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

Пояснювальна записка

до комплексної магістерської кваліфікаційної роботи

на тему «Інформаційна технологія розпізнавання нетипових ситуацій на відео: виявлення нетиповості ситуації»

Виконав: студент 2 курсу, групи 2КН-19 м
спеціальності 122 «Комп'ютерні науки»
Дерев’янко М. Ю.
Керівник: к.т.н., доц. Колесницький О.К.
Рецензент: канд. техн. наук, доцент
Рейда О.М.

Вінниця - 2020 року

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

_____
(підпис)
“____” _____2020 року

ЗАВДАННЯ

на комплексну магістерську кваліфікаційну роботу на здобуття кваліфікації магістра зі спеціальності___122 – «Комп'ютерні науки»_____

08-22.МКР.025.19.000.ПЗ

Магістранта групи 2КН-19м Дерев’янка Мирослав Юрійовича

Тема комплексної магістерської кваліфікаційної роботи: «Інформаційна технологія розпізнавання нетипових ситуацій на відео: виявлення нетиповості ситуації»

Вхідні дані: растрове зображення розміром 448х448, набір визначених об’єктів – список з координатами та розмірами, тестова вибірка з нетиповими ситуаціями не менше 50 ситуацій.

Короткий зміст частин магістерської кваліфікаційної роботи:

1. Графічна: схема алгоритму визначення нетипових ситуацій, загальна структурна схема програми, структура інформаційної технології розпізнавання нетипових ситуацій на відео, приклад виконання програми.
2. Текстова (пояснювальна записка): вступ, аналіз сучасного рівня розвитку інформаційної технології розпізнавання нетипових ситуацій на відео, розробка інформаційної технології розпізнавання нетипових ситуацій на відео, програмна реалізація інформаційної технології розпізнавання нетипових ситуацій на відео, 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 нетипових ситуацій на відео. Постановка задач дослідження			Аналітичний огляд літературних джерел, задачі досліджень, розділ 1 ПЗ
2	Розробка методу та інформаційної технології розпізнавання нетипових ситуацій на відео			Метод, інформаційна технологія, розділ 2
3	Програмна реалізація розробленої інформаційної технології, тестування та оцінка параметрів			Програмне забезпечення, розділ 3
4	Підготовка економічної частини			розділ 4
5	Апробація та/або впровадження результатів дослідження			тези доповідей/акт впровадження
6	Оформлення пояснювальної записки, графічного матеріалу та презентації			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. екон. наук., доц. каф. ЕПВМ
(підпис) наук. ступінь, вчене звання (посада)
_____М. В. Бальзан_____
ініціали та прізвище
“____” _____20___ р.

Дата попереднього захисту роботи “____” _____20___ р.

Рецензент _____ _____к.т.н., 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. Дерев’янко_____
(підпис) ініціали та прізвище
“____” _____20___ р.

АНОТАЦІЯ

У даній комплексній магістерській кваліфікаційній роботі було виконано розробку та програмну реалізацію інформаційної технології виявлення нетипових ситуацій на відео, в частині виявлення нетипових ситуацій. Було здійснено аналіз методів відстеження та аналізу сцен, було обрано метод аналізу, що ґрунтується на квадратичному відхилені. Програмна реалізація виконувалася за допомогою бібліотеки OpenCV з використанням мови Python, графічний інтерфейс був викоанний на мові програмування JavaScript з використанням HTML/CSS.

Вхідною інформацією модуля є вихідне зображення відеопотоку, набір об’єктів та налаштування, а вихідними – зображення з виявленою нетиповою ситуацією. Результати тестування показали збільшення точності на 9% та збільшення швидкості роботи майже у 7 разів за допомогою викорстання багатопоточності та технології CUDA.

ABSTRACT

In this complex master's qualification work, the development and software implementation of information technology for detecting atypical situations on video, in terms of detecting atypical situations, was performed. The analysis of methods of tracking and analysis of scenes was carried out, the method of the analysis based on a quadratic deviation was chosen. The software implementation was performed using the OpenCV library using the Python language, the graphical interface was implemented in the JavaScript programming language using HTML / CSS..

The input information of the module is the output image of the video stream, a set of objects and settings, and the output information is the image with the detected atypical situation. The test results showed an increase in accuracy by 9% and an increase in speed of almost 7 times through the use of multithreading and CUDA technology.iii

ЗМІСТ

ВСТУП

Актуальність теми дослідження. На сьогоднішній день небезпека дорожньо-транспортних пригод посідає одне з передових місць у всьому світі. За даними Всесвітньої організації охорони здоров'я щороку гинуть більше 1,2 млн людей по всьому світу. При цьому отримують травми близько 20-50 млн. В Україні ця проблема також є суттєвою, за даними статистики 2019 року в країні відбулося 15450 дорожньо-транспортних пригод, у яких постраждало 2446 людей, з яких загинуло 334 людини.

Відсутність загальнодоступних, точних та швидких програмних засобів для детектування нетипових ситуацій на дорозі не дозволяє виконувати постіний цілодобовий контроль над дорогами міста. Для реалізації такого моніторингу потрібно залучення великої кількості ресурсів, а особливо людських ресурсів. Для цілодобового моніторингу необхідне вровадження трьох змін, які б давали змогу виконувати спостереження у нічний час. Крім того збільшення кількості камер спостереження також призводить до збільшення потреб у людських кадрах. В даній ситуації доцільним є створення програмного засобу для автоматизованого моніторингу стану на дорозі і розпізнавання нетипових ситуацій, щоб значно оптимізувало цей процес.

Моніторинг вулиць та доріг сьогоднішній день нечасто відбувається із застосуванням апаратних та програмних засобів. Такі засоби, орієнтовані на визначення збір та виведення інформації, наприклад отримання потокового відео з камер спостереження. Але саме фіксування та визначення ситуацій, що можуть статися, повністю покладається на оператора. Ця людина спостерігає за моніторами. Зрозуміло, що такий підхід містить досить сильний людський фактор. Також така робота змушує оператора знаходитись перед моніторами досить довгий час, що може негативно впливати на його здоров'я.

Інтелектуальна система моніторингу матиме ряд переваг на відмінну від звичайного моніторингу із залученням людської праці. По-перше, дана система значно скоротить використання людських ресурсів для проведення

моніторингу, окрім того вона зможе здійснювати аналіз в режимі 24/7, шо складно організувати використовуючи персонал. По-друге, система при належній точності, зможе виявляти ситуації, які може пропустити спостерігач, це в свою чергу виключає суб’єктивний влив на результати. Вона зможе значно збільшити виявлення правопорушень, що в свою чергу зробить дороги більш безпечними для учасників руху. По-третє, дана система матиме механізм сповіщення оператора про виявлення ситуацій, що залишить у оператора лише обовязок приймати рішення про ту чи іншу подію, але забере в нього обов’язок сидіти постіно перед моніторами. Це підвищить умови праці і зменшить негативний вплив на здоров’я.

Існуюча система із залученням великої кількості персоналу для моніторингу і спостереженням за станом дороги не є досить ефективною і є ресурсоємною, а також не відповідає вимогам навіть невеликого за маштабами міста.

Отже, задача автоматизації розпізнавання нетипових ситуацій носить важливий прикладний характер, і для її розв’язання відсутні достатньо ефективні програмні методи, що надає можливість застосування засобів штучного інтелекту для досягнення ефективнішого моніторингу та спостереження за станом дороги.

Тема магістерської кваліфікаційної роботи є актуальною, тому що існує потреба у сторенні інтелектуальної інформаціної технології, яка б мала змогу автоматизувати та полекшити процес спостереження. Дана технологія буде корисною для державних установ та міських адміністрацій, що хотіли б покращити проживання людей у своєму місті та знизити показники ДТП та смертності постраждалих у них.

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 та швидкодії розпізнавання нетипових ситуацій на відео програмними засобами в частині визначення нетиповості ситуації за рахунок застосування статистичного аналізу параметрів трекінгу об’єктів.

Для досягнення поставленої мети необхідно розв’язати такі завдання:

- провести аналіз проблеми розв’язання задачі розпізнавання нетипової ситуації на відео;
- розглянути існуючі методи вирішення задачі розпізнавання нетипової ситуації на відео та обрати й обґрунтувати вибір методу, який задовольняє мету даної магістерської кваліфікаційної роботи;
- розробити математичну модель інформаційної системи розпізнавання нетипових ситуацій на відео та удосконалити її згідно з метою роботи;
- сформулювати стадії інформаційної технології та на їх основі розробити структуру та алгоритм роботи програмного засобу;
- виконати програмну реалізацію запропонованої інформаційної технології розпізнвання нетипових ситуацій на відео;
- провести тестування програмного продукту та виконати аналіз отриманих результатів.

Об’єкт дослідження – процес розпізнавання нетипових ситуацій на відео з використанням штучних нейронних мереж.

Предмет дослідження – інформаційна технологія та програмні засоби розпізнавання нетипових ситуацій на відео з використанням штучних нейронних мереж та достовірність і швидкодія їх роботи.

Методи дослідження. У роботі використані наступні методи наукових досліджень: системний аналіз для аналізу структури інформаційної системи, теорія нейронних мереж для реалізації інформаційної технології розпізнавання

нетипових ситуацій на відео, методи математичної статистики для розробки математичної моделі визначення нетипової ситуації, об’єктно-орієнтоване програмування для реалізації програмного продукту.

Наукова новизна одержаних результатів полягає в наступному:

- вперше запропоновано інформаційну технологію розпізнавання нетипових ситуацій на відео, яка використовує глибоку згорткову нейрону мережу, а також удосконалений метод визначення нетиповості ситуації, що дозволяє підвищити достовірність та швидкодію розпізнавання нетипових ситуацій на відео;
- удосконалено метод визначення нетиповості ситуації, який відрізняється використанням статистичного аналізу параметрів трекінгу об’єктів, що дозволяє підвищити достовірність розпізнавання нетипових ситуацій на відео;
- удосконалено процес розпізнавання нетипових ситуацій на відео за рахунок паралелізації та використання обрахункових потужностей графічного ядра, що дозволяє досягнути вищої продуктивності роботи мережі та, відповідно, підвищення швидкодії розпізнавання нетипових ситуацій на відео.

Практичне значення одержаних результатів полягає у наступному:

1. Розроблено новий алгоритм розпізнавання нетипових ситуацій, який використовує дані з потокового відео для визначення ситуацій.
2. Розроблено математичну модель для визначення нетипових ситуацій.
3. Розроблено програмний засіб для розпізнавання нетипових ситуацій

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 магістранта. Усі результати, наведені у магістерській кваліфікаційній роботі, отримані самостійно. У роботах, опублікованих у співавторстві, автору належать такі результати: [1] – дослідження застосування згорткової нейронної мережі у задачі розпізнавання нетипових ситуацій. [2] – експериментальні дослідження методу визначення нетипових ситуацій; [3] – розробка алгоритму та дослідження ефективності визначення нетипових ситуацій;

Апробація результатів роботи. Результати роботи були апробовані на 1 міжнародній науково-практичній конференції [2], 1 регіональній науково-технічній конференції [3]: XX Міжнародній науково-практичній конференції ІОН 2020 (м. Вінниця, Україна, 2020 р.), XLIX Науково-технічна конференція факультету інформаційних технологій та комп'ютерної інженерії, (м. Вінниця, Україна, 2020 р.).

Публікації. За результатами магістерської кваліфікаційної роботи опубліковано 1 стаття у фаховому журналі [1], 3 тези доповіді конференцій [2, 3, 4], подано заявку на авторське свідоцтво на твір (програму).

1 АНАЛІЗ СУЧАСНОГО РІВНЯ РОЗВИТКУ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ РОЗПІЗНАВАННЯ НЕТИПОВИХ СИТУАЦІЙ НА ВІДЕО

1.1 Постановка проблеми визначення нетипових ситуацій на відео

Як було загадно у вступі, проблема автоматизації розпізнавання нетипових ситуацій на відео є досить актуальною через виникнення великої кількості дорожньо транспортних пригод на дорогах України та у всьому світі загалом [5, 6].

Для того щоб проаналізувати проблему визначення нетипових ситуацій на відео потрібно визначити основні дані, якими потрібно керувати, та середовище. Для цього проаналізуємо схему зображену на рис.1.1.

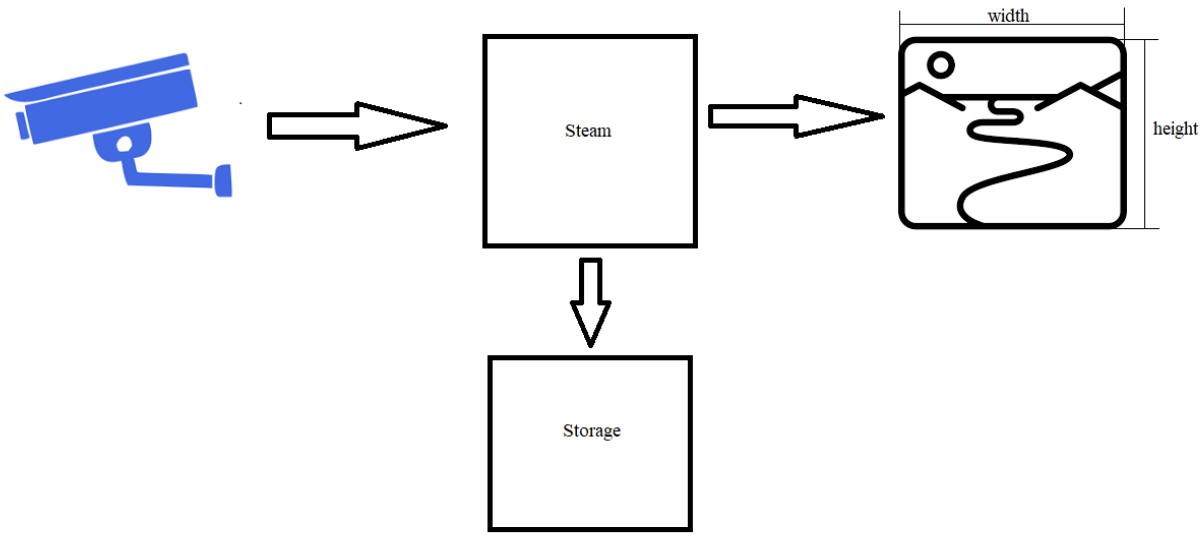

Рисунок 1.1 – Схематичне зображення середовища та вхідних даних розпізнавання нетипових ситуацій на відео

У більшості систем спостереження існує записуючий засіб, зазвичай цей засіб це відеокамера чи камера спостереження. Дана камера формує потік даних. Ці дані являють собою набором растрових зображень, що має певну роздільну здатність, тобто висоту та ширину. Інколи дані записуються на

внутрішнє сховище записуючого пристрою, або ж на зовнішнє велике сховище. Звукові дані дуже рідко використовуються саме під час спостереження. По-перше віддаленість камер від місця, за яким ведеться спостереження. По друге, наявністю шумів, що можуть перешкоджати хорошому звуку. Отже, основні дані, які навяні під час вирішення проблеми виявлення нетипових ситуацій, це потік растрових двохвимірних зображень.

Розгланемо також наше растрове зображення. Кожен кадр відео являє собою сцену, тобто набір обєктів, які розташовані у двохвимірному просторі, приклад такої сцени зображений на рис. 1.2.

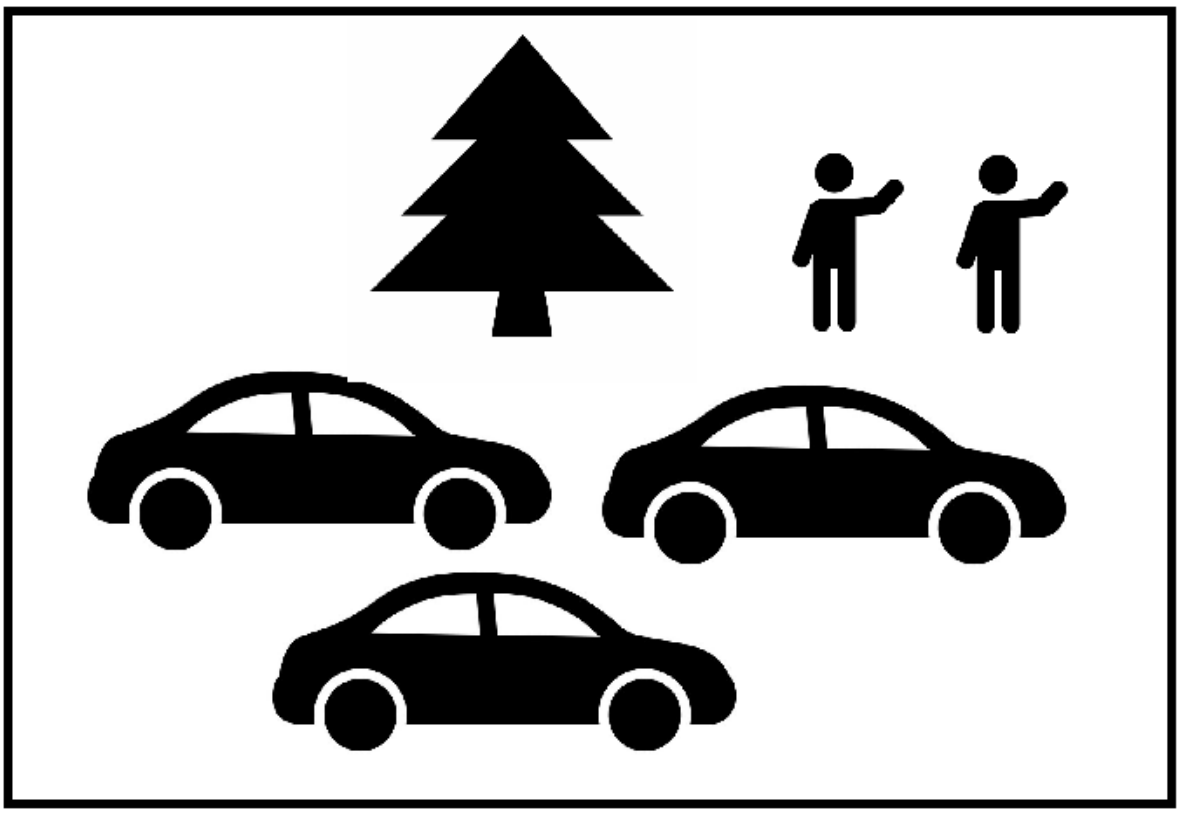

Рисунок 1.2 – Приклад сцени, що містить кадр вхідного потоку даних

На сцені ми маємо набір обєктів. Кожен об’ект має свої характеристтики, а саме розмір і розташування на зображенні. Перш за все потрібно розуміти, які обєкти є актуальними і пріоритетними для визначення нетипових ситуацій. Тобто має місце бути фільтрування об’єктів. Очевидно що статичні об’єкти такі

дерево, будівлі, паркани, тощо – є неприорітетними. Динамічні і рухомі обєкти є основними факторами, що можуть спричиняти нетипові ситуації. Це означає, що потрібно виконувати детектування динамічних об’єктів на сцені. Але щоб визначити те, що обєкт є динамічним і властивості його руху потрібно проаналізувати не одне зображення, а декілька. Тобто потрібно відслідковувати зміну положення чи розмірів об’єкту від однієї сцени до іншої. Отже, зображення містить набір об’єктів, які мають своє положення та свої розміри. Ці дані є основними при задачі виявлення нетипових ситуацій на відео.

