МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра комп'ютерної інженерії

Гергардт Сергій Олександрович

«Алгоритм детекції рухомих об'єктів у відеопотоці на основі адаптивного ковзаючого вікна / Algorithm for moving objects detecting in a video stream based on an adaptive sliding window»

спеціальність: 123 - Комп'ютерна інженерія
освітньо-професійна програма - Комп'ютерна інженерія
Кваліфікаційна робота

Виконав студент групи КІм-22
С.О. Гергардт
_____

Науковий керівник:
к.т.н., доц. Ю.М. Батько
_____

Кваліфікаційну роботу допущено
до захисту:

"___" _____ 20___ р.

Завідувач кафедри
_____О. М. Березький

Тернопіль – 2021

ЗМІСТ

ВСТУП

Актуальність роботи. На сьогоднішній день велика кількість наукових досліджень проводиться якраз в рамках комп'ютерного зору, який ястаннім часом отримує все більше визнання та застосування. Зростаючий інтерес до аналізу руху людини останнім часом сильно мотивований покращення алгоритмів та моделей комп'ютерного зору, доступністю недорогого обладнання, такого як відео камери та різноманітні нові перспективні програми які можуть проводити як ідентифікацію особи так і візуальне спостереження. Метою виявлення руху є розпізнавання зміни позиції предмету на двох послідовних зображеннях. Крім того, знаходження руху об'єктів може сприяти розпізнаванню об'єктів. Таким чином, основна мета дослідження полягає в тому, щоб розпізнати пікселі, що належать одному об’єкту. Проте процес детекції сильно залежиться від деяких сторонніх факторів, серед яких:

- технічні можливості камери, а також стабільність її положення є ключовою в процесі детекції руху;
- стабільне світло, без мерехтіння;
- висока частота кадрів і роздільна здатність камери.

Аналіз руху людського тіла був цікавим дослідникам через його різноманітність на яку впливають і такі фактори як фізична продуктивність, оцінка, медична діагностика, віртуальна реальність. В даний час для виявлення рухомих об'єктів використовується в основному кадровий метод віднімання, метод віднімання фону та метод оптичного потоку.

Метод оптичного потоку полягає в обчисленні поля оптичного потоку зображення і виконанні кластера обробка відповідно до характеристик розподілу оптичного потоку зображення. Метод фонового віднімання полягає у використанні методу відмінності поточного зображення та фонового зображення для виявлення переміщення об'єктів. Даний підхід є дуже чутливими до змін зовнішнього довкілля і має погану здатність до захисту від

перешкод. У рамковому методі віднімання наявність рухомих об'єктів визначається шляхом обчислення різниці між двома послідовні зображення. Будь-яка система виявлення руху на основі фону завжди буде чутлива до таких зовнішніх факторів як:

- зашумленість зображення через неякісне джерело зображення;
- поступові зміни умов освітлення в сцені;
- невеликі рухи нестатичних об’єктів, таких як гілки дерев та кущі тощо.

Таким чином для побудови програмної системи відеоспостереження з елементави детекції рухомих обєктів необхідно спроектувати та реалізувати алгоритм на основі ковзаючого вікна. При цьому необхідно вихначити граничну область на якій відбувається спостереження постійно та центральну область на якій спостреження відбувається тільки під час виявлення рухомого обєкту. Тому задача створення алгоритму детекції рухомих обєктів на основі ковзаючого вікна є актуальною.

Метою роботи є розробка алгоритму детекції рухомих об'єкті на основі ковзаючого вікна для комп’ютерних систем відеоспостереження.

Для досягнення даної мети ставились наступні завдання:

- провести аналіз та класифікацію завдань комп’ютерного зору;
- проаналізувати методи кодування та передачі сигналів у відеопотоках;
- провести аналітичний огляд наявних програмних засобів для виділеня та класифікації об’єктів на зображеннях;
- проаналізувати існуючі методи та підходи до виділення та класифікації об’єктів на зображеннях;
- розробити алгоритм детекції обєктів на зображенні на основі ковзаючого вікна;
- реалізувати програмну систему детекції обєктів на зображенні на основі ковзаючого вікна, провести її тестування та порівняти з подібними програмами.

Об’єкт дослідження – процес аналізу цифрових зображень в системах відеоспостереження.

Предмет дослідження – методи і алгоритми розпізнавання цифрових зображень у відеопотоці.

Наукова новизна одержаних результатів визначається наступним чином:

– проведено комплексний аналіз та класифікацію алгоритмів виділення об’єктів на цифрових зображеннях, що надало можливість підкреслити їх переваги та недоліки, а також розробити власний алгоритм детекції руха на основі ковзного вікна;

– розроблено алгоритм детекції рухомих об’єктів на основі аналізу граничних полів області спостереження, що дозволило зменшити обчислювальну складність процесу детекції рухомих об’єктів.

Практична цінність одержаних результатів полягає в тому, що:

– розроблено та проведено теоретичне дослідження програмної системи відоспостереження з елементами автоматичної детекції руху на основі ковзаючого вікна, що дозволило в подальшому програмно реалізувати та провести дослідження розроблених алгоритмів;

– реалізовано програмне забезпечення для детекції рухомих об’єктів на основі цифрових бібліотек OpenCV та Image AI та з використанням алгоритмів комп’ютерного зору.

Публікації та апробація до випускної кваліфікаційної роботи. За результатами наукових досліджень, проведених у випускній кваліфікаційні роботі, підготовлено тези доповіді «Аналіз алгоритмів попередньої обробки зображень систем автоматизованого моніторингу» обсягом 1 сторінка на V Науково-практичній конференції молодих вчених і студентів «Інтелектуальні комп’ютерні системи та мережі», а також «Аналіз алгоритмів сегментації для систем автоматизованого аналізу зображень» обсягом 1 сторінка на V Науково-практичній конференції молодих вчених і студентів «Інтелектуальні комп’ютерні системи та мережі».

1 ПРОГРАМНІ ТА АПАРАТНІ СИСТЕМИ АНАЛІЗУ ТА ОБРОБКИ ВІДЕОПОТОКІВ

1.1 Комп’ютерний зір, основні поняття та функції

Алгоритми комп’ютерного зору на даний момент є однією з найбільш трансформаційних і потужних систем штучного інтелекту у світі. Системи комп’ютерного зору використовуються в автономних транспортних засобах, роботів-навігаціях, системах розпізнавання облич тощо. Щоб повністю зрозуміти, як працюють системи комп’ютерного зору, давайте спочатку обговоримо, як люди розпізнають об’єкти. Найкращим поясненням того, як люди розпізнають об’єкти, є модель, яка описує початкову фазу розпізнавання об’єктів як таку, коли основні компоненти об’єктів, такі як форма, колір і глибина, спочатку інтерпретуються мозком. Сигнали від ока, які надходять у мозок, аналізуються, щоб спочатку витягнути краї об’єкта, і ці краї об’єднуються в більш складне уявлення, яке завершує форму об’єкта.

Системи комп’ютерного зору працюють дуже подібно до зорової системи людини: спочатку розрізняють краї об’єкта, а потім з’єднують ці краї разом у форму об’єкта. Велика відмінність полягає в тому, що, оскільки комп’ютери інтерпретують зображення як числа, системі комп’ютерного зору потрібен спосіб інтерпретації окремих пікселів, які складають зображення. Система комп’ютерного зору призначатиме значення пікселям на зображенні, і, досліджуючи різницю значень між однією областю пікселів та іншою областю пікселів, комп’ютер може розрізняти краї. Наприклад, якщо розглянуте зображення має відтінки сірого, то значення будуть варіюватися від чорного (відображається 0) до білого (представленого 255). Раптова зміна діапазону значень пікселів поруч один з одним буде вказувати на край.

Цей основний принцип порівняння значень пікселів також можна виконати з кольоровими зображеннями, коли комп’ютер порівнює відмінності між різними каналами кольору RGB. Отже, система комп’ютерного зору вивчає

значення пікселів для інтерпретації зображення, а звідси архітектура системи комп’ютерного зору матиме наступну структуру яку наведемо на прикладі згорткової нейронної мережі.

Згорткові нейронні мережі (CNN). Основним типом штучного інтелекту, який використовується в задачах комп'ютерного зору, є той, що базується на згорткових нейронних мережах. Згортки – це математичні процеси, які мережа використовує для визначення різниці значень між пікселями. Якщо уявити сітку значень пікселів, то це ще менша сітка яка переміщується по цій головній сітці. Значення під другою сіткою аналізуються мережею, тому мережа розглядає лише кілька пікселів за раз. Це часто називають технікою «розсувних вікон». Значення, що аналізуються за допомогою розсувного вікна, підсумовуються мережею, що допомагає зменшити складність зображення та полегшити для мережі вилучення шаблонів.

Згорткові нейронні мережі поділяються на дві різні секції:

- згорткові;
- повністю зв’язані.

Згорткові шари мережі – це екстрактори ознак, завдання яких – аналізувати пікселі в зображенні та формувати їх уявлення, з яких щільно пов’язані шари нейронної мережі можуть вивчати шаблони. Згорткові шари починаються з простого вивчення пікселів і виділення низькорівневих ознак зображення, таких як краї. Пізніші згорткові шари з’єднують краї разом у більш складні форми. Відповідно в кінці мережа матиме представлення країв і деталей зображення, які вона зможе передати повністю пов’язаним шарам.

При роботі з згортковими нейронними мережами слід виділити ряд ключових етапі які необхідно виділити для успішного виконання завдання. Перелік етапів для успішної роботи з згортковими нейроними мережами наведено на рисунку 1.1.

Згортка. згортка виконується на зображенні для визначення певних елементів зображення. Згортка допомагає розмити, збільшити різкість, виявити

краї, зменшити шум тощо на зображенні, що може допомогти машині дізнатися конкретні характеристики зображення.

Об’єднання. Закручене зображення може бути занадто великим, і тому його потрібно зменшити. Об’єднання в основному виконується для зменшення зображення без втрати функцій або візерунків.

Згладжування. Згладжування перетворює двовимірну матрицю ознак у вектор ознак, який можна подати в нейронну мережу або класифікатор.

Повне підключення. Повне підключення просто відноситься до процесу подачі сплощеного зображення в нейронну мережу.

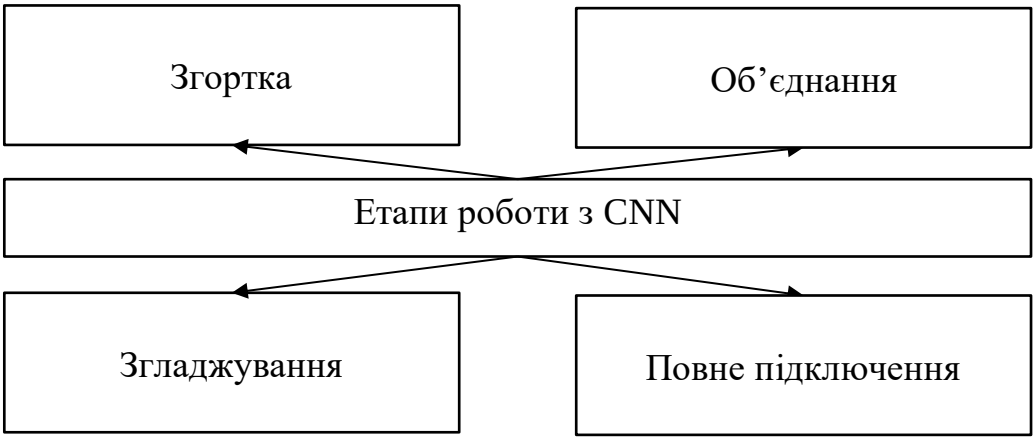

Рисунок 1.1 – Ключові етапи при роботі з нейронними мережами

У той час як згорткова нейронна мережа може витягувати шаблони із зображень сама по собі, точність системи комп’ютерного зору можна значно підвищити, додавши до зображень анотації.

Анотація зображення – це процес додавання метаданих до зображення, які допомагають класифікатору виявляти важливі об’єкти на зображенні. Використання анотації зображень важливо, коли системи комп’ютерного зору повинні бути високоточними, наприклад, під час керування автономним транспортним засобом або роботом.

Існують різні способи анотування зображень для покращення продуктивності класифікатора комп’ютерного зору. Анотація зображень часто

виконується за допомогою обмежувальних рамок, блоку, який оточує краї цільового об’єкта та наказує комп’ютеру зосередити свою увагу всередині рамки.

Семантична сегментація – це інший тип анотації зображення, який діє шляхом присвоєння класу зображення кожному пікселю в зображенні. Іншими словами, кожен піксель, який можна вважати «травою» або «деревою», буде позначено як належний до цих класів. Ця техніка забезпечує точність на рівні пікселів, але створення анотацій семантичної сегментації є більш складним і трудомістким, ніж створення простих обмежувальних рамок. Існують також інші методи анотації, такі як лінії та точки.

Машинне навчання є однією з найбільш швидкозростаючих технологічних галузей, але незважаючи на те, як часто вживаються слова «машинне навчання», може бути важко зрозуміти, що саме таке машинне навчання.

Машинне навчання не відноситься до однієї речі, це загальний термін, який можна застосувати до багатьох різних концепцій і технік. Розуміння машинного навчання означає знайомство з різними формами аналізу моделі, змінними та алгоритмами. Хоча термін машинне навчання можна застосувати до багатьох різних речей, загалом цей термін відноситься до дозволу комп’ютеру виконувати завдання, не отримуючи явних порядкових інструкцій для цього. Спеціалісту з машинного навчання не потрібно писати всі кроки, необхідні для вирішення проблеми, оскільки комп’ютер здатний «навчатися», аналізуючи закономірності в даних і узагальнюючи ці закономірності на нові дані.

Системи машинного навчання складаються з трьох основних частин:

- входи;
- алгоритми;
- виходи.

Вхідні дані – це дані, які подаються в систему машинного навчання, а вхідні дані можна розділити на мітки та функції. Характеристики – це відповідні змінні, змінні, які будуть проаналізовані, щоб дізнатися закономірності та

зробити висновки. Тим часом мітки — це класи/описи, надані окремим екземплярам даних.

Функції та мітки можна використовувати в двох різних типах машинного навчання: навчання з наглядом і навчання без нагляду (рисунок 1.2).

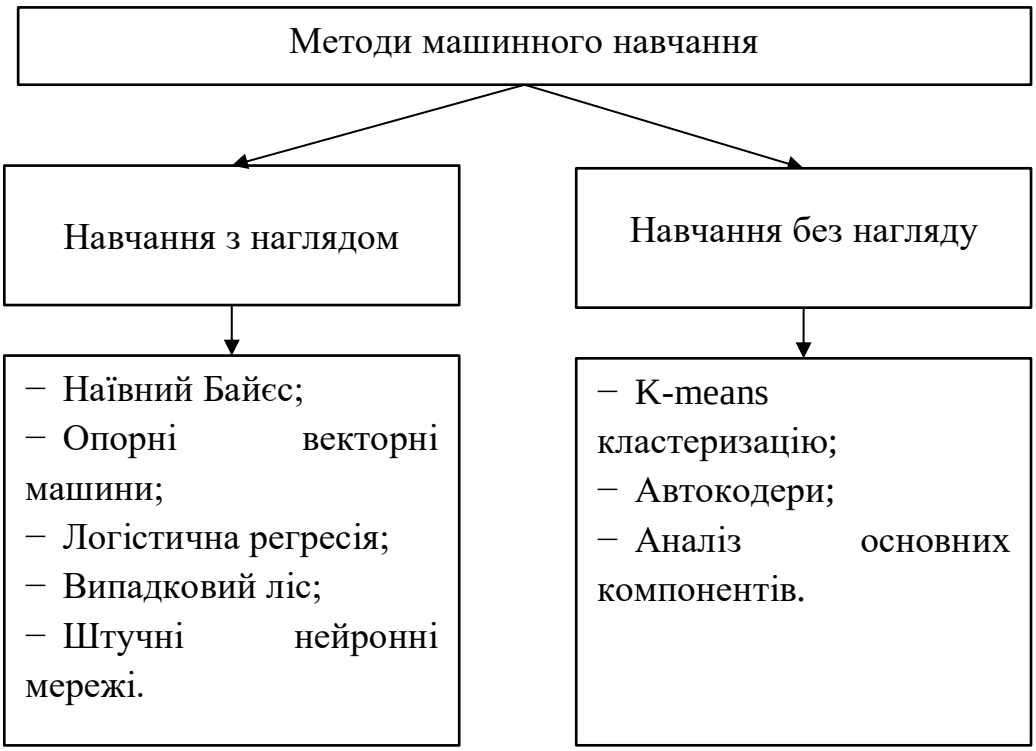

Рисунок 1.2 – Класифікація методів машинного навчання

Навчання без нагляду проти контролюваного навчання. При навчанні з керівництвом вхідні дані супроводжуються основною істиною. Проблеми з навчанням під керівництвом мають правильні вихідні значення як частину набору даних, тому очікувані класи відомі заздалегідь. Це дає змогу спеціалісту з даних перевірити продуктивність алгоритму, перевіривши дані в тестовому наборі даних і побачивши, який відсоток елементів було правильно класифіковано.

Навпаки, проблеми з навчанням без нагляду не мають основних позначок істини. Алгоритм машинного навчання, навчений виконувати завдання навчання

без нагляду, повинен мати можливість самостійно виводити відповідні закономірності в даних.

Алгоритми навчання з наглядом зазвичай використовуються для задач класифікації, де є великий набір даних, заповнений екземплярами, які необхідно відсортувати в один із багатьох різних класів. Іншим типом навчання з керівництвом є завдання регресії, де значення, виведене алгоритмом, є безперервним, а не категоричним.

Тим часом алгоритми навчання без нагляду використовуються для таких завдань, як оцінка щільності, кластеризація та навчання репрезентації. Ці три завдання потребують моделі машинного навчання, щоб зробити висновок про структуру даних, для моделі немає попередньо визначених класів.

Давайте коротко розглянемо деякі з найпоширеніших алгоритмів, які використовуються як у навчанні без нагляду, так і в навчанні з наглядом.

Поширені алгоритми навчання під наглядом включають:

- наївний Байєс;
- опорні векторні машини;
- логістична регресія;
- випадковий ліс;
- штучні нейронні мережі.

Машини опорних векторів — це алгоритми, які поділяють набір даних на різні класи. Точки даних групуються в кластери шляхом малювання ліній, що відокремлюють класи один від одного. Точки, знайдені з одного боку від прямої, будуть належати до одного класу, а точки по інший бік лінії — до іншого класу. Машини опорних векторів мають на меті максимізувати відстань між лінією та точками, знайденими по обидва боки від лінії, і чим більше відстань, тим впевненіше класифікатор, що точка належить одному класу, а не іншому.

Логістична регресія – це алгоритм, який використовується в завданнях двійкової класифікації, коли точки даних необхідно класифікувати як належні до одного з двох класів. Логістична регресія працює, позначаючи точку даних 1 або 0. Якщо сприйняте значення точки даних становить 0,49 або нижче, воно

класифікується як 0, а якщо воно дорівнює 0,5 або вище, воно класифікується як 1.

Алгоритми дерева рішень діють шляхом поділу наборів даних на все менші і менші фрагменти. Точні критерії, які використовуються для поділу даних, залежить від інженера з машинного навчання, але мета полягає в тому, щоб остаточно розділити дані на окремі точки даних, які потім будуть класифіковані за допомогою ключа.

Алгоритм Random Forest — це, по суті, багато окремих класифікаторів дерева рішень, пов’язаних разом у більш потужний класифікатор.

Наївний байесовский класифікатор обчислює ймовірність того, що дана точка даних сталася на основі ймовірності попереднього події, що сталася. Він заснований на теоремі Байєса і розділяє точки даних у класи на основі їх обчисленої ймовірності. При реалізації наївного байєсового класифікатора передбачається, що всі предиктори мають однаковий вплив на результат класу.

Штучна нейронна мережа, або багатошаровий персептрон, алгоритми машинного навчання , натхненні структури і функціями людського мозку. Штучні нейронні мережі отримали свою назву через те, що вони складаються з багатьох вузлів/нейронів, пов’язаних між собою. Кожен нейрон маніпулює даними за допомогою математичної функції. У штучних нейронних мережах є вхідні шари, приховані шари та вихідні шари.

Прихований шар нейронної мережі – це місце, де дані фактично інтерпретуються та аналізуються на наявність шаблонів. Іншими словами, це те, де вчиться алгоритм. Більше нейронів, об’єднаних разом, створюють складніші мережі, здатні вивчати більш складні шаблони.

Типи неконтрольованого навчання. Алгоритми навчання без нагляду включають:

- K-means кластеризацію;
- автокодери;
- аналіз основних компонентів.

Кластеризація K-середніх – це методика класифікації без нагляду, яка працює шляхом поділу точок даних на кластери або групи на основі їхніх особливостей. Кластеризація K-середніх аналізує ознаки, знайдені в точках даних, і розрізняє в них закономірності, які роблять точки даних, знайдені в кластері даного класу, більш схожими один на одного, ніж вони є на кластери, що містять інші точки даних. Це досягається шляхом розміщення можливих центрів для кластера, або центроїдів, на графіку даних і перепризначення положення центроїда, доки не буде знайдено положення, яке мінімізує відстань між центроїдом і точками, які належать до класу цього центроїда. Дослідник може вказати потрібну кількість кластерів.

Аналіз основних компонентів – це метод, який зменшує велику кількість функцій/змінних до меншого простору/меншої кількості функцій. «Основні компоненти» точок даних вибираються для збереження, тоді як інші ознаки стискаються в менше представлення. Зв'язок між вихідними зіллями даних зберігається, але оскільки складність точок даних простіша, дані легше кількісно оцінити та описати.

Автокодери – це версії нейронних мереж, які можна застосувати до завдань навчання без нагляду. Автокодери здатні приймати дані довільної форми без міток і перетворювати їх у дані, які може використовувати нейронна мережа, створюючи в основному власні мічені навчальні дані. Мета автокодера полягає в тому, щоб перетворити вхідні дані та перебудувати їх якомога точніше, тому мережа спонукає визначати, які функції є найважливішими та витягувати їх.

1.2 Аналіз технологій та протоколів передачі відео

Протокол потокового відео – це стандартизований метод доставки для розбиття відео на частини, відправлення його приймачу та повторної збірки.

Більшість цифрових відео призначені для двох речей: зберігання та відтворення. Це призводить до двох основних параметрів, а саме невеликого розміру файлу та універсального відтворення. Більшість відеофайлів не призначені для потокової передачі, а це означає, що для потокового відео спочатку потрібно конвертувати його в потоковий файл. Це передбачає розбиття його на невеликі частини. Потім ці фрагменти надходять послідовно і відтворюються в міру їх отримання. Якщо програма транслює відео в реальному часі то вихідне відео надходить прямо з камери. Інакше воно надходить із файлу формату VOD.

Протоколи потокової передачі можуть стати набагато складнішими. Багато з них, наприклад, є протоколами «адаптивного бітрейту». Ця технологія забезпечить найкращу якість, яку може підтримувати глядач у будь-який момент. Деякі протоколи зосереджені на зменшенні затримки або затримки між подією, що відбувається в реальному житті, і тим, коли вона відтворюється на екрані глядача. Деякі протоколи працюють лише в певних системах, а інші протоколи зосереджені на управлінні цифровими правами (DRM).

Термін «кодек» відноситься до технології стиснення відео. Логічно, різні потокові кодеки використовуються для різних цілей. Наприклад, Apple ProRes часто використовується для редагування відео. H.264, найпоширеніший відеокодек, широко використовується для онлайн-відео.

Як і у випадку з кодеком, термін «формат» також може ввести в оману в контексті протоколів потокового відео. У багатьох випадках формат просто відноситься до формату контейнера відеофайлу. Поширені формати контейнерів включають .mp4, .m4v та .avi.

По суті, формат контейнера функціонує як «коробка», яка зазвичай містить відеофайл, аудіофайл і метадані. Однак формат контейнера не є таким центральним поняттям для прямих трансляцій.

HTTP Live Streaming (HLS). Протокол HLS, або HTTP Live Streaming, був розроблений Apple і підтримує медіа-програвачі, веб-браузери, мобільні пристрої та медіа-сервери. Apple спочатку випустила цей протокол у 2009 році,

щоб дозволити їм видалити Flash з iPhone. З тих пір HLS став найбільш широко використовуваним протоколом потокової передачі.

По-перше, настільні браузери, смарт-телевізори, а також мобільні пристрої Android і iOS підтримують HLS. Відеопрогравачі HTML5 також підтримують потокове передавання HLS.

Це дозволяє потоку охопити якомога більше глядачів, що робить HLS найбезпечнішим протоколом на сьогодні для масштабування прямої трансляції для широкої аудиторії. Що стосується функцій, стандарт HLS також підтримує трансляцію з адаптивним бітрейтом, динамічно забезпечуючи найкращу можливу якість відео в будь-який момент. З останніми оновленнями цей стандарт тепер підтримує найновіший і найкращий кодек H.265, який забезпечує вдвічі кращу якість відео при тому ж розмірі файлу, що й H.264 . Наразі єдиним недоліком HLS є те, що затримка може бути відносно високою. Однак існують методи зменшення затримки HLS.

Плюси використання HLS:

- висока сумісність: оскільки HLS сумісний з відеопрогравачем HTML5, протокол HLS підходить для потокової передачі практично на будь-який пристрій і операційну систему з підтримкою Інтернету;
- безпечний: HLS відомий безпечним потоковим передаванням;
- висока якість: HLS створює надвисокоякісні відеопотоки завдяки технології потокового потокового адаптивного бітрейту;

Мінуси використання HLS :

- висока затримка: HLS не підтримує таку низьку затримку, як деякі інші бажані протоколи;
- не підходить для прийому: HLS не найкращий варіант для прийому, оскільки HLS-сумісні кодери недоступні або недорогі.

Протокол обміну повідомленнями в реальному часі (RTMP). Спочатку розроблений Macromedia на початку потокової передачі, протокол RTMP досі широко використовується.

Сьогодні RTMP в основному використовується для отримання прямих трансляцій за допомогою кодера з підтримкою RTMP . RTMP рідко використовується як протокол потокового відео для глядача, як це було колись. Це тому, що це залежить від плагіна Flash, який зараз повністю застарів.

Плюси використання RTMP:

- Низька затримка. Низька затримка дозволяє вашому прямому відеопотоку підтримувати стабільне з'єднання та відеоканал для глядача, навіть якщо інтернет-з'єднання ненадійне. Це дає вашим глядачам менше «затримок» під час перегляду ваших відео з хитким інтернет-з'єднанням, дозволяючи їм швидко відновити потік, як тільки їх інтернет-з'єднання стабілізується.

- Адаптивний: адаптований канал означає, що ваші глядачі не можуть дивитися ваші стрічки в одному лінійному напрямку. З вмістом, розміщеним на сервері RTMP, канал дозволяє їм пропускати та перемотувати частини стрічки назад або приєднуватися до прямої трансляції після її початку.

- Гнучкість: RTMP дозволяє інтегрувати різноманітні відеоформати в один цілісний пакет, безперешкодно поєднуючи аудіо, відео та текст. Крім того, ви можете мати кілька варіантів медіа-каналів, таких як потокове аудіопотоки MP3 і AAC або потокове відео MP4, FLV і F4V.

Мінуси використання RTMP:

- Не підтримується HTML5: RTMP підтримується програвачами Flash, форматом, який вже давно застарів. Програвачі HTML5 швидко стають сучасним стандартом, але RTMP не може відтворюватися на програвачах HTML5 без такого конвертера, як HLS.

- Проблеми з пропускною здатністю: потоки RTMP можуть бути особливо вразливими до проблем із низькою пропускною здатністю. Це може спричинити часті неприємні переривання ваших потоків, які зіпсують враження для ваших глядачів.

- HTTP несумісний: не можливо безпосередньо передавати RTMP-канал через з'єднання HTTP. Щоб використовувати потік RTMP на своєму веб-сайті,

потрібно підключитися до спеціального сервера, наприклад Flash Media Server, і використовувати сторонню мережу доставки вмісту (CDN).

Web Real-Time Communications (WebRTC) – це відеопроект з відкритим вихідним кодом, який може транслювати потоки із затримкою в реальному часі. Цей проект був розроблений для підтримки VoIP і був придбаний Google для підтримки інструментів відеочату Google. WebRTC технічно є потоковим проектом, а не потоковим протоколом. Однак його часто об’єднують із бажаними протоколами, оскільки є багато збігів.

Плюси використання WebRTC:

- відкритий вихідний код: оскільки WebRTC є відкритим вихідним кодом, його можна налаштувати відповідно до ваших конкретних потреб;
- затримка в режимі реального часу: WebRTC підтримує потокове передавання з затримкою в режимі реального часу, що означає, що ваше відео передається на екрани ваших глядачів практично в режимі реального часу.

Мінуси використання WebRTC:

- трохи футуристично: технологія потокового передавання ще не повністю наздогнала WebRTC, тому є деякі проблеми з сумісністю налаштування потокової передачі

Secure Reliable Transport (SRT) — це відносно новий протокол потокової передачі від Haivision, лідера в індустрії потокового онлайн-трансляції. Цей протокол з відкритим кодом відомий своєю чудовою безпекою, надійністю, сумісністю та можливістю потокової передачі з низькою затримкою.

Наразі існують певні обмеження на потокове передавання за допомогою SRT, оскільки інше потокове обладнання та програмне забезпечення ще не розроблено для підтримки цього протоколу.

Плюси використання SRT:

- безпека: SRT включає найсучасніші інструменти безпеки та конфіденційності, щоб мовники могли бути впевнені, що їхні потоки залишаються безпечними та надійними;

- сумісний: SRT не залежить від пристроїв і операційної системи, що означає, що він може доставляти потоки на більшість пристроїв з підтримкою Інтернету;

- низька затримка: потокове передавання з низькою затримкою є головною перевагою для професійних мовників. SRT забезпечує низьку затримку потокового передавання за допомогою технології корекції помилок.

Мінуси використання SRT:

- поки що не підтримується широко: як і WebRTC, SRT все ще трохи футуристичний. Індустрії потокової передачі потрібно буде наздогнати, перш ніж цей протокол стане стандартом.

Протокол потокової передачі в реальному часі (RTSP)

Можливо, менш відомий протокол потокового відео – Real-Time Streaming Protocol (RTSP) був вперше опублікований у 1998 році. RTSP був розроблений для керування потоковими медіа-серверами в розважальних і комунікаційних системах, зокрема. У 2016 році стала доступна оновлена версія RTSP 2.0. Загалом, він відомий як протокол потокового відео для встановлення та керування медіа-сеансами між кінцевими точками. RTSP в чомусь схожий на протокол HTTP Live Streaming (HLS), який ми розглянемо нижче. Однак передача даних у прямому ефірі – це не те, що RTSP виконує самостійно. Натомість сервери RTSP часто працюють у поєднанні з транспортним протоколом реального часу (RTP) і протоколом керування реальним часом (RTCP) для доставки медіа-потоків.

Плюси використання RTSP:

- сегментована потокова передача: замість того, щоб змушувати ваших глядачів завантажувати ціле відео перед його переглядом, потік RTSP дозволяє їм переглядати ваш вміст до завершення завантаження;

- налаштування. Використовуючи інші протоколи, такі як протокол керування передачею (TCP) і протокол дейтаграм користувача (UDP), ви можете створювати власні програми для потокового відео.

Мінуси використання RTSP:

- менш популярний: у порівнянні з іншими протоколами потокової передачі медіа , RTSP набагато менш популярний. Більшість відеопрогравачів і потокових служб не підтримують потокове передавання RTSP, що ускладнює трансляцію потоку у вашому браузері. Щоб транслювати потік RTSP, ви повинні використовувати окремий сервіс потокової трансляції RTSP.

- HTTP несумісний: як і RTMP, ви не можете безпосередньо передавати RTSP через HTTP. Через це не існує простого і зрозумілого способу потокової передачі RTSP у веб-браузері, оскільки RTSP призначений більше для потокового відео в приватних мережах, таких як системи безпеки в компанії.

Динамічне адаптивне потокове передавання через HTTP (MPEG-DASH). Хоча воно ще не широко використовується, цей протокол має деякі великі переваги. По-перше, він підтримує трансляцію з адаптивним бітрейтом . Це означає, що глядачі завжди отримають найкращу якість відео, яку може підтримувати їх поточна швидкість інтернет-з'єднання. Це має тенденцію коливатися від секунди до секунди, і DASH може не відставати.

MPEG-DASH усуває деякі давні технічні проблеми з доставкою та стисненням. Ще одна перевага полягає в тому, що MPEG-DASH є «незалежним від кодеків», тобто його можна використовувати майже з будь-яким форматом потокового кодування . Він також підтримує Encrypted Media Extensions (EME) і Media Source Extension (MSE), які є заснованими на стандартах API для керування цифровими правами на основі браузера (DRM).

Плюси використання MPEG-DASH:

- адаптивний: актуальним DASH є його підтримка адаптивної потокової передачі бітрейту, яка чудово підходить для доставки високоякісних потоків користувачам з різною швидкістю Інтернету;

- відкритий вихідний код: MPEG-DASH є відкритим вихідним кодом і не має постачальників, що означає, що користувачі можуть налаштувати його відповідно до своїх конкретних потреб.

Мінуси використання MPEG-DASH:

- обмежена підтримка: MPEG-DASH не сумісний з пристроями Apple/iOS, що може бути досить проблематичним для мовників;

- безмайбутнє: хоча колись були надії на майбутнє, де DASH був переважним протоколом, шанси на це стають все стрункішими.

Хоча протоколи потокової передачі та пов’язані з ними технології є дещо складними, вони цілком доступні, якщо їх розбити на менші, більш зрозумілі ідеї.

1.3 Програмні засоби для виділення та класифікації зображень

Одним з найважливіших застосувань комп’ютерного зору є виділення та розпізнавання обличчя. Для багатьох інтернет-компаній, таких як Facebook і Google, виділення та розпізнавання облич є одним із найважливіших модулів у системах штучного інтелекту. Розпізнавання облич також має значний вплив для компаній, що працюють у системах біометричної ідентифікації. Вони знаходять застосування в таких секторах, як банківська справа, охорона здоров’я, туризм та роздрібної торгівлі.

Google Cloud Vision. Платформи розпізнавання облич, налаштовані на хмарні системи, мають величезну перевагу. Якщо програма працює на хмарному ядрі Google, інтегрувати Google Cloud Vision у продукт або програму буде дуже легко. Проект може похвалитися багатьма попередньо навченими моделями та функціями API, щоб стати потужним інструментом для багатьох програмістів комп’ютерного зору. Існує додаткова перевага доступності AutoML Vision, яка може допомогти програмам навчати моделі бачення клієнтів. Google Cloud Vision здатний розпізнавати кілька облич на зображенні разом із такими основними атрибутами, як емоційний стан та використання пов’язки на голову людиною. Також повідомляється, що Google працює над додаванням функції розпізнавання обличчя до набору інструментів.

IBM Watson Visual Recognition — це дуже потужна і надійна програма промислового масштабу, яку можна використовувати в багатьох програмах комп'ютерного зору. Вона має можливість точно та швидко позначати, класифікувати та тренувати дані/набори даних бачення за допомогою машинного навчання. На початку через деякі події IBM відключила функції, які могли б допомогти програмістам ідентифікувати ім'я та тип людини, наданої зображенням, але двигун все ще залишається дуже потужним місцем для навчання найсучасніших програм розпізнавання та виявлення обличчя. Наразі підтримуються формати зображень .gif, .jpg, .png і .tif з максимальним розміром 10 МБ із рекомендованою щільністю пікселів 32 x 32 на дюйм.

API Microsoft Face. Microsoft може похвалитися найкращою командою AI з експертами з комп'ютерного зору, які створили потужні інтелектуальні додатки, орієнтовані на хмару. Microsoft Face API допомагає розробникам виявляти схожі обличчя та порівнювати їх, організовувати зображення облич у схожі групи для групування їх. Face API також дає розробникам можливість ідентифікувати раніше позначених людей на зображеннях, що може привести до потужних програм безпеки та Інтернету. API Face від Microsoft надає перевірку обличчя як послугу, за допомогою якої можна перевірити ймовірність того, що два різні обличчя є однією людиною, і повернути оцінку. Служба розпізнавання облич від API має можливість виявляти одне або кілька людських облич на зображенні та отримувати для обличчя прямокутник із 27 орієнтирами для одного обличчя.Додаткові функції Microsoft Face API включають розпізнавання емоцій для облич, які можуть виявляти емоції, наприкладзлість, презирство, огида, страх, щастя серед іншого.

Amazon Rekognition за підтримки хмарних сервісів AWS – ще один гігант у сфері розпізнавання обличчя. Amazon Rekognition – це найпростіший спосіб додати функції, пов'язані з обробкою зображень або відео, до вашої програми, особливо якщо ви запускаєте програму в хмарі AWS. Служба може ідентифікувати обличчя, людей та діяльність, серед багатьох інших речей, коли надано вміст зображення. Amazon Rekognition є дуже популярним для його осіб

аналітична служба. Він має досить точний механізм розпізнавання облич, який також працює з багатьма видами зображень і відео. Розробник може використовувати цю службу для створення функцій додатків, які можуть перевіряти користувачів, підраховувати кількість людей і створювати програми для громадської безпеки. AWS, як найбільший хмарний сервіс у світі, використовує величезні обсяги даних для покращення своїх моделей машинного навчання. Послуга має додаткові переваги, як-от можливість запускати процеси в пакетному режимі та в режимі реального часу.

Kairos – компанія, що спеціалізується на розпізнаванні та розпізнаванні облич. На відміну від інших API, які представляють собою набір інструментів і сервісів, об'єднаних в один, Kairos надає виключно рішення для розпізнавання облич і, отже, є одним із найкращих рішень Face AI у світі. Багато клієнтів вважають за краще розміщувати Kairos API на власних серверах і, отже, бути в безпеці та контролювати свої дані. Ця функція також дає перевагу Кайросу над конкурентами. Kairos має незліченну кількість функцій аналізу облич, таких як розпізнавання, ідентифікація та перевірка обличчя. Інші послуги включають виявлення емоцій та розпізнавання рис обличчя, які можуть виявляти певні частини, такі як очі, ніс та інші.

ImageAI. Бібліотека ImageAI має на меті надати розробникам безліч алгоритмів комп'ютерного зору та методологій глибокого навчання для виконання завдань, пов'язаних із виявленням об'єктів та обробкою зображень. Основна мета бібліотеки ImageAI – забезпечити ефективний підхід до кодування проектів виявлення об'єктів за допомогою кількох рядків коду.

Більшість доступних блоків коду написані за допомогою мови програмування Python разом із популярною структурою глибокого навчання Tensorflow. Станом на червень 2021 року ця бібліотека використовує бекенд PyTorch для обчислення завдань обробки зображень.

Бібліотека ImageAI підтримує безліч операцій, пов'язаних з виявленням об'єктів, а саме розпізнавання зображень, виявлення об'єктів зображення, виявлення відеооб'єктів, аналіз виявлення відео, навчання та висновки з

розпізнавання користувацьких зображень, а також навчання та визначення користувацьких об’єктів. Функція розпізнавання зображень може розпізнавати до 1000 різних об’єктів на окремому зображенні.

Завдання виявлення зображень і відео об’єктів допоможе виявити 80 найпоширеніших об’єктів, які можна побачити в повсякденному житті. Аналіз виявлення відео допоможе обчислити своєчасний аналіз будь-якого конкретного об’єкта, який виявляється на відео або в режимі реального часу. У цій бібліотеці також можна ввести власні зображення для навчання власних зразків. Ви можете навчити набагато більше об’єктів для завдання виявлення об’єктів за допомогою новіших зображень і наборів даних.

GluonCV є однією з кращих бібліотечних структур з більшістю реалізацій впроваджених для глибоких алгоритмів навчання для різних областей застосування комп'ютерного зору. Основна мета цієї бібліотеки – допомогти ентузіастам цієї галузі досягти продуктивних результатів за короткий проміжок часу. Вона містить одні з найкращих функцій із великим набором навчальних наборів даних, методів впровадження та ретельно розроблених API.

Фреймворк бібліотеки GluonCV підтримує велику кількість завдань, які можна виконати з нею. Ці проекти включають завдання класифікації зображень, задачі виявлення об’єктів у зображенні, відео чи в режимі реального часу, семантичну сегментацію та сегментацію екземплярів, оцінку пози для визначення пози конкретного тіла та розпізнавання дій для виявлення типу людської діяльності, що виконується. Ці функції роблять цю бібліотеку однією з найкращих бібліотек виявлення об’єктів для досягнення швидших результатів.

Ця структура забезпечує всі найсучасніші методики, необхідні для виконання вищезгаданих завдань. Він підтримує як MXNet, так і PyTorch і має широкий набір навчальних посібників і додаткової підтримки, з якої ви можете почати вивчати численні концепції. Він містить велику кількість навчальних моделей, з яких ви можете досліджувати та створювати конкретну модель машинного навчання за вашим вибором для виконання конкретного завдання.

Detectron2. Фреймворк Detectron2, розроблений дослідницькою командою Facebook (FAIR), вважається бібліотекою нового покоління, яка підтримує більшість найсучасніших методів виявлення, методів виявлення об’єктів та алгоритмів сегментації. Бібліотека Detectron2 — це фреймворк для виявлення об’єктів на основі PyTorch. Бібліотека дуже гнучка і розширювана, надаючи користувачам безліч високоякісних алгоритмів і методів реалізації. Вона також підтримує численні програми та виробничі проекти на Facebook.

Бібліотека Detectron2, розроблена на PyTorch компанією FaceBook, має величезні додатки, і її можна навчати на одному або кількох графічних процесорах для отримання швидких та ефективних результатів. За допомогою цієї бібліотеки ви можете реалізувати кілька високоякісних алгоритмів виявлення об’єктів для досягнення найкращих результатів. Ці найсучасніші технології та алгоритми виявлення об’єктів, які підтримує бібліотека, включають

DensePose, пірамідні мережі з паноптичними функціями та численні інші варіанти сімейства моделей Mask R-CNN.

Бібліотека Detectron2 також дозволяє користувачам легко навчати власні моделі та набори даних. Процедура встановлення нижче досить проста. Єдині залежності, які вам потрібні для наступного, це PyTorch і COCO API. Якщо ви отримаєте наступні вимоги, ви можете переходити до встановлення моделі Detectron2 і легко навчати безліч моделей.

YOLOv3_TensorFlow. Модель YOLO v3 є однією з успішних реалізацій серії YOLO, яка була випущена в 2018 році. Третя версія YOLO покращує попередні моделі. Продуктивність цієї моделі краща, ніж у попередників, як за швидкістю, так і за точністю. На відміну від інших архітектур, він також може гідно працювати на менших об’єктах з хорошою точністю. Єдина основна проблема в порівнянні з іншими основними алгоритмами - це компроміс між швидкістю і точністю.

Бібліотека YOLOv3_TensorFlow є однією з найперших реалізацій архітектури YOLO для обробки та обчислень виявлення об’єктів. Вона забезпечує надзвичайно швидкі обчислення GPU, ефективні результати та

конвеєри даних, перетворення ваги, прискорений час навчання та багато іншого. Хоча бібліотеку можна отримати за посиланням, наведеним у наступному розділі, підтримка для цього фреймворка припинена (як і для більшості інших), і тепер замість цього підтримується PyTorch.

Darkflow натхненний фреймворком darknet і в основному є перекладом, який відповідає мові програмування Python і TensorFlow, щоб зробити його доступним для більш широкого кола аудиторій. Darknet – це рання реалізація бібліотеки виявлення об'єктів із C і CUDA. Процедури встановлення та роботи цієї бібліотеки досить прості та легкі у виконанні. Фреймворк також підтримує обчислення як CPU, так і GPU завдань виявлення об'єктів для досягнення найкращих результатів у будь-якому сценарії.

Структура темного потоку вимагає деяких основних потреб для її реалізації. Деякі з цих основних вимог: Python3, TensorFlow, Numpy і Opencv. З цими залежностями ви можете легко почати обчислювальні завдання, пов'язані з виявленням об'єктів. За допомогою бібліотеки dark flow можна виконувати багато завдань. Framework Dark flow має доступ до моделей YOLO, і є можливість завантажити власні ваги для різних моделей.

Деякі із завдань, які допомагає вам виконати бібліотека darkflow, включають аналіз анотацій, проектування мережі відповідно до певної конфігурації, побудову графіків із потоком, навчання нової моделі, навчання на спеціальному наборі даних, створення файлу в режимі реального часу або відео. , використовуючи фреймворк Darkflow для інших подібних програм, і, нарешті, він також дозволяє зберігати ці моделі у форматі protobuf (.pb).

На сьогоднішній день виявлення об'єктів залишається одним із найважливіших програм глибокого навчання та комп'ютерного зору. Існує багато вдосконалень алгоритмів і вдосконалень у методології виявлення об'єктів.

Почалося з таких алгоритмів, як гістограма орієнтованих градієнтів, представлена ще в 1986 році для виділення простих об'єктів на зображеннях із

пристойною точністю. Тепер же існують сучасні архітектури, такі як Faster R-CNN, Mask R-CNN, YOLO та RetinaNet.

Обмеження для виявлення об’єктів не обмежуються зображеннями, оскільки їх можна ефективно виконувати на відео та відео в реальному часі з високою точністю. У майбутньому на нас чекає набагато більше успішних алгоритмів і бібліотек для виявлення об’єктів.

1.4 Постановка задач дослідження

Під час проведених досліджень в даному розділі було наведено результати аналізу задач комп’ютерного зору, приклади та сфери їх використання. Досліджено принципи створення, кодування та передачі сигладів за допомогою відеопотоків, а також виділено їх особливі характеристики. Досліджено програмні засоби для виділення та класифікації обєктів на цифрових зображеннях у відеопотоках.

Для досягнення поставленої мети необхідно розв’язати наступні задачі.

- провести аналіз та класифікацію завдань комп’ютерного зору;
- проаналізувати методи кодування та передачі сигналів у відеопотоках;
- провести аналітичний огляд наявних програмних засобів для виділеня та класифікації об’єктів на зображеннях;
- проаналізувати існуючі методи та підходи до виділення та класифікації об’єктів на зображеннях;
- розробити алгоритм детекції обєктів на зображенні на основі ковзаючого вікна;
- реалізувати програмну систему детекції обєктів на зображенні на основі ковзаючого вікна, провести її тестування та порівняти з подібними програмами.

1.5 Висновки до розділу

Проведено аналіз та класифікацію завдань комп’ютерного зору, що надало можливість окреслити основні завдання та напрямки роботи алгоритмів комп’ютерного зору при обробці цифрових зображень.

Проаналізовано методи кодування та передачі сигналів у відеопотоках, що дозволило визначити оптимальні алгоритми кодування для створення прорамної системи з елементами детекції руху.

Проведено аналіз наявних програмних засобів для виділеня та класифікації об’єктів на зображеннях який дозволив виділити основні структурні модулі, та встановити інтерфейси обміну даними між окремими модулями.

2 МЕТОДИ ТА АЛГОРИТМИ ВИДІЛЕННЯ ТА КЛАСИФІКАЦІЇ ОБ’ЄКТІВ НА ЦИФРОВИХ ЗОБРАЖЕННЯХ

2.1 Методи та алгоритми виділення об’єктів на зображенні

Методи виділення об'єктів і класифікація зображень є важливими методами, коли мова йде про роботу в області застосування комп'ютерного зору (рисунок 2.1). Ці методи допомагають машинам розуміти та ідентифікувати об’єкти та середовища в реальному часі за допомогою цифрових зображень як вхідних даних. Протягом багатьох років методи комп’ютерного зору використовувалися в кількох сферах, включаючи охорону здоров’я, виробництво, роздрібну торгівлю, і це лише деякі з них.

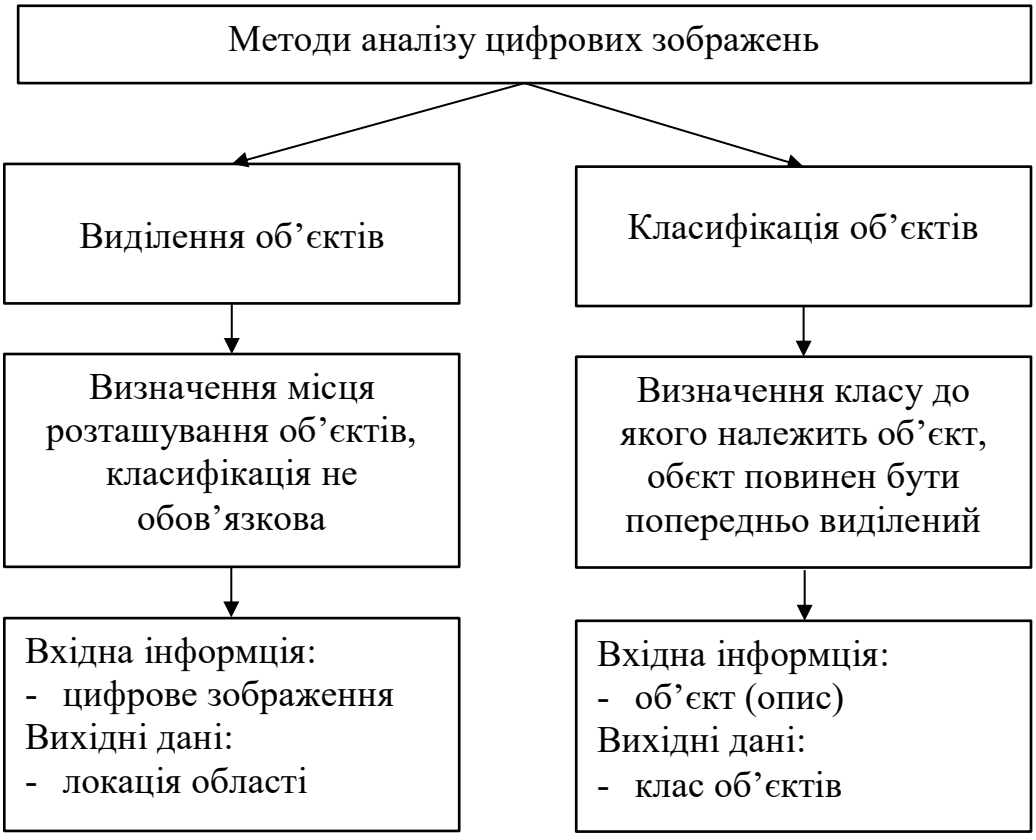

Рисунок 2.1 – Класифікація алгоритмів виділення (детекції) та класифікації об’єктів на зображенні

Оскільки такі методи, як класифікація зображень та виявлення об’єктів, пов’язані з ідентифікацією об’єктів у цифрових зображеннях, часто виникає плутанина: що насправді є цими двома методами та чим ці дві методики відрізняються один від одного.

Класифікація зображень – це техніка, яка використовується для класифікації або прогнозування класу конкретного об’єкта на зображенні. Основна мета цієї техніки – точно визначити особливості зображення.

Загалом, методи класифікації зображень можна розділити на параметричні та непараметричні або контрольовані та неконтрольовані, а також жорсткі та м’які класифікатори. Для контрольованої класифікації ця методика дає результати на основі створеної межі рішення, яка в основному покладається на вхідні та вихідні дані, надані під час навчання моделі. Але, у випадку неконтрольованої класифікації, методика дає результат на основі власного аналізу вхідного набору даних; функції не подаються безпосередньо в моделі.

Основні кроки, що пов’язані з методами класифікації зображень – це визначення відповідної системи класифікації, виділення ознак, вибір хороших навчальних зразків, попередня обробка зображення та вибір відповідного методу класифікації, обробка після класифікації та, нарешті, оцінка загальної точності. У цій техніці вхідні дані зазвичай є зображенням певного об’єкта, а вихідні дані – це передбачені класи, які визначають і відповідають вхідним об’єктам. Згорткові нейронні мережі (CNN) є найпопулярнішою моделлю нейронної мережі, яка використовується для задачі класифікації зображень.

В повсякденному житті користувачі постійно зустрічаються з технологіями які включають в себе елементи виявлення/детекції об’єктів. Наприклад, смартфон можна розблокувати за допомогою розпізнавання обличчя. Або під час відеоспостереження магазинів чи складів системи визначають підозрілі дії.

Основні напрямки застосувань методів та алгоритмів виявлення об'єктів наведено на рисуноку 2.2).

Рисунок 2.2 – Сфери застосування алгоритмів виділення об’єктів

Розпізнавання номерних знаків – використання технології виявлення об’єктів і оптичного розпізнавання символів (OCR) для розпізнавання буквено-цифрових символів на автомобілі. Користувач можете використовувати виявлення об’єктів для отримання зображень і виявлення транспортних засобів на певному зображенні. Після того, як модель розпізнає номерний знак, технологія OCR працює над перетворенням двовимірних даних у машинно-кодований текст.

Виявлення та розпізнавання облич – як обговорювалося раніше, одним з основних застосувань виявлення об’єктів є виявлення та розпізнавання облич. За допомогою сучасних алгоритмів користувачі можуть виявляти людські обличчя на зображенні або відео. Тепер можна навіть розпізнавати обличчя лише за одним навченим зображенням завдяки одноразовим методам навчання.

Відстеження об’єктів – під час перегляду гри в футбол або баскетбол м’яч активно перміщується, що інколи викликає певні проблеми для відеотрансляції. У цих ситуаціях добре відстежувати рух м’яча разом із відстанню, яку він долає. Для цього відстеження об’єктів може гарантувати, що маємо безперервну інформацію про напрямок руху м’яча.

Автомобілі з автономним керуванням – для автономних автомобілів дуже важливо вивчати різні елементи навколо автомобіля під час руху. Модель виявлення об’єктів, навчена на кількох класах для розпізнавання різних об’єктів,

стає життєво важливою для хорошої продуктивності автономних транспортних засобів.

Робототехніка – багато завдань, як-от підняття важких вантажів, операції з підбору та розміщення та інші роботи в режимі реального часу, виконуються роботами. Виявлення об’єктів є важливим для роботів, щоб виявляти речі та автоматизувати завдання.

Визначення проблеми виявлення об’єктів полягає в тому, щоб визначити, де розташовані об’єкти на даному зображенні, наприклад, локалізація об’єкта, і до якої категорії належить кожен об’єкт, тобто класифікація об’єктів. Простіше кажучи, виявлення об’єктів є різновидом техніки класифікації зображень, і крім класифікації, ця методика також визначає розташування екземплярів об’єктів з великої кількості попередньо визначених категорій у природних зображеннях.

Ця техніка має можливість шукати певний клас об’єктів, таких як автомобілі, люди, тварини, птахи тощо, і успішно використовується в зображеннях наступного покоління, а також у системах обробки відео. Останні досягнення в цій техніці стали можливими лише з появою методологій глибокого навчання.

Методи виявлення об’єктів можна використовувати в реальних проектах, таких як виявлення обличчя, виявлення пішоходів, виявлення транспортних засобів, виявлення дорожніх знаків, відеоспостереження тощо

Конвеєр традиційних моделей виявлення об'єктів можна в основному розділити на три етапи: вибір інформативної області, виділення ознак і класифікація. Існує кілька популярних моделей для виявлення об’єктів на основі глибокого навчання, які використовуються організаціями та науковими колами для досягнення ефективності, а також точних результатів у виявленні об’єктів із зображень. Серед популярних моделей:

- швидкий R-CNN;
- MobileNet;
- гістограма орієнтованих градієнтів (HOG);
- You Only Live Once (YOLO);

- Mark-RCNN;
- RetinaNet та інші.

Виявлення об’єктів стало свідком швидких революційних змін у сфері комп’ютерного зору. Його участь у поєднанні класифікації об’єктів, а також локалізації об’єктів робить його однією з найскладніших тем у сфері комп’ютерного зору.

Швидкий R-CNN. Написаний на Python і C++ (Caffe), метод Fast Region-Based Convolutional Network або Fast R-CNN є навчальним алгоритмом для виявлення об’єктів. Цей алгоритм в основному усуває недоліки R-CNN і SPPnet, одночасно покращуючи їх швидкість і точність.

Переваги швидкого R-CNN:

- вища якість виявлення (mAP), ніж R-CNN, SPPnet;
- навчання одноетапне, з використанням багатозадачної втрати;
- навчання може оновлювати всі рівні мережі;
- для кешування функцій не потрібне дискове сховище.

Швидший R-CNN. Швидший R-CNN – це алгоритм виявлення об’єктів, схожий на R-CNN. Цей алгоритм використовує мережу пропозицій регіону (RPN), яка спільне з мережею виявлення спільних згорткових функцій повного зображення економно, ніж R-CNN і Fast R-CNN. Мережа пропозицій регіону – це, по суті, повністю згортка мережа, яка одночасно прогнозує межі об’єкта, а також оцінки об’єктності в кожній позиції об’єкта та наскрізь навчається для створення високоякісних пропозицій регіонів, які потім використовуються Fast R. -CNN для виявлення об'єктів.

Гістограма орієнтованих градієнтів (HOG) – це в основному дескриптор ознак, який використовується для виявлення об’єктів під час обробки зображень та інших методів комп’ютерного зору. Техніка дескриптора гістограми орієнтованих градієнтів включає випадки орієнтації градієнта в локалізованих частинах зображення, таких як вікно виявлення, область інтересу (ROI), серед іншого. Однією з переваг HOG-подібних функцій є їхня простота і легше зрозуміти інформацію, яку вони несуть.

Метод згорткової мережі на основі регіонів (RCNN) — це комбінація пропозицій регіону з нейронними мережами згортки (CNN). R-CNN допомагає локалізувати об’єкти з глибокою мережею та навчати моделі високої ємності лише з невеликою кількістю анотованих даних виявлення. Він забезпечує чудову точність визначення об’єктів, використовуючи глибоку мережу ConvNet для класифікації пропозицій об’єктів. R-CNN має можливість масштабуватися до тисяч класів об’єктів, не вдаючись до приблизних методів, включаючи хешування.

Повністю згорткові мережі на основі регіонів або R-FCN — це детектор на основі регіону для виявлення об’єктів. На відміну від інших регіональних детекторів, які використовують дорогу підмережу для кожного регіону, наприклад Fast R-CNN або Faster R-CNN, цей детектор на основі регіону є повністю згортковим, і майже всі обчислення використовуються для всього зображення.

R-FCN складається із спільних, повністю згорткових архітектур, як у випадку FCN, яка, як відомо, дає кращий результат, ніж Faster R-CNN. У цьому алгоритмі всі вагові шари, які можна вивчати, є згортковими і призначені для класифікації ROI за категоріями об’єктів і фонами.

Single Shot Detector (SSD) — це метод виявлення об’єктів на зображеннях за допомогою однієї глибокої нейронної мережі. Підхід SSD дискретує вихідний простір обмежувальних рамок на набір блоків за замовчуванням з різними співвідношеннями сторін. Після дискретизації метод масштабується відповідно до розташування карти об’єктів. Мережа Single Shot Detector поєднує прогнози з кількох карт об’єктів з різною роздільною здатністю, щоб природно обробляти об’єкти різного розміру.

- SSD повністю виключає генерацію пропозицій і наступні етапи повторної вибірки пікселів або функцій і інкапсулює всі обчислення в одній мережі.

- Легко навчати і легко інтегрувати в системи, які потребують компонента виявлення.

- SSD має конкурентоспроможну точність щодо методів, які використовують додатковий крок пропозиції об’єкта, і він набагато швидший, забезпечуючи уніфіковану структуру як для навчання, так і для висновку.

Об’єднання просторових пірамід (SPP-net) — це мережева структура, яка може генерувати представлення фіксованої довжини незалежно від розміру/масштабу зображення. Вважається, що об’єднання пірамід є стійким до деформацій об’єктів, а SPP-net покращує всі методи класифікації зображень на основі CNN. Використовуючи SPP-net, дослідники можуть обчислити карти ознак із всього зображення лише один раз, а потім об’єднати ознаки в довільних областях (підзображення), щоб створити представлення фіксованої довжини для навчання детекторів. Цей метод дозволяє уникнути багаторазового обчислення згорткових ознак.

You Only Look Once або YOLO є одним із популярних алгоритмів виявлення об’єктів, які використовуються дослідниками по всьому світу. За словами дослідників з Facebook AI Research, уніфікована архітектура YOLO надзвичайно швидка. Базова модель YOLO обробляє зображення в режимі реального часу зі швидкістю 45 кадрів в секунду, в той час як менша версія мережі, Fast YOLO обробляє приголомшливі 155 кадрів в секунду, при цьому досягаючи вдвічі кращого mAP, ніж інші детектори реального часу. Цей алгоритм перевершує інші методи виявлення, включаючи DPM і R-CNN, при узагальненні від природних зображень до інших доменів, таких як ілюстрація.

За останні кілька років було досягнуто великого успіху в контрольованому середовищі для проблеми виявлення об'єктів. Проте проблема залишається невирішеною в неконтрольованих місцях, зокрема, коли об’єкти поміщають у довільні позі в захаращеному та закритому середовищі. Проте на зважаючи на наведені недоліки на сьогоднішній день технології які базуються на використанні нейронних мереж широко використовуються в різних галузя. Це можна пояснити швидким рівнем автоматизації процесів та здешевленням апаратних засобів для реалізації програмно-апаратних комплексів. А отже, і в подальшому буде спостерігатись активні розробки в даному напрямку.

2.2 Алгоритми класифікації об’єктів на зображенні

Класифікація та розпізнавання зображень – це те, що дає змогу досягти багатьох найбільш вражаючих досягнень технологій з елементами штучного інтелекту.

Методики класифікації зображень під наглядом включають техніку паралелепіпеда, класифікатор мінімальної відстані, класифікатор максимальної імовірності та інші. На сьогодні можна виділити декілька типів методів класифікації зображень які зазначено нижче:

- класифікація зображень на основі інформації, отриманої від різних датчиків;
- класифікація зображень на основі характеру навчальної вибірки, яка використовується в класифікації;
- класифікація зображень на основі різних параметрів даних;
- класифікація зображень на основі природи піксельної інформації, яка використовується в даних;
- класифікація зображення на основі кількості вихідних даних, створених для кожного елемента просторових даних;
- класифікація зображень на основі природи просторової інформації.

Методи класифікації зображень в основному можна розділити на дві різні категорії (рисунок 2.3).

Рисунок 2.3 – Групи алгоритмів класифікації об’єктів

Пікселі є базовими одиницями зображення, а аналіз пікселів є основним способом класифікації зображень. Однак алгоритми класифікації можуть використовувати лише спектральну інформацію в окремих пікселях для класифікації зображення або досліджувати просторову інформацію (близькі пікселі) разом зі спектральною інформацією. Методи класифікації на основі пікселів використовують лише спектральну інформацію (інтенсивність пікселя), тоді як методи класифікації на основі об'єктів враховують як спектральну інформацію пікселя, так і просторову інформацію.

Існують різні методи класифікації, які використовуються для класифікації на основі пікселів. До них належать мінімальна відстань до середнього, максимальна ймовірність та мінімальна відстань Махаланобіса. Ці методи вимагають, щоб середні значення та дисперсії класів були відомі, і всі вони працюють, досліджуючи «відстань» між середніми класами та цільовими пікселями.

Методи класифікації на основі пікселів обмежені тим фактом, що вони не можуть використовувати інформацію з інших сусідніх пікселів. На відміну від цього, методи класифікації на основі об'єктів можуть включати інші пікселі, і тому вони також використовують просторову інформацію для класифікації елементів. Зауважимо, що «об'єкт» стосується лише суміжних областей пікселів, а не того, чи є цільовий об'єкт у цій області пікселів чи ні.

Попередня обробка даних зображення для виявлення об'єктів. Найновіші та надійні системи класифікації зображень переважно використовують схеми класифікації на рівні об'єктів, і для цих підходів дані зображення мають бути підготовлені особливим чином. Об'єкти/регіони потрібно вибрати та попередньо обробити.

Перш ніж зображення та об'єкти/області в цьому зображенні можуть бути класифіковані, дані, які містять це зображення, мають бути інтерпретовані комп'ютером. Зображення необхідно попередньо обробити та підготувати для введення в алгоритм класифікації, і це робиться за допомогою виявлення

об’єктів. Це важлива частина підготовки даних і зображень для навчання класифікатора машинного навчання .

Виявлення об’єктів здійснюється за допомогою різноманітних методів і прийомів. Почнемо з того, чи є кілька об’єктів, що цікавлять, чи один об’єкт, що цікавить, впливає на те, як виконується попередня обробка зображення. Якщо є лише один цікавий об’єкт, зображення піддається локалізації зображення. Пікселі, що містять зображення, мають числові значення, які інтерпретуються комп’ютером і використовуються для відображення правильних кольорів і відтінків. Навколо об’єкта, який цікавить, малюється об’єкт, відомий як обмежувальна рамка, яка допомагає комп’ютеру знати, яка частина зображення є важливою і які значення пікселів визначають об’єкт. Якщо на зображенні є кілька об’єктів, що представляють інтерес, використовується техніка, яка називається виявленням об’єктів, щоб застосувати ці обмежувальні рамки до всіх об’єктів на зображенні.

Іншим методом попередньої обробки є сегментація зображення. Сегментація зображення функціонує шляхом поділу всього зображення на сегменти на основі подібних ознак. Різні області зображення матимуть подібні значення пікселів у порівнянні з іншими областями зображення, тому ці пікселі групуються разом у маски зображення, які відповідають формі та межам відповідних об’єктів у зображенні. Сегментація зображень допомагає комп’ютеру виділити ознаки зображення, які допоможуть йому класифікувати об’єкт, як це роблять обмежувальні рамки, але вони забезпечують набагато точніші мітки на рівні пікселів.

Після завершення виявлення об’єкта або сегментації зображення мітки наносяться на відповідні області. Ці мітки разом із значеннями пікселів, що містять об’єкт, подаються в алгоритми машинного навчання, які вивчатимуть шаблони, пов’язані з різними мітками.

Алгоритми машинного навчання. Після того, як дані були підготовлені та позначені, дані подаються в алгоритм машинного навчання, який тренується на

даних. Нижче наведено деякі з найпоширеніших видів алгоритмів класифікації зображень машинного навчання.

K-Nearest Neighbours – це алгоритм класифікації, який досліджує найближчі навчальні приклади та розглядає їх мітки, щоб визначити найбільш імовірну мітку для даного тестового прикладу. Коли справа доходить до класифікації зображень за допомогою KNN, вектори ознак і мітки навчальних зображень зберігаються, і лише вектор ознак передається в алгоритм під час тестування. Потім вектори ознак навчання та тестування порівнюються один з одним для подібності.

K-Nearest Neighbours (KNN) — це концептуально простий, але дуже потужний алгоритм, і з цих причин це один з найпопулярніших алгоритмів машинного навчання.

Для ілюстрації роботи досліджуваного алгоритму візуалізуємо набір даних на 2*D* площині (рисунок 2.4). Відображимо на графіку купу точок даних, розподілених уздовж графіка невеликими кластерами. KNN досліджує розподіл точок даних і, залежно від аргументів, наданих моделі, розділяє точки даних на групи. Потім цим групам присвоюється мітка. Основне припущення, яке робить модель KNN, полягає в тому, що точки даних/екземпляри, які існують в безпосередній близькості один від одного, дуже схожі, а якщо точка даних знаходиться далеко від іншої групи, вона не схожа на ці точки даних. Зробивши припущення алгоритм проводить наступні перевірки.

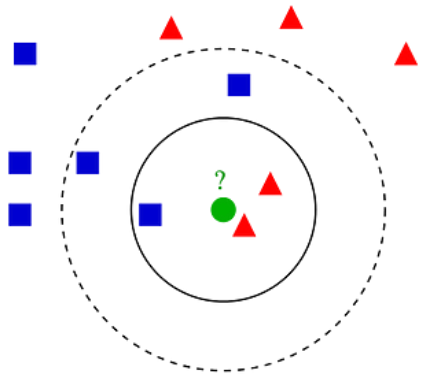

Рисунок 2.4 – Схема роботи алгоритму K-Nearest Neighbours

Модель KNN обчислює подібність, використовуючи відстань між двома точками на графіку. Чим більше відстань між точками, тим менше вони схожі. Існує кілька способів обчислення відстані між точками, але найпоширенішим показником відстані є просто евклідова відстань (відстань між двома точками на прямій).

KNN – це алгоритм навчання з наглядом, тобто приклади в наборі даних повинні мати мітки, призначені для них/їх класи повинні бути відомі. Є дві інші важливі речі, які потрібно знати про KNN. По-перше, KNN є непараметричним алгоритмом. Це означає, що під час використання моделі не робиться жодних припущень щодо набору даних. Швидше, модель побудована повністю з наданих даних. По-друге, при використанні KNN набір даних не поділяється на навчальний та тестовий набори. KNN не робить узагальнень між навчальним набором і набором тестування, тому всі навчальні дані також використовуються, коли моделі просять зробити передбачення.

Алгоритм KNN проходить такі основні фази під час виконання:

1. Встановлення *K* на вибрану кількість сусідів.

2. Обчислення відстані між наданим/тестовим прикладом і прикладами набору даних.

3. Сортування обчислених відстаней.

4. Отримання міток найкращих *K* записів.

5. Повернення прогнозу щодо тестового прикладу.

На першому кроці користувач вибирає *K*, і він повідомляє алгоритму, скільки сусідів (скільки оточуючих точок даних) слід враховувати при винесенні судження про групу, до якої належить цільовий приклад. На другому кроці зверніть увагу, що модель перевіряє відстань між цільовим прикладом і кожним прикладом у наборі даних. Відстані потім додаються до списку та сортуються. Після цього перевіряється відсортований список і повертаються мітки для верхніх елементів *K*. Іншими словами, якщо *K* встановлено на 5, модель перевіряє мітки 5 верхніх найближчих точок даних до цільової точки даних. Під час візуалізації прогнозу щодо цільової точки даних має значення, чи є завдання

регресією чи класифікацієюзавдання. Для задачі регресії використовується середнє з верхніх міток *K*, тоді як у разі класифікації використовується режим верхніх *K* міток (рисунок 2.5).

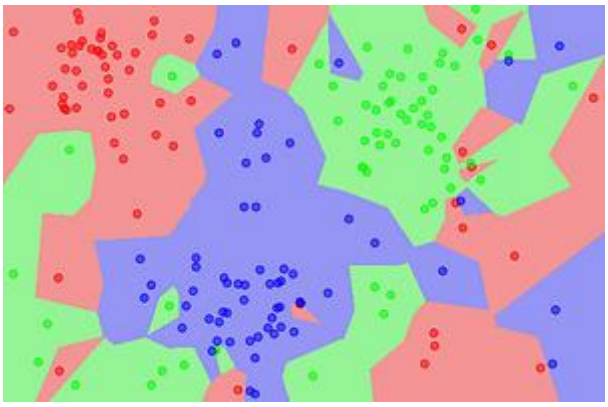

Рисунок 2.5 – Приклад результату роботи алгоритму K-Nearest Neighbours

Точні математичні операції, які використовуються для виконання KNN, відрізняються залежно від вибраної метрики відстані. Якщо ви хочете дізнатися більше про те, як обчислюються показники, ви можете прочитати про деякі з найпоширеніших показників відстані, такі як Евклідова, Манхеттенська та Мінковський.

Основне обмеження під час використання KNN полягає в тому, що може бути обрано неправильне значення *K* (неправильна кількість сусідів, які слід враховувати). Якщо це станеться, прогнози, які повертаються, можуть істотно відхилитися. Дуже важливо, щоб при використанні алгоритму KNN було вибрано правильне значення для *K*. Очевидно, що користувач хочете вибрати значення для *K*, яке максимізує здатність моделі робити прогнози щодо невидимих даних, одночасно зменшуючи кількість помилок, які вона допускає.

Нижчі значення *K* означають, що передбачення, надані KNN, менш стабільні та надійні. Щоб зрозуміти, чому це так, розглянемо випадок, коли маємо 7 сусідів навколо цільової точки даних. Припустимо, що модель KNN працює зі значенням *K* 2 (алгоритму необхідно перевірити двох найближчих сусідів, щоб зробити прогноз). Якщо переважна більшість сусідів (п’ять із семи)

належать, наприклад, до класу *Blue*, але два найближчі сусіди просто *Red*, модель передбачить, що прикладом запиту є червоний. Незважаючи на припущення моделі, у такому сценарії синій був би кращим здогадом.

Очевидно, що і обирати найвище значення *K* теж не раціонально. Це пов'язано з тим, що вказівка моделі враховувати занадто багато сусідів також знизить точність. Оскільки радіус, який розглядає модель KNN, збільшується, вона зрештою почне розглядати точки даних, які ближче до інших груп, ніж вони є цільовою точкою даних, і почне відбуватися неправильна класифікація. Наприклад, навіть якщо початково обрана точка була в одній із червоних областей вище, якщо *K* було встановлено занадто високим, модель простягнеться в інші області, щоб розглянути точки. При використанні моделі KNN, різні значення *K* намагаються побачити, яке значення дає моделі найкращу продуктивність.

До переваг моделі KNN слід віднести:

- KNN можна використовувати як для завдань регресії, так і для завдань класифікації, на відміну від деяких інших алгоритмів навчання з наглядом.

- KNN дуже точний і простий у використанні. Його легко інтерпретувати, розуміти та реалізовувати.

- KNN не робить жодних припущень щодо даних, що означає, що їх можна використовувати для широкого спектру проблем.

Серед недоліків:

- KNN зберігає більшість або всі дані, а це означає, що для моделі потрібно багато пам'яті та її обчислювально дорого. Великі набори даних також можуть призвести до того, що прогнозування займе багато часу.

- KNN виявляється дуже чутливим до масштабу набору даних, і його можна відкинути невідповідними функціями досить легко в порівнянні з іншими моделями.

K-Nearest Neighbours – один із найпростіших алгоритмів машинного навчання. Незважаючи на те, що KNN є простим за принципом, це також потужний алгоритм, який дає досить високу точність у більшості проблем. Коли

ви використовуєте KNN, обов’язково експериментуйте з різними значеннями K, щоб знайти число, яке забезпечує найвищу точність.

Алгоритми класифікації на основі KNN надзвичайно прості і вони досить легко мають справу з кількома класами. Однак KNN однаково обчислює схожість на основі всіх ознак. Це означає, що він може бути схильний до неправильної класифікації, якщо надано зображення, де лише підмножина ознак важлива для класифікації зображення.

Машини опорних векторів – це метод класифікації, який розміщує точки в просторі, а потім малює розділові лінії між точками, розміщуючи об’єкти в різних класах залежно від того, на яку сторону розділової площини падають точки. Машини опорних векторів здатні виконувати нелінійну класифікацію за допомогою техніки, відомої як трюк ядра. Хоча класифікатори SVM часто є дуже точними, істотним недоліком класифікаторів SVM є те, що вони, як правило, обмежуються як розміром, так і швидкістю, при цьому швидкість погіршується при збільшенні розміру.

Багатошарові персептрони, також звані моделями нейронних мереж, є алгоритмами машинного навчання, натхненними людським мозком. Багатошарові персептрони складаються з різних шарів, які з’єднані один з одним, так само, як нейрони в мозку людини з’єднані між собою. Нейронні мережі роблять припущення про те, як вхідні функції пов’язані з класами даних, і ці припущення коригуються в ході навчання. Прості моделі нейронної мережі, такі як багатошаровий персептрон, здатні вивчати нелінійні зв’язки, і в результаті вони можуть бути набагато точнішими, ніж інші моделі. Однак моделі MLP страждають від деяких помітних проблем, як-от наявність неопуклих функцій втрат.

Найбільш часто використовуваним алгоритмом класифікації зображень останнім часом є згортка нейронна мережа (CNN). CNN – це налаштовані версії нейронних мереж, які поєднують багатошарові нейронні мережі зі спеціалізованими шарами, які здатні виділяти ознаки, найбільш важливі та релевантні для класифікації об’єкта (рисунок 2.6).

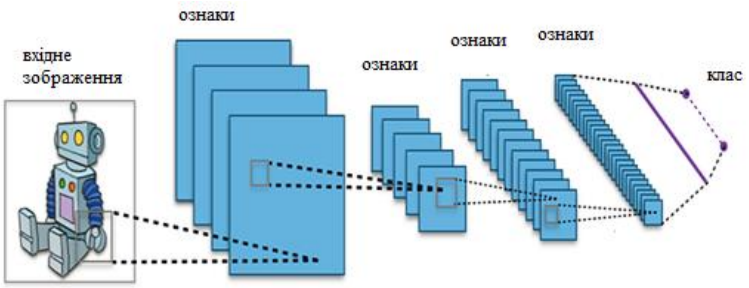

Рисунок 2.6 – Схема роботи згорткових нейронних мереж

CNN можуть автоматично виявляти, створювати та вивчати особливості зображень. Це значно зменшує потребу вручну позначати та сегментувати зображення, щоб підготувати їх до алгоритмів машинного навчання. Вони також мають перевагу перед мережами MLP, оскільки можуть мати справу з неопуклими функціями втрат.

Згорткові нейронні мережі отримали свою назву через те, що вони створюють «згортки». CNN працюють, беручи фільтр і ковзаючи його по зображенню. Ви можете розглядати це як перегляд розділів ландшафту через рухоме вікно, зосереджуючись лише на об’єктах, які можна переглядати у вікні в будь-який момент. Фільтр містить числові значення, які помножуються на значення самих пікселів. Результатом є новий кадр або матриця, наповнена числами, які представляють вихідне зображення. Цей процес повторюється для вибраної кількості фільтрів, а потім кадри об’єднуються в нове зображення, яке є трохи меншим і менш складним, ніж вихідне зображення. Техніка під назвою об’єднання використовується для вибору лише найважливіших значень у зображенні,і мета полягає в тому, щоб згорткові шари в кінцевому підсумку витягли лише найбільш помітні частини зображення, які допоможуть нейронній мережі розпізнати об’єкти на зображенні.

Згорткові нейронні мережі складаються з двох різних частин. Згорткові шари – це те, що витягує особливості зображення та перетворює їх у формат, який шари нейронної мережі можуть інтерпретувати та навчатися. Ранні згорткові шари відповідають за виділення основних елементів зображення, таких

як прості лінії та межі. Середні згорткові шари починають захоплювати більш складні форми, як-от прості вигини та кути. Пізніші, глибші згорткові шари витягують високорівневі характеристики зображення, які передаються в нейронну мережу CNN і про що дізнається класифікатор.

2.3 Алгоритм виділення рухомих об’єків на основі ковзного вікна

При робробці алгориму детекції окремих об’єктів в відеопотоці на основі ковзного вікна були враховані усі перваги та недоліки алгоритмів які використовуються в уже існуючих системах відеоспостереження та цифрових бібліотеках функції. Оскільки даний алгоритм повинен працювати в режимі реального часу, то під час його проектування було максимально мінімізовано кількість операцій які необхідно виконати для досягнення позитивного результату. Іншою важливою проблемою яку необхідно було вирішити є різні значення освітленя при обробці відеопотоку, а саме можливість включення/виключення сторонніх джерел освітлення. Це приводило до значних шумоутворень під час проведення аналізу окремих зображень відеопотоку. Ще однією проблемою яка впливає на результати робти алгоритму детекціє є роздільна здатність камери спостереження. При низькій роздільній здатності об’єкти на зображенні займають меншу кількість точок, що з однієї сторони зменшує кількість обчислень необхідних для отримання результату, але це може мати негативний вплив на кінцевий результат роботи алгоритму. Проте і занадто висока роздільна здатність призводить до значного зростання математичних та логічних обчислень. Тому для прискорення роботи алгоритму при розгляді відеопотоку до уваги береться не ціле зображення а лише його частини, а саме краї картинки. Блок-схема алгоритму наведена на рисунку 2.7.

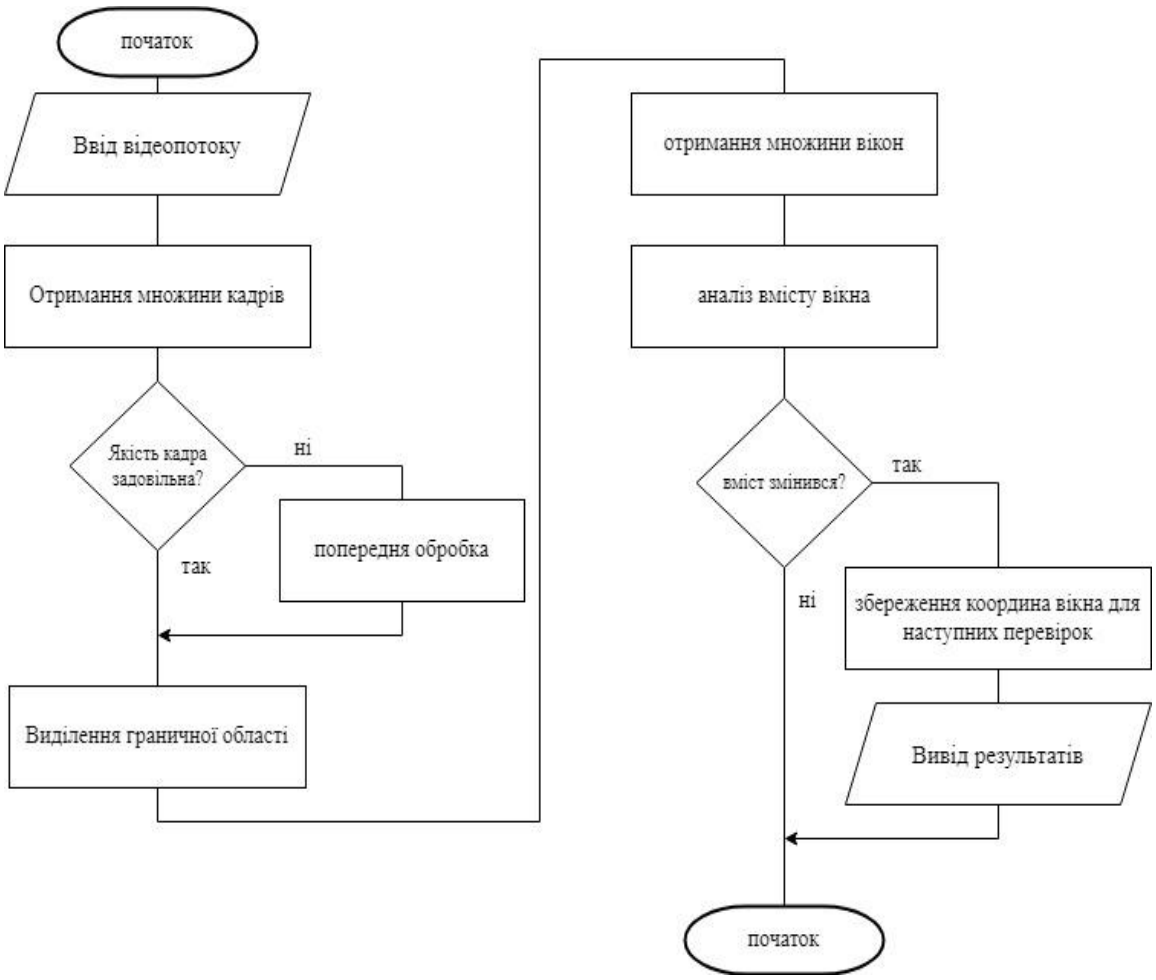

Рисунок 2.7 – Блок-схема алгоритму перетворення елементів на зображенні

Основна ідея даного алгоритму полягає в тому, що при спостереженні деякої області рухомих об’єкт повинен спочатку появитись на границях цієї області. Дане припущення дозволяє значно зменшити кількість необхідних превірок, оскільки більшу частину зображення не превіряємо. Перевірка центральної частини зображення здійснюється тільки у випадку коли об’єкт був ідентифікований на зображенні і в подільшому проводиться його супровід в межах аналізованої площі.

Алгоритм містить наступні кроки:

1) Активація роботи програми, встановлення параметрів роботи (розміри граничної оласті спостереження та розміри ковзного вікна).

2) Отримання першого кадру відеопотоку.

3) Розбиття граничної області першого кадру на окремі вікна заданого розміру та формування масиву спостереження.

4) Отримання наступного кадру відеопотоку та розбиття його граничної області на окремі вікна.

5) Перевірка значень вікон поточного кадру зі значеннями вікон масиву спостереження які мають подібні координати.

6) Якщо у вікні з ідентичними координатами не відбулось змін, то перехід на крок 8.

7) Якщо вікна з ідентичними координатами відрізняються (більше 20% точок у вікні має інше значення за заданий поріг *T*) одне від одного то вважається, що відбувся деякий рух в даній області, дане вікно заноситься в масив вікон з потенційним виявленням.

8) Проводиться перевірка області вікна з масиву потенційних виявлень з метою мінімізації точок які зберегли свій стан незмінним та формується остаточне вікно спостереження за об’єктом.

9) Провоодиться перевірка сусідніх областей з масиву потенційних виявлень при цьому розміри вікна збільшуються на 25%, для прослідковування руху детектоваого об’єкту.

10) Інформація про виявлені об’єкти та напрям їх руху.

11) Проводитсья заміна значень вікон поточного кадра з масивом вікон спостереження.

12) Якщо спостереження завершено то вихід з алгоритму інакше перехід на крок 4.

Використання принципу аналізу не цілої області інтересу а лише її граничних часнин є однією з переваг запропонованого алгориму. Оскільки зменшення кількості перевірок дозволяє зменшити кількість математичних операцій, особливо у випадку коли руху на області спостереження не відувається або у випадку коли кількість рухомих обєктів є мінімальною.

До основних переваг розробленого алгоритму відносяться:

- висока швидкість роботи за рахунок зменшення операцій первірки;

- низькі вимоги до апаратного забезпечення;
- простота та зрозумілість реалізації;
- можливість автономної роботи без втручання людини.

Недоліки:

- результат роботи алгоритму залежить від початкових параметрів роміру вікна та швидкості оновлення вхідних кадрів;
- промлемне детектувння, якщо об’єкт рухаєтсья або з дуже великою або з дуже малою швидкістю.

В результаті моделювання робти запропонованого алгоритму він показав хороші результати по швидкості роботи та точності детекції рухомих об’єктів.

2.4 Висновки до розділу

Проведно аналітичний огляд алгоритмі виділення та класифікації обєєктів на цифрових зображень, що дозволило в подальшому розробити алгоритм виділення рухомих об’єктів у відеопотоці.

Розроблено алгоритм детекції рухомих обєктів у відеопотоці, на основі використання принципів ковзного вікна на границі зображення, щозволило провести структурне проектування та реалізації програмної системи детекції рухомих об’єктів на зображеннях відеопотоку.

3 ПРОГРАМНА СИСТЕМА ДЕТЕКЦІЇ ОБ’ЄКТІВ НА ЦИФРОВИХ ЗОБРАЖЕННЯХ

3.1 Структура програмної детекції рухомих об’єктів

Виявлення об'єктів є технологічно складною і практично корисною проблемою в області комп’ютерного зору. Виявлення об'єктів займається виявленням присутності різних окремих об'єктів в образі. Великих успіхів досягнуто в контрольованому середовищі для об'єкта, але проблема залишається невирішеною в неконтрольованих місцях, зокрема, коли об’єкти розміщені у довільних позах у захаращеному та закритому середовищі. Як наприклад, можна легко навчити домашнього робота розпізнