МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Харківський національний університет
імені В. Н. Каразіна
Факультет радіофізики, біомедичної
електроніки та комп'ютерних систем
Кафедра Кафедра фізичної і біомедичної електроніки та комплексних
інформаційних технологій

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ _____
підпис ініціали, прізвище

"____" _____20___ року

Кваліфікаційна робота магістра

на тему: «АВТОМАТИЗОВАНА СИСТЕМА ПОШУКУ ПОТЕРПІЛИХ НА ОСНОВІ ШТУЧНОГО ІНТЕЛЕКТУ»

Виконав: студент ІІ курсу магістратури, групи РЕ-61
спеціальності 153 Мікро- та наносистемна техніка
освітньо-професійна програма «Фізична та біомедична електроніка»

Денис ПИВЕНЬ

Керівник
д. фіз.-мат. наук,
проф. Віктор КАТРИЧ

Консультант
ст. викладач. Євгеній АНТОНЕНКО

2022 рік

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Харківський національний університет імені В.Н.Каразіна

Факультет Радіофізики, біомедичної електроніки та комп'ютерних систем
Кафедра ФБМЕ та КІТ
Спеціальність 153 Мікро- та наносистемна техніка
Освітньо-професійна програма Фізична та біомедична електроніка
Рівень вищої освіти другий (магістерський)

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ _____
підпис ініціали, прізвище
"____" _____20___ року

З А В Д А Н Н Я
НА ДИПЛОМНУ РОБОТУ

_____Півень Денис Олегович_____
(прізвище, ім'я, по батькові студента)

1. Тема роботи Автоматизована система пошуку потерпілих на основі штучного інтелекту

керівник роботи _____
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від "___"_____20__року №___

2. Строк подання студентом роботи _____

3. Перелік питань, які потрібно розробити:
3.1 Розробити метод аналізу відеопотоку з ціллю пошуку потерпілих.
3.2 Розробити алгоритм детектування силуетів.
3.3 Розробити алгоритм аналізу для пошуку потенційних потерпілих.

4. План роботи

№ з/п	Назви етапів роботи
1.	Аналіз технічного завдання
2.	Огляд літератури
3.	Обгрунтування обраних методів розробки
4.	Розробка методу пошуку потерпілих
5.	Розробка алгоритмічного забезпечення
6.	Оформлення пояснювальної записки
7.	Підготовка до захисту

5. Дата видачі завдання _____

Студент _____ _____

підпис прізвище ініціали,

Керівник роботи _____ _____

підпис прізвище ініціали,

РЕФЕРАТ

Півень.Д.О. Автоматизована система пошуку потерпілих на основі штучного інтелекту. Кваліфікаційна робота магістра. Харківський національний університет імені В.Н.Каразіна, 2022, 52 с., 12 рис., 8 джерел.

АВТОМАТИЗОВАНА СИСТЕМА ПОШУКУ ПОТЕРПІЛИХ НА ОСНОВІ ШТУЧНОГО ІНТЕЛЕКТУ.

Об’єкт дослідження – пошук потерпілих використовуючи штучний інтелект.

Предмет дослідження – аналіз систем, для пошуку потерпілих використовуючи БПЛА.

Мета роботи – розробка застосунку для пошуку силуетів потенційних людей у незвичайних ситуаціях.

Методи дослідження: стандартні методи аналізу C++ з використанням штучного інтелекту.

Результати дослідження: на основі вивчення наявних засобів та методі пошуку потерпілих під час незвичайних обставин, за допомогою БПЛА, а також, використовуючи технічні можливості C++ та бібліотеки OpenCV. Розроблено алгоритм детектування силуетів людей. Запропоновані методи використання створеної програми.

Результати роботи можуть бути застосовані при розробці спеціалізованих систем пошуку, а також БПЛА для пошуку цілей.

Abstract

Piven.D.O. Automated victim search system based on artificial intelligence. Master's qualification work. V.N. Karazin Kharkiv National University, 2022, 52 p., 12 fig., 8 sources.

AUTOMATED VICTIM SEARCH SYSTEM BASED ON ARTIFICIAL INTELLIGENCE.

The object of the research is to search for victims using artificial intelligence. The subject of the study is the analysis of systems for searching for victims using UAVs. The purpose of the work is to develop an application for finding silhouettes of potential people in unusual situations. Research methods: standard C++ analysis methods using artificial intelligence. Research results: based on the study of available means and methods of searching for victims in unusual circumstances, using UAVs, as well as using the technical capabilities of C++ and the OpenCV library. An algorithm for detecting silhouettes of people has been developed. Proposed methods of using the created program. The results of the work can be applied in the development of specialized search systems, as well as UAVs for finding targets.

ЗМІСТ

ВСТУП

В останні роки професійні служби швидкого реагування почали використовувати нові технології на місці катастроф, щоб врятувати більше життів. Все частіше, вони використовують роботів для пошуку місць стихійних лих і використовують мобільні телефони як для локалізації, так і для спілкування з пораненими.

Однією з найбільш поширених і успішно використовуваних платформ роботів у світі є безпілотні літальні апарати (БПЛА) які дозволяють дистанційний огляд і картографування. Вони здатні забезпечити високу роздільну здатність і часто потрібна мінімальна інфраструктура для польоту.

Безпілотники зараз використовуються з різних причин і вважаються стандартними в усіх агентствах пошуку по всьому світу. Однак операції на базі безпілотників, які передбачають ручний моніторинг операторів, створюють ряд фундаментальних перешкод. Керуючи літальним апаратом, оператор повинен переглядати зображення в реальному часі на крихітному екрані. Оскільки людина, яку шукають, така маленька, порівняно з рештою середовища, вона займає лише кілька пікселів на екрані. Навіть для тих, кого навчили, підтримувати тривалу концентрацію та увагу важко. Людей, яких розшукують, часто ховають рослини, або інші перешкоди, що ускладнює пошук навіть за гарної погоди. Ці проблеми можна пом’якшити за допомогою автоматизованої системи.

Пожежі, повені, землетруси та цунамі є прикладами стихійних лих, а біологічні атаки, ядерні вибухи та терористичні атаки є прикладами техногенних катастроф. Стихійні лиха викликають надзвичайні ситуації, а також фізичний і суспільний хаос. За даними Міжнародної федерації товариств Червоного Хреста і Червоного Півмісяця, кількість людей, які загинули внаслідок стихійних лих, значно коливалася протягом останнього десятиліття: від 14 389 у 2015 році до 314 503 у 2010 році. Від природних катаклізмів щороку страждає понад 100 мільйонів людей. Уряд, підприємства та гуманітарні організації намагаються

зменшити кількість цих сумних смертей. У такому надзвичайному стані для виживання постраждалим найважливіше їжа, вода та ліки. На додачу, повинно бути джерело або надійна мережа для зв'язку з жертвами. Для будь-якої рятувальної операції мережева система аварійного відновлення є найважливішою для постраждалих і членів рятувальної команди під час порятунку жертв, щоб спілкуватися з ними та дізнаватися про стан жертв. Пошук і виявлення тих, хто вижив, а також їх порятунок іноді є частиною операції з надання допомоги при катастрофі. Наразі за такого підходу члени рятувальної команди здійснюють ручний пошук у цих зонах, що може завдати шкоди їх життю також. Це також займає дуже багато часу. Більшість лих вимагає широкомасштабної рятувальної операції. У більшості таких спроб порятунку беруть участь людські команди, яких легко розбити та які відчайдушно потребують допомоги. Члени рятувальних команд іноді змушені працювати у ворожих і небезпечних умовах.

З часом роботи та системи штучного інтелекту виявилися надзвичайно надійними в багатьох сферах. Ці єдині в своєму роді машини можна використовувати для пошуку жертв, а також для збору даних, які можуть допомогти покращити або оптимізувати пошуково-рятувальну діяльність. У результаті останніх досягнень у галузі робототехніки та операцій зросла кількість досліджень, спрямованих на вивчення можливості використання роботів-рятувальників для пошуково-рятувальних місій.

Щоб прискорити процес збору інформації кілька БПЛА можуть бути в повітрі одночасно. Це збільшує площу, яку можна спостерігати в заданій кількості, але вимагає координації, щоб забезпечити ефективне використання ресурсів. Однак наразі таке розгортання потребує трудомістких, індивідуальних БПЛА союзника з дистанційним керуванням. Зважаючи на це, існує прагнення до використання кількох роботів, що працюють з певним рівнем автономності, щоб зменшити навантаження операторів. Один підхід для використання кількох роботів таким чином напівавтономна робота під наглядом невеликої кількості

професіоналів (потрібні лише люди-експерти для прийняття важливих рішень). Платформи БПЛА також дозволяють розгортати різноманітні групи роботів, спонукаючи їх поєднувати свої індивідуальні здібності, щоб бути більш ефективними. Наприклад, БПЛА з фіксованим крилом здатні літати швидше і перевозити більше корисного навантаження, але коли вони це роблять, їх слід розгортати з більшою кількістю заходів безпеки (для важких пристроїв потрібні пілоти безпеки). З іншого боку, невеликі гвинтокрилі БПЛА більш маневрені та можуть наближатися до цілей і надавати зображення об’єктів на місцевості.

Метою дипломної роботи, є аналіз існуючих методів пошуку жертв, а також створення алгоритму пошуку на основі C++.

1. ОГЛЯД МЕТОДІВ АНАЛІЗУ ПОТОКОВОГО ВІДЕО

Контейнер і кодек є двома компонентами будь-якого відеофайлу. Формат відео – це контейнер, який зберігає аудіо, відео, субтитри та будь-які інші метадані. Кодек кодує та декодує мультимедійні дані, такі як аудіо та відео.

Під час створення відео відеокодек кодує та стискає відео, тоді як аудіокодек робить те саме зі звуком. Після цього закодовані відео та аудіо синхронізуються та зберігаються в мультимедійному контейнері - у форматі файлу.

1.1 Аналіз популярних протоколів і форматів відео

Протокол потокового відео – це стандартизований метод доставки для розбиття відео на частини, надсилання його глядачеві та повторного збирання. Протоколи потокового відео – це правила та методи, які використовуються для розбиття відеофайлів на дрібні частини, щоб їх можна було доставити глядачам.

Перш ніж йти далі, давайте розглянемо «чому» за протоколами потокового відео. Більшість цифрового відео призначено для двох речей: зберігання та відтворення. Це призводить до двох основних міркувань, а саме невеликого розміру файлу та універсального відтворення.

Більшість відеофайлів не призначені для потокового передавання, а це означає, що для потокового передавання відео потрібно спочатку перетворити його на потоковий файл. Це передбачає розбиття відео на невеликі частини. Потім ці фрагменти надходять послідовно та відтворюються в міру отримання. Якщо ви транслюєте відео в прямому ефірі, вихідне відео надходить безпосередньо з камери. В іншому випадку він надходить із файлу для вмісту відео на вимогу VOD.

Це базове пояснення того, як працюють потокові протоколи. Протоколи потокової передачі можуть стати набагато складнішими. Наприклад, багато протоколів є «адаптивним бітрейтом». Ця технологія забезпечить найкращу якість, яку глядач може підтримувати в будь-який момент часу.