Для більш ефективної постановки задачі потрібно розбити її на менші підзадачі, які простіші у вирішенні. Можна виокремити основні під задачі:

1) Отримання списку виявлених об’єктів
2) Збір інформації про зміни станів об’єктів
3) Аналіз змін та виявлення нетипових ситуацій

Як було сказано вище для початку потрібно визначити усі релевантні обєкти на сцені (зображенні), які необхідні для визначення нетипових ситуацій на відео. Для цього потрібно розпізнати та визначити розімри та розташування обєктів. Для виконання цієї під задачі буде використовуватися нейрона мережа [7]. Але так як на зображені будуть знаходитися багато потенційних об’єктів потрібно використовувати згорткову нейрону мережу, адже вона досить ефективно здатна знаходити за заданим ядром об’єкти [8].

Далі потрібно виконати збір інформації між сценами для того, щоб визначити динаміку (рух) обєктів на відео. Для цього потрібно індентифікувати вже виявлені обєкти з попередньої сцени на теперішній сцені, та визначити зміни, які сталися під час переходу. Для вирішення потрібно виконувати отриманням та збереженням станів об’єкту. Важливим на цьому етапі є отримання даних саме того об’єкту, якого потрібно, при цьому він здатний змінювати своє положення та розміри з часом. Вхідними даними даної під задачі є список вже виявлених об’єктів та наступне зображення, з якого потрібно визначити наступний стан об’єкту. Можна припустити, що для отримання положення на теперішньому зображені ми можемо використати

попередні дані про розташування об’єкту. Тому основними характеристиками, що можна отримати, є розташування об’єкту, що задається двома величинами x та y, адже отримані дані лише з 2D зображення, та розмірами, що задаються також двома величинами w та h, що є шириною та довжиною. Зрозуміло, що точність цих величин обмежена дискретним значеннями роздільної здатності зображення і можливо визначити лише до пікселя.

Завершенням є задача аналізу змін, які відбуваються на сценах з обєктами. Це пебзпосередня задача визначення нетипових ситуацій. Потрібно визначити за базовими характеристиками об’єкту, що відбулася якась нетипова аьо аномальна ситуація. Для прикладу можна навести різку зміну швикості чи напрямку руху. Тут вхідними даними виступають накопичені стани об’єктів. Особливістю даного етапу полягає у тому, що потрібно визначити основні механізми, що допоможуть фільтрувати аномальні об’єкти від звичайних.

1.2 Аналітичний огляд відомих методів та засобів визначення нетипових ситуацій на відео

Особливістю під задачі збору інформації, що було визначено у попередньому розділі, є отримання даних з динамічних об’єктів, що рухаються у часі, для цього потрібно отримати розташування об’єкту в теперішній момент часу ґрунтуючись на попередньому. Цю задачу можна вирушити з використанням відстеження.

Відстеженням (англ. tracking) називають визначення розташування рухомого об’єкта (декількох об’єктів) з часом за допомогою відеокамери. Алгоритм аналізує кадри відео і видає положення рухомих цільових об’єктів відносно кадру. [9]

Основною проблемою відстеження є зіставлення положень цільового об’єкта на послідовних кадрах, особливо якщо об’єкт рухається швидко порівняно з частотою кадрів. Таким чином, системи відстеження зазвичай

використовують модель руху, котра описує, як може змінюватись зображення цільового об’єкта при різноманітних його рухах.

Прикладами таких простих моделей руху є:

– відстеження плоских об’єктів, модель руху – двовимірне перетворення (афінне перетворення або гомографія) зображення об’єкта (наприклад, початкового кадру)

– коли цільовим є жорсткий тривимірний об’єкт, модель руху визначає вигляд залежно від його положення у просторі та орієнтації

– зображення деформованого об’єкта може бути покрите сіткою, рух об’єкта задають положенням вершин цієї сітки

Головне завдання алгоритму відстеження – це послідовний аналіз кадрів відео для оцінювання параметрів руху. Ці параметри характеризують положення цільового об’єкта.

Система візуального спостереження складається з двох основних частин:

– Подання та локалізація цільового об’єкта (Target Representation and Localization)

– Фільтрування та об’єднання даних (Filtering and Data Association)

Подання та локалізація цільового об’єкта є здебільшого висхідним процесом (англ. bottom-up process), тобто послідовним і його наступні кроки не зачіпають попередніх.

Фільтрування та об’єднання даних є здебільшого низхідним процесом (англ. top-down process), котрий включає об’єднання апріорної інформації про сцену або об’єкт, що співвідноситься з динамікою об’єкта та обчисленням різних гіпотез. Обчислювальна складність цих алгоритмів зазвичай висока.

Прикладом алгоритму фільтрування можна навести фільтр Кальмана [10].

Суть роботи фільтру Кальмана полягає у використанні наявних даних та у використанні випадкових величин теорії ймовірності, а саме дисперсії випадкової величини. Тобто всі неточності відстеження координат об’єкту приймаються як випадкова величина [11]. При цьому алгоритм є ітераційним прагне до мінімізації похибок під час пошуку об’єкту відстеження. На кожній

ітерації обирається значення, яке називають коефіцієнтом Кальмана, що відображає степінь важливості між прогнозованими даними та отриманими. У відстеженні фільтр Кальмана дозволяє визначити знаходження об’єкту на наступній ітерації базуючись на його попередніх станах.

Алгоритми відстеження об’єктів є ефективним способом збору інформації про виявлені об’єкти, а також це дасть змогу зменшити навантаження на згорткову нейрону мережу, адже вона має складну структуру і процес обробки зображення може бути вимогливим до ресурсів і часу. Тому для збору інформації вже виявлених об’єктів у інтелектуальній системі є можливість використання відстеження як ефективного засобу отримання зміни розташування об’єктів та їх основних характеристик.

Також існує велика різноманітність алгоритмів відстеження. Освновними з них є:

- BOOSTING– цей алгоритм є одним із перших. У своїй роботі використовує схожий алгоритм, що застосовується для забезпечення машинного навчання за каскадами Хаара. Спочатку задається початковий стан обєкту, що алгоритмом сприймається як первинний. Після знаходження нового положення обєкту він доповнює первинний стан новими даними. Основна проблема даного аглоритму полягає у тому, що він працює досить повільно і вважається застарілим, але його використовують як еталон у порівнняні з іншими алгоритмами
- MIL (Multiple Instance Learning) – цей алгоритм грунтується на принципі алгоритму BOOSTING, але він замість одного первинного стану генерує декілька можливих розсташувань об’єкту. Це дозволяє збільшити точність відслідковування об’єкту. Даний алгоритм також працює набагато швидше, ніж попередній, але повільно в порівнняні з більщ сучасними.
- KCF (Kernelized Correlation Filters) – цей алгоритм оптимізує роботу MIL алгоритму. Основна ідеї полягає в тому, що декілька первинних станів, які визначаються в MIL, мають зони перекриття. Саме ця особливість

використовується, щоб прискорити роботу і зробити відстеження більш точним.

- TLD (Tracking, learning, and detection) – цей алгоритм базується на «навчанні», тобто попередені стани об’єкту дозволяють алгоритму коригувати свою роботу і ефективніше знаходити наступні стани. Він чудово здатний шукати об’єкти які частково перекриваються іншими об’єктами, але інколи не «розуміє», який об’єкт відслідковувати, якщо два чи більше схожих предмети знаходяться поруч.
- MEDIANFLOW – цей алгоритм базується на принципі відстеження об’єкту в обидва напрямки. Тобто з попереднього стану вираховується майбутній стан, а з теперішнього попередній, після вираховується різниця цих двох станів. Даний алгоритм хороший для відстеження невеликого руху.
- GOTURN – цей алгоритм використовує глибоку згорткову нейронну мережу. Наступні стани шукаються за допомогою нейронної мережі, на вхід подаються попередні стани і за рахунок цього вираховується наступний. Проблема цього алгоритму в мережі, вона досить громіздка і повільна, якщо відстежувати декілька об’єктів.
- MOSSE (Minimum Output Sum of Squared Error) – цей алгоритм грунтується на кореляції початкового стану. На осові початкового стану побується список фільтрів, які використовується для визначення наступного стану об’єкту. Він легкий у реалізації, а також він такий же точний, як і інші складні трекери, і швидший.
- CSRT (Discriminative Correlation Filter with Channel) – алгоритм використовує карту просторової надійності для регулювання фільтра до частини вибраної області з кадру, щоб відстежити об’єкт. Це забезпечує збільшення та локалізацію вибраної області та покращує відстеження не прямокутних областей або об'єктів.

Зі всіх алгоритмів найкраще для розпізнавання об'єктів на відео є CSRT тому, що він має найкращі співвідношення швидкості та точності, стабільний до збоїв. Застарілі алгоритми такі як BOOSTING є повільними для обробки

відео в реальному часі. Більш сучасні алгоритми мають схильність до одного з паарметрів, тобто алгоритм працює швидко, але не зважди точно і коректно відстежуються стани. І навпаки, якщо алгоритм точний він повільніший за інші. Як було сказано вище CSRT відображає собою «золоту» середину цих двох показників, що є дуже важливими у задачі розпізнавання нетипових ситуацій на відео, тому саме він буде використовуватись для відстеження об’єктів.

Отже, найкращим методом збору інформації та моніторингу динаміки об’єктів є відстеження, що дозволяє знаходити наступний стан об’єкту на основі його попередніх станів, що в задачі визначення нетипових ситуацій на відео є однією з головниз підзадач.

1.3 Обгрунтування вибору аналогу програми визначення нетипових ситуацій на відео

Ідея автоматизації моніторингу за станом доріг є не новою і в різні періоди часу проводилися дослідження, щодо використання інформаційних технологій задля автоматизації даного процесу.

Наприклад у дослідженні Automatic and real-time Pothole detection and Traffic monitoring system using Smartphone Technology пропонувалася ідея використання мобільних телефонів та вбудованому у них GPS для збирання інформації про трафік на дорогах задля подальшого аналізу і оптимізації керування потоком транспорту [12]. Труднощі ідеї полягали у тому, щоб зацікавити водіїв транспортних засобів використовувати мобільний додаток, адже щоб дані були адекватними потрібно, щоб більшість водіїв посилали дані про своє переміщення для обробки. Дана ідея є цікавою, але вона не дає змоги виявляти нетипові ситуації на дорозі тим більше у реальному часі. Результатами дослідження був прототип аплікації на смартфон, приклад якої наведено на рис 1.3.

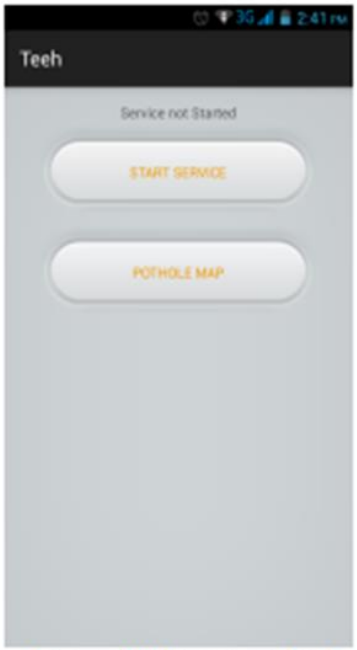

Figure 10 Screen grab of Main Fragment View

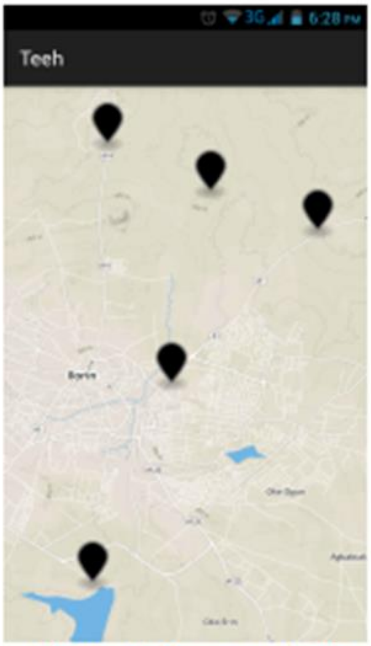

Figure 11 Screen grab of Map Fragment View

Рисунок 1.3 – – Приклад роботи програми моніторингу трафіку

Ще цікава ідея щодо моніторингу стану доріг була подана у дослідженні Automated Sensing System for Monitoring of Road Surface Quality by Mobile Devices [13]. Вона повторювала суть попереднього способу, але полягала у визначенні якості дорожнього покриття. Тобто мобільний додаток використовував вбудований акселератор, що визначає положення пристрою відносно землі, він визначає порушення у положенні автомобіля, що може викликатись дефектами дорожнього покриття. Дані збираються і аналізуються для визначення найбільш проблемних ділянок дороги. Але дана система також не може визначати аномальні ситуації на дорозі в режимі реального часу. Приклад роботи програми наведено на рис 1.4.

Найбільш схожу ідею моніторингу трафіку на дорозі запропоновано у дослідженні Automatic Daytime Road Traffic Control and Monitoring System [14]. Суть полягає у тому, що за допомогою камер спостереження і використання технологій штучного зору відбувалось визначення положення, відстані та розмірів транспорту.

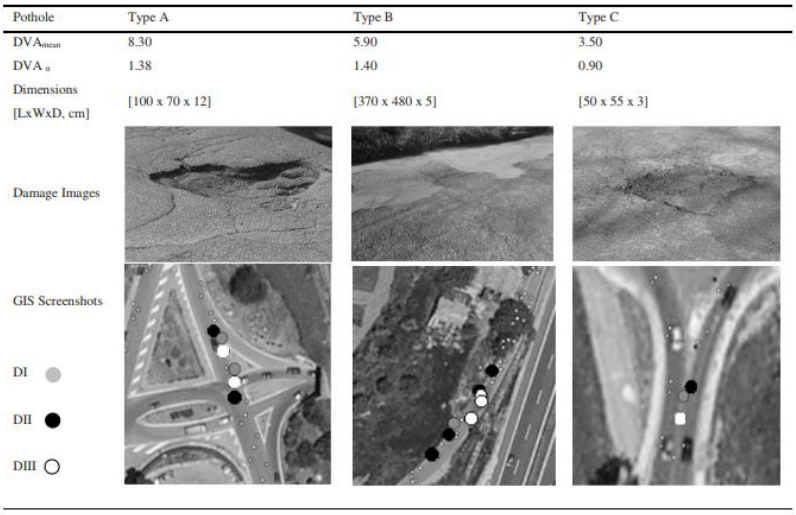

Рисунок 1.4 – Приклад роботи програми моніторингу стану дороги

Дані використовуються для ідентифікації типу транспорту (легковий автомобіль, вантажівка, тощо), після цього збираються дані завантаженості дороги, кількість транспорту за певний період часу, тощо. Ці дані використовувались для оптимізації контролю графіком руху. Дана система мала змогу проводити моніторинг у режимі реального часу, але немає можливості визначення нетипових ситуацій, що потрібно реалізувати у даній роботі. Приклад роботи програми наведено на рис 1.5.

Також існує «Інтелектуальна система моніторингу дорожнього руху», що розроблена в Україні, і яка виконує деякий набір функцій, що схожі на ті, що планується реалізувати у системі, що проектується. Ресурс зображено на рис 1.6.

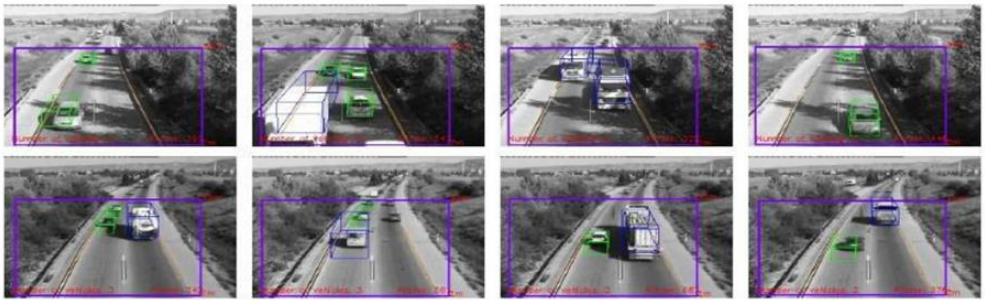

Рисунок 1.5 – Приклад роботи програми моніторингу за допомогою штучного зору

Дана програма надає ряд функціоналу:

– відеомоніторінг інтенсивності дорожнього руху;
– розпізнавання державних номерних знаків транспортних засобів;
– GPS-моніторінг та керування транспортом [15].

Перевагами даної системи є велика кількість можливостей та інформації, що вона здатна збирати. Але недоліками є складність системи та велика потреба ресурсів. Дана система є найбільш близькою по функціоналу, і тому вона буде відігравати роль основного аналога, на властивості якої треба опиратись під час проектування.

Відповідно до вище сказано потрібно розробити програму, що була б легкою у використанні, потребувала мінімальну кількість ресурсів, та мала змогу з досить високою точністю визначати нетипові ситуації на дорозі з використанням камер відеоспостереження. Саме в цьому полягає доцільність розробки системи.

Рисунок 1.6 – Зображення сайту НАУ

Отже, існує велика кількість способів використання інформаційних технологій для моніторингу доріг та розроблені системи, що реалізують дані ідеї, але існує потреба у створенні системи, яка має кращі властивості відповідно до аналогів, та має змогу виявляти нетипові ситуації на дорозі.

1.4 Висновок

У цьому розділі ми визначили вхідні дані та середовище системи для вирішення задачі розпізнавання нетипових ситуацій на відео. На основі цього задача була розбита на основні підзадачі, які потрібно вирішити:

1) Отримання списку виявлених об’єктів
2) Збір інформації про зміни станів об’єктів
3) Аналіз змін та виявлення нетипових ситуацій

Для задачі отримання списку об’єктів на зображені вирішено використовувати нейронні мережі, що здатні виконувати обробку зображень. Одним із видів таких мереж є згорткові нейронні мережі.

Для задачі збору інформації про об’єкти та відслідковування їх динаміки, було вирішено використовувати алгоритми відстеження об’єктів на відео. Було обрано алгоритм CSRT як такий, що має найкраще співвідношення швидкості до точності.

Наступним етапом буде обгрунтування математичної моделі, що надасть можливість виконувати аналіз змін, і на основі цього аналізу виявляти аномалії та розпізнавати нетипові ситуації. Також потрібно виконати оптимізацію основних компонентів системи.

