порівнянні з аналогом достовірність розпізнавання зросла на 3.5%.

4 ЕКОНОМІЧНА ЧАСТИНА

4.1 Оцінювання комерційного потенціалу розробки

Метою проведення технологічного аудиту є оцінювання комерційного потенціалу розробки. Для проведення технологічного аудиту було залучено 2-х незалежних експертів. Такими експертами будуть Пересенчук Д.А. та Дядюсь Н.С.

Здійснюємо оцінювання комерційного потенціалу розробки за 12-ма критеріями за 5-ти бальною шкалою. Результати оцінювання комерційного потенціалу розробки наведено в таблиці 4.1.

Таблиця 4.1 – Результати оцінювання комерційного потенціалу розробки

Критерії	Прізвище, ініціали, посада експерта	
	Пересенчук Д.А.	Дядюсь Н.С.
	Бали виставлені експертами	
1	3	5
2	5	5
3	4	4
4	5	4
5	3	3
6	4	5
7	5	4
8	3	3
9	4	5
10	4	4
11	5	5
12	5	3
Сума балів	50	50
Середньоарифметична сума балів СБ	$СБ = \frac{\sum_1^2 СБ_i}{2} = 50$	

Отже, з отриманих даних таблиці 4.1 стає зрозуміло, що нова розробка має високий рівень комерційного потенціалу.

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

Для розробки нового програмного продукту необхідні такі витрати. Основна заробітна плата для розробників визначається за формулою (4.1):

$$З_о = \frac{М}{Т_р} * t \qquad (4.1)$$

де М – місячний посадовий оклад конкретного розробника;

Тр – кількість робочих днів у місяці,

Тр = 22 дні;

t – число днів роботи розробника, t = 50 днів.

Розрахунки заробітних плат для керівника і програміста наведені в таблиці 4.2.

Таблиця 4.2 – Розрахунки основної заробітної плати

Працівник	Оклад М, грн	Оплата за робочий день, грн	Число днів роботи, t	Витрати на оплату праці, грн.
Науковий керівник	7000	318,18	5	1590,9
Інженер програміст	5000	227,27	30	6818,1
Всього:				8409

Розрахуємо додаткову заробітну плату:

$$З_{дод}=0,1*8409=840,9(грн.)$$

Нарахування на заробітну плату операторів $Н_{зп}$ розраховується як 22% від суми їхньої основної та додаткової заробітної плати:

$$Н_{зп}=(З_о+З_{дод})*β/100, \quad (4.2)$$

$$Н_{зп}=(8409+840,9)*22/100=2034,98(грн.)$$

Розрахунок амортизаційних витрат розраховується за такою формулою:

$$А=\frac{Ц*Н_а}{100}*\frac{Т}{12} \quad (4.3)$$

де Ц – балансова вартість обладнання, грн;

$Н_а$ – річна норма амортизаційних відрахувань % (для програмного забезпечення 25%);

Т – Термін використання (Т=1 міс.).

Таблиця 4.3 – Розрахунок амортизаційних відрахувань

Найменування програмного забезпечення	Балансова вартість, грн.	Норма амортизації, %	Термін використання, міс.	Величина амортизаційних відрахувань, грн.
Персональний комп’ютер	10000	25	1	208,33
Всього:				208,33

Розрахуємо витрати на комплектуючі. Витрати на комплектуючі розрахуємо за формулою:

$$К=\sum_1^n Н_і * Ц_і * К_і, \quad (4.3)$$

де n – кількість комплектуючих;

$Н_і$ - кількість комплектуючих і-го виду;

$Ц_і$ – покупна ціна комплектуючих і-го виду, грн;

$К_і$ – коефіцієнт транспортних витрат (приймемо $К_і$ = 1,1).

Таблиця 4.4 - Витрати на комплектуючі, що були використані для розробки ПЗ.