Отже, якщо у глядача низька швидкість Інтернету, йому буде доставлено відео нижчої якості, а якщо у глядача швидкість Інтернету вища, йому буде доставлено відео вищої якості.

Деякі протоколи зосереджені на зменшенні затримки або затримки між подією, що відбувається в реальному житті, і її відтворенням на екрані глядача. Деякі відео-протоколи працюють лише в певних системах, а інші відео-протоколи зосереджені на управлінні цифровими правами (DRM).

1.1.1 Протоколи передачі відео

HLS (HTTP LIVE STREAMING) - це адаптивний протокол зв'язку потокового медіа на основі HTTP. Ця специфікація широко підтримується потоковими серверами таких постачальників, як Adobe, Microsoft, Real Networks тощо. Популярність пристроїв iOS і ця підтримка технології, пов'язаної з розповсюдженням, також призвели до збільшення підтримки з боку від Google в Android 3.0.

На дуже високому рівні HLS працює, як і всі адаптивні потокові технології, ви створюєте кілька файлів для розповсюдження в програвач, який може адаптивно змінювати потоки для оптимізації відтворення. Оскільки технологія базується на HTTP, величезна інфраструктура потокового сервера не потрібна, тому вся логіка перемикання знаходиться на плеєрі. Як працює сервер HLS:

Вхідний потік кодується/перекодується. Джерело вхідного відео та аудіо кодується (або транскодується) у контейнер транспортного потоку MPEG-2.

Створюються одиничні або варіантні профілі виводу. Зазвичай один вхідний потік буде перекодований до кількох вихідних роздільних

здатностей/швидкостей передачі даних, залежно від типів клієнтських пристроїв, для яких призначено потік. Усі ці профілі зазвичай називаються «Варіантний список відтворення».

Вхідні потоки сегментовані. Усі потоки, що містяться в профілях, потрібно сегментувати та зробити доступними для доставки на клієнтський пристрій через HTTP. Програмний або апаратний пристрій, який виконує сегментацію (сегментатор), також створює індексний файл (також відомий як «Головний список відтворення»), який використовується для відстеження окремих відео/аудіо сегментів.

Клієнт завантажує «Головний список відтворення» через URL-адресу, яка ідентифікує потік. Зазвичай він містить URL-адреси варіантів списку відтворення (з різними бітрейтами та роздільною здатністю), щоб клієнт міг вибрати бажаний для завантаження. Усі ці файли містять URL-адресу для клієнта, де можна отримати фактичні фрагменти медіа-потоку. Для певного потоку клієнт потім отримує кожен фрагмент потоку в порядку. Щойно клієнт має мінімальну кількість завантажених і буферизованих фрагментів, він відображає їх користувачеві як відтворення медіа.

«Щоб забезпечити підтримку різних категорій пристроїв і різної якості відео, нам потрібно виконати кілька наборів перекодування (іншими словами, повторне кодування) для одного каналу відео. Оскільки все це окреме перекодування потребує як потужності процесора, так і часу, тож, зрештою, клієнтські пристрої отримуватимуть живу стрічку з величезною затримкою (залежно від кількості різних виконаних перекодувань). Оскільки основна ідея Live Feed полягає в тому, щоб мати канали в режимі реального часу, тож він стає величезним вузьким місцем, підтримуючи адаптивний потоковий підхід.»[1]

Real Time Messaging Prootocol (RTMP) – с початку розроблений Macromedia на початку потокового передавання, протокол RTMP досі широко використовується.

Сьогодні RTMP здебільшого використовується для прийому прямих трансляцій за допомогою кодера з підтримкою RTMP. Простими словами, коли ви налаштовуєте свій кодувальник, щоб надіслати ваш канал відео на вашу потокову платформу, це відео досягне платформи через протокол RTMP. Згодом цей вміст досягає кінцевого глядача в іншому протоколі, зазвичай HLS. RTMP використовується разом з іншими протоколами потокового відео. RTMP рідко використовується як протокол потокового відео для глядача, як це було раніше. Це тому, що він залежить від плагіна Flash, який зараз повністю застарів. Якщо він використовується, він зазвичай поєднується з іншим протоколом, наприклад HLS. «Внутрішньокадрове стиснення здійснюється шляхом використання надлишкового простору в кадрі. Ця надмірність викликана подібністю між одним пікселем і пікселями навколо нього. Внутрішньокадрове стиснення складається з процесу перетворення та квантування, у процесі перетворення є дискретне косинусне перетворення (DCT), яке використовується для виконання процесу перетворення з часової області в просторову область. Процес квантування використовується для скорочення результату перетворення; наступним процесом є кодування з використанням кодування довжини серії (RLE) та ентропійного кодування.»[2]

Плюси використання RTMP:

Низька затримка: дозволяє вашому прямому відеопотоку підтримувати стабільне з'єднання та відео для глядача, навіть якщо з'єднання з Інтернетом ненадійне. Це надає глядачам менше «затримок» під час перегляду ваших відео з нестабільним інтернет-з'єднанням, дозволяючи їм швидко відновити трансляцію, коли їхнє інтернет-з'єднання стабілізується.

Адаптивність: адаптивний канал означає, що ваші глядачі не прив'язані до перегляду ваших каналів в одному лінійному напрямку. З вмістом, розміщеним на сервері RTMP, канал дозволяє пропускати та перемотувати частини каналу або приєднуватися до прямого ефіру після його початку. Глядачі часто очікують такого типу контролю над вмістом, який вони переглядають.

Гнучкість: RTMP дозволяє інтегрувати різноманітні відеоформати в один єдиний пакет, плавно поєднуючи аудіо, відео та текст. Крім того, ви можете мати кілька варіантів медіаканалів, наприклад потокове передавання аудіопотоків MP3 і AAC або потокове передавання відео MP4, FLV і F4V.

Мінуси використання RTMP:

Не підтримується HTML5: RTMP підтримується програвачами Flash, форматом, який на шляху до старіння. Програвачі HTML5 швидко стають сучасним стандартом, але RTMP не може відтворюватися на програвачах HTML5 без конвертера, такого як відеопротокол HLS.

Проблеми з пропускною здатністю: потоки RTMP можуть бути особливо вразливими до проблем із низькою пропускною здатністю. Це може призвести до частих неприємних перерв у ваших трансляціях, які зіпсують враження для ваших глядачів.

HTTP несумісний: ви не можете безпосередньо передавати канал RTMP через HTTP-з'єднання. Щоб використовувати потік RTMP на своєму веб-сайті, вам потрібно підключитися до спеціального сервера, наприклад Flash Media Server, і використовувати сторонню мережу доставки вмісту (CDN).

Web Real - Time Communications - це відеопроект із відкритим кодом, який підтримує потокове передавання із затримкою в реальному часі. Цей проект було розроблено для підтримки протоколу голосового зв'язку через Інтернет (VoIP), і його було придбано Google для підтримки інструментів Google для відеочату.

Технічно WebRTC - це потоковий проект, а не потоковий протокол. Однак його часто об'єднують із протоколами потокового відео, яким надається перевага, оскільки існує багато збігів.

WebRTC є цінним у налаштуваннях потокового передавання, які потребують затримки в реальному часі. Однорангова потокова передача, яку зазвичай називають «веб-конференцією» або «відеоконференцією», є одним із найпопулярніших випадків використання WebRTC.

Деякі популярні програми та програми, які використовують WebRTC, включають Snapchat, Facebook, WhatsApp та інші соціальні медіа-платформи, які підтримують відеочат.

Плюси використання WebRTC:

- З відкритим вихідним кодом: оскільки WebRTC є відкритим вихідним кодом, його можна налаштувати відповідно до ваших конкретних потреб потокового передавання

- Затримка в реальному часі: WebRTC підтримує потокове передавання із затримкою в реальному часі, що означає, що ваше відео переміщується на екрани ваших глядачів практично в реальному часі.

Мінуси використання WebRTC:

- Нова технологія: WebRTC є нещодавньою розробкою, тому решта ринку ще не адаптувалася. Ви можете виявити деякі проблеми з сумісністю налаштувань потокового передавання. «Щоб налагодити зв'язок між різними користувачами та/або пристроями, WebRTC вимагає свого роду механізм сигналізації або підтримку протоколів. Але Інженерна робоча група Інтернету (IETF) і Консорціум Всесвітньої павутини (W3C) ще не погодили остаточний механізм сигналізації або остаточний протокол API для тестування WebRTC і регулювання архітектури зв'язку.» [3]

1.1.2 Формати відео

Будь - який відеокліп, який ви переглядаєте на смартфоні, комп'ютері, телевізорі чи планшеті, має певний формат файлу. Якщо ви хочете, щоб ваші відео добре показувалися на будь-якій платформі, ви повинні розуміти, як працює кожен відеоформат. Наприклад, формат відео для веб-розробки відрізнятиметься від формату, який ви використовуєте для своїх соціальних мереж.

«Низка міжнародних стандартів була встановлена на основі різних програм, технологій та епохи. Незважаючи на те, що ці стандарти можуть працювати для спектру програм, кожен оптимізований для певного класу програм. Іноді невиправдано й невигідно створювати будь-який окремий медіа-сервер або термінал для підтримки всіх типів кодування та декодування. Методи транскодування, які перетворюють один формат в інший, переважно в стиснутому домені, вирішують проблему міжстандартної працездатності.» [4]

MPEG - 4 Part 14 або MP4 є одним із найперших форматів цифрових відеофайлів, представлених у 2001 році. Більшість цифрових платформ і пристроїв підтримують MP4. Формат MP4 може зберігати аудіофайли, відеофайли, нерухомі зображення та текст. Крім того, MP4 забезпечує високу якість відео, зберігаючи відносно невеликі розміри файлів.

MOV - це популярний формат відеофайлів, розроблений Apple. Його розроблено для підтримки програвача QuickTime. Файли MOV містять відео, аудіо, субтитри, часові коди та інші типи медіа. Він сумісний із різними версіями QuickTimePlayer як для Mac, так і для Windows. Оскільки це відеоформат дуже високої якості, файли MOV займають значно більше місця в пам’яті комп’ютера.

Відеоформат WMV розроблено компанією Microsoft і широко використовується в медіапрогравачах Windows. Формат WMV забезпечує невеликий розмір файлу з кращим стисненням, ніж MP4. Ось чому він популярний для потокового онлайн-відео. Хоча він несумісний з пристроями Apple, користувачі можуть завантажити Windows Media Player для свого iPhone або Mac.

FLV - це формат файлів, який використовується Adobe Flash Player. Це один із найпопулярніших і універсальних відеоформатів, який підтримується всіма відеоплатформами та браузерами. Формат FLV є хорошим вибором для онлайн-платформ потокового відео, таких як YouTube. Вони мають відносно невеликий розмір файлу, що полегшує їх завантаження. Єдиним недоліком є те, що він несумісний з багатьма мобільними пристроями, такими як iPhone.

Формат файлу AVI був представлений у 1992 році компанією Microsoft і досі широко використовується. Відеоформат AVI використовує менше стиснення, ніж інші відеоформати, такі як MPEG або MOV. Це призводить до дуже великих розмірів файлів, приблизно 2-3 ГБ на хвилину відео. Це може бути проблемою для користувачів з обмеженим простором для зберігання. Ви також можете створювати відеофайли AVI без стиснення. Це робить файли без втрат. Файл без втрат зберігає свою якість з часом, незалежно від того, скільки разів ви відкриваєте чи зберігаєте файл. Крім того, це виключило використання кодеків у відеоплеєрах.

Advanced Video Coding High Definition (AVCHD) - це формат, який використовується для відтворення HD-відео та цифрового запису. Цей відеоформат був розроблений Panasonic і Sony для професійного запису відео високої чіткості. AVCHD також дозволяє зберігати години високоякісного відео, використовуючи лише невелику кількість даних, використовуючи H.264/MPEG-4.

1.2 Засоби розпізнавання фрагментів у відео-потоці

За останнє десятиліття була проведена значна робота в галузі машинного навчання, особливо в області комп’ютерного зору. Від передових алгоритмів класифікації, таких як Inception від Google, до піонерської роботи Яна Гудфеллоу над Generative Adversarial Networks для генерування даних із шумів, чимало галузей були досліджені багатьма відданими дослідниками з усього світу. Цікаво, що в першій половині десятиліття найбільш піонерські роботи в галузі комп’ютерного зору стосувалися обробки зображень, таких як класифікація, виявлення, сегментація та генерація, тоді як область обробки відео була менш глибоко вивчена.

Однією з очевидних причин невеликого дисбалансу є те, що відео по суті є послідовністю зображень (кадрів) разом. Однак це визначення не може охопити повне уявлення про те, що таке обробка відео, і це тому, що обробка відео додає новий вимір до проблеми: часовий вимір. Відео – це не лише послідовність зображень, це скоріше послідовність пов'язаних зображень. Хоча це здається мінімальною різницею, дослідники можуть використовувати цей вимір багатьма способами, які не застосовуються до окремих зображень. Крім того, через складність відеоданих (розмір, пов'язані анотації) і дороге обчислення навчання та логічного висновку було важче пробитися в цій галузі. Однак нещодавно, з випуском ImageNet VID та інших масивних наборів відеоданих у другій половині десятиліття, з'являється все більше дослідницьких статей, пов'язаних із відео. У цьому посібнику ми здебільшого досліджуватимемо дослідження відеодетектування, точніше, як дослідники можуть досліджувати часовий вимір.

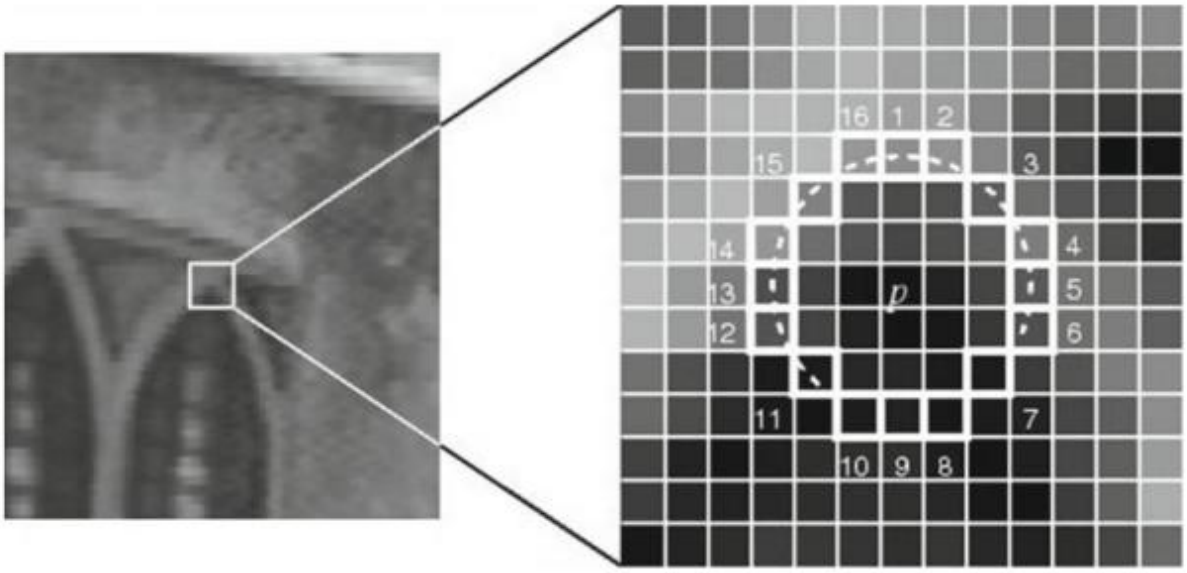

Рисунок 1.1 - Метод 3D згортки

Першими методами (Методи пост-обробки), які з'явилися, були модифікації етапу пост-обробки конвеєра виявлення об'єктів. Це тому, що він вимагає менше інфраструктури та не вимагає змін в архітектурі моделі. Методи пост-обробки все ще являтимуть собою процес виявлення кожного кадру, і тому

вони не матимуть підвищення продуктивності (обробка може зайняти трохи більше часу). Однак це може значно підвищити точність. Відомим методом є Seq-NMS (Sequence Non-Maximal Suppression), який застосовує модифікацію достовірності виявлення на основі інших виявлень на «доріжці» за допомогою динамічного програмування. Наприклад, слабші передбачення позитивного об’єкта можуть бути викликані оклюзією, розмиттям руху або іншими дефектами, але оскільки це буде присутнє в «доріжці» (критерій накладення), отриманій із попередніх кадрів, впевненість підвищиться. Це ефективно мінімізує кількість неправильних виявлень між кадрами або виявлення випадкових стрибків і стабілізує вихідний результат.

Методи 3D згортки. Наприклад, першим природним бажанням розробника, який має досвід класифікації зображень, було б подумати про якусь 3D-згортку на основі 2D-згортки, яка виконується на зображеннях. Така архітектура є ймовірною: ітерація через n кадрів як вхідних даних для моделі та виведення послідовних виявлень на послідовних кадрах. Це, безумовно, потенційний напрямок для виявлення, оскільки він може витягувати низькорівневі функції для просторово-часових даних, але згортка нейронної мережі з 3D-згортками в основному доведена як корисна та плідна, коли мова йде про обробку 3D-зображень. Тому ці моделі є скоріше проривом у сфері медичної візуалізації та менш актуальними для відео-детектування.

«Останнім часом методи, засновані на глибокому навчанні, досягли великого успіху в області комп’ютерного зору, особливо в тих проблемах розпізнавання високого рівня. У той час як розробити ефективну систему виявлення дій, яка використовує нейронну мережу для даних на основі скелета, недостатньо вивчено.»[5]

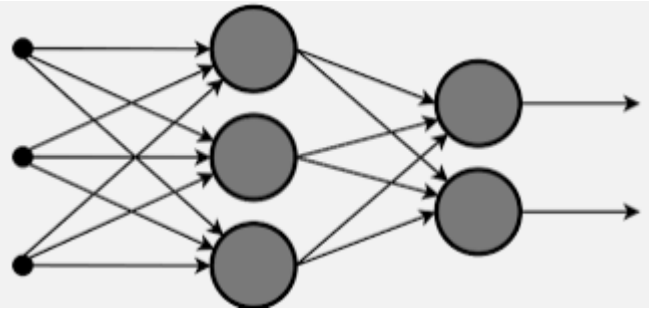

Рисунок 1.2. - Метод 3D згортки

Інші, хто має більше досвіду роботи з послідовними даними, можуть подумати про використання рекурентної нейронної мережі, такої як LSTM. RNN - це спеціальні типи мереж, створені для обробки послідовних, у тому числі тимчасових даних. Область, яка отримала значну користь від цієї архітектури, це обробка природної мови. Наприклад, AWD-LSTM працює на рівні з найсучаснішою моделлю трансформатора BERT, але має набагато менші параметри.

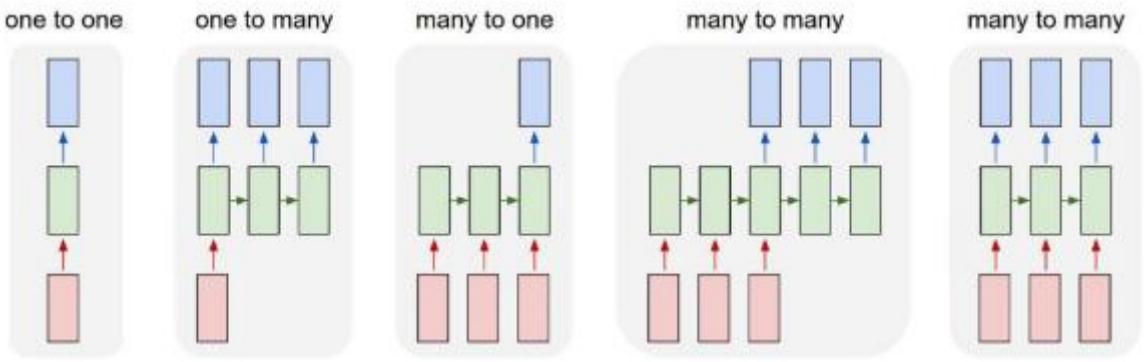

Рисунок 1.3 - Рекурентні нейронні мережі

«Багато існуючих методів набору даних Imagenet VID організовують виявлення окремих кадрів у доріжки або трубочки та оновлюють виявлення за допомогою пост-обробки. Seq-NMS пов’язує прогнози з високою достовірністю в послідовності по всьому відео. TCNN використовує оптичний потік для відображення виявлень у сусідніх кадрах і пригнічує прогнози з низьким рівнем

достовірності, а також включає алгоритми відстеження. Однак жоден із цих методів не робить висновок онлайн і не зосереджується на ефективності. Замість того, щоб оперувати кінцевими результатами виявлення, наш підхід безпосередньо включає часовий контекст на рівні функцій і не вимагає жодних етапів посто-бробки чи онлайн-навчання. Оскільки наш метод можна розширити, застосовуючи будь-який із цих підходів до наших результатів виявлення, наша мережа більш схожа на базовий однокадровий детектор, який використовується цими методами, ніж самі методи.

Недавня робота, D&T, поєднує відстеження та виявлення шляхом додавання операції відстеження RoI та багатозадачних втрат на парах кадрів. Однак їхні експерименти зосереджені на дорогих високоточних моделях, тоді як наш підхід розроблено для мобільних середовищ. Навіть з агресивними часовими кроками їхню модель неможливо запустити на мобільних пристроях через вартість базової мережі Resnet-101, яка вимагає в 79 разів більше обчислень, ніж наша більша модель, і в 450 разів більше, ніж наша менша.»[6]

Оцінка оптичного потоку (Optical Flow) – це метод оцінки видимого руху об’єктів між двома кадрами відео, викликаного камерою (фоном) або рухом об’єкта. Зазвичай результатом є двовимірне векторне поле, де кожен вектор представляє вектор зміщення пікселя від першого кадру до другого. Optical Flow - це галузь дослідження комп’ютерного зору, яка вивчалася з 1980-х років і нещодавно відродилася як цікава сфера глибокого навчання, започаткована Flownet. До цього оригінальні методи були диференціальними. Наприклад, метод Лукаса-Каннаде передбачає, що потік є фактично постійним у локальному околиці пікселя, який розглядається, і розв’язує базові рівняння оптичного потоку для всіх пікселів у цьому околі за допомогою критерій найменших квадратів. Але завдяки новим досягненням і новим наборам даних оптичного потоку, таким як Sintel, з’являється все більше і більше архітектур, одна швидша й точніша за іншу.

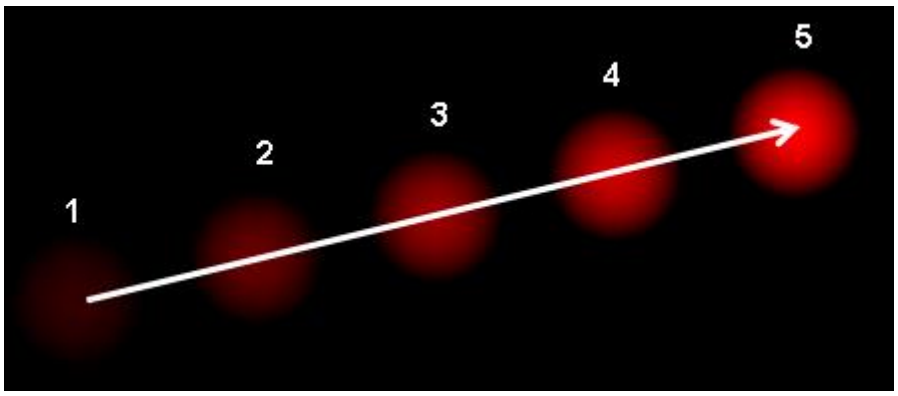

Рисунок 1.4 - Оцінка оптичного потоку

Архітектура функціонує з концепцією розрідженого ключового кадру. Оскільки мережа оптичного потоку може бути відносно невеликою, час обробки та обчислювальна потужність, необхідні для таких мереж, менші, ніж для детекторів об’єктів. Таким чином, конвеєр функціонує як цикл з n кадрів. Перший кадр називається ключовим. Це кадр, який виявляє детектор об’єктів. Оскільки тепер детектори забезпечують точне виявлення всіх об’єктів, виявлення буде залежати від алгоритмів оптичного потоку. Після отримання векторів зміщення відомо про виявлення наступних n-1 кадрів, і цикл повторюється.

Багато-кадрове агрегування функцій для точності. Методом підвищення точності виявлення відео є багато-кадрове агрегування функцій. Існують різні способи реалізації цього, але всі вони обертаються навколо однієї ідеї: щільно обчислене виявлення кожного кадру з одночасним викривленням характеристик із сусідніх кадрів на поточний кадр та агрегуванням із зваженим усередненням. Таким чином, поточний кадр матиме перевагу від поточних кадрів, а також деяких подальших кадрів для кращого виявлення. Тоді це може вирішити проблеми з рухом і кадруванням об’єктів із відеокадру.

1.3 Аналіз бібліотек ропізнавання об’єктів у відео-потоці

Бібліотека Simd - це безкоштовна бібліотека для обробки зображень і машинного навчання з відкритим вихідним кодом, розроблена для програмістів на C і C++. Він надає багато корисних високопродуктивних алгоритмів для обробки зображень, таких як: перетворення формату пікселів, масштабування та фільтрація зображення, вилучення статистичної інформації із зображень, виявлення руху, виявлення об’єктів (каскади класифікаторів HAAR і LBP) і класифікація, нейронна мережа.

Алгоритми оптимізовані з використанням різних розширень ЦП SIMD. Зокрема, бібліотека підтримує такі розширення ЦП: SSE, AVX, AVX-512 і AMX для x86/x64, VMX(Altivec) і VSX(Power7) для PowerPC, NEON для ARM.

Бібліотека Simd має C API, а також містить корисні C++ класи та функції для полегшення доступу до C API. Бібліотека підтримує динамічне та статичне зв’язування, 32-розрядні та 64-розрядні Windows і Linux, компілятори MSVS, G++ і Clang, системи збірки MSVS project і CMake.

«У поточних мовах C і C++ немає SIMD і пунктів, а також анотацій векторних функцій для підтвердження векторизації для звичайних функцій C/C++ і складних циклів, векторизація в основному покладається на аналіз залежностей даних, а також підказки компілятора від виробника (наприклад, #pragma ivdep підтримується компіляторами IBM*, Cray*, Intel:registered:). Це призводить до обмеженого використання апаратних блоків SIMD для багатьох реальних програм.

Ми визначаємо набір конструкцій SIMD, пунктів і анотацій, щоб увімкнути векторизацію SIMD для функцій і циклів C/C++, які компілятор не може автоматично векторизувати з підказками компілятора або без них. Наші векторні розширення SIMD для вираження паралелізму векторів мають схожий вигляд і відчуття, що й OpenMP* для вираження паралелізму потоків."»[7]

Boost Generic Image Library (GIL) - це бібліотека C++11, яка абстрагує представлення зображень від алгоритмів і дозволяє писати код, який може працювати з різноманітними зображеннями з продуктивністю, подібною до рукописного написання для певного типу зображення.

Magick++ - це об’єктно-орієнтований C++ API для бібліотеки обробки зображень GraphicsMagick, найповнішого доступного пакета обробки зображень із відкритим кодом.

Magick++ забезпечує інтегровану підтримку стандартної бібліотеки шаблонів (STL), яка є частиною стандартної мови C++, тому доступні потужні контейнери (наприклад, deque, vector, list і map) можна використовувати для написання програм, подібних до тих, що можливі за допомогою PERL & PerlMagick. Сумісні з STL версії шаблонів операцій у стилі списків GraphicsMagick надаються, щоб операції можна було виконувати над кількома зображеннями, що зберігаються в контейнерах STL.

Бібліотека OpenCV є однією з найбільш широко використовуваних бібліотек в обробці зображень, яка розроблена і випущена як програмне забезпечення з відкритим кодом корпорацією Intel. OpenCV складається з базових операцій із зображеннями, таких як логічні та арифметичні операції, а також включає складні операції, такі як виявлення та відстеження об’єктів. Впровадження такої корисної бібліотеки OpenCV на вбудованому процесорі дає змогу досягти розпізнавання зображень у режимі реального часу, необхідного для кількох вбудованих програм. Тому ми виконуємо паралельну реалізацію бібліотеки OpenCV на Cell Broadband Engine (Cell), який є одним із найпоширеніших високопродуктивних вбудованих процесорів, який, як очікується, буде використаний у багатьох програмах. У цьому документі описано деталі CVCell, оптимізованої реалізації бібліотеки OpenCV для процесора Cell Broadband Engine.

1.4 Обґрунтування вибору бібліотеки OpenCV

OpenCV (Open Source Computer Vision Library) - бібліотека комп'ютерного зору та машинного навчання з відкритим кодом. OpenCV було створено, щоб забезпечити загальну інфраструктуру для програм комп'ютерного зору та прискорити використання машинного сприйняття в комерційних продуктах. Будучи ліцензованим продуктом Apache 2, OpenCV дозволяє легко використовувати та змінювати код.

Бібліотека містить понад 2500 оптимізованих алгоритмів, що включає повний набір як класичних, так і найсучасніших алгоритмів комп'ютерного зору та машинного навчання. Ці алгоритми можна використовувати для виявлення та розпізнавання облич, ідентифікації об'єктів, класифікації дій людей у відео, відстеження рухів камери, відстеження рухомих об'єктів, вилучення 3D-моделей об'єктів, створення 3D-хмар точок із стереокамер, з'єднання зображень для отримання високої роздільної здатності. зображення цілої сцени, знаходити подібні зображення в базі даних зображень, видаляти червоні очі із зображень, зроблених за допомогою спалаху, стежити за рухами очей, розпізнавати пейзаж і встановлювати маркери для накладання на нього доповненої реальності тощо. OpenCV має понад 47 тисяч користувачів, спільнота та оціночна кількість завантажень перевищує 18 мільйонів. Бібліотека широко використовується компаніями, дослідницькими групами та державними органами.

Поряд із такими відомими компаніями, як Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota, які використовують бібліотеку, є багато стартапів, таких як Applied Minds, VideoSurf і Zeitera, які широко використовують OpenCV. Застосування OpenCV охоплює широкий діапазон: від з'єднання зображень вуличного перегляду, виявлення вторгнень у відеоспостереження в Ізраїлі, моніторингу шахтного обладнання в Китаї, допомоги роботам у навігації та підбиранні об'єктів у Willow Garage, виявлення нещасних випадків утоплення в басейні в Європі, використання інтерактивного мистецтва в Іспанія та Нью-Йорк,

перевірка злітно-посадкових смуг на наявність сміття в Туреччині, перевірка етикеток на продуктах на фабриках по всьому світу та швидке виявлення облич у Японії.

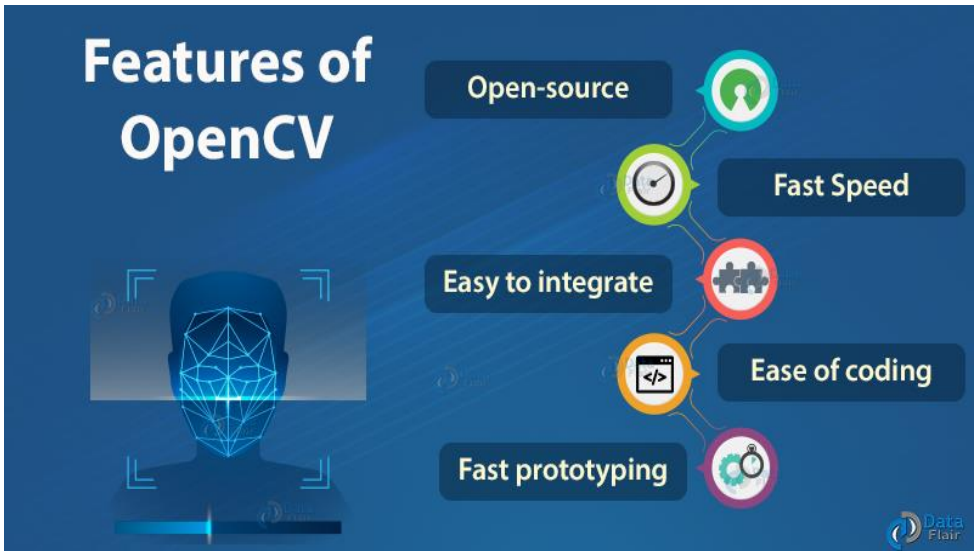

Рисунок 1.5 - Переваги OpenCV

Він має інтерфейси C++, Python, Java і MATLAB і підтримує Windows, Linux, Android і Mac OS. OpenCV здебільшого орієнтується на програми бачення в реальному часі та використовує переваги інструкцій MMX і SSE, якщо вони доступні. Зараз активно розробляються повнофункціональні інтерфейси CUDA і OpenCL. Існує понад 500 алгоритмів і приблизно в 10 разів більше функцій, які створюють або підтримують ці алгоритми. OpenCV написаний на C++ і має шаблонний інтерфейс, який бездоганно працює з контейнерами STL.

Тому, ця бібліотека була обрана через її популярність, функціональність, інтерфейс, а також алгоритми.

2 АНАЛІЗ ЗАСТОСУВАННЯ ДРОНІВ ДЛЯ НАДАННЯ ЕКСТРЕННОЇ ДОПОМОГИ ПОТЕРПІЛИМ

Використання безпілотних літальних комплексів (БЛА) стає все більш частим під час пошуково-рятувальних (SAR) операцій з пошуку зниклих безвісти. Ці системи виявилися особливо корисними для операцій, які виконуються в дикій місцевості, тобто у відкритих і гірських районах. Успішне впровадження цих систем можливе завдяки потенціалу безпілотних літальних апаратів (БПЛА), які допомагають значно прискоротити робочий час і, отже, дозволяють набагато швидше знаходити втрачених людей. Це вкрай важливо для підвищення їхніх шансів на виживання в екстремальних умовах (відмова від гідратації, їжі та ліків, а також гіпотермії).

Використовуючи глибоке навчання, угорські дослідники Лю та Сіраньї (2021) продемонстрували, що дрони мають здатність точно та надійно розпізнавати ряд людських жестів на відстані, що може допомогти в майбутніх пошуково-рятувальних операціях.

«Безпілотні літальні апарати (БПЛА), широко відомі як дрони, стають все більш поширеними та постійно набувають нових функцій. Нині вони є одним із найбільш інноваційних елементів у діяльності рятувальних служб, у тому числі Державної пожежної охорони (ДФС).» [8]

Безпілотники також використовуються як критично важливий інструмент для реагування на катастрофи та боротьби з ними. У таких випадках безпілотники можуть використовуватися для спостереження за надзвичайними ситуаціями, телекомунікаційних послуг, пошуково-рятувальних операцій і доставки екстрених матеріалів і допомоги в райони, куди медичний персонал не може безпечно дістатися. Наприклад, безпілотники змогли надати таку екстрену медичну допомогу та допомогу людям на Гаїті та Тайвані після землетрусів 2010 та 2016 років, на Філіппінах після тайфуну Хайян у 2013 році та в Непалі, який стикається з частими повенями, зсувами та лавинами. У США плани доставки

вантажів на острів Окракоук, віддалене острівне село в Аутер-Бенкс штату Північна Кароліна, перебувають у процесі перевірки щодо сценаріїв реагування на урагани. Безпілотники, оснащені відео, телекомунікаційними можливостями та тепловізором, можуть бути особливо корисними в таких подіях. Також було показано, що дрони можуть бути особливо цінним інструментом для боротьби зі стихійними лихами у віддалених районах, таких як канадська Арктика.

У нещодавньому огляді досліджень, які досліджують доцільність використання безпілотних літальних апаратів для надання телемедичної допомоги, дослідження Бхатта та інших показало, що безпілотні літальні апарати можуть бути можливим і економічно ефективним механізмом для надання екстреної телекомунікаційної та телемедичної допомоги пацієнтам із обмеженим або затримковим доступом через відстань, географія або інфраструктура. Зокрема, дрони, оснащені двостороннім відеозв'язком і біосенсорами (наприклад, для вимірювання температури, частоти серцевих скорочень або артеріального тиску)14, можуть бути особливо застосовні для використання в телемедицині для діагностики, оцінки та навіть лікування у віддалених районах.

2.1 Проведення досліджень операцій БПЛА на прикладі Польщі

Проведені дослідження зосереджені на характеристиках, специфіці та перспективах застосування БПЛА в рятувальних операціях. 8 липня 2021 року на Базі підготовки та аварійно-рятувальних робіт відбулися випробування щодо застосування БПЛА в аварійно-рятувальних роботах, підготовлені ПФО.

Інновації в Головній школі пожежної служби в Новий Двір Мазовецький. Випробовувалися апаратно-програмні компоненти концептуальної системи VFD, а також можливість використання БПЛА та АРМ ГКС.

МЧСБ брала участь у змодельованих операціях. Учасники цих симуляцій координували дії, використовуючи попередній перегляд та інтегровану з БПЛА

систему оповіщення. Випробування підтвердили взаємодоповнюваність обмінюваної інформацією, яку можна використовувати в операціях, а також рівень взаємодії між SFS та VFD.

Учасники були розділені на дві групи з ідентифікаторами Альфа та Дельта. Стартуючи з різних місць на полігоні, вони повинні були досягти певної точки за командами, отриманими по радіо від керівника рятувальної операції (HRO), який керував ними, порівнюючи ортофотокарту місцевості з їх положенням, яке вказували безпілотники. . Крім того, збої зв'язку моделювали в групі Альфа. HRO міг передавати команди та повідомлення лише через динамік, встановлений на дроні DJI Mavic 2 Enterprise Dual. Однак зворотного радіоканалу не було, і точність виконаних команд оцінювалася HRO на основі зображення дронів. Було виконано дві частини цього етапу навчань. У першому учасники дослідження вирушили в дію без попередньої підготовки і не маючи в своєму розпорядженні будь-якого локаційного обладнання (GPS, компаса). Координувати їх було нелегко роботи, а команди, вказівки, орієнтири виявилися неоднозначними. Незважаючи на ці труднощі, обидві команди успішно вийшли на стартову точку. З другої спроби учасники навчань вже знали місцевість і були оснащені локаційними приладами; тому час досягнення цілі значно скоротився. Випробування з використанням БПЛА: AUTEL EVO II і DJI Mavic 2 Enterprise Dual проводилися відповідно до інструкцій, наданих у внутрішніх робочих механізмах VFD. Було проведено моделювання пошуково-рятувальних робіт, під час яких були команди скоординовано із застосуванням БПЛА. Використовувався прототип робочої станції GCS, який дозволяв у режимі реального часу переглядати БПЛА на робочій станції HRO.

На першому етапі навчань було виготовлено точний ортофотоплан місцевості. Для цього використовувався БПЛА AUTEL EVO II з камерою з роздільною здатністю 8К, який виконали політ на висоті 90 м в режимі серійних надірних зображень (камера спрямована перпендикулярно до поверхні землі). Маршрут польоту готували сходовим методом. Цей метод дозволяє рівномірно

охоплювати територію з накладанням кадрів інші – на 50–70%, щоб створити дуже точну ортофотокарту певної ділянки, яка потім може слугувати основою для подальшої оперативної діяльності. Подібний метод був успішно застосований під час пожежі в національному парку Бебжа в Польщі в 2020 році.

У рамках дослідження було виконано три сценарії вправ: 1- відтворення ортофотокарти в польових умовах; 2-координація наземних груп; 3 - спостереження за місцевістю за допомогою тепловізора. У кожному випадку польотом керували три людини (оператор БПЛА, технік-спостерігач і керівник групи).

Техніка вимірювання часу дозволяла записувати експеримент на відеокамеру, а потім зчитувати час окремих завдань. У день експерименту в місті Новий Двір Мазовецький були такі погодні умови: безхмарно, невеликі опади (1 мм), температура вдень 29 ^{o}C, тиск 1019,5 гПа, швидкість вітру 20,25 км/год, напрям південно-східний. Використовувався пошуковий алгоритм із залученням цілевказівних груп HRO, оснащених GPS та компасом, а також проводились перевірки ефективності дій без цього обладнання та у разі збою зв’язку.

8 липня 2021 року на території Навчально-рятувальної інноваційної бази MSFS у місті Новий Двір Мазовецький відбулися випробування для перевірки характеристик, особливостей та перспектив використання дронів у рятувальних операціях. Під час виконання вправ апаратна та програмного забезпечення, що використовується ПВД, а також можливості використання БПЛА та АРМ ГКС.

Сценарії запланованих навчань передбачали картографування місцевості - виготовлення карти в польових умовах, координацію наземних груп, оцінку місцевості (загорання об’єкта) за допомогою тепловізора, точне спостереження за визначеним об’єктом (районом та особою).

Учасників дослідження залучали до різноманітних імітаційних дій. Вони координували дії за допомогою встановлених гучномовців та попереднього перегляду БПЛА. Метою спільних навчань було вдосконалення прийомів і навичок координації дій та взаємодії.

Система пошуку зниклих безвісти SARUAV, присвячена, зокрема, відкритим просторам, основною частиною якої є алгоритм виявлення силуетів людей на знімках, отриманих з БПЛА, використовується в Польщі 4 підрозділами ПВД та 2 групами МВРС.

Рішення SARUAV було розроблено та протестовано у співпраці з MVRS Jurassic Group. Польові випробування системи вказали на високу продуктивність алгоритму, і результати були опубліковані в . Система отримала дуже позитивні відгуки, серед іншого, від MVRS, а також від компанії FlyTech UAV з Кракова, яка виробляє професійні БПЛА BIRDIE, які можна використовувати в системі порятунку SAR. Ефективність системи також було доведено в реальних ситуаціях, коли час мав вирішальне значення.

2.2 Структура ,організація та складові БПЛА операцій

Організаційна структура відділу БПЛА під час виконання операцій випливає не лише з норм авіаційного законодавства (специфічної ролі оператора БПЛА), а й з реальних експлуатаційних потреб.

Оператор БПЛА є особою, відповідальною за виконання кожного польоту та приймає остаточне рішення щодо можливості виконання польоту в даному місці. На нього покладається обов'язок здійснити технічний огляд БПЛА перед зльотом, контролювати його та забезпечити безпеку виконуваної місії.

Відповідальністю техніка, який також виконує функції спостерігача, є підготовка іншого необхідного обладнання, включаючи роботу робочої станції GCS, встановлення зв’язку з іншими підрозділами та спостереження за польотом БПЛА. Під час вправи він/вона залишається з однієї з команд курсантів для оцінки їх виступу. Роль техніка також полягає в тому, щоб забезпечити безпеку зони виконання польотів, щоб переконатися, що зльоти та посадки виконуються безпечним способом для сторонніх осіб.

Рисунок. 2.1 - Підготовка польоту БПЛА

Керівник групи діє в першу чергу як зв’язкова ланка між підрозділом БПЛА та HRO (або іншими службами). Його/її завдання - бути своєрідним інформаційним фільтром, щоб оператори могли повністю зосередитися на своїх завданнях. Він/вона також відповідає за координацію діяльності різних операторів, які, керуючи БПЛА в одному просторі, повинні співвідносити свої дії, щоб забезпечити належний рівень безпеки польоту.

Важливим питанням при проведенні операцій з використанням безпілотних літальних комплексів є час передачі даних. В організації оперативної діяльності прийнятий метод, який передбачає паралельні дії після прибуття в зону проведення операції за таким оперативним алгоритмом:

1. Оператори готують БПЛА до процедури запуску в найкоротший оперативний час, дотримуючись відповідних правил безпеки.

2. Технік готує робочу станцію GCS та активує системи зв’язку, включаючи щоглу LTE/5G Wi-Fi.

3. Керівник льотної групи розробляє детальний план дій з іншими учасниками, залученими до операції, та контролює повноту процедур запуску.

Все це дозволяє запустити БПЛА протягом 3 хвилин після прибуття в район дії. Перше зображення в АРМ ГКС можна отримати вже на 6-й хвилині, а

зображення в мобільних пристроях учасників операції – на 8-й хвилині після початку операції.

Блок VFD оснащений наступними технологіями та програмним забезпеченням :

1. Два БПЛА: DJI Mavic 2 Enterprise Dual і AUTEL EVO II.

2. PIX4Dreact - додаток дозволяє картографувати зону дії (складання фотокарти). На основі польоту БПЛА створюється точна та актуальна ситуаційна карта в польових умовах. Live preview RTMP - зображення з дронів видно на моніторах (ноутбук, робоча станція GCS) і на смартфонах учасників. Також є можливість передати зображення через Інтернет.

3. Доступ до Інтернету - налаштовано мережу Wi-Fi MESH з доступом до Інтернету. Це дозволяє швидко передати інформацію рятувальникам.

4. ЛОКАЛЬНА WWW - робоча станція GCS має власний веб-сервер, який дає змогу записувати та представляти інформацію, яка є актуальною для операції (наприклад, звіт про пошук).

5. SARUAV - підтримує пошук зниклих безвісти шляхом чисельного моделювання переміщень та аналізу зображень з БПЛА.

6. Тепловізор - один із обладнаних нею БПЛА разом із телеметричною системою дозволяє точно визначати температуру в полі зору камери.

7. Радіозв'язок NFRS - мобільний радіозв'язок великої потужності з високопродуктивною антеною, що дозволяє підтримувати зв'язок у складних умовах.

8. Внутрішній радіозв'язок - зв'язок всередині групи операторів БПЛА відбувається на виділеному обладнанні та частотах, щоб не створювати перешкод або перешкод іншим службам.

Особливої уваги заслуговує робоча станція GCS, розроблена членами VFD (дивіться робочу схему на малюнку 3), яка дозволяє:

1. Мультиплікація (тиражування) зображень дрона та їх передача на інші пристрої як в Інтернеті, так і в локальній мережі.

2. Створення локальної мережі Wi-Fi для доступу до Інтернету та локальної інформації, а також відео з БПЛА.

3. Налаштування зв'язку також у складних умовах місцевості, де звичайні мобільні або портативні радіостанції не справляються з вимогами зв'язку (оснащується мобільною радіостанцією стандарту радіозв'язку SFS з антеною з високим коефіцієнтом посилення енергії).

Ключовим питанням при складанні робочої станції GCS є правильне розділення всіх з’єднань і усунення будь-яких потенційних радіоперешкод.

І мережа Wi-Fi, і зв’язок БПЛА працюють у стандартах 2,4 і 5 ГГц. Тому важливо вибрати оптимальний слот (канал) для з’єднань, щоб уникнути можливих перешкод у зв’язку між БПЛА та контрольним обладнанням. Це можна зробити за допомогою високоякісних мережевих пристроїв Ubiquiti, які постійно перевіряють зайнятість мережі.

Усі кабелі та з’єднання, які використовуються на робочій станції GCS, мають бути екрановані, щоб мінімізувати можливі перешкоди. Постійне джерело живлення та високоякісні перетворювачі та стабілізатори напруги гарантують стабільну роботу всього рішення.

3. РОЗРОБКА МОДУЛЯ РОЗПІЗНАВАННЯ ОБ'ЄКТІВ ЗА ДОПОМОГОЮ КОМП'ЮТЕРНОГО ЗОРУ

Алгоритм розпізнавання зображень, він же класифікатор зображень, приймає зображення або частину зображення як вхідні дані та виводить ділянку, яка містить шукане зображення. Тобто, результатом є мітка класу.

В основі алгоритму розпізнавання стоїть двійковий класифікатор. Сам термін «двійковий класифікатор» передбачає визначення, чи є частина зображення, що аналізується, шуканим елементом (наприклад, обличчям) чи фоном.

Наступна діаграма ілюструє етапи створення традиційного класифікатора зображень (Рис. 3.1).

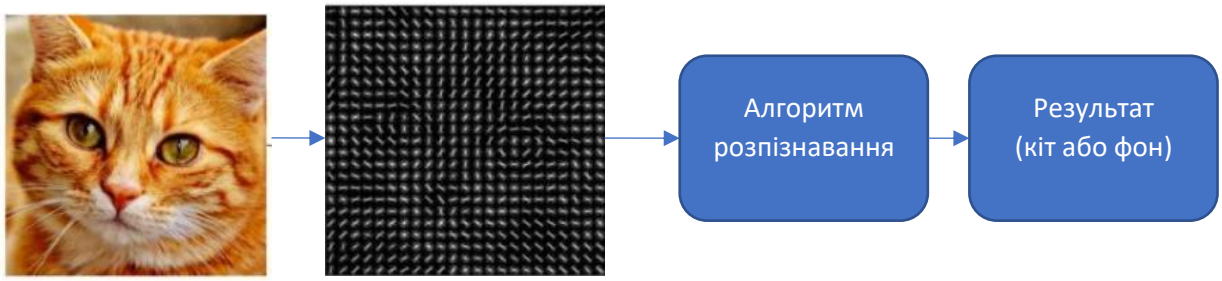

Рисунок. 3.1 – Структурна схема задачі двійкової класифікації

Цікаво, що багато традиційних алгоритмів класифікації зображень комп'ютерного зору дотримуються цього конвеєра, тоді як алгоритми на основі глибокого навчання повністю обходять етап вилучення ознак. Давайте розглянемо ці кроки докладніше.

Часто вхідне зображення попередньо обробляється для нормалізації ефектів контрастності та яскравості. Дуже поширеним етапом попередньої обробки є віднімання середнього значення інтенсивності зображення та ділення на стандартне відхилення. Іноді гамма-корекція дає трохи кращі результати. Під час роботи з кольоровими зображеннями трансформація колірного простору може допомогти отримати кращі результати.

Оскільки в системах пошуку можуть використовуватися різні камери, в тому числі і тепловізійні, проведено оцінку, включаючи колірні простори відтінків сірого, RGB і LAB, із вирівнюванням по степеневому закону (гамма). Ці нормалізації мають лише помірний вплив на продуктивність, можливо тому, що наступна нормалізація дескриптора досягає подібних результатів. Ми використовуємо інформацію про колір, якщо вона доступна. Кольорові простори RGB і LAB дають порівняльні результати, але обмеження відтінками сірого знижує продуктивність на 1,5% при 10-4 FPPW. Гамма-стиск квадратного кореня кожного колірного каналу покращує продуктивність при низькій FPPW (на 1% при 10-4 FPPW), але логарифмічне стиснення є надто сильним і погіршує його на 2% при 10-4 FPPW.

Таким чином, не зрозуміло, яку попередню обробку використовувати. У рамках попередньої обробки вхідне зображення або фрагмент зображення також обрізається та змінюється до фіксованого розміру. Це важливо, оскільки наступний крок, виділення ознак, виконується на зображенні фіксованого розміру.

Вхідне зображення містить забагато додаткової інформації, яка не потрібна для класифікації. Тому першим кроком у класифікації зображень є спрощення зображення шляхом вилучення важливої інформації, що міститься на зображенні, і вилучення решти. Наприклад, якщо на меті є знайти профіль людини на зображеннях, спостерігається значна варіація значень яскравості компонент пікселів RGB. Однак, запустивши детектор країв на зображенні, ми можемо спростити зображення. Так, можна зробити висновок, що виявлення країв зберігає важливу інформацію, відкидаючи несуттєву. Крок називається вилученням ознак. У традиційних підходах комп’ютерного зору розробка цих функцій має вирішальне значення для продуктивності алгоритму. Виявляється, ми можемо зробити набагато краще, ніж просте виявлення країв, і знайти функції, що є набагато надійнішим.

Деякі добре відомі функції, які використовуються в комп’ютерному зорі, — це функції, подібні до Хаара, представлені Віолою та Джонсом, гістограма орієнтованих градієнтів (HOG), масштабно-інваріантне перетворення ознак (SIFT), прискорена надійна функція (SURF), тощо.

Як конкретний приклад, розглянемо вилучення ознак за допомогою гістограми орієнтованих градієнтів (HOG).

3.1 Гістограма орієнтованих градієнтів

Алгоритм виділення ознак перетворює зображення фіксованого розміру на вектор ознак фіксованого розміру. У випадку виявлення пішоходів дескриптор функції HOG обчислюється для фрагмента зображення розміром 64 × 128 і повертає вектор розміром 3780. Слід зауважити, що вихідний розмір цього фрагмента зображення становив 64 x 128 x 3 = 24 576, тобто зменшено до 3780 за допомогою дескриптора HOG.

HOG базується на ідеї, що зовнішній вигляд локального об’єкта можна ефективно описати за допомогою розподілу (гістограми) напрямків країв (орієнтованих градієнтів). Нижче наведено кроки для обчислення дескриптора HOG для зображення 64×128.

Розрахунок градієнта здійснюється наступним чином. обчисліть градієнтні зображення x і y, g_x і g_y, з вихідного зображення. Це можна зробити, відфільтрувавши оригінальне зображення за допомогою наступних ядер (Рис 3.2).

Рисунок. 3.2 – До обчислення градієнта

Використовуючи зображення градієнта g_x і g_y, ми можемо обчислити величину та орієнтацію градієнта за допомогою наступних рівнянь:

$$g = \sqrt{g_x^2 + g_y^2}$$

$$\theta = arctan\frac{g_y}{g_x}$$

Обчислені градієнти є «беззнаковими», тому θ знаходиться в діапазоні від 0 до 180 градусів.

Далі, має сенс розділити зображення, наприклад, на 8×8 клітинок.

Обчисліть гістограму градієнтів у цих клітинках 8×8: у кожному пікселі клітинки 8×8 ми знаємо градієнт (величину та напрямок), отже, ми маємо 64 величини та 64 напрямки — тобто 128 чисел. Гістограма цих градієнтів забезпечить більш корисне та компактне представлення. Далі ми перетворимо ці 128 чисел на гістограму з роздільною здатністю 9 бітів. Розряди (біти) гістограми відповідають напрямкам градієнтів 0, 20, 40 … 160 градусів. Кожен піксель відповідає за один або два біти на гістограмі. Якщо напрямок градієнта на пікселі дорівнює точно 0, 20, 40 … або 160 градусів, то градієнт подається пікселем у кошик. Піксель, у якому напрямок градієнта не дорівнює точно 0, 20, 40 … 160 градусів, розподіляє свій напрямок між двома найближчими контейнерами на основі відстані від контейнера.

Нормалізація блоку відбувається наступним чином. Гістограма, розрахована на попередньому кроці, не дуже стійка до змін яскравості. Множення інтенсивності зображення на постійний коефіцієнт також масштабує значення гістограми. Щоб протистояти цим ефектам, ми можемо нормалізувати гістограму — тобто подумати про гістограму як про вектор із 9 елементів і розділити кожен елемент на величину цього вектора.

У попередніх кроках з’ясовано, як обчислити гістограму для клітинки 8×8, а потім нормалізувати її для блоку 16×16. Щоб обчислити остаточний вектор ознак для всього зображення, блок 16 × 16 переміщується з кроком 8 (тобто 50% перекриття з попереднім блоком) і 36 чисел (що відповідають 4 гістограмам у блоці 16 × 16), обчислених за кожен крок об’єднується, щоб створити кінцевий вектор ознак.

Вхідне зображення має розмір 64 × 128 пікселів, і ми рухаємося на 8 пікселів за раз. Отже, ми можемо зробити 7 кроків у горизонтальному напрямку та 15 кроків у вертикальному напрямку, що в сумі дає 7 x 15 = 105 кроків. На кожному кроці ми обчислюємо 36 чисел, що робить довжину кінцевого вектора 105 x 36 = 3780.

3.2 Алгоритм навчання для класифікації

Після перетворення зображення на вектор ознак, необхідно передати його до алгоритму класифікації, як вхідні дані та отримати мітку класу на виході (присутність або відсутність шуканого об’єкта).

Перш ніж алгоритм класифікації почне видавати двійковий результат, потрібно навчити його, показавши якомога більше зображень, які містять шуканий елемент. Загальний принцип навчання полягає в тому, що алгоритми навчання розглядають вектори ознак як точки у багатовимірному просторі та намагаються знайти площини або поверхні, які розділяють простір таким чином, щоб усі приклади, що належать до одного класу, були з одного боку площини або поверхні.

Розглянемо алгоритм навчання під назвою Support Vector Machines (SVM).

Машина опорних векторів (SVM) є одним із найпопулярніших керованих алгоритмів двійкової класифікації. Хоча ідеї, що використовуються в SVM, існують з 1963 року, поточна версія була запропонована в 1995 році Кортесом і Вапником.

На попередньому кроці ми дізналися, що дескриптор HOG зображення є вектором ознак довжиною 3780. Ми можемо розглядати цей вектор як точку в 3780-вимірному просторі. Візуалізація простору з більшою вимірністю неможлива, тому уявимо, що вектор ознак є лише двовимірним.

У нашому спрощеному варіанті маємо 2D-точки, що представляють два класи (наприклад, силует людини і фон). На Рис. 3.3 представлені два класи двома різними видами точок. Усі чорні крапки належать до одного класу, а білі — до іншого. Під час навчання ми надаємо алгоритму багато прикладів із двох класів. Таким чином, ми повідомляємо алгоритму координати двовимірних точок, а також те, чорна чи біла точка.

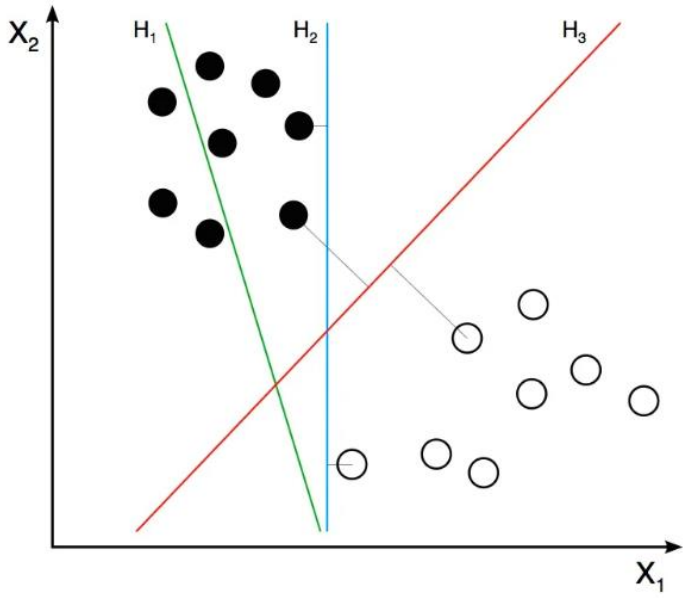

Рисунок. 3.3 – До пояснення опису алгоритму класифікації

Різні алгоритми навчання визначають, як розділити ці два класи різними способами. Лінійний SVM намагається знайти найкращу лінію, яка розділяє два класи. На малюнку вище H_1, H_2 і H_3 є трьома лініями в цьому двовимірному просторі. H_1 не розділяє два класи і тому не є хорошим класифікатором. H_2 і H_3 розділяють два класи, але H_3 є кращим класифікатором, ніж H_2, тому що H_3

чіткіше розділяє два класи. H_3 вибирається таким чином, щоб лінія поділу знаходилася на максимальній відстані від членів двох класів.

Враховуючи 2D-функції на малюнку вище, SVM знайде лінію H_3. Якщо буде отримано новий двовимірний вектор ознак, що відповідає зображенню, якого алгоритм ніколи раніше не бачив, можна просто перевірити, з якого боку лінії лежить точка, і призначити їй відповідну мітку класу. Якщо вектори об’єктів є тривимірними, SVM знайде відповідну площину, яка максимально розділяє два класи. Якщо вектор ознак знаходиться в 3780-вимірному просторі, SVM знайде відповідну гіперплощину.

Якими мають бути дії, якщо об'єкти, що належать до двох класів, не можна розділити за допомогою гіперплощини? У таких випадках SVM все одно знаходить найкращу гіперплощину, вирішуючи задачу оптимізації, яка намагається збільшити відстань гіперплощини від двох класів, одночасно намагаючись переконатися, що багато навчальних прикладів правильно класифіковані.

Лістинг програми з використанням вищезазначеного алгоритму, який реалізовано у бібліотеці OpenCV приведено у Додатку А. Програмний додаток реалізовано на мові високого рівня C++. Додаток є консольним і не містить графічного інтерфейсу. Для тестування системи і навчання алгоритму, використано серію тепловізійних зображень. Приклади фото для аналізу приведено на Рис. 3.4.

Рисунок 3.4 – Приклади тепловізійних зображень для аналізу

4. РЕЗУЛЬТАТИ ТЕСТУВАННЯ СИСТЕМИ

4.1 Збільшення вірогідності розпізнавання

Для тестування системи використано фрагменти з фото, які містять шуканий об’єкт. Модуль навчання доповнено системою перетворення вхідного зображення, використано два типи перетворень: обертання і створення штучної перспективи. Приклад серії перетворень приведено на Рис. 4.1.

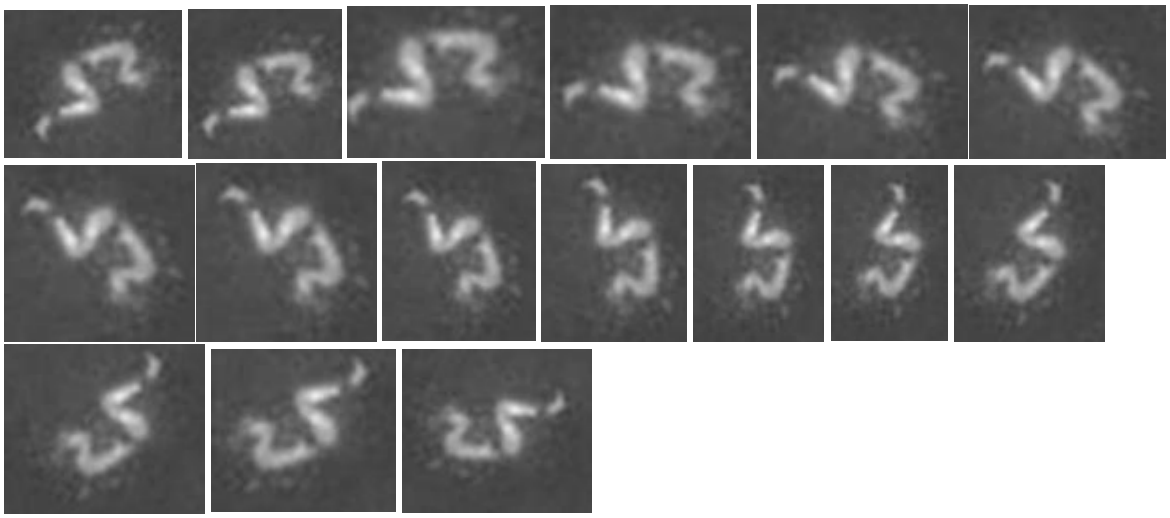

Рис. 4.1. – Штучне обертання об’єкта для процесу навчання

Експериментально встановлено, що при дефіциті зображень для навчання, доцільно прибігати до методів перетворення зображень, які зводяться до обертання площини (зображення) у тривимірному просторі. Таким чином штучно можна змінити кут спостереження (фотографування). Тобто, застосовується матриця повороту для виконання власного ортогонального перетворення в евклідовому просторі.

4.2 Тестування системи.

Результат розпізнавання об’єкта на фото, яке не входило до стеку зображень для навчання приведено на Рис. 4.2.

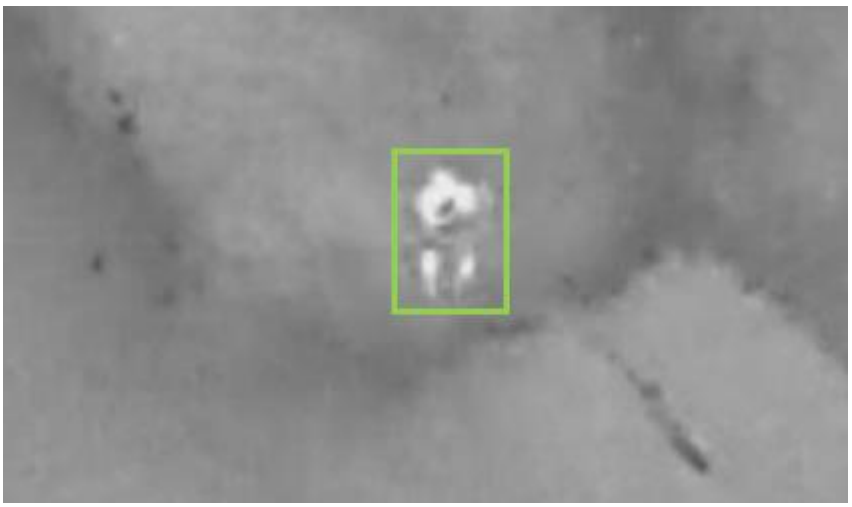

Рис. 4.2. – Результат розпізнавання об’єкта

ВИСНОВКИ

Для вирішення задачі розпізнавання об’єктів було розроблено програмний додаток на мові високого рівня C++. Основними результатами роботи є наступні.

1. Проведено огляд існуючих проблем виявлення постраждалих. Розглянуті сучасні формати відео, а також варіанти їх протоколів передачі і засобів розпізнавання об’єтків у відеопотоці. Проведено огляд бібліотек розпізнавання об’єктів і обґрунтовано вибір бібліотеки OpenCV.

2. Проведено аналіз існуючих методів використання БПЛА на основі проведених експериментів. Наведена структура, організація та складові БПЛА операцій.

3. Розроблено консольний додаток на мові високого рівня C++ для пошуку потенційних потерпілих. Проведено навчання системи для розпізнавання профіля людини на основі тепловізійних зображень.

4. Проведено навчання для класифікації тепловізійних, зображень за допомогою алгоритму SVM.

5. Наведені результати тестування програми. Проведено оптимізацію коду для досягненні мінімальної затримки, а також навантаження системи. Проведено оцінку можливості переносу додатка для виконання на одноплатному комп’ютері.

Таким, чином результати роботи можуть бути застосовані при розробці програмного забезпечення, для пошуку потерпілих за допомогою БПЛА. Розроблений програмний додаток придатний значно зменшує час обробки відео та виявлення профілей людини. Розроблений додаток має можливості модифікації для навчання та пошуку об’єктів інших типів.

ПЕРЕЛІК ПОСИЛАНЬ

1. Chakraborty, P., Dev, S., & Naganur, R. H. (2015). Dynamic HTTP Live Streaming Method for Live Feeds. 2015 International Conference on Computational Intelligence and Communication Networks (CICN). -1395c.

2. Nurrohman, A., & Abdurohman, M. (2018). High Performance Streaming Based on H264 and Real Time Messaging Protocol (RTMP). 2018 6th International Conference on Information and Communication Technology (ICoICT). -175c.

3. Edan, N. M., Al-Sherbaz, A., & Turner, S. (2017). Design and evaluation of browser-to-browser video conferencing in WebRTC. 2017 Global Information Infrastructure and Networking Symposium (GIIS). -78c.

4. Wang Xing Guo, Zheng Wei Guo, & Ahmad, I. (n.d.). MPEG-2 to MPEG-4 transcoding. Proceedings of Workshop and Exhibition on MPEG-4 (Cat. No.01EX511). -83c.

5. Bo Li, Huahui Chen, Yucheng Chen, Yuchao Dai, & Mingyi He. (2017). Skeleton boxes: Solving skeleton based action detection with a single deep convolutional neural network. 2017 IEEE International Conference on Multimedia & Expo Workshops (ICMEW). -613c.

6. Zhu, M., & Liu, M. (2018). Mobile Video Object Detection with Temporally-Aware Feature Maps. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. – 5697c.

7. Tian, X., Saito, H., Girkar, M., Preis, S. V., Kozhukhov, S. S., Cherkasov, A. G., Geva, R. (2012). Compiling C/C++ SIMD Extensions for Function and Loop Vectorizaion on Multicore-SIMD Processors. 2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum. -2351c

8. Túsnio, N.; Wróblewski, W. The Efficiency of Drones Usage for Safety and Rescue Operations in an Open Area: A Case from Poland. Sustainability 2022. -16c

ДОДАТОК А
Лістинг програмного коду

```
#include  "opencv2/text.hpp"
#include  "opencv2/highgui.hpp"
#include  "opencv2/imgproc.hpp"

#include  <vector>
#include  <iostream>
#include  <iomanip>
#include  <thread>

#include <windows.h>

#include "Detector.h"
#include "Recognizer.h"

using namespace std;
using namespace cv;
using namespace cv::text;

void groups_draw(Mat& src, vector<Rect>& groups, cv::Point shift);
void er_show(vector<Mat>& channels, vector<vector<ERStat> >& regions);
int NumberDigit(std::string in);

HANDLE hSerial;
std::mutex g_mutex;

void ReadCOM(bool& isCath)
{
    DWORD iSize;
    char sReceivedChar[3];
    isCath = false;

    std::lock_guard<std::mutex> lock(g_mutex);
    while (true)
    {
        ReadFile(hSerial, &sReceivedChar, 1, &iSize, 0); //получаем 1
байт
        if (iSize > 0)
        {
            // если что-то принято, выводим
            for (int i = 0; i < iSize; ++i)
            {
                std::cout << std::to_string(sReceivedChar[i]) << "  ";
            }
            std::cout << std::endl;
            isCath = true;
        }
    }
    std::lock_guard<std::mutex> unlock(g_mutex);
}
```

```
int main(int argc, const char* argv[])
{
    //Обьявление и инициализация порта

    LPCTSTR sPortName = L"COM3";
    hSerial = ::CreateFile(sPortName, GENERIC_READ | GENERIC_WRITE, NULL,
NULL, OPEN_EXISTING, 0, NULL);


    if (hSerial == INVALID_HANDLE_VALUE)
    {
        if (GetLastError() == ERROR_FILE_NOT_FOUND)
        {
            std::cout << "serial port does not exist.\n";
        }
        std::cout << "some other error occurred.\n";
    }

    DCB dcbSerialParams = { 0 };
    dcbSerialParams.DCBlength = sizeof(dcbSerialParams);
    if (!GetCommState(hSerial, &dcbSerialParams))
    {
        std::cout << "getting state error\n";
    }
    dcbSerialParams.BaudRate = CBR_9600;
    dcbSerialParams.ByteSize = 8;
    dcbSerialParams.StopBits = ONESTOPBIT;
    dcbSerialParams.Parity = NOPARITY;
    if (!SetCommState(hSerial, &dcbSerialParams))
    {
        std::cout << "error setting serial port state\n";
    }


    //Ініціалізація камери
    cv::VideoCapture camera(0);
    if (!camera.isOpened())
    {
        std::cout << "ERROR: Could not open camera" << std::endl;
        return 1;
    }




    //Обьявление потока. В него будем передавать функцию чтения COM порта
    thread Worker;



    Mat frame;
    //Инициализация ROI
    Rect ROI = cv::Rect(cv::Point(0, 0), cv::Point(0, 0));

    cv::Mat src;
```

```
    bool isData = false;
    bool isCOM = false;
    bool isCrop = false;


    cv::namedWindow("Display window", WINDOW_AUTOSIZE);
    std::string outText;
    cv::Mat ToTess;




    vector<Mat> channels;
    vector<Rect> groups_boxes;



    int count_img = 0;
    int count_not_recog = 0;
    std::string dispROI = "0";
    std::string dispCount = "0";
    std::string NoRecogCount = "0";
    std::string PATH_DATA_IMG = "DataImg\\Recognize\\";
    std::string PATH_DATA_NO_RECOG = "DataImg\\NO_Recognize\\";


    Tesserct_Init();

    while (1)
    {
        camera >> frame;

        if (waitKey(1) == 's')
        {
            ROI = selectROI("Display window", frame, false);
            src = frame(ROI);
        }


        Worker=std::thread(&ReadCOM,  std::ref(isCOM));
        Worker.detach();

        if(ROI.width!=0 && isCOM)   //
        {
            channels = ComputeChannel(src, 1);
            groups_boxes = TextBox(channels, src);

            if (0 == groups_boxes.size())
            {
                cout << "TEXT DOES NOT DETECTED!" << endl;
                isData = false;
                count_not_recog++;
                cv::imwrite(PATH_DATA_NO_RECOG +
std::to_string(count_not_recog) + ".jpg", frame);
            }
            else
```

```
            {
                for (int i = 0; i < groups_boxes.size(); ++i)
                {
                    ToTess = src(groups_boxes[i]);
                    outText = GetText(ToTess);
                    cout << outText << endl;

                    if ((double)NumberDigit(outText) /
(double)outText.length() >= 0.7 && outText.length() > 12)
                    {
                        isData = true;
                        cout << "Data Find and Recognized!";
                        cv::rectangle(frame, ROI.tl(), ROI.br(),
Scalar(255, 0, 0), 2, 8);
                        groups_draw(frame, groups_boxes, ROI.tl());
                        count_img++;
                        cv::imwrite(PATH_DATA_IMG +
std::to_string(count_img) + ".jpg", frame);
                    }
                    else
                    {
                        isData = false;
                        cout << "Data Find and NOT Recognized!";
                        count_not_recog++;
                        cv::imwrite(PATH_DATA_NO_RECOG +
std::to_string(count_not_recog) + ".jpg", frame);
                    }

                    ToTess.release();
                }
            }
        }

        isData = false;

        dispROI = "Rect (TL, W, H): (" + std::to_string(ROI.tl().x) + ","
+ std::to_string(ROI.tl().x) + "); " + std::to_string(ROI.width) + "; " +
std::to_string(ROI.height);
        dispCount = "Detected data: " + std::to_string(count_img);
        NoRecogCount= "NOT detected data: " +
std::to_string(count_not_recog);
        cv::putText(frame, dispROI, cv::Point(10, 20),
cv::FONT_HERSHEY_DUPLEX, 0.5, cv::Scalar(0, 255, 0), 1, false);
        cv::putText(frame, dispCount, cv::Point(10, 40),
cv::FONT_HERSHEY_DUPLEX, 0.5, cv::Scalar(0, 255, 0), 1, false);
        cv::putText(frame, NoRecogCount, cv::Point(10, 60),
cv::FONT_HERSHEY_DUPLEX, 0.5, cv::Scalar(0, 255, 0), 1, false);
        cv::imshow("Display window", frame);


        if (waitKey(10) == 27) break;

    }
```

```
    Clear_Tesseract();
    CloseHandle(hSerial);

    if (!groups_boxes.empty())
    {
        groups_boxes.clear();
    }


    return 0;

}

void groups_draw(Mat& src, vector<Rect>& groups, cv::Point
shift=cv::Point(0,0))
{
    for (int i = (int)groups.size() - 1; i >= 0; i--)
    {
        if (src.type() == CV_8UC3)
            rectangle(src, groups.at(i).tl()+shift, groups.at(i).br() +
shift, Scalar(0, 0, 255), 2, 8);
        else
            rectangle(src, groups.at(i).tl() + shift, groups.at(i).br() +
shift, Scalar(255), 1, 8);
    }
}

void er_show(vector<Mat>& channels, vector<vector<ERStat> >& regions)
{
    for (int c = 0; c < (int)channels.size(); c++)
    {
        Mat dst = Mat::zeros(channels[0].rows + 2, channels[0].cols + 2,
CV_8UC1);
        for (int r = 0; r < (int)regions[c].size(); r++)
        {
            ERStat er = regions[c][r];
            if (er.parent != NULL) // deprecate the root region
            {
                int newMaskVal = 255;
                int flags = 4 + (newMaskVal << 8) + FLOODFILL_FIXED_RANGE
+ FLOODFILL_MASK_ONLY;
                floodFill(channels[c], dst, Point(er.pixel %
channels[c].cols, er.pixel / channels[c].cols),
                    Scalar(255), 0, Scalar(er.level), Scalar(0), flags);
            }
        }
        char buff[20]; char* buff_ptr = buff;
        //sprintf(buff, "channel %d", c);
        imshow(buff_ptr, dst);
    }
    waitKey(-1);
}

int NumberDigit(std::string in)
{
```

```
    int Num = 0;
    for (int i = 0; i < in.size(); ++i)
        if (isdigit(in[i])) ++Num;

    return Num;
}
```