2 РОЗРОБКА ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ РОЗПІЗНАВАННЯ НЕТИПОВИХ СИТУАЦІЙ НА ВІДЕО

2.1 Обґрунтування вибору теоретичної основи інформаційної технології розпізнавання нетипових ситуацій

Для визначення об’єктів на зображенні (кадру) найкращим методом є нейронна мережа, що на вхід приймає пікселі зображення, а на виході буде видавати розташування визначених об’єків. Для цієї цілі анйчастіше використовуються глибокі нейронні мережі (deep neural networks). Для задачі розпізнавання нетипових ситуацій на відео найкращим підвидом глибокої нейронної мережі є згорткова нейронна мережа [8]. Суть мережі полягає у тому, що її внутрішні шари агрегують вхідний шар, зводячи його до мінімального ядра. Дані ядра об’єктів подаються як вхід для звичайної класифікуючої мережі, що детектує якому об’єкту відповідає дане ядро. Переваги даного підходу у тому, що існуються алгоритми, які здатні знайти всі об’єкти на зображенні за один прохід мережі, що прискорює її роботу. Один з таких алгоритмів це YOLO (you only looks one) [16]. Це дозволить обробляти відео у реальному часі.

Наступним етапом потрібно визначити найкращий метод відстеження об’єктів. Як було сказано у першому розділі потрібно відстежувати лише динамічні об’єкти, адже саме вони несуть потенційну небезпеку, що спричиняють нетипові ситуації. Для цього потрбіно детектувати переміщення обєктів між кадрами та запам’ятовувати їх стани. Для відстеження існує безліч готових алгоритмів, які мають свої переваги і недоліки. Ці алгоритми були описані в розділі 1.2. Найкращим алгоритмом було обрано CSRT алгортим через свою швидкість, непогану точність та стійкість до збоїв.

Але відстеження неможливо виконувати постійно, тому що будь який алгоритм при довготривалій роботі починає «втрачати» об’єкт, який він відстежує. Також на сцені (зображені), деякі об’єкти з’являються, а деякі

зникають. Це означає, що детектування обєктів потрібно виконувати з деякою періодичністю, щоб знаходити нові, а також корегувати вже наявні. Тут виникає проблема з ідентифікації об’єктів. Нейронна мережа повертає список визначених об’єктів, але при цьому вона не «розуміє», які з них уже були детектовані нею. Щоб це уникнути потрібно розробити механізм індентифікації.

Під час відстеження об’єкт може змінювати своє положення та розміри, якщо припустити, що новий детектований об’єкт знаходиться на місці одного із наявних обєктів, тоді цей новий об’єкт з великою вірогідністю і є цей обєкт. Порівняння геометричних площ двох об’єктів не є ефективним методом, тому що об’єкти часто змінюють свої розміри, тобто віддаляютьтся або ж приближаються. Щоб це уникнути потрбіно порівнювати геометричні центри областей, де знаходяться об’єкти. Приклад зображено на рисунку 2.1

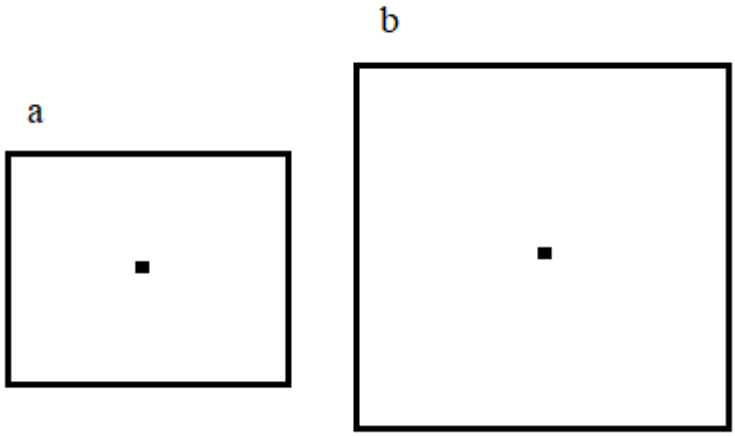

Рисунок 2.1 – Геометричні центри двох об’єктів

Як видно з рисунку обєкти a і b мають різні розміри, але центр у них однаковий. Тобто, якщо накласти ці два об’єкти їх центри будуть збігатись. У даному випадку, якщо дане порівняння виконати з новими детектованими і вже наявними об’єктами і центри будуть збігатись, то з великою ймовірністю це

один об’єкт. Це пояснюється тим, що неможливо розпізнати ті об’єкти, які повністю або майже повністю перекриваються іншими об’єктами.

Але ідеальний збіг центрів майже неможливий, тому постає ситуація, яка зображена на рисунку 2.2.

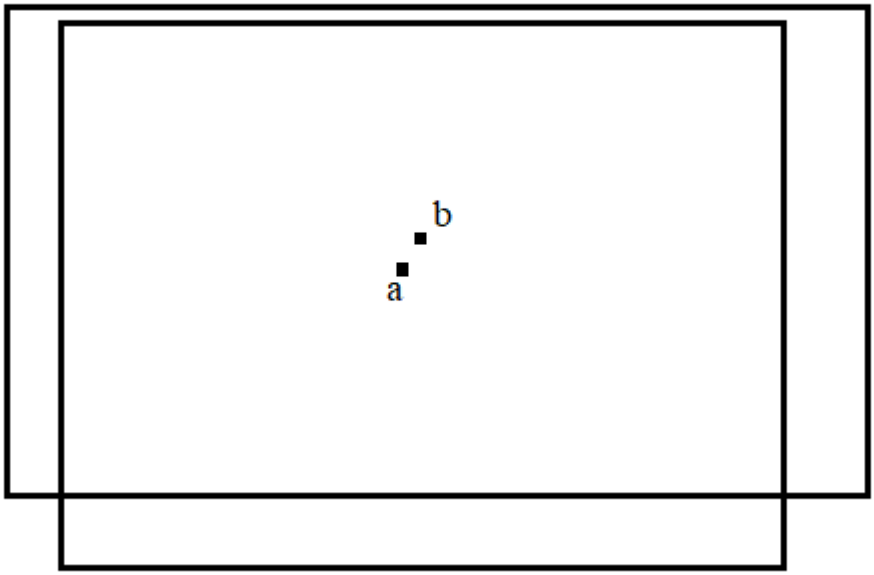

Рисунок 2.2 – Геометричні центри двох подібних об’єктів

У даному випадку центри дуже близькі, але не збігаються точно. Але зрозуміло, що чим ближче один центр до іншого тим більша ймовірність того, що це один об’єкт. Під час ідентифікації об’єктів потрібно визначити мінімальний допустимий рівень ймовірності порівняння.

Отже, для детектування об’єктів буде використовуватись згорткова нейронна мережа. Але для корегування відстеження та виявлення нових потрібно виконувати повторну детекцію. Для індетифікації обєктів між детекціями буде використовуватись порівняння геометричних центрів. Наступний етапом буде обгрунтування методу виявлення власне аномалій у об’єктів.

2.2 Обґрунтування вибору методу виявлення нетипових ситуацій

Відповідно до вищесказаного у пункті 1.1 визначення аномалій є заключною та основною задачею даної інформаційної технології. При цьому потрібно працювати з об’єктами, отриманими в 2D системі координат. Основними визначеними характеристиками є:

Положення об’єкту (x та y)

Розміри об’єкту (w та h)

Так як основні об’єкти є динамічними і рухаються з часом ми можемо визначити зміни координат об’єкта, що є параметром швидкості, при цьому визначити можна швидкість по X та Y координаті відповідно.

Основна ідея визначення аномалій полягає у накопиченні середнього значення швидкості об’єкту та визначенні дисперсії швидкості, що являє собою середнє сподівання зміни швидкості [10]. Якщо зміна швидкості або дисперсія не відповідає заданому порогу, можна констатувати аномалію. Приклад руху об’єкту зображено на рис 2.3.

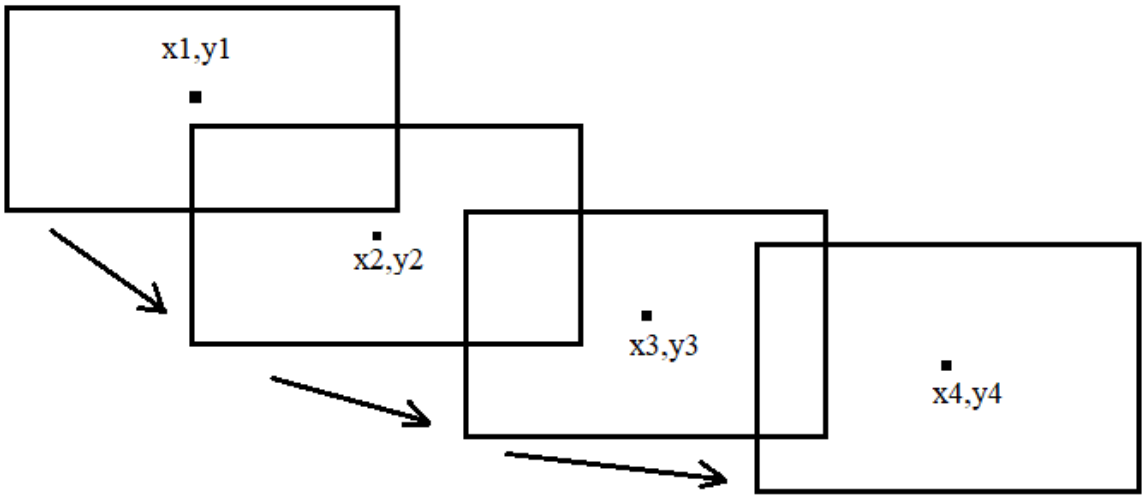

Рисунок 2.3 – Рух області об’єкту між кадрами

Базуючись на цій ідеї визначимо основні параметри, що будуть використовуватись:

– U^X швидкість по X
– U^Y швидкість по Y
– U^X_{avg} середня швидкість по X
– U^Y_{avg} середня швидкість по Y
– dU^X зміна швидкості по X
– dU^Y зміна швидкості по Y
– σ^X дисперсія швидкості по X
– σ^Y дисперсія швидкості по Y

Тобто якщо існує теперішний момент часу t і є визначені попередні стани t-1 можна визначити формули:

$$dU^X(t) = |U^X(t) - U^X(t-1)|$$

$$U^X_{avg}(t) = \left|\frac{\left(dU^X + U^X_{avg}(t-1)\right)}{2}\right|$$

$$\sigma^X = \sqrt{\left|dU^X(t)^2 - U^X_{avg}(t)^2\right|}$$

Аналогічним чином визначаються параметри і для Y координати:

$$dU^Y(t) = |U^Y(t) - U^Y(t-1)|$$

$$U^Y_{avg}(t) = \left|\frac{\left(dU^Y + U^Y_{avg}(t-1)\right)}{2}\right|$$

$$\sigma^Y = \sqrt{\left|dU^Y(t)^2 - U^Y_{avg}(t)^2\right|}$$

Як було сказано вище потрібно ввести значення порогу θ_x та θ_y, що являють собою значенням, перевищивши яке можна сказати, що даний об’єкт є аномальним При цьому поріг має обмежувати значення зміни не лише згори, а

й знизу, адже при незначних змінах, які прагнуть до нуля, значення середньої швидкості є настільки малим, що при незначному прискорені об’єкту може, дати досить вагоме відхилення, що може перевищити заданий поріг. Самі значення порогів є досить ситуативним значенням, і залежать від величини параметрів об’єктів, тобто об’єкти, що розташовані близько до точки спостереження можуть мати більші значення зміни, ніж ті, що розташовані подалі. Відповідно до цього значення порогу задають міру «чутливості» системи, більш чутлива система реагуватиме на менш незначні зміни об’єктів. Тому ці параметри мають бути динамічними і має бути можливість задання їх користувачем.

Відповідно до вищесказаного якщо отримані зміни дисперсії та швидкості не відповідають заданим порогам, то даний об’єкт вважається аномальним, після чого потрібно зберегти інформацію про нього для подальшої обробки користувачем.

Отже, було визначено які основні характеристики отримуються на вході під час виконання задачі аналізу, а також визначено основні параметри, за якими відбуватиметься визначення нетипових ситуацій.

2.3 Аналіз та вдосконалення роботи програми для визначення нетипових ситуацій на відео

В попередніх розділах ми визначии основні методи, які будуь використовуватись під час визначення нетипових ситуацій на відео. На одному зображенні буде багато об’єктів, це означає, що чим більше об’єктів відстежуєтсья тим повільніше працюватиме алгоритм. Також це стосується нейронної мережі. Тобто чим більше вхідне зображення тим довше буде він оброблятись. Для прискорення роботи можливо використовувати багатопоточність та технології обрахунків на GPU.

Багатопоточність (multithreading) – це технологія, яка довзоляє використовувати декілька ядр процесора, а не лище один за допомогою

створення потоків та процесів. Потоки та процеси – це елементи коду програми, що здатні виконувати паралельно основній програмі. [17]. Так як більшість сучасних процесорів є багатоядровими це може стати чудовою оптимізацією роботи алгоритму. Основна ідея полягає у тому, щоб обрахунки на списком об’єктів розподілити на декілька процесів та ядр. Зазвичай під час роботи в одному потоці всі обрахунки відбуваються послідовно. Приклад такої обробки об’єктів зображено на рис 2.4.

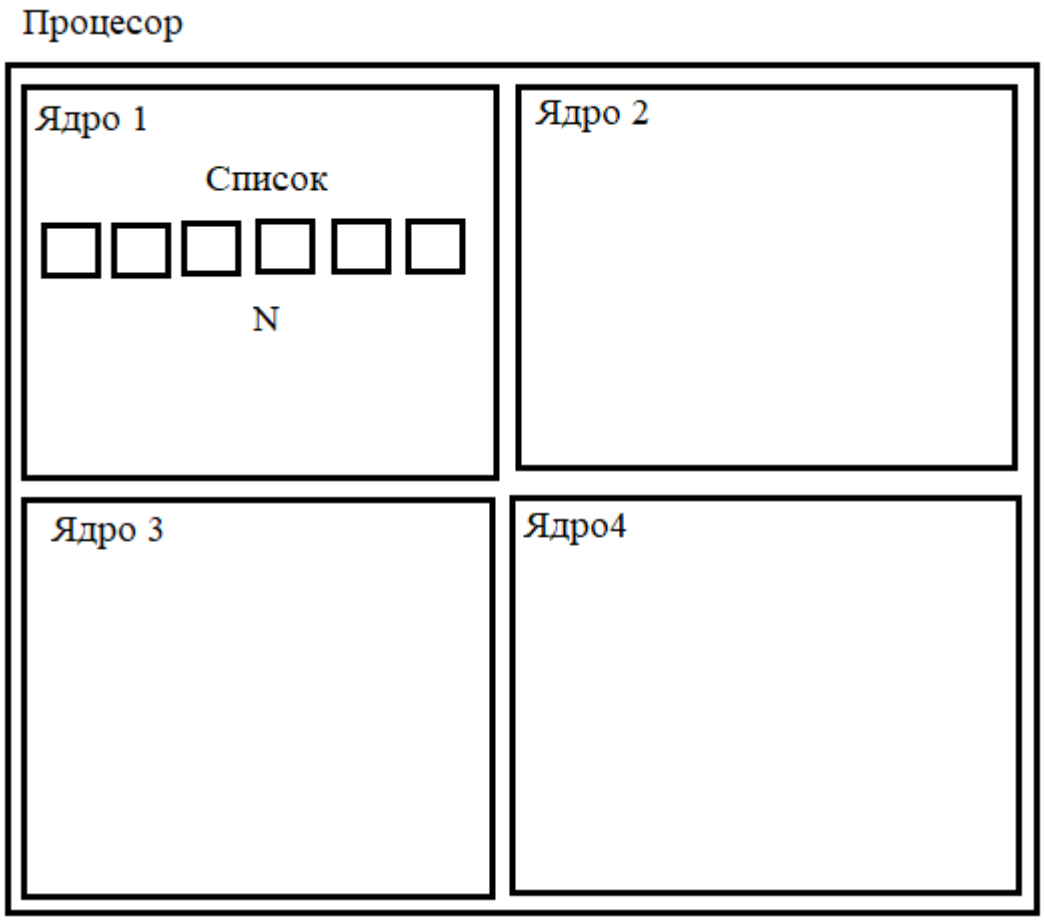

Рисунок 2.4 – Приклад обробки списку об’єктів

На рисунку схематично зображено процесор, що містить 4 ядра. Всі обрахунки відбуваються у головному потоці, який зазвичай запускається на основному ядрі процесора, при цьому інші ядра не використовуються автоматично, тому вони не працюють. Внаслідок цього втрачається

обрахунковий потенціал процесора. Тому список з розміром N буде оброблятись доволі довго, якщо число N буде довлі великим.

Багатопоточність доволить розділити цю обробку на декілька потоків, які вже мають можливість виконувати на декількох ядрах. Схематично це зображено на рисунку 2.5.

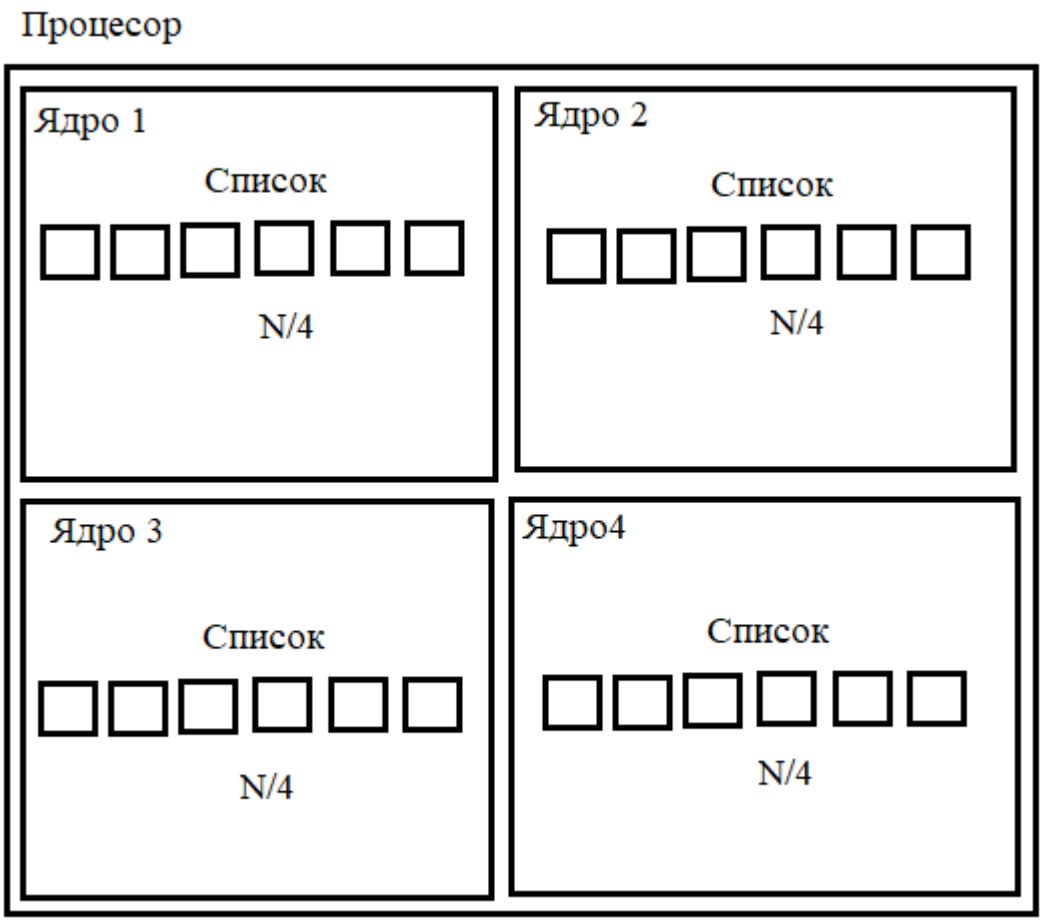

Рисунок 2.5 – Приклад обробки списку з декількома потоками.

В даному випадку список розбивається на рівномірні частини, кожна з частин обробляєтсья на свому ядру, що в загальному збільшує загальну швидкість обробки.

Основна проблема підходу багатопоточності грунтується на сихронізації та підготовці даних. Сихронізація потоків відбувається у моменти, які називаються критичні зони, це місця, де декілька потоків можуть «заважати один одному», або ж будуть доступатись до ресурсів, які можуть бути доступні

лише одному ядру в одиницю часу. Також сихронізація використовується дла агрегування та збір даних з потоків в одному місці [18]. Тобто для обробки списку на декількох ядрах потрібно виконати ряд операцій:

- Підготовка даних – розбиття масиву даних на рівномірні частини, підготовка потоків та сторення процесів.
- Обробка – безпосереднє виконання потоків та їх сихронізація, якщо потрібно.
- Збір даних – сихронізація всіх потоків у одному місці програми та обєднання результатів

Логічно припустити, що багатопоточність буде ефективно тільки якщо підготовка та збір результатів набагато менше чим виконання обробки в одному потоці. Це можливо в двох випадках: розмір масиву досягає великих значень (тисячі, сотні тисяч) і якщо обробка одного об’єкту в масиві забирає доволі багато часу. Під час визначення нетипових ситуацій на відео місце має другий випадок. На зображенні можуть міститься декілька десятків об’єктів, але обробка алгоритмів відстеження займає доволі велику частину часу.

Використання багатопоточності під час обробки масивів даних дозволить прискорити процес розпізнавання нетипових ситуацій на відео.

Ще однією технологією для прискорення роботи системи є технології використання графічних ядер. Тобто замість використання масивних ядер CPU можна використовувати маленькі ядра GPU. Ефективність такого методу пояснюєься тим, що в GPU міститься тисячі ядер, що здатні обробляти інформацію. Дана відмінність у архітектурі схематично показана на рис 2.6

Ядра графічного процесору невеликі, але на них можна розподілити обробку велику кількість однотипних операцій. При цьому кожне ядро відповідає певному блоку ядер, які керуються своїм керуючим механізом.

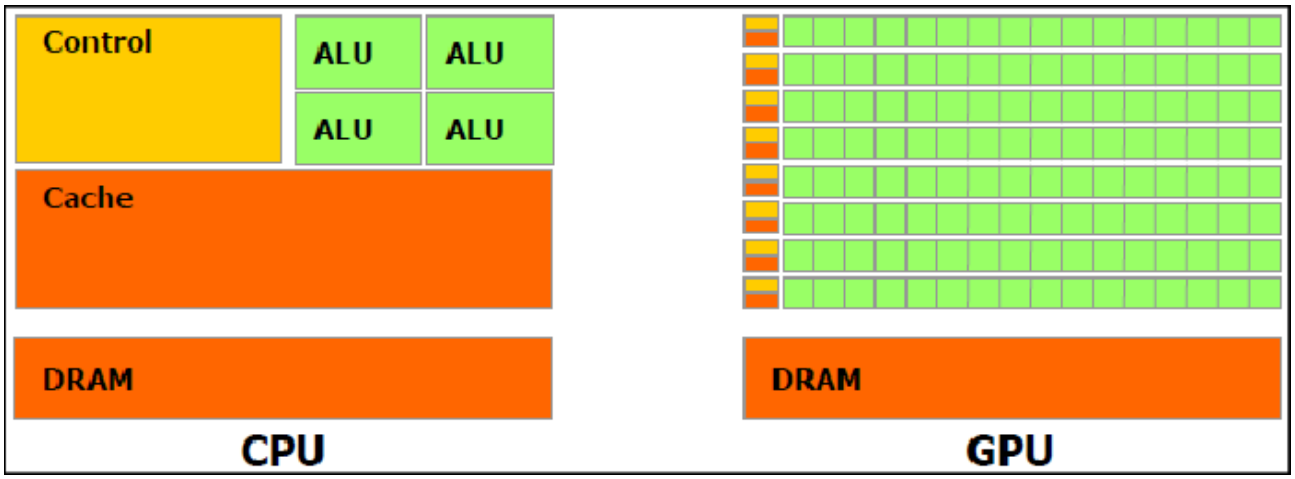

Рисунок 2.6 – Порівняння будови CPU та GPU

Дані технології знайшли своє місце у нейромережевих технологіях. Великі глибокі нейронні мережі, які містять велику кількість синапсів та шарів, де виконуються елементарні математичні операції, розбиваються на блоки, які можна виконувати на графічних ядрах. Однією з технологій, що доволяє вільно використовувати графічні процесори для обрахунків є CUDA [19]. Ця технологія була розроблена виробником графічних процесорів NVIDIA, що активно пітримує розвиток нейрокомп'ютерних технологій, тому вони сприяють використанню GPU для оптимізації навчання та роботи складних мереж. Технологія використання GPU дозволить значно прискорити роботу нейронної мережі, що в свою чергу дозволиьт використовувати більш складну структуру та більшу кількість вхідних нейронів, що збільшить точність детектування об'єктів.

Отже, сновними методами оптимізації розпізнавання нетипових ситуацій на відео є використання багатопоточності та GPU технологій.

2.4 Складові інформаційної технології розпізнавання нетипових ситуацій на відео

Структура інформаційної технології розпізнавання нетипових ситуацій на відео на основі згорткової нейронної мережі зображена на рис. 2.7.

Вхідна інформація для інформаційної технології розпізнавання нетипових ситуацій на відео є потокове відео, що розбивається на покадрові растрові зображення, що обробляються і перетворюються на вхідний вектор для подачі на вхід згорткової нейронної мережі. Розпізнавання обєктів на зображенні та їх класифікація буде використовуватись глибока згорткова нейронна мережа. Після чого утворений список визначених об’єктів буде відстежуватись за допомогою одного з алгоритмів відстеження, а також буде виконуватись обрахунок основних характеристик об’єктів (швидкість, прискорення). Далі відбувається процес обрахунку змін швидкості і дисперсії, різка зміна яких буде сигналізувати про наявність нетипової ситуації. В завершення буде виконуватись процес збереження результатів та повідомлення оператора.

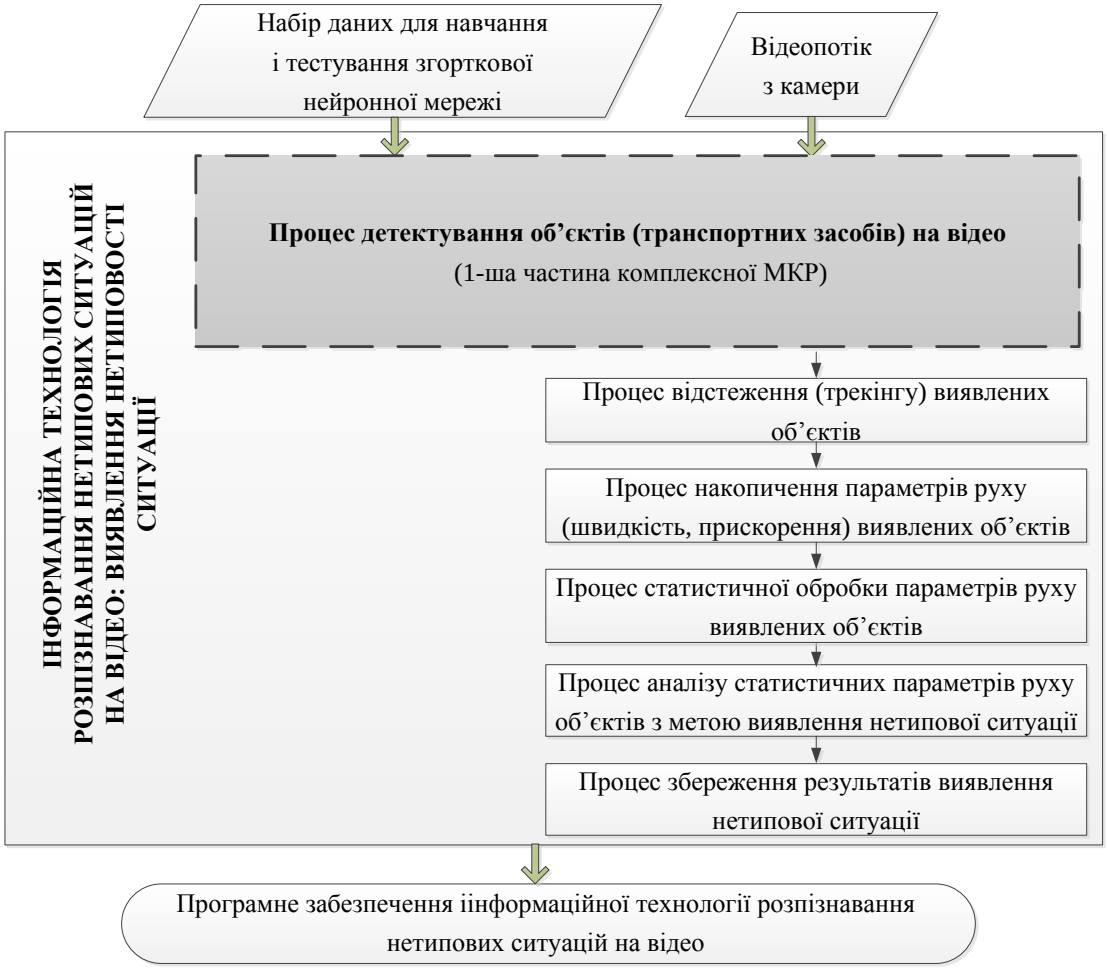

Рисунок 2.7 – Структура інформаційної технології розпізнавання нетипових ситуацій на відео

Отже, розроблена структура інформаційної технології розпізнавання нетипових ситуацій на відео може бути використана для подальшої розробки програмних засобів.

2.5 Висновок

У даному розділі було виконано обгрунтування методу індетифікації об’єктів між детекціями. Дана процедура потрібна для знаходження нових об’єктів та корегування вже існуючих.

Основним методом розпізнавання нетипових ситуацій було обрано аналіз основних змін об’єктів, а саме зміна його основних характеристик: розміри та розсташування. З цих характеристик можливо знайти швидкість руху та прискорення об’єктів, а також дисперсію змін. Стрімка зміна цих показників буде сигналом того, що відбулася аномалія, що може призвести до нетипової ситуації.

Також було виконано розбиття інформаційної технології на компоненти та описано основні задачі, які стоять перед цими компонентами. Також описані їх взаємозв’язки.

Було виконано аналіз та оптимізацію методів вирішення задачі розпізнавання нетипових ситуацій на відео. Освновними є оптимізації швидкості роботи: багатопоточність та технології обробки на графічному ядрі GPU. GPU обробка допоможе прискорити детектування об’єктів за допомогою згорткової нейронної мережі. Багатопоточність допоможе розподілити обробку списків даних на всі ядра процесора, що прискорить відстеження великої кількості об’єктів на зображенні та виконання обрахунків і аналізу.

Наступним етапом розробки інформаційної технології розпізнавання нетипових ситуацій на відео є власне проектування та програмна реалізація технології за допомогою сучасних мов програмування та технологій, а також тестування та аналіз точності та швидкості роботи системи.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ РОЗПІЗНАВАННЯ НЕТИПОВИХ СИТУАЦІЙ НА ВІДЕО

3.1 Розробка структури програмних засобів розпізнавання нетипових ситуацій на відео

Першим етапом програмної реалізації системи розпізнавання нетипових ситуацій на відео потрібно провести проектування основної структури застосунку. Для початку потрібно визначити основний дизайн архітектури програми, яка буде реалізовуватись. Перш за все для зручної роботи з програмою потрібно реалізувати графічний інтерфейс користувача, також потрібно імплементувати основне ядро програми, яке буде виконувати основну логіку та реалізовуватиме основні модулі, які були описані в пункті 2.4.

Спроектуємо загальну структурну схему основних модулів системи, що виконуватимуть основні дії, які потрібні для виконання задачі. Дана структурна схема наведена на рис. 3.1

На структурній схемі виділені основні модулі системи, що являють собої основні її складові.

Модуль введення являє собою способи введення вхідних даних, дані можуть поступати у систему у вигляді відео або потокового відео у режимі реального часу. Даний модуль займатиметься отриманням з цих даних фреймів(зображень), що будуть подаватися на вхід мережі.

Модуль обробки зображень являє собою процес перетворення зображення до вигляду прийнятного для подачі на вхід нейронної мережі. Тут відбуватимуться процеси зміни розміру вхідного зображення, додавання маски тощо. Після обробки зображення буде перетворюватися на вхідний вектор нейронної мережі.

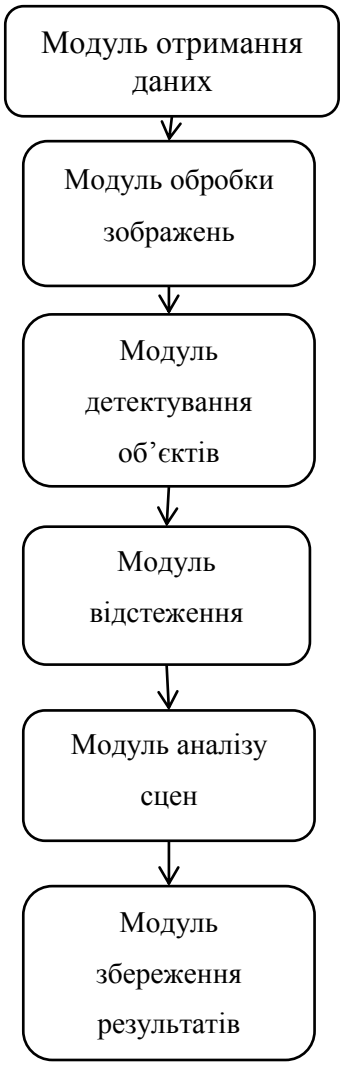

Рисунок 3.1 – Структурна схема системи розпізнавання нетипових ситуацій на відео

Модуль детектування являє собою згорткову нейронну мережу, що буде виконувати на заданому вхідному векторі пошук об’єктів.

Модуль відстеження буде виконувати спостереження за вже виявленими об’єктами та збір інформації про зміну їх положення при отриманні наступного фрейму.

Модуль аналізу виконуватиме виявлення аномалій серед виявлених об’єктів.

Модуль збереження результатів буде зберігати отримані аномалії, для подальшої обробки їх користувачем.

Дана структурна схема допомагає зрозуміти основні задачі, які будуть викнувати той чи інший структурний елемент технології, а також значно підвищить розуміння взаємозв’язків між ними і спростить подальшу реалізацію.

Для подальшого проектування потрібно розробити архітектуру програми, для цього застосуємо популярний шаблон архітектури як MVC.

MVC (model - view - controller) – архітектурний шаблон, який використовується під час проектування та розробки програмного забезпечення. Він передбачає розділення системи на три взаємопов'язані частини: модель (model) даних, вигляд (view), що уособлює собою інтерфейс користувача та модуль керування (controller). Застосовується для відокремлення даних (моделі) від інтерфейсу користувача (вигляду) так, щоб зміни інтерфейсу користувача мінімально впливали на роботу з даними, а зміни в моделі даних могли здійснюватися без змін інтерфейсу користувача [20]. Основна схема взаємодії MVC архітектури зображено на рис 3.2.

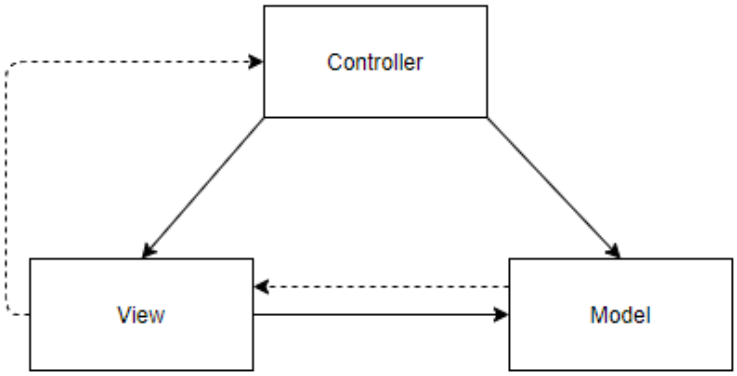

Рисунок 3.2 – Схематичне зображення взаємодії шаблону MVC

Як можна спостерігати зі схеми графічний інтерфейс не працює напряму з даними, а виконує всі дії через керування. Це дозволяє відокремити ці два поняття, що значно спрощує розробку та пітримку графічного інтерфейсу і також ядра маніпулювання даними.

На основі даного підходу розробки виконаємо проектування архітектури системи розпізнавання нетипових ситуацій на відео. У веб застосунках для обміну повідомленнями між модулем керування та графічним інтерфейсом зазвичай використовують API – програмний інтерфейс застосунку, що використовується для доступу до даних та обробки інформації. Одним з підвидів API є підхід REST.

REST – це архітектурний підхід до побудови комунікації у мережі де використовується HTTP протоколи передачі даних, що стандартизує основні механізми спілкування між клієнтом та сервером [21]. Даний підхід є непоганим, але він містить ряд недоліків: потрібно передавати багато метаданих такий як HTTP заголовки, що розповідають про пакет його вміст та основні характеристики, потрібно кожного разу створювати запити зі всіма необхідними даними і потім очікувати і парсити відповіді від сервера. Даний підхід чудово підходить для Веб затосунку, але в даному випадку нам потрібний швидкий канал комунікації між UI та основними моделями даних. Одним з таких підходів є використання WebSocket каналів.

WebSocket – це протокол, яка базується на HTTP, який довзоляє відкривати прямий двонаправлений канал передачі даних між двома точками і виконувати спілкування між ними за допомогою повідомлень (Event) [22]. Він є досить швидким тому, що він відкриває канал при ініціалізації підключення, після чого повідомлення не навантажуються непотрібними метаданими, а також двонаправленість цього канала дозволяє швидко доставляти дані від клієнта до сервера і навпаки. Цей протокол є чудовим механізмом, що допоможе у реалізації каналу зв’язку у архітектури системи.

Схематично архітектура системи зображеня на рис 3.3.

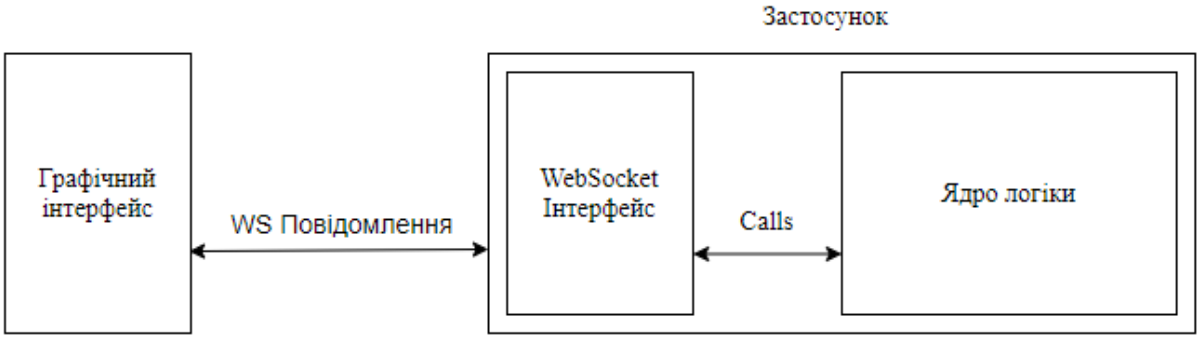

Рисунок 3.3 – Схематичне зображення архітектури системи

Як можна спостерігати з схеми, зображеної вище, архітектура системи розпізнавання нетипових ситуацій на відео здіснена за допомогою шаблону MVC. Це означає, що графічний інтерфейс (View) за допомогою WebSocket протоколу звязується з WebSocket інтерфейсом, що виконує задачу модуля керування (Controller), який буде здійснювати виклики основного ядра логіки, тобто моделі (Model). Весь процес обробки буде виконуватись саме в ядрі.

Подальше проектування буде відноситись більшим чином до ядра і моделі, щоб показати основні взаємозвязки всередині системи між основними внутрішніми модулями.

Для кращого розуміння основної логіки роботи системи у проектуванні часто використовують DFD (Data flow diagram) діаграма [23]. Дані діаграми дозволяють подати у графічному вигляді переміщення даних між модулями, їх перетворення з одних типів у інші, що у свою чергу спрощує розуміння функціонування програми всіма розробниками. Тому виконаємо побудову DFD діаграми основних потоків даних та опишемо основні процеси, що на ній відображені. Діаграма зображена на рис. 3.4

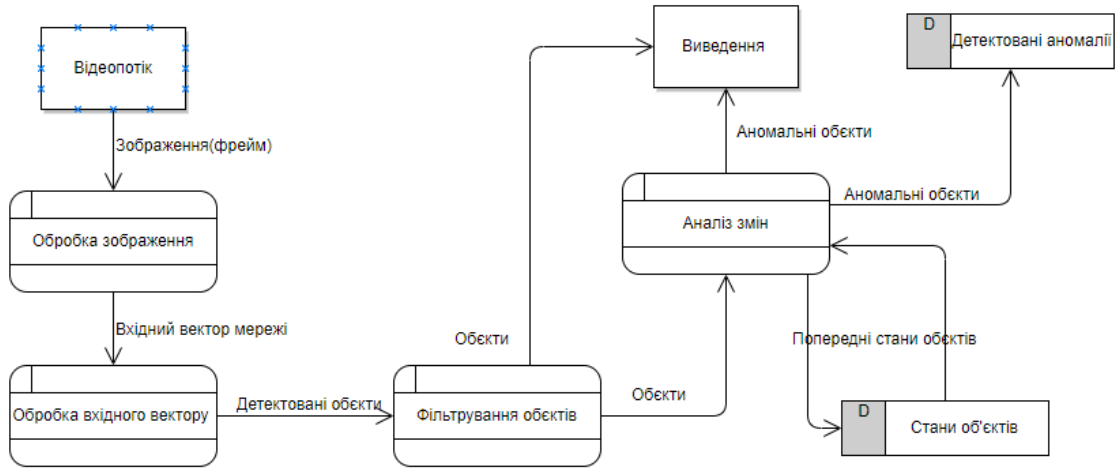

Рисунок 3.4 – DFD діаграма системи розпізнавання нетипових ситуацій на відео

Дані надходять з зовнішній сутностей та перетворюють походу проходження їх через систему. Обробка зображення виконує перетворення вхідного зображення на вхідний вектор нейронної мережі, Обробка вхідного вектору являє собою нейрону мережу, що на виході дає масив виявлених об’єктів, які в свою чергу фільтруються від шумів. Аналіз змін отримує на вхід об’єкти та їх теперішні стани, а також стани, що були до цього – це потрібно для виявлення аномальних змін. Після цього об’єкти виводяться у заданий спосіб через Виведення, а виявлені аномалії зберігаються.

Сховище станів об’єктів являє собою локальне сховище(ОЗУ, хеш, тощо) для тимчасового зберігання та швидкого доступу.

Дана діаграма допоможе зрозуміти, як дані перетворюються при обробці їх основними функціональними модулями програми, що по суті являє собою набором вхідних і вихідних даних кожної частини.

Інколи наявна кількість інформації буває недостатньою, тоді використовують IDEF0 діаграми [24]. Вони окрім вхідних та вихідних даних, також описують сигнали управління і методи та інструменти, що використовуються у функціональних частинах. Такий підхід дозволяє зрозуміти, що саме впливає на кожен окремий модуль і що використовується у процесі його роботи, що у свою чергу дозволяє зрозуміти як реалізовується сам

процес обробки даних. Виконаємо розробку IDEF0 діаграми для модуля аналізу сцен, щоб на більш нижчому рівні абстракції описати його роботу. Дана діаграма наведена на рис 3.5.

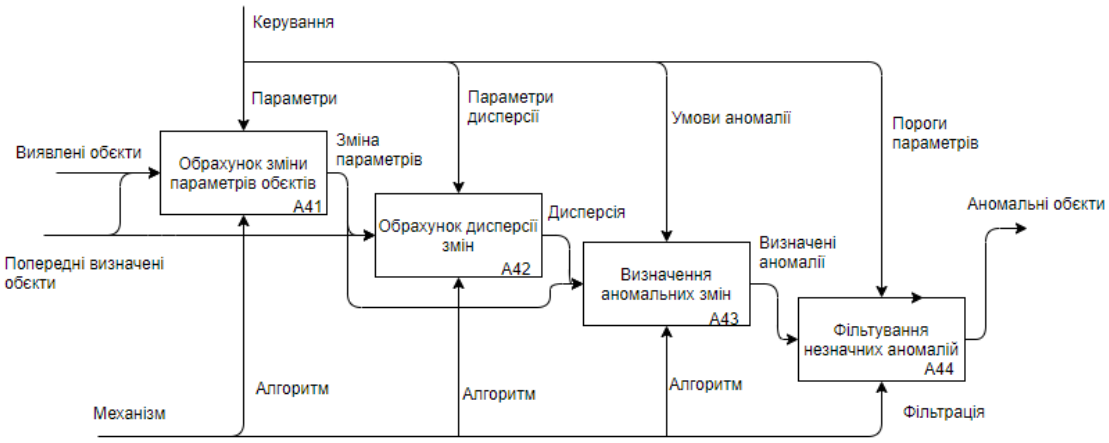

Рисунок 3.5 – IDEF0 діаграма модуля аналізу сцен програми розпізнавання нетипових ситуацій на відео

На даній схемі можна спостерігати, які саме механізми використовуватимуться під час аналізу сцен.

На першому етапі відбувається визначення змін основних параметрів об’єкту на основі наявної інформації і попередніх станів.

На другому етапі відбувається обрахунок дисперсії змін параметрів, що являє собою накопичення динаміки зміни параметрів.

На третьому по заданих умовах відбувається визначення чи є об’єкт аномальним.

На останньому відбувається фільтрування аномалій, що є незначними і не мають бути визначені як аномалії.

Після структурних та функціональних властивостей системи потрібно зрозуміти як саме будуть отримуватись вхідні дані даної системи і як вона буде взаємодіяти безпосередньо з користувачем, адже система є відкритою і має обмінюватись інформацією з зовнішнім середовищем. Зазвичай для графічного представлення різних аспектів роботи об’єкту, що проектуєтеся,

використовують UML – Unified Modeling Language [25]. Дана уніфікована мова містить набір діаграм, які дозволяють графічно зобразити багато аспектів системи починаючи від функціональних та структурних закінчуючи фізичним розташуванням компонентів системи.

Для опису основні взаємодії між компонентами в системі використовують діаграму послідовностей [25]. Даний тип діаграм зображують, які взаємодії відбуваються у системі з часом і які відповіді формує кожна функціональна система. Це розширить розуміння того, як користувач взаємодіє з системою. Дана схема наведена на рис 3.6.

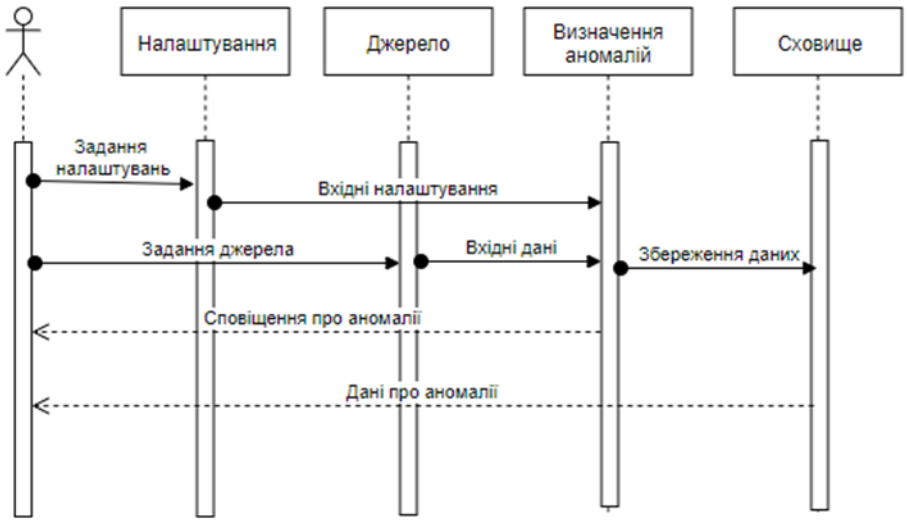

Рисунок 3.6 – UML діаграма послідовностей

Дана діаграма поєднує у собі діаграму варіантів використання та основні процеси, що протікають у системі і показує, які дії протікають у часі в результаті роботи.

Однією з найголовніших діаграм в UML є діаграми класів [25]. Даний вид діаграми використовується для представлення графічним способом основних сутностей програми їх властивостей та поведінки. Діаграма класів зазвичай повторює використовується для зображення предметної області об’єкту, що розробляється, за принципами об’єктно-орієнтованого програмування – ООП [25]. Даний підхід програмування зображає всі предмети через класи, що

містять опис про його властивості та поведінку об’єкту. Тому побудуємо діаграму класів опираючись на принципи ООП. Дана діаграма наведена на рис 3.7.

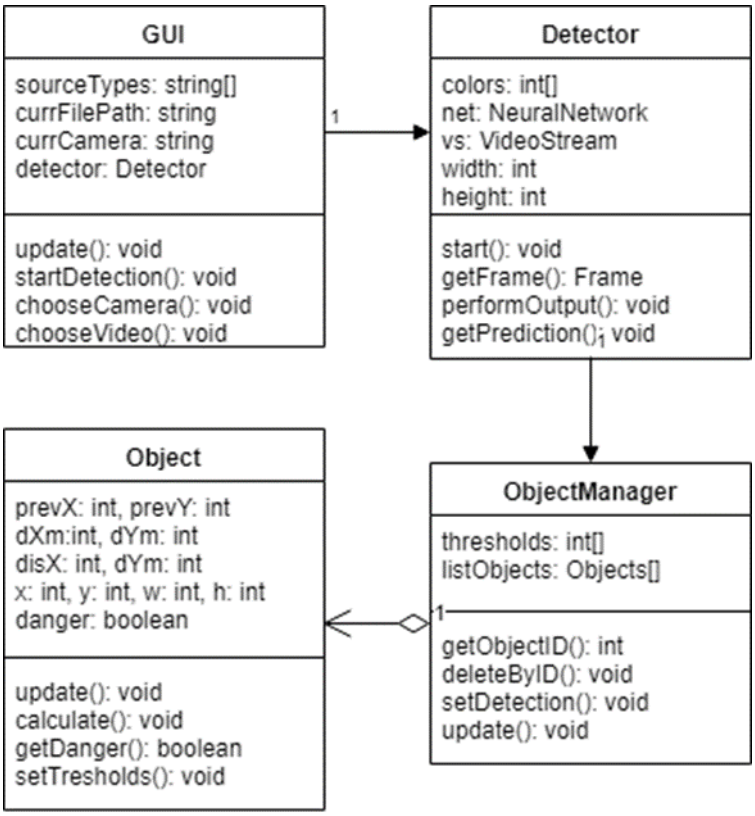

Рисунок 3.7 – UML діаграма класів

Дана діаграма описує основні класи, що будуть реалізовані у програмі. Серед таких сутностей можна виділити:

1. GUI
2. Detector
3. ObjectManager
4. Object

Клас GUI репрезентує реалізацію графічного інтерфейсу і служить основним мостом взаємодії користувача та системи. Даний клас містить набір основних властивостей, а саме він має зберігати типи вхідний джерел, шлях до відео, або номер обраної камери, також він має містити екземпляр класу Detector, тому він має тип з’єднання з цим класом як асоціація. Також цей клас має реалізовувати механізми введення налаштувань користувача та способи

отримання інформації. Серед основних поведінкових аспектів виділено можливість введення основних джерел інформації користувачем та можливість розпочати роботу основного модуля.

Клас Detector є основним класом і являє собою місцем роботи всіх функціональних модулів. Саме в даному класі буде відбуватися отримання вхідних даних відповідно до вибраного джерела, а також виконання перетворення вхідних зображень на вхідний вектор. Тут також буде відбуватись детектування за допомогою нейронної мережі і отримання вихідних об’єктів. Після цього ці об’єкти будуть передаватись до екземпляру класу ObjectManager, тому тип зв’язку між цими класами є асоціація. Також даний клас має керувати налаштуваннями, що отримані від GUI і відповідно до цього керувати станом детектування і отриманням результатів. Тому цей клас є основним відповідно до інших класів.

Клас ObjectManager є допоміжним класом відносно інших класів і виконує функцію управління за списком виявлених об’єктів. У цьому класів відбуватиметься зберігання виявлених об’єктів, присвоєння їм ідентифікаційного номеру та передача основних параметрів, отриманих від користувача. Також має бути реалізована можливість отримувати та видаляти об’єкти зі списку, та обновляти стани об’єктів. Окрім того даний клас має мати можливість розрізняти нові об’єкти від уже збережених, що є дуже важливим, адже повторне детектування за допомогою нейронної мережі повертає новий набір об’єктів, серед яких можуть бути уже наявні у списку. Даний клас зберігає список класу Object, тому тип зв’язку між ними є композиція.

Клас Object виконує опис виявленого об’єкту і зберігає основні його властивості такі як попередній стан, теперішній стан, значення зміни швидкості, значення дисперсії, середні значення зміни швидкості. Також даний клас зберігає основні пороги та налаштування, що використовуються при аналізу об’єкту. Саме в даному класі буде реалізований основний алгоритм роботи модуля аналізу сцен, тому даний клас має мати здатність його виконувати. Також даний клас містить стан, що сигналізує чи є даний клас

аномальним чи ні, що буде використовуватися при збереженні результатів. Загалом клас є найменшою структурною одиницею, але в ньому має реалізовуватись основна логіка роботи програми, тому він є досить важливим.

Отже, в результаті проектування програмного модуля аналізу сцен було спроектовано структурну схему програми, та проаналізовано основні зв’язки між структурними одиницями. Також було виконано проектування DFD та IDEF0 діаграм, що показують, які дані використовуватимуться як вони перетворюються і де зберігаються, а також основні функціональні блоки та механізми та інструменти їх реалізації. Після цього було розроблено UML-діаграми варіантів використання та послідовностей, що описують як користувач зможе взаємодіяти з системою. Було розроблено UML діаграма класів, що описує основні класи, які потрібно реалізувати, їх властивості та поведінку.

3.2 Розробка алгоритму роботи програмних засобів розпізнавання нетипових ситуацій на відео

Основна ідея роботи алгоритму полягає у підрахунку змін характеристик об’єкта аналізу між двома сусідніми станами. Так як система веде роботу з двовимірними зображеннями алгоритм буде враховувати зміну положення об’єкту по двом координатам, що в свою чергу має фізичну інтерпретацію швидкості руху. При цьому швидкість буде вираховуватись по x та y координатах окремо. При отримані змін швидкостей теперішнього стану і порівнявши їх з попереднім станом. Значна зміна швидкості по модулю свідчить про те, що з об’єктом сталась аномалія, наприклад різка зміна напрямку руху, різке гальмування або прискорення, що при звичайному стані спостерігатись не має. Для того, щоб відкинути малі аномалій, які можуть виникати при русі об’єкту, потрібно ввести поняття порогу, що описував би мінімальну зміну, яку вже можна вважати аномалією. Даний параметр залежить від багатьох зовнішніх факторів, таких як відстань до об’єкту від токи

спостереження, кут спостереження тощо. Тому цей показник користувач має калібрувати під час налаштування системи. Для введення додаткового параметру аналізу потрібно підраховувати дисперсію зміни швидкості, що буде накопичувальним елементом динаміки руху об’єкту. З фізичної точки зору це буде описувати прискорення. Зрозуміло, що при значній зміні дисперсії можна також, говорити про аномалію. Аналогічно до зміни швидкості потрібно ввести поріг, щоб фільтрувати незначні зміни.

Робота алгоритму починається з отримання вхідних даних, що показано як блок 1, серед вхідних даних основними є теперішні показники розміщення об’єкту та його розмірів, а також попередній стан. Після цього відбувається підрахунок зміни швидкості та дисперсії, на схемі ці процеси наведені блоком 2 та 3 відповідно. Після у блоці 4 відбувається перевірка чи є отримані значення суттєвими і чи є сенс виконувати подальші перевірки, якщо ні переходимо в кінець алгоритму, якщо так відбувається перехід до блоку 5. У цьому блоці відбувається перевірка значення зміни швидкості, якщо воно значне то можна констатувати аномалію і переходимо до блоку 7, якщо ж ні то переходимо до блоку 6. У блоці 6 відбувається перевірка чи є значна зміна дисперсії, якщо так переходимо до блоку 7, інакше алгоритм закінчується. У блоці 7 відбувається збереження інформації про аномальний об’єкт для подальшого аналізу користувачем, після чого алгоритм закінчується.

Дані кроки виконуються для будь-якого виявленого обєкту при переході між його станами. Схема алгоритму наведена на рис 3.8.

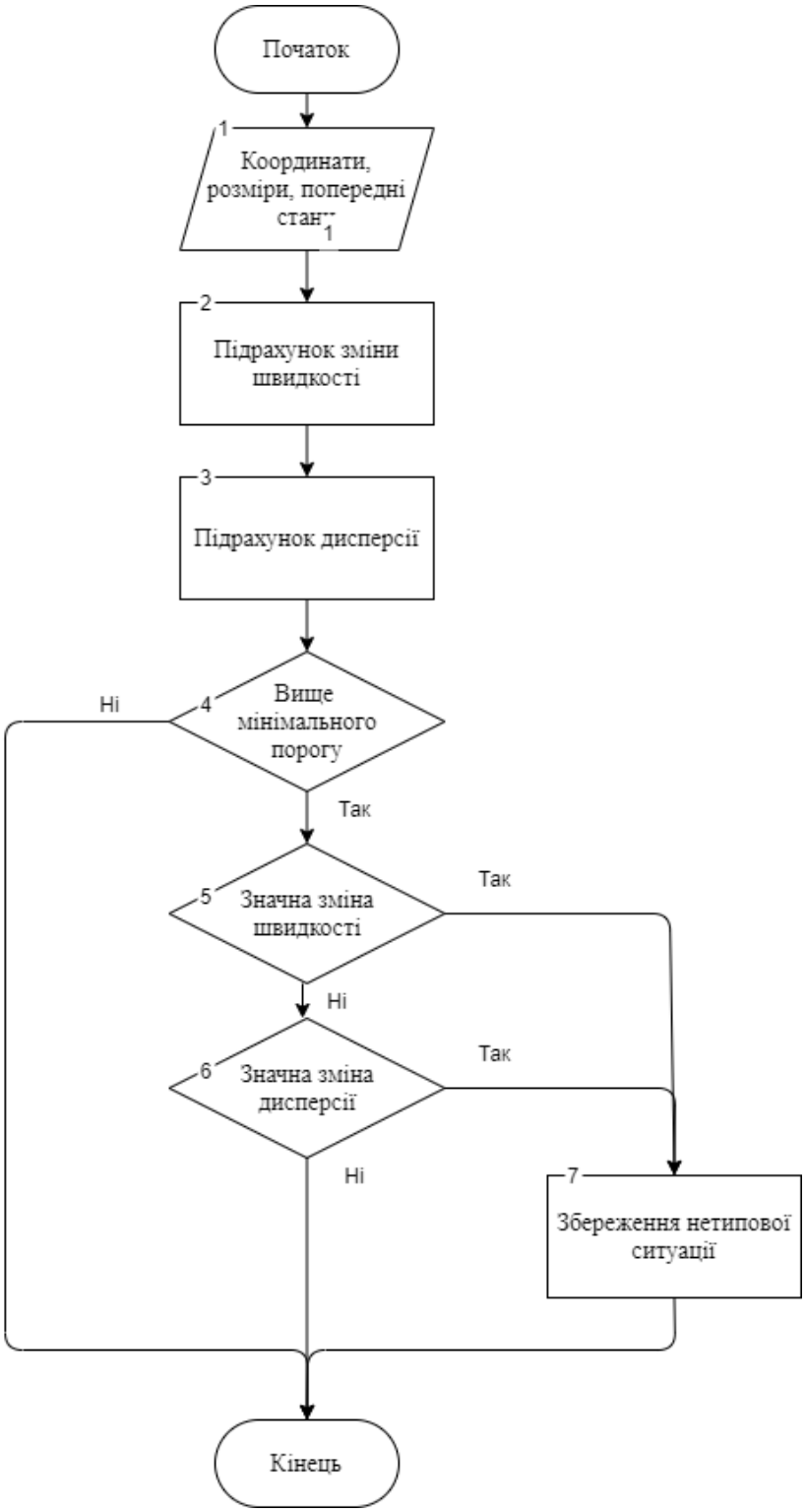

Рисунок 3.8 – Схема алгоритму визначення нетипових ситуацій

Отже, дана схема описує основний алгоритм, за допомогою якого виконуватиметься визначення аномалій у даному об’єкті.

3.3 Обґрунтування вибору мови та середовища програмування програмних засобів розпізнавання нетипових ситуацій на відео

Для створення даної системи потрібно обрати мову програмування. Існує багато мов програмування, що підтримують роботу з нейронними мережами. Проведемо порівняння трьох найпопулярніших з них Python, Java та C++. Коротка характеристика мов програмування наведеній у таблиці 3.1

Таблиця 3.1 – Порівняльна таблиця мов програмування

	Java	C++	Python
Imperative	+	+	+
ООП	+	+	+
Functional	-	+	+
Typing	static	static	dynamic
Compilation	+	+	-

Основними перевагами C++ є швидкість виконання програми, універсальність (підтримка різних технологій, включаючи ООП, мета програмування та директивне програмування) [27].

Основні недоліки: складність мови, безпека мови(велика ймовірність виникнення помилок під час створення коду) та слабка підтримка модулів

Перевагами Java є універсальність коду(програми запускаються на різних пристроях), краще реалізоване ООП, велика кількість виключень та перевірок, що покращують безпеку коду та автоматичне керування пам’яттю (збір «сміття») [28].

Основні недоліки: повільніше виконання програми, не має підтримки вказівників, менша кількість можливостей з використанням різних технологій.

Переваги Python полягає у строгій динамічній типізації, простий синтаксис та широкий набір функціоналу, можливість швидкого переносу

програм на інші платформи, відкритий код. Але найбільшою перевагою Python є його популярність у роботі з нейронними мережами та аналітичними системами, що призвело до появи безліч бібліотек та інструментів роботи з ними, при цьому багато з них є у відкритому коді, що дозволяє використовувати їх можливості усім.

Також Python підтримує структури даних високого рівня разом із динамічною семантикою та динамічним зв'язуванням роблять її привабливою для швидкої розробки програм, а також як засіб поєднування наявних компонентів. Python підтримує модулі та пакети модулів, що сприяє модульності та повторному використанню коду. Інтерпретатор Python та стандартні бібліотеки доступні як у скомпільованій, так і у вихідній формі на всіх основних платформах. В мові програмування Python підтримується кілька парадигм програмування, зокрема: об'єктно-орієнтована, процедурна, функціональна та аспектно-орієнтована [29]. Також Python дозволяє створювати WebSocket канали, що потрібно для комунікування з графічним інтерфейсом.

Усі мови мають свої переваги та недоліки, але найкращою мовою для реалізації системи є мова Python, адже ця мова має непогані можливості та реалізації механізмів роботи з нейронними мережами та великою кількістю даних.

Для розробки графічного інтерфейсу системи можна скористатись технологіями реалізованими на Python. Але стандартні бібліотеки мови не довзоляють зробити зручний сучасний інтерфейс, а більш складні бібліотеки складні у використанні. Але однією з переваг MVC є відокремленість вигляду програми, тобто інтерфейсу, що дозволить нам виконати реалзацію графічного інтерфейсу на іншому наборі технологій. У WEB програмуванні великої поплярності набули технологої HTML/CSS з використанням мови JavaScript. Дані технології дозволяють створювати графічні інтерфейси майже будь-якої складності починаючи від WEB сторінок до телефонних (гібридних) та компютерних застосунків.

HTML – це спеціальна мова розмітки, що доволяє описувати базові шаблони вигляду застосунку [30].

CSS – це ехнологія каскадних стилів для HTML, що дозволяє створювати стилі та модифікувати вигляд HTML розмітки [31]. Зручний інструмент у створені графічних інтерфейсів.

Але самі по собі HTML/CSS дають змогу створити лише зручний дизайн, для взаємодії інтерфейсу з сервером потрібна логіка, найкраще з цією задачою справляється JavaScript. Дана мова є скриптовою з нестрогою типізацією та асихронним кодом. Має можливість стоврення WebSocket каналу, що є вважливим, і запускається в системах де є браузери, або інші рушії такі як Node.js [32]. За допомогою даних технологій буде реалізовуватись інтерфейс користувача.

Для розробки даної системи було обрано середовище програмування Visual Studio Code. Деякі відомості про середовище розробки:

Visual Studio Code — засіб для створення, редагування та зневадження сучасних веб-застосунків і програм для хмарних систем. Visual Studio Code розповсюджується безкоштовно і доступний у версіях для платформ Windows, Linux і OS X [33].

За основу для Visual Studio Code використовуються напрацювання вільного проекту Atom, що розвивається компанією GitHub. Зокрема, Visual Studio Code є надбудовою над Atom Shell, що використовують браузерний рушій Chromium і Node.js.

Редактор містить вбудований інструмент відслідковування помилок, інструменти для роботи з Git і засоби рефакторингу, навігації по коду.

Найголовніше, що VS Code підтримує роботу багатьох мов програмування, а найголовніше роботу з Python та JavaScript. Також середовище дозволяє встановлювати розширення, що спрощують роботу з будь-якою мовою чи технологією (HTML, CSS і т.д).

Отже, VS Code найкраще підходить для розробки на обраній мові програмування і має усі інструменти та засоби, які потрібні для програмної реалізації системи.

В процесі вибору мови програмування була основною совою розробки була обрана мова Python. Дана мова програмування має велику кількість бібліотек та інструментів для роботи з нейронними мережами. Однією з таких бібліотек є OpenCV.

OpenCV (англ. Open Source Computer Vision Library, бібліотека комп'ютерного зору з відкритим кодом) — бібліотека функцій та алгоритмів комп'ютерного зору, обробки зображень і чисельних алгоритмів загального призначення з відкритим кодом. [16]

Бібліотека надає засоби для обробки і аналізу вмісту зображень, у тому числі розпізнавання об'єктів на фотографіях (наприклад, осіб і фігур людей, тексту тощо), відстеження руху об'єктів, перетворення зображень, застосування методів машинного навчання і виявлення загальних елементів на різних зображеннях.

Дана бібліотека містить модуль роботи з глибокими нейронними мережами, до яких відноситься і згорткова нейронна мережа, dnn (deep neural network), а також засоби обробки зображень.

Також бібліотека підтримує ряд алгоритмів відстеження:

– Boosting
– CSRT
– GOTURN
– KCF
– MIL
– TLD
– MOSSE

Кожен з цих алгоритмів відрізняється по точності та швидкості роботи. Звичайно чим швидший алгоритм тим менш точнішим він є. Найточнішим серед них є CSRT алгоритм, який буде використовуватись у системі.

CSRT (Discriminative Correlation Filter with Channel and Spatial Reliability) алгоритм базується на використанні карт просторової надійності, що налаштовує підтримку фільтра на частину об'єкта, придатну для відстеження. Це дозволяє збільшити область пошуку та покращити відстеження не прямокутних об'єктів. Оцінки надійності відображають якісно каналових фільтрів і використовуються як коефіцієнти зважування ознак в локалізації [16].

Отже, опис основних функції та класів, що будуть використовуватися з бібліотеки OpenCV, наведені в табл.3.2.

Таблиця 3.2 – Основні класи та методи бібліотеки OpenCV

Ім’я	Опис
dnn	Клас, що містить інструменти для роботи з глибокими нейронними мережами.
dnn.readNet	Метод, що дозволяє зчитувати конфігурацію та модель ваг мережі, підтримує різні імплементації, що працюють з моделями навченими на різних платформах (наприклад Tensorflow). Повертає об’єкт мережі.
dnn.blobFromImage	Метод, що формує вхідний вектор мережі, базуючись на заданій моделі
dnn.NMSBoxes	Метод, що фільтрує серед вихідних даних мережі неточні, а також групує множині виявлені об’єкти, якщо вони накладаються.
VideoCapture	Метод, що дозволяє виконувати зчитування відеопотоку з обраного джерела.
resize	Метод, що дозволяє змінювати розміри зображення.
imshow	Метод, що дозволяє виконувати виведення зображень на екран.

Продовження таблиці 3.2

rectangle	Метод, що дозволяє виводити прямокутник виявленого об’єкту.
TrackerCSRT_create()	Метод, що повертає об’єкт CSRT відстеження..
Tracker.init	Метод об’єкту CSRT, що задає початкове положення на зображені.
Tracker.update	Метод об’єкту CSRT, що виконує пошук нового положення об’єкту базуючись на новому зображені.
Net.setInput	Метод об’єкту dnn, що дозволяє вказати вхідний вектор даних.
Net.forward	Метод об’єкту dnn, виконує проходження вхідного вектору по мережі, повертає вихідний вектор мережі.

Отже, основні інструменти для роботи з глибокими нейронними мережами та обробки вхідних даних є реалізованими у бібліотеці OpenCV, що являє собою зручний спосіб реалізації деяких модулів системи, що дозволить зосередити основні сили на розробці аналітичного модуля.

3.4 Програмна реалізація інформаційної технології розпізнавання нетипових ситуацій на відео

Проведемо програмну реалізацію основних методів модуля аналізу сцен та опишемо принципи їх роботи та особливості реалізації за допомогою мови програмування Python.

Отже, під час проектування модуля було визначено, що основна логіка та алгоритм роботи модуля буде зосереджений у класі Object, що являє собою сутність виявленого об’єкту. Тому визначимо основні властивості цього класу, які будуть використовуватись під час аналізу.

```
MIN_CHANGE_DU = 3
MAX_TIMES_CHANGE_DU = 2
MIN_DIS_VALUE = 3
MAX_TIMES_CHANGE_DIS = 2
# previous state of  X and Y
prevX = None
prevY = None
# middle changes X and Y
dXm = None
dYm = None
# dispercy Dx and Dy
disDx = None
disDy = None
bbox = None
dangerSpeed = False
dangerAccuracy = False
```

Тут поля MIN_CHANGE_DU, MAX_TIMES_CHANGES_DU, MIN_DIS_CHANGE, MAX_TIMES_CHANGES_DIS відповідають порогам, що обмежують зміну швидкості та дисперсії знизу та згори відповідно, при чому верхня межа означає у скільки разів відхилення від теперішнього значення середнього значення буде відповідати нетиповому стану об’єкту. Поля prevX та prevY зберігають попередні координати об’єкту, а dXm та dYm – зберігають значення середніх значень зміни координат тобто швидкості. Значення полів disDx та disDy використовуються для зберігання дисперсії змін. Поле bbox – це структура, що приходить під час відстеження і містить чотири цілих числа, а саме (x, y, w, h), тобто координати і розміри об’єкту, що відстежується, dangerSpeed та dangerAccuracy відповідають стану об’єкту, коли він стає аномальним один з цих полів приймає значення істини.

Тепер розгладнемо основні методи, що реалізовані у класі.

```
def __init__(self, id, frame, bbox, confidence):
    self.id = id
    self.confidence = confidence
    self.bbox = bbox
    self.tracker = cv2.TrackerCSRT_create()
    self.tracker.init(frame, bbox)
```

Метод __init__ викликається при створенні екземпляру класу Object у Python є стандартним методом для всіх класів, при цьому будь-який метод класу у Python першим аргументом приймає посилання на екземпляр класу, що відкриває доступ до внутрішніх його методів та полів. Кожен екземпляр має своє id, а також початковий стан bbox. Також в даному класі створюється екземпляр класу Tracker, що реалізує відстеження у бібліотеці OpenCV, при чому він приймає початкове положення об’єкту, а також зображення фрейму – це потрібно для роботи алгоритму.

```
def updateTracker(self, frame, withSafeState):
    (success, box) = self.tracker.update(frame)
    if success == True:
        self.bbox = box
        self.calculation(box, withSafeState)
    else:
        print("FALSE", box)
    return (success, box, self.id, self.isDanger())
```

Даний метод виконує оновлення стану відстеження. Тобто при зміні кадру потрібно виконати пошук даного об’єкту на новому кадрі на основі попереднього. Даний функціонал виконує метод update, при цьому він повертає два значення: перше логічне значення, що означає чи успішно було виконано відстеження нового положення об’єкту, друге – нове значення bbox, що містить координати і розміри об’єкту. При цьому саме в цьому методі виконується виклик методу calculation, що виконує підрахунок значень, за якими відбувається аналіз. Коли даний метод повертає, що об’єкт було втрачено, потрібно даний об’єкт видалити зі списку об’єктів, що знаходиться в класі ObjectManager.

```
(x, y) = self._getCenter(bbox)
if self.prevX != None and self.prevY != None:
    if self.dXm != None and self.dYm != None:
        oldDx = self.dXm
        oldDy = self.dYm
        currDx = abs(x - self.prevX)
        currDy = abs(y - self.prevY)
        if (abs(currDx) > self.MIN_CHANGE_DU and abs(oldDx) >
        self.MIN_CHANGE_DU) and (abs(currDy) > self.MIN_CHANGE_DU
        and abs(oldDy) >
        self.MIN_CHANGE_DU):
            if abs(currDx) > self.MAX_TIMES_CHANGE_DU * abs(oldDx) or
            abs(currDy) > self.MAX_TIMES_CHANGE_DU * abs(oldDy):
                self.dangerSpeed = True
                print("Speed", self.id)
                print(currDx, oldDx)
                print(currDy, oldDy)
        else:
            self.dangerSpeed = False

        self.dXm = (currDx + oldDx) / 2
        self.dYm = (currDy + oldDy) / 2
        self._calcDis(currDx, currDy)
        self.prevX = x
        self.prevY = y
    else:
        self.dXm = abs(x - self.prevX)
        self.dYm = abs(y - self.prevY)
else:
    self.prevX = x
    self.prevY = y
```

В даному фрагменті коду відбувається підрахунок основних числових значень, що використовуються в алгоритмі. Рахується теперішня зміна координат об’єкта – currDx та currDy, що являє собою по суті швидкість, адже зміна відбувається в період зміни кадру. Саме це значення порівнюється з порогом, якщо воно менше нижчої межі, далі перевірка не відбувається, при чому значення беруться по модулю – abs(currDx). Якщо ж дане значення переважає попереднє середнє значення oldDx чи oldDy у задану кількість разів, стан об’єкта dangerSpeed змінюється на істинне, що у майбутньому буде

використовуватись для збереження даних про аномальний об’єкт. Після перевірки обновляються дані середньої зміни dXm та dYm, а також змінюється попереднє значення prevX та prevY, що будуть використовуватися у подальших кадрах. В загальному принцип досить простий і полягає лише в підрахунку значень і перевірці через умовний оператор if else.

Схожим принципом рахується зміна дисперсії, відрізняється лише дані та формули, по яким відбувається підрахунок.

```
        newdisDx = math.sqrt( currDx**2 + self.dXm**2 )
        newdisDy = math.sqrt( currDy**2 + self.dYm**2 )
        if abs(disX) > self.MIN_DIS_VALUE and abs(self.disDx) > self.MIN_DIS_VALUE
        and abs(disY) > self.MIN_DIS_VALUE and abs(self.disDy) >
        self.MIN_DIS_VALUE:
            if  abs(disX) > self.MAX_TIMES_CHANGE_DIS * abs(self.disDx) or
            abs(disY) > self.MAX_TIMES_CHANGE_DIS * abs(self.disDy):
                self.dangerAccuracy = True
                print("Accuracy", self.id)
                print(disX, self.disDx)
                print(disY, self.disDy)
        else:
            self.dangerAccuracy = False
        self.disDx = newdisDx
        self.disDy = newdisDy
    else:
        self.disDx = disX
        self.disDy = disY
```

Для початку підраховується нова дисперсія newdisDx та newdisDy. Щоб виконувати математичні операції в Python існує стандартна бібліотека math. Для того, щоб отримати корінь, використовується функція sqrt. Конструкція dXm**2 означає піднесення до квадрату. Після як було вирахувана нова дисперсія проводяться аналогічні операції, спочатку порівнюється з нижнім порогом, щоб відкинути незначні зміни, після цього порівнюють з верхнім, і якщо дане значення вище змінюють стан dangerAccuracy стає істинним.

Отже, в даному розділі було проведено опис основних програмних частин, що були реалізовані мовою програмування Python та реалізують основні частини алгоритму по аналізу сцен.

3.5 Тестування та аналіз результатів роботи програмних засобів розпізнавання нетипових ситуацій на відео

Під час розробки системи були визначені ряд завдань, які система має виконувати:

– Можливість виявлення об’єктів
– Можливість відстеження та збір інформації
– Можливість визначення аномалій

Для тестування спроможності системи виконувати задані задачі виберемо відео, на яких виникає ДТП, та проаналізуємо чи зможе програма виявити їх.

Для тестування будемо зберігати основні параметри пораховані системою, щоб в подальшому проаналізувати результати.

Проаналізуємо аномальну ситуацію, що показана на рис. 3.9.

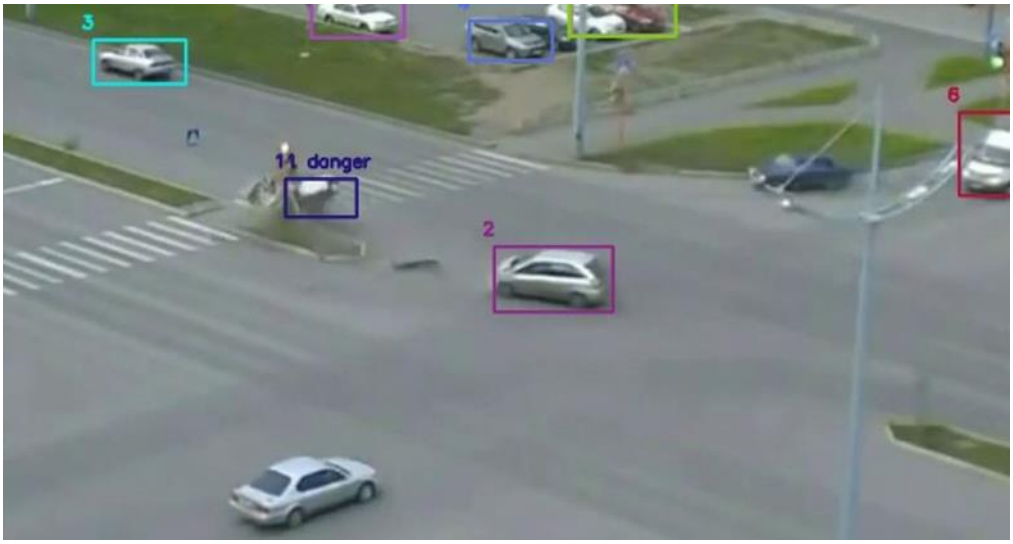

Рисунок 3.9 – Виявлена нетипова ситуація

Виконаємо аналіз зміни даного об’єкту по X координаті. Отримані системою дані наведені в таблиці 3.3

Таблиця 3.3 – Дані отримані під час тестування

Фрейм	1	2	3	4	5	6	7	8	9	10	11	12
X	157	168	179	198	208	222	234	268	278	297	357	468
Фрейм	13	14	15	16	17	18	19	20	21	22	23	24
X	520	586	578	574	578	579	582	589	601	612	621	630

Побудуємо графіки зміни швидкості та дисперсії базуючись на отриманих даних. Дані графіки наведені на рис. 3.10 та рис 3.11.

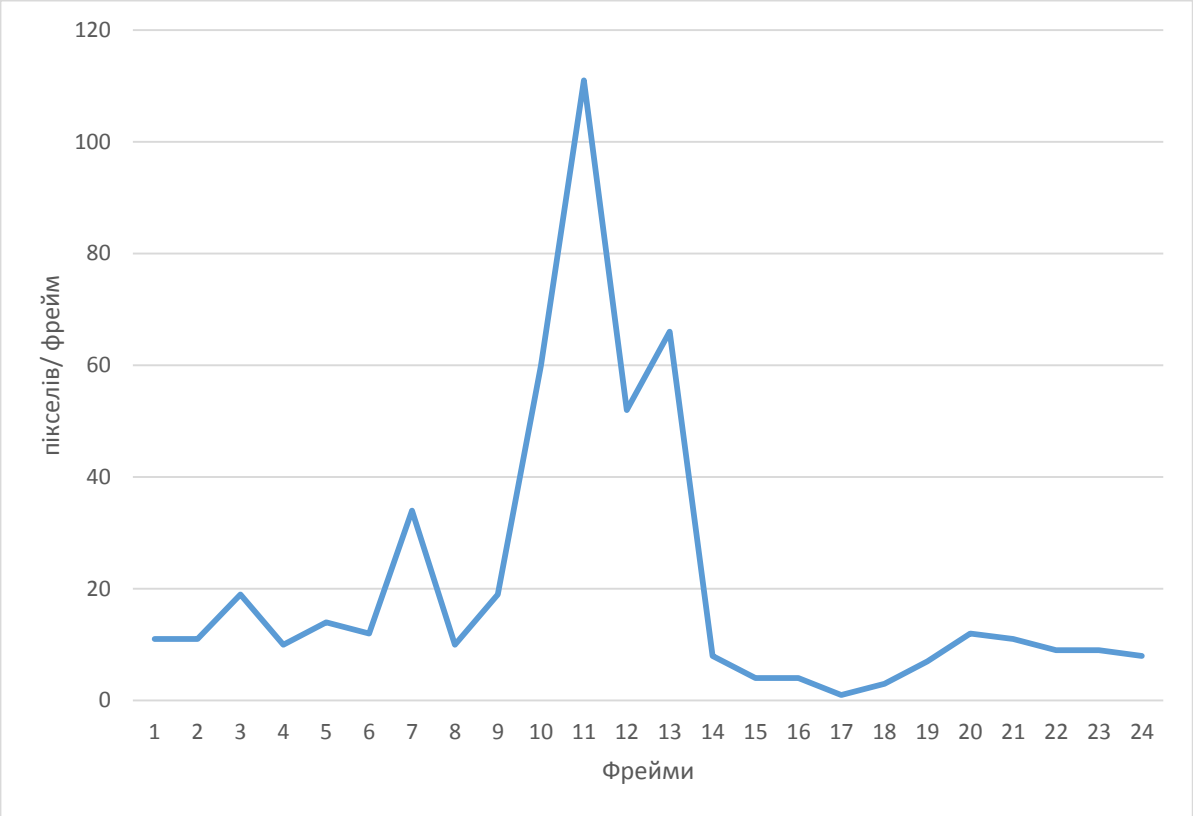

Рисунок 3.10 – Графік зміни середньої швидкості

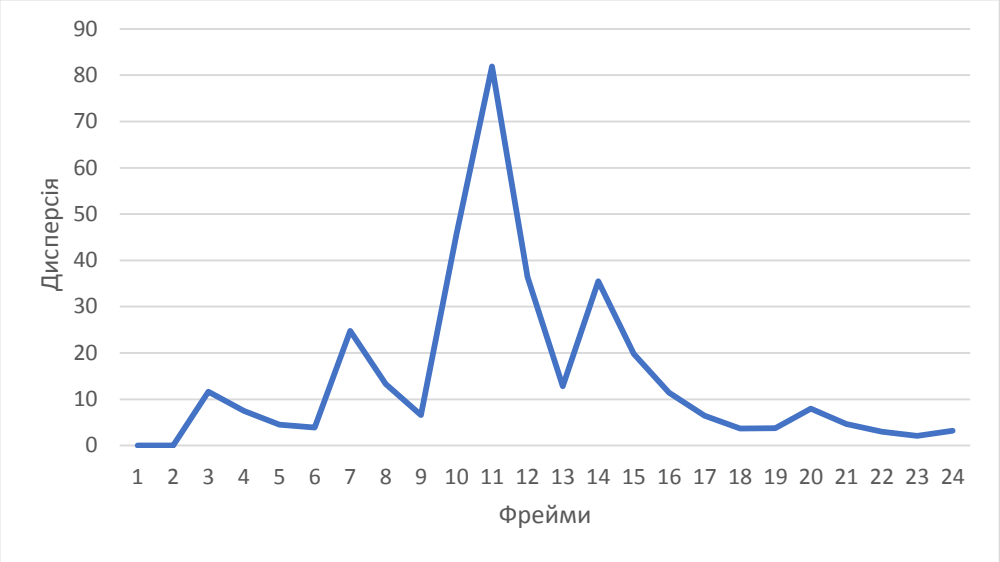

Рисунок 3.11 – Графік зміни дисперсії

Аналізуючи графіки можна зробити висновок, що при зіткненні двох об’єктів один із них отримав значне прискорення по X координаті, на графіку це видно за рахунок стрімкого зростання графіку. Також можна спостерігати, що після цього відбувалася стрімка зупинка об’єкту, що відповідає стрімкому спаду. Відповідно до цього спрацював поріг, що являє собою максимальне зростання або спадання графіку, що знаходиться в межах норми.

Звідси можна зробити висновок, що система має здатність виявляти стрімкі зміни швидкості об’єктів, що можуть вказувати на нетипові ситуації.

Загальні тестування системи наведені в таблиці 3.4.

Таблиця 3.4 – Загальне тестування системи

	Аналог		Програма	
Час	Денний	Нічний	Денний	Нічний
Загальна кількість	35	15	35	15
Виявлені	27	9	29	11
Точність	77%	60%	82%	73%
Середня точність	68,5%		77,5%	

Загальне тестування показало, що точність виявлення об’єктів в середньому складає 77,5%, що є досить непоганим результатом.

З 50 протестований аварій було обрано 35, що відбуваються у денний час доби, а 15 – у нічний. Це пояснюється тим, що денний час довший відносно нічного.

З 35 протестованих аварій днем система змогла визначити 29, що складає точність більше 82%. У нічний час з 15 ситуацій визначено було 11 ситуацій, що складає приблизно 73%. Аналог зміг днем виявити лише 27 ситуацій, що складає 77%, а у нічний – 9, що становить 60%. У середньому аналог має точність приблизно 68,5%, а розроблена система 77,5%, звідси можна зробити висновок, що система збільшила точність виявлення нетипових ситуацій в порівнянні з аналогічною системою.

Найкраще система визначає динамічні ситуації, де зміна положення об’єктів суттєва, ситуації де зіткнення були незначними система ігнорувала. Також на точність впливала точність відстеження об’єктів.

Також проведемо тестування швидкості роботи програми. Для цього виконаємо запуск програми на відео з стандартними параметрами розширення. Запуск буде відбуватись у звичайному однопоточному режимі, у режимі багатопоточності на 8 ядрах, а також у режимі багатопоточності з використанням технології CUDA. Результати тестування наведені на рис. 3.12.

Як можна спостергати під час використання 1 ядра і потоку швидкість програми складає приблизно 1.5 кадра на секунду. Що не є хорошим результатом. Після розпаралелювання обробки масивів даних швикість зросла майже до 3.5 кадрів на секунду.

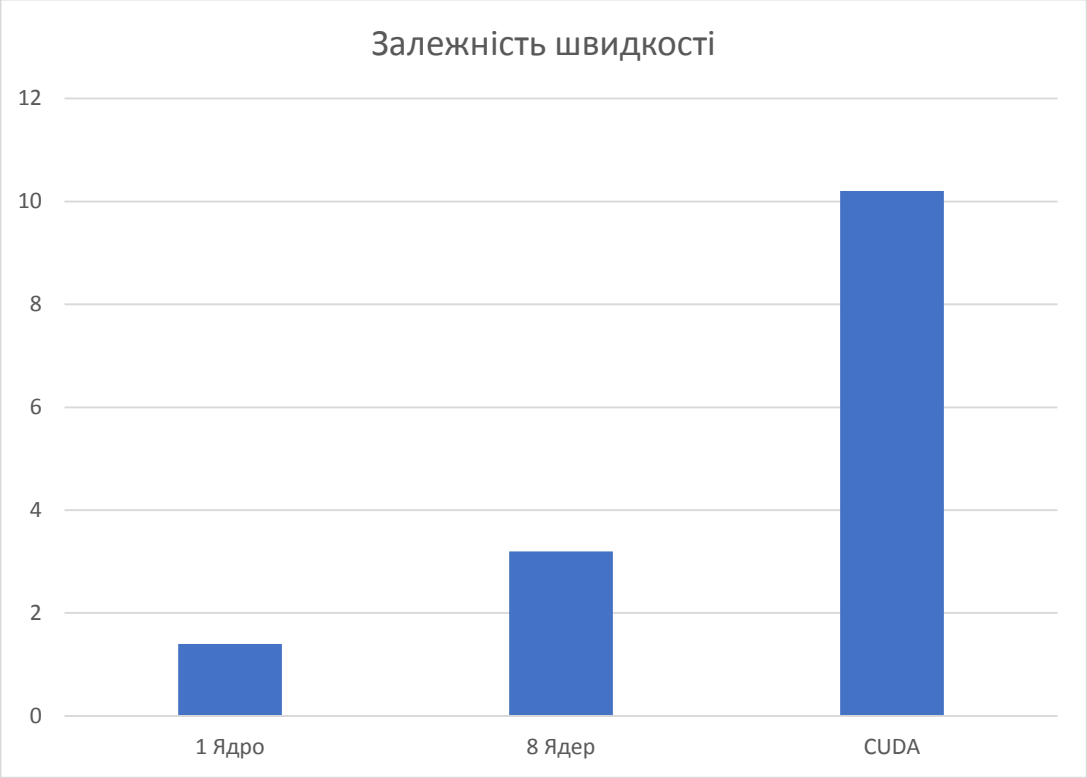

Рисунок 3.12 – Тестування швидкості програми

Найкращі результати були отримані при додаванні обчислень за допомогою CUDA, що довзолило прискорити нейронну мережу. В цьому випадку швидкість складає трошки більше 10 кадрів на секунду. В загальному швидкість роботи програми не є достатньо високою, що пояснюється складністю алгоритмів відстеження та нейронної мережі. Також на даний параметр сильно впливає технічні характеристики системи, на якій запускається програма, тобто чим потужніший процесор та графічне ядро тим швидше працює програма. В подальшому потрібно покращувати методи багатопоточності і проводити заходити щодо спрощення і оптимізації алгоритмів і нейронної мережі.

3.6 Висновок

В результаті програмної реалізації системи розпізнавання нетипових ситуацій на відео було виконано етап проектування системи. Проектування

виконувалось з використанням MVC підходу. Було прийнято рішення винести графічний інтерфейс та зв’язати логічне ядро з інтерфейсом за допомогою WebSocket протоколу. Також були розроблені DFD та IDEF0 діаграми, а також UML діаграми застосунку. Було описано і проаналізовано основний алгоритм роботи програми.

В подальшому було обрано мову програмування та технології, які будуть використовуватись. Основною мовою програмування було обрано Python через зручність роботи з нейронними мережами. Також було прийняте рішення використання бібліотеки OpenCV для роботи з нейронними мережами. Для розробки графічного інтерфейсу було обрано мову програмування JavaScript, а також допоміжні технології HTML та CSS. Середовищем було обрано Visual Studio Code.

Під час програмної реалізації було виконано опис основних частин коду та методів, що використовувались під час програмної реалізації. В результаті роботи було виконано тестування роботи системи. Основна точність роботи програми становить 77.5%, що є досить хорошим результатом але недостатнім.

Також було проведено тестування швидкості роботи програми. В однопоточному режимі швидкість становить близько 1.5 кадрів на секунду, при багатопоточному режимі – 3.5, а при використані CUDA – 10. Це непоганий результат, але невідповідає станадартним 60 кадрів на секунду.

В загальному програмна реалізація була виконана належним чином і розроблена система показала непогані результати швидкості і точності. Але потрібно продумувати заходи щодо оптимізації точності і швидкості.

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 Колесницький О.К. та . Рейда О.М.

Виконаємо оцінювання комерційного потенціалу розробки за допомогою 12-ти критерій, для оцінювання будемо використовувати 5-ти бальну шкалу.

Результати оцінювання комерційного потенціалу розробки відображено в таблиці 4.1.

Таблиця 4.1 – Результати оцінювання комерційного потенціалу розробки

Критерії	Прізвище, ініціали, посада експерта	
	1. Експерт 1	2. Експерт 2
	Бали, виставлені експертами:	
1	4	4
2	4	3
3	4	4
4	4	3
5	3	4
6	4	4
7	4	3
8	4	4
9	4	4
10	4	4
11	3	4
12	4	4
Сума балів	$СБ_1 = 47$	$СБ_2 = 45$
Середньоарифметична сума балів $\overline{СБ}$	$\overline{СБ} = \frac{\sum_{1}^{3} СБ_i}{2} = 46$	

Отже, аналізуючи отримані результати оцінювання можна зробити висновок, що розробка має досить високий рівень комерційного потенціалу.

4.2 Прогнозування витрат на виконання науково-дослідної, дослідно-конструкторської та конструкторсько-технологічної роботи

Щоб виконати розробку даної системи потрібно витратити велику кількість матеріальних ресурсів. Спрогнозуємо основні витрати, які будуть необхідні в процесі розробки.

Перш за все необхідні витрати на заробітню плату персоналу, що буде займатись розробкою і підтримкою системи. Основна заробітна плата для розробників визначається за формулою (4.1):

$$З_о = \frac{М}{Т_р} \cdot t, \quad (4.1)$$

де М- місячний посадовий оклад конкретного розробника;

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

t - число днів роботи розробника, t = 45 днів.

Розрахунки заробітних плат основних кадрів 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	10000	1047,62	10	4761,9
Інженер-програміст	5000	714,29	45	10714,28
Всього:				15476,19

Розрахуємо додаткову заробітну плату:

$$З_{дод} = 0,1 \cdot 15476,1905 = 1547,61 \text{ (грн.)}$$

Нарахування на заробітну плату операторів НЗП розраховується як 22% від суми їхньої основної та додаткової заробітної плати:

$$Н_{зп}= (З_о + З_р)\cdot \frac{\beta}{100}, \quad (4.2)$$

$$Н_{зп}= (15476{,}1905+1547{,}61905)\cdot \frac{22}{100} = 3745{,}23 \text{ (грн.)}.$$

Розрахунок амортизаційних витрат для програмного забезпечення виконується за такою формулою:

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

де Ц – балансова вартість обладнання, грн;

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

Т – Термін використання (Т=3 міс.).

Таблиця 4.3 – Розрахунок амортизаційних відрахувань

Найменування програмного забезпечення	Балансова вартість, грн.	Норма амортизації, %	Термін використання, міс.	Величина амортизаційних відрахувань,грн
Потужний комп’ютер	32000	25	2	1333
Всього:				1333

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

$$K = \sum_1^n Н_і \cdot Ц_і \cdot К_і, \quad (4.4)$$

де n – кількість комплектуючих;

Ні - кількість комплектуючих і-го виду;

Ці – покупна ціна комплектуючих і-го виду, грн;

Кі – коефіцієнт транспортних витрат (приймемо Кі = 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.
Флешка	шт.	300	1	300
Пачка паперу	уп.	500	1	500
Ручка	шт.	25	3	75
Всього з урахуванням транспортних витрат				962,5

Витрати на силову електроенергію розраховуються за формулою:

$$В_е=В\cdot П\cdot Ф\cdot К_п \quad (4.5)$$

де В – вартість 1кВт-години електроенергії (В=2,5 грн/кВт);

П – установлена потужність комп'ютера (П=1,6кВт);

Ф – фактична кількість годин роботи комп'ютера (Ф=210 год.);

$К_п$ – коефіцієнт використання потужності ($К_п< 1$, $К_п = 0,8$).

$$В_е= 2,5 \cdot 1,6 \cdot 210 \cdot 0,8 = 672 \text{ (грн.)}$$

Розрахуємо інші витрати $В_{ін}$.

Під додатковими витратами $І_в$ можна взяти суму яка на 100…300% більша від суми основної заробітної плати розробників та робітників, які були виконувати дану роботу, тобто:

$$В_{ін} = (1..3) \cdot (З_о + З_р). \quad (4.6)$$

Отже, сума інших витрат становить:

$$В_{ін} = 1 * (15476,19 + 1547,61) = 17023,8 \text{ (грн).}$$

Підрахуємо суму всіх статей витрат, щоб отримати загальні витрати на виконання виконання даної частини роботи:

$$В = З_о + З_д + Н_{зп} + А + К + В_е + І_в$$

В = 15476,19+1547,61 +3745,23+1333+962,5+672+17023,8 = 40760,35 (грн.)

Образуємо вартість наукової роботи, використану для реалізації, за формулою:

$$В_{заг} = \frac{В_{ін}}{\alpha} \quad (4.7)$$

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

$$В_{заг} = \frac{40760,35}{1} = 40760,35 \text{ (грн.)}$$

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

$$ЗВ = \frac{В_{заг}}{\beta} \quad (4.8)$$

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

Отже, розрахуємо загальні витрати:

$$ЗВ = \frac{40760,35}{0,9} = 45289,28 \text{ (грн.)}$$

4.3 Прогнозування комерційних ефектів від реалізації результатів розробки

Виконаємо прогнозування рівню прибутку, який можна отримати від реалізації розробки системи. Зростання чистого прибутку потрібно оцінювати з врахуванням наявної вартості грошей, що забезпечить основним користувачам системи надходження додаткових коштів, що лозволять покращити фінансові результати діяльності.

Спершу потрібно отримати оцінку 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, можна розрахувати за формулою:

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

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 30 грн (що автоматично спричинить збільшення чистого прибутку підприємства на 30 грн). Кількість користувачів, які будуть користуватись інформаційною технологією,

збільшиться: протягом першого року – на 250 користувачів, протягом другого року – на 300 користувачів, протягом третього року – 350 користувачів. Реалізація інформаційної технології до впровадження результатів наукової розробки складала 900 користувачів, а прибуток, що отримував розробник до впровадження результатів наукової розробки – 450 грн.

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

Отже, збільшення чистого прибутку протягом першого року складатиме:

$$\Delta П_1 = 30 \cdot 250 + (450 + 30) \cdot 250 = 127500 \text{ (грн.)}$$

Протягом другого року:

$$\Delta П_2 = 30 \cdot 300 + (450 + 30) \cdot (250 + 300) = 273000 \text{ (грн.)}$$

Протягом третього року:

$$\Delta П_3 = 30 \cdot 350 + (450 + 30) \cdot (250 + 300 + 350) = 442500 \text{ (грн.)}$$

4.4 Розрахунок ефективності вкладених інвестицій та періоду їх окупності

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

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

$$Е_{абс} = (ПП - PV), \qquad (4.10)$$

Схематично рух платежів (інвестицій та додаткових ресурсів) зображено на рис 4.1.

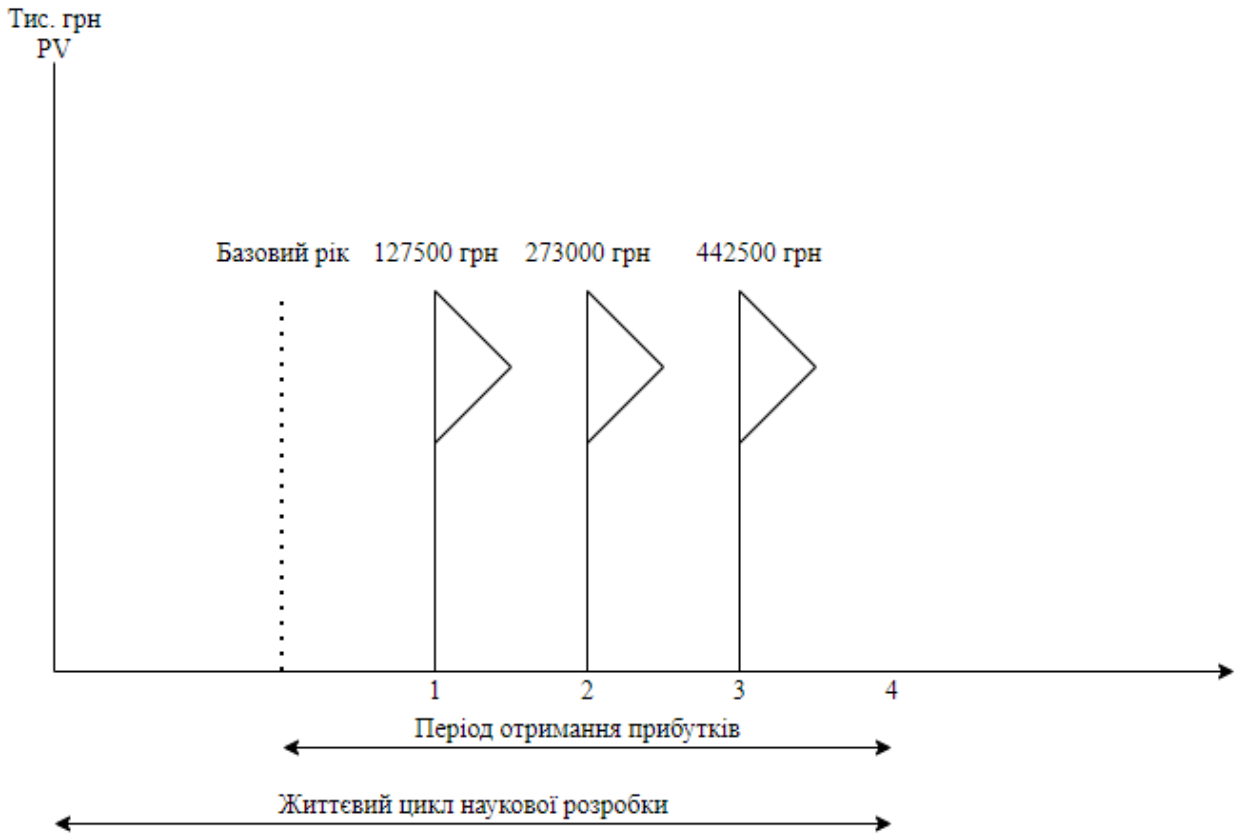

Рисунок 4.1 – вісь часу з фіксацією платежів, що мають місце під час розробки та впровадження результатів НДДКР

Розрахуємо вартість чистих прибутків за формулою:

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

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;іі

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:

$$ПП = \frac{45289,28}{(1+0,1)^0} + \frac{127500}{(1+0,1)^2} + \frac{273000}{(1+0,1)^3} + \frac{442500}{(1+0,1)^4} = 658003,58 \text{ (грн.)}$$

Тоді розрахуємо $E_{абс}$:

$$E_{абс} = 658003,58 - 45289,28 = 612714,29 \text{ (грн.)}$$

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

Виконаємо розрахунок відносної (щорічної) ефективності вкладених в наукову розробку інвестицій $E_в$ за формулою:

$$E_в = \sqrt[T]{1 + \frac{E_{абс}}{PV}} - 1 \qquad (4.12)$$

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

PV – теперішня вартість інвестицій PV = ЗВ, грн;

$T_ж$ – життєвий цикл наукової розробки, роки.

Отже, будемо мати:

$$E_в = \sqrt[3]{1 + \frac{612714,29}{45289,28}} - 1 = 1,44 \ (144\%)$$

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:

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

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 Ев = 144% > 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 $Т_{ок}$ розраховується за формулою:

$$Т_{ок} = \frac{1}{Е_{в}} \quad (4.14)$$

$$Т_{ок} = \frac{1}{1,44} = 0,69 \text{ року}$$

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 пітверджено, 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 45289,28 грн.

Після чого було проведено підрахунок можливого прибутку після впровадження інформаційної технології. Перший рік використання планується отримання прибутку у 127500 грн , другий – 273000 грн, третій – 442500 грн. Загальна ефективність розробки становить 144%. Розрахований період окупності становить 0,69 року.

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

ВИСНОВКИ

У даній магістерській кваліфікаційній роботі було виконано розробку інформаційної технології розпізнавання нетипових ситуацій на відео. В результаті аналізу, було виявлено основні задачі, які потрібно виконати та обрано основні методи вирішення. Для задачі збору інформації було обрано метод відстеження об’єкту на зображенні, а метод аналізу базується на зміні основних характеристик. Були обґрунтовані та запропоновані основні формули, за якими відбуватиметься визначення нетипових ситуацій.

Було виконано проектування за допомогою мови моделювання UML, а також розроблено основну архутектуру, яка базується на патерну MVC, та алгоритм роботи аналізу сцен. Для реалізації даної програми було обрано мову програмування Python, з використанням бібліотеки OpenCV, а для виконання графічного інтерфейсу мову програмування JavaScript з підтримкою HTML/CSS. В результаті було отримано інтелектуальний модуль, що здатний виявляти нетипові ситуації ґрунтуючись на зміні станів об’єкту.

Тестування модуля показало, що модуль має точність приблизно 77,5%, при цьому точність аналога складає приблизно 68,5%. Дана точність показує непоганий результат, але потрібне запровадження оптимізаційних заходів, що могли б покращити точність. Також було виконане тестування швидкості роботи системи. За допомогою використання багатопоточності та технологій CUDA, що дозволяє обробки на графічному ядрі, вдалося досягти швидкості обробки 10 кадрів на секунду, що майже в 7 разів більше за швидкість використання 1 потоку.

В загальному можна констатувати збільшення достовірності виявлення нетипових ситуацій на 9%, та швидкості роботи програми у 7 разів, тому основну мету роботи було досягнуто.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. О. Колесницький, С. Кукунін, М. Дерев’янко, і А. Преподобний Мендеш Да Майа, Розпізнавання нетипових ситуацій на дорозі за допомогою згорткової нейронної мережі, OEIET, vol 38, № 2, с. 38-44, Бер 2020. DOI: https://doi.org/10.31649/1681-7893-2019-38-2-38-44

2. Преподобний Мендеш да Майа А. А., Колесницький О. К., Дерев'янко М. Ю. «Визначення аномальних ситуацій за допомогою нейронних мереж», в Матеріали конференції «XLVIII Науково-технічна конференція підрозділів Вінницького національного технічного університету (2019)», Вінниця, 2019. С. 801-803 [Електронний ресурс]. Режим доступу: https://conferences.vntu.edu.ua/index.php/all-fitki/index/pages/view/zbirn2019 Дата звернення: Черв. 2019

3. Колесницький О. К., Преподобний Мендеш да Майа А. А., Дерев’янко М. Ю., «Дослідження ефективності використання згорткової нейронної мережі для визначення нетипових ситуацій на дорозі», в Матеріали конференції «XLIX Науково-технічна конференція підрозділів Вінницького національного технічного університету (2020)», Вінниця, 2020, с. 983-986. [Електронний ресурс]. Режим доступу: https://conferences.vntu.edu.ua/index.php/allvntu/index/pages/view/zbirn2020 Дата звернення: Черв. 2020.

4. Колесницький О., Дерев’янко М., Преподобний Мендеш да Майа А. А. «Використання згорткової нейронної мережі для визначення нетипових ситуацій на дорозі», «ІНТЕРНЕТ-ОСВІТА-НАУКА-2020», XII Міжнародна науковопрактична конференція ІОН-2020, 26-29 травня, 2020 : Збірник праць. – Вінниця : ВНТУ, 2020 – с. 55-57. Режим доступу: https://conferences.vntu.edu.ua/index.php/allvntu/index/pages/view/zbirn2020 Дата звернення: Черв. 2020

5. who.int [Електронний ресурс] : [Веб-сайт]. – Режим доступу: https://www.who.int/violence_injury_prevention/road_safety_status/report/ru/ – Назва з екрана.

6. patrol.police.gov.ua [Електронний ресурс] : [Веб-сайт]. – Режим доступу: http://patrol.police.gov.ua/statystyka/ – Назва з екрана.

7. Jeannette Lawrence, Sylvia Luedeking Introduction to neural networks :design, theory and applications. Nevada City, Calif. : California Scientific Software, 1994.

8. Frank Millstein Convolutional Neural Networks in Python: Beginner's Guide to Convolutional Neural Networks in Python. CreateSpace Independent Publishing Platform, 2018. 120 с.

9. Frank Millstein Convolutional Neural Networks in Python: Beginner's Guide to Convolutional Neural Networks in Python. CreateSpace Independent Publishing Platform, 2018. 120 с.

10. Charles K. Chui, Guanrong Chen Kalman Filtering: With Real-Time Applications. Berlin: Springer Science & Business Media, 2009. 229 с.

11. Щербакова Г.Ю. Теорія ймовірностей конспект лекцій. Одеса: Наука і техніка, 2005. 68 с.

12. ResearchGate [Електронний ресурс] : [Веб-сайт]. – Режим доступу: https://www.researchgate.net/publication/281628399_Automatic_and_real-time_Pothole_detection_and_Traffic_monitoring_system_using_Smartphone_Technology – Назва з екрана.

13. ResearchGate [Електронний ресурс] : [Веб-сайт]. – Режим доступу: https://www.researchgate.net/publication/261132505_Automated_Sensing_System_for_Monitoring_of_Road_Surface_Quality_by_Mobile_Devices – Назва з екрана.

14. ResearchGate [Електронний ресурс] : [Веб-сайт]. – Режим доступу: https://www.researchgate.net/publication/224364818_Automatic_Daytime_Road_Traffic_Control_and_Monitoring_System – Назва з екрана.

15. nau.edu.ua [Електронний ресурс] : [Веб-сайт]. – Режим доступу: https://nau.edu.ua/ua/menu/science/naukovi-rozrobki/intelektualna-sistema-monitoringu-dorozhnogo-ruxu.html – Назва з екрана.

16. YOLO: Real-Time Object Detection [Електронний ресурс] : [Веб-сайт]. – Режим доступу: https://pjreddie.com/darknet/yolo (дата звернення 03.04.2020) – Назва з екрана.

17. Quan Nguyen. Mastering Concurrency in Python: Create faster programs using concurrency, asynchronous, multithreading, and parallel programming. Packt Publishing, 2018. 446 с.

18. Richard Carver, Kuo-Chung Tai. Modern Multithreading: Implementing, Testing, and Debugging Multithreaded Java and C++/Pthreads/Win32 Programs. John Wiley & Sons, 2005. 480 с.

19. Боресков А., Харламов А. Основы работы с технологией CUDA. Москва, ДМК пресс, 2010. 232 с.

20. Krasner, G.E. and S.T. Pope. A cookbook for using the Model-View-Controller user interface paradigm in Smalltalk-80. Journal of Object-Oriented Programming. SIGS Publications, New York, USA, 1988. 49 с.

21. Mark Masse. REST API Design Rulebook: Designing Consistent RESTful Web Service Interfaces. O'Reilly Media, Inc, 2011. 116 с.

22. Andrew Lombardi. WebSocket: Lightweight Client-Server Communications. O'Reilly Media, Inc, 2015. 144 с.

23. Frederic P Agnes F. John M. Data Flow Diagram: Information System, Visualization, Computer Data Processing, Flowchart, Control Flow Diagram, Data Island, Dataflow, Functional Flow Block Diagram. Alphascript Publishing,2010.80 с.

24. Dennis M. The Engineering Design of Systems: Models and Methods. New York: Wiley, 2011. 536 с.

25. Дж. Рамбо, М. Блаха UML 2.0. Об’єктно-орієнтована розробка і моделювання. 2 видання. Пітер, 2007. 544 с.

26. Горбунова Т. Объектно-ориентированное программирование и проектирование: учебное пособие. Москва: МГОУ, 2011. 99 с.

27. Виллемер А. Программирование на C++. Москва:Эскмо, 2013.528 с.

28. Нимейер П. Леук Д. Программирование на Java: Исчерпывающее руководство для профессионалов. Москва: Эскмо, 2014. 1216 с.

29. Прохоренок Н. Python. Самое необходимое. Санкт-Петербург: “БХВ-Петербург”, 2010. 416 с.

30. Петюшкин А. В. HTML в Web-дизайне. БХВ-Петербург, 2005. 400с.

31. Дронов В. А. HTML 5 и CSS 3: Современный Web-дизайн. БХВ-Петербург, 2011. 416 с.

32. David Flanagan. JavaScript: The Definitive Guide. O'Reilly Media, Inc, 2006. 1032 с.

33. Ovais M. Ahmed K., Khusro H. Developing Multi-Platform Apps with Visual Studio Code: Get up and running with VS Code by building multi-platform, cloud-native, and microservices-based apps. Packt Publishing Ltd, 2020. 334 с.

34. OpenCV. Open Source Computer Vision [Електронний ресурс] : [Веб-сайт]. – Режим доступу: https://docs.opencv.org/master/ (дата звернення 01.10.2020) – Назва з екрана.

35. GitHub. OpenCV issues [Електронний ресурс] : [Веб-сайт]. – Режим доступу: https://github.com/opencv/opencv/issues (дата звернення 02.10.2020) – Назва з екрана.

36. Python. Documentation [Електронний ресурс] : [Веб-сайт]. – Режим доступу: https://docs.python.org/3/ (дата звернення 01.10.2020) – Назва з екрана.

37. MDN JavaScript Documentation [Електронний ресурс] : [Веб-сайт]. – Режим доступу: https://developer.mozilla.org/uk/docs/Web/JavaScript (дата звернення 10.10.2020) – Назва з екрана.ss

38. Козловський В. О. Методичні вказівки до виконання студентами-магістрантами економічної частини магістерських кваліфікаційних робіт / Уклад. Вінниця: ВНТУ, 2012. – 22 с.