Найменування матеріалу	Одиниці виміру	Ціна, грн	Витрачено	Вартість витрачених матеріалів, грн.
Зовнішній HDD накопичувач	шт.	2000	1	2000
Флешка	шт.	160	1	160
Камера	шт.	200	1	200
Всього з урахуванням транспортних витрат:				2596

Витрати на силову електроенергію розраховуються за формулою:

$$В_е=В*П*Ф*К_п, \quad (4.5)$$

де В – вартість 1кВт-години електроенергії (В=1,68 грн/кВт);

П – установлена потужність комп’ютера (П=0,09кВт);

Ф – фактична кількість годин роботи комп’ютера (Ф=88 год.);

Кп – коефіцієнт використання потужності (Кп< 1, Кп = 0,7).

$$В_е=1,68*0,09*88*0,7=9,31(грн.)$$

Розрахуємо інші витрати $В_{ін}$.

Інші витрати $І_в$ можна прийняти як (100…300)% від суми основної заробітної плати розробників та робітників, які були виконували дану роботу, тобто:

$$В_{ін}=(1..3)*(З_о + З_{дод}). \quad (4.6)$$

Отже, розрахуємо інші витрати:

$$В_{ін}=1* (8409+840,9)=9249,9(грн.)$$

Сума всіх попередніх статей витрат дає витрати на виконання даної частини роботи:

$$В= З_о + З_{дод} + Н_{зп} + А + К + Е_в + І_в$$

$$В=8409 + 840,9 + 2034,98 + 208,33 + 2596 + 9,31 + 9249,9 = 23348,42 (грн.)$$

Розрахуємо загальну вартість наукової роботи $В_{заг}$ за формулою:

$$В_{заг}=В/\alpha \quad (4.7)$$

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

$$В_{заг}=23348,42/1=23348,42.$$

Прогнозування загальних витрат ЗВ на виконання та впровадження результатів виконаної наукової роботи здійснюється за формулою:

$$ЗВ=В_{заг}/\beta \quad (4.8)$$

де β – коефіцієнт, який характеризує етап (стадію) виконання даної роботи.

Отже, розрахуємо загальні витрати:

$$ЗВ= 23348 / 0,9 = 25942,69(грн.)$$

4.3 Прогнозування комерційних ефектів від реалізації результатів розробки

Спрогнозуємо отримання прибутку від реалізації результатів нашої розробки. Зростання чистого прибутку можна оцінити у теперішній вартості грошей. Це забезпечить підприємству (організації) надходження додаткових коштів, які дозволять покращити фінансові результати діяльності.

Оцінка зростання чистого прибутку підприємства від впровадження результатів наукової розробки. У цьому випадку збільшення чистого прибутку підприємства $\Delta П_i$ для кожного із років, протягом яких очікується отримання позитивних результатів від впровадження розробки, розраховується за формулою:

$$\Delta П_i= \sum_1^n (\Delta П_я * N + П_я \Delta N)_i \quad (4.9)$$

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

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

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

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

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

В результаті впровадження результатів наукової розробки витрати на виготовлення інформаційної технології зменшаться на 25 грн (що автоматично спричинить збільшення чистого прибутку підприємства на 25 грн), а кількість користувачів, які будуть користуватись збільшиться: протягом першого року – на 150 користувачів, протягом другого року – на 125 користувачів, протягом третього року – 100 користувачів. Реалізація інформаційної технології до впровадження результатів наукової розробки складала 500 користувачів, а прибуток, що отримував розробник до впровадження результатів наукової розробки – 400 грн.

Спрогнозуємо збільшення чистого прибутку від впровадження результатів наукової розробки у кожному році відносно базового.

Отже, збільшення чистого продукту $\Delta П_1$ протягом першого року складатиме:

$$\Delta П_1 = 25 \cdot 500 + (400 + 25) \cdot 150 = 76250 \text{ грн.}$$

Протягом другого року:

$$\Delta П_2 = 25 \cdot 500 + (400 + 25) \cdot (150 + 125) = 129375 \text{ грн.}$$

Протягом третього року:

$$\Delta П_3 = 25 \cdot 500 + (400 + 25) \cdot (150 + 125 + 100) = 171875 \text{ грн.}$$

4.4 Розрахунок ефективності вкладених інвестицій та період їх окупності

Визначимо абсолютну і відносну ефективність вкладених інвестором інвестицій та розрахуємо термін окупності.

Абсолютна ефективність $E_{абс}$ вкладених інвестицій розраховується за формулою:

$$E_{абс} = (\Delta П_i - PV), \qquad (4.10)$$

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

т – період часу, протягом якого виявляються результати впровадженої НДДКР, 3 роки;

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

t – період часу (в роках) від моменту отримання чистого прибутку до точки 2, 3,4.

Рисунок, що характеризує рух платежів (інвестицій та додаткових прибутків) буде мати вигляд, рисунок 4.1.

Рисунок 4.1 - Вісь часу з фіксацією платежів, що мають місце під час розробки та впровадження результатів НДДКР

Розрахуємо вартість чистих прибутків за формулою:

$$ПП = \sum_1^m \frac{\Delta П_i}{(1+\tau)^t} \qquad (4.11)$$

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

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

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

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

Отже, розрахуємо вартість чистого прибутку:

$$ПП = \frac{25942,69}{(1+0,1)^0} + \frac{76250}{(1+0,1)^2} + \frac{129375}{(1+0,1)^3} + \frac{171875}{(1+0,1)^4} = 391568,79(грн.)$$

Тоді розрахуємо $Е_{абс}$:

$$Е_{абс} = 391568,79 – 25942,69 = 365626,1(грн.)$$

Оскільки $Е_{абс} > 0$, то вкладання коштів на виконання та впровадження результатів НДДКР буде доцільним.

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $Е_в$ за формулою:

$$Е_в = \sqrt[Т]{1 + \frac{Е_{абс}}{PV}} - 1, \qquad (4.12)$$

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

PV – теперішня вартість інвестицій PV = ЗВ, грн;

$Т_ж$ – життєвий цикл наукової розробки, роки.

Тоді будемо мати:

$$Е_в = \sqrt[3]{1 + \frac{391568,79}{25942,69}} - 1 = 1,52 \text{ або } 152\ \%.$$

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

$$\tau = d + f \quad (4.13)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2020 році в Україні d = 0,2;

f – показник, що характеризує ризикованість вкладень, величина f = 0,1.

$$\tau = 0,2 + 0,1 = 0,3$$

Оскільки $E_в$ = 152% $>\tau_{мін}$ = 0,3 = 30%, то у інвестор буде зацікавлений вкладати гроші в дану наукову розробку.

Термін окупності вкладених у реалізацію наукового проекту інвестицій. Термін окупності вкладених у реалізацію наукового проекту інвестицій $T_{ок}$ розраховується за формулою:

$$T_{ок}=1/1,52=0,66 \text{ року.}$$

Обрахувавши термін окупності даної наукової розробки, можна зробити висновок, що фінансування даної наукової розробки буде доцільним.

4.5 Висновок

Отже, було проведено оцінювання комерційного потенціалу розробки, в результаті якого, було підтверджено, що розробка має досить високі можливості для комерційного використання. Також було виконано розрахунок основних витрат, які були необхідні під час науково-дослідної роботи, проектування та програмної реалізації. Загальні витрати становлять Після чого

було проведено підрахунок можливого прибутку після впровадження інформаційної технології. Перший рік використання планується отримання прибутку у 76250 грн , другий – 129375 грн, третій – 171875 грн. Загальна ефективність розробки становить 152%. Розрахований період окупності становить 0,66 року.

В загальному можна відзначити, що розробка має досить високий комерційний потенціал.

ВИСНОВКИ

В ході виконання магістерської кваліфікаційної роботи розроблено інформаційну технологію розпізнавання зображень дорожніх знаків на основі дескрипторів. Під час аналізу предметної області було відзначено, що розпізнавання зображень дорожніх знаків має можливість знайти своє застосування у багатьох сучасних сферах людської діяльності та значно підвищує рівень безпеки на дорозі. На жаль на сьогоднішній момент не існує такого пристрою який зміг би бути зручним у використанні, мав високі показники точності розпізнавання та був доступних для широких слоїв населення. Визначено основні проблеми при розпізнаванні дорожніх знаків. Аналіз даної предметної області показав, що найоптимальнішим варіантом вирішення даної проблеми є використання дескрипторів та бібліотек комп'ютерного зору.

Розроблено архітектуру програмного продукту, з покращенням, яке полягає, у одночасному використанні бібліотеки комп'ютерного зору та нового типу дескриптора, що дозволило підвищити достовірність розпізнавання та скоротити час роботи. В ході проведення кваліфікаційної роботи було запропоновано структуру інформаційної технології.

Було розроблено алгоритм програмного продукту, у основі якого було реалізовано програмний засіб розпізнавання зображень дорожніх знаків на основі SIFT дескрипторі за допомогою мови програмування C# та середовища розробки Visual Studio 2019 common, та з використанням бібліотеки комп'ютерного зору OpenCV.

Після проведення тестування програмного продукту результати показали, що архітектура SIFT дескрипторів дала можливість покращити достовірність розпізнавання в середньому на 3,5%. В результаті виконання даної магістерської кваліфікаційної роботи поставлені задачі були виконані в повній мірі. Тому, мета магістерської роботи досягнута.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Василенко М. Ю., Колесницький О. К., «Аналіз методів розпізнавання зображення об’єктів», в *Матеріали конференції «XLIX Науково-технічна конференція підрозділів Вінницького національного технічного університету (2020)»*, Вінниця, 2020, с. 901-902. [Електронний ресурс]. Режим доступу: https://conferences.vntu.edu.ua/index.php/allvntu/index/pages/view/zbirn2020 Дата звернення: Черв. 2020.

2. . Навчальні курси дисципліни дорожнього транспорту [Електронний ресурс] – Режим доступа: https://zp.edu.ua/navchalni-kursydyscypliny-kafedry-transportnyh-tehnologiy

3. Загальний курс транспорту: підруч. для студ. вищ. навч. закл. напрямку підготовки «Транспорт. технології»: присвяч. світлій пам'яті колиш. зав. каф. «Транспорт. технології» проф. Л. Ю. Яцківського / М. Ф. Дмитриченко, І. І. Кельман, Є. К. Вільковський та ін. ; М-во освіти і науки, молоді та спорту України, Нац. транспорт. ун-т. — Л. : [б. в.], 2011. — 524 с. : іл. — Бібліогр.: с. 519—521 (34 назви). — ISBN 978-617-629-026-1.

4. Цент громадського здоров’я МОЗ України. Кількість ДТП у світі зростає [Електронний ресурс] – Режим доступа: https://phc.org.ua/news/kilkist-dtp-u-sviti-zrostae-vooz.

5. Сегментація зображень засобами openCV [Електронний ресурс] – Режим доступа: http://ai-tern.in.ua/Segmentation.html.

6. Записки системного администраторпа. Что такое RGB, CMYK, HSV+HSL, Lab — цветовые модели и параметры [Електронний ресурс] – Режим доступа: https://sonikelf.ru/znakomimsya-s-cvetom-vsyo-o-cvetovyx-modelyax-rgb-cmyk-hsvhsl-lab/.

7. Opel Astra OPC Arden Lightning II › Бортжурнал › Обзор и освещение 2 или умный Opel Eye [Електронний ресурс] – Режим доступа: https://www.drive2.ru/l/2394702/.

8. Road Sign Information and sign display [Електронний ресурс] – Режим доступа: https://www.volvocars.com/uk/support/manuals/v90/2018w46/driver-support/road-sign-information/road-sign-information-and-sign-display.

9. HSL и HSV (цветовые модели) [Електронний ресурс] – Режим доступа: https://science.wikia.org/ru/wiki/HSL_%D0%B8_HSV_(цветовые модели).

10. Theory perceive the color, introduction to graphic of computer

11. Object Recognition from Local Scale-Invariant Features [Тескт] / David G. Lowe

12. Tuytelaars, T., & Schmid, C. (2007, October). Vector quantizing feature space with a regular lattice. In 2007 IEEE 11th International Conference on Computer Vision (pp. 1-8). IEEE.

13. Mikolajczyk, K., & Schmid, C. (2005). A performance evaluation of local descriptors.

14. Rublee, E., Rabaud, V., Konolige, K., & Bradski, G. R. (2011, November). ORB: An efficient alternative to SIFT or SURF. In ICCV (Vol. 11, No. 1, p. 2).

15. НОУ ИНТУИТ. Детекторы и дескрипторы ключевых точек. Алгоритмы классификации изображений [Електронний ресурс] – Режим доступа: https://intuit.ru/studies/courses/10621/1105/lecture/17983?page=2.

16. Построение SIFT-дескрипторов и нахождение особых точек на изображениях / А.С. Сафонов.

17. Форсайт, Д. А., & Понс, Ж. (2004). Компьютерное зрение. Современный подход. М.: Изд. дом „Вильямс“, 928 с.

18. OpenCV foundation. (2017). OpenCV: Image Thresholding. Retrieved from [Електронний ресурс] – Режим доступу https://docs.opencv.org/3.3.0/d7/d4d/tutorial_py_thresholding.html

19. Microsoft | Visual studio [Електронний ресурс] – Режим доступа: https://visualstudio.microsoft.com/ru/

20. Розробка мобільних додатків від А до Я: повний гайд [Електронний ресурс] – Режим доступа: https://dan-it.com.ua/uk/rozrobka-mobilnih-dodatkiv-vid-a-do-ja-povnij-gajd/.

21. Оновлення для платформи Microsoft .NET Native [Електронний ресурс] – Режим доступа: https://www.microsoft.com/uk-ua/download/developer-tools.aspx.

22. І розробка програмного забезпечення, і веб-розробка починаються з одного рядка коду [Електронний ресурс] – Режим доступа: https://svcministry.org/uk/dictionary/brackets-vs-notepad/.

23. The entire Pro Git book, written by Scott Chacon and Ben Straub [Електронний ресурс] – Режим доступа: https://git-scm.com/book/uk/v2/Вступ-про-систему-контролю-версий/

24. Облачная платформа Azure включает более 200 продуктов и облачных служб [Електронний ресурс] – Режим доступа: https://azure.microsoft.com/ru-ru/overview/what-is-azure/.

25. Microsoft оголосила, що Visual Studio 2019 для Windows і Mac вже у відкритому доступі [Електронний ресурс] – Режим доступа: https://codeguida.com/post/1754.

26. Going forward, the .NET team is using https://github.com/dotnet/runtime to develop the code and issues formerly in this repository. [Електронний ресурс] – Режим доступа: https://github.com/dotnet/coreclr.

27. Code-Lice.ru Портал для програмирования [Електронний ресурс] – Режим доступа: https://code-live.ru/tag/cpp-manual/.

28. Java oracle [Електронний ресурс] – Режим доступа: https://www.java.com/ru/download/.

29. Emgu CV is a cross platform .Net wrapper to the OpenCV image processing library. Allowing OpenCV functions to be called from .NET compatible languages [Електронний ресурс] – Режим доступа: http://www.emgu.com/wiki/index.php/Main_Page.

30. Emgu CV is a cross platform .Net wrapper for OpenCV [Електронний ресурс] – Режим доступа: https://sourceforge.net/projects/emgucv/files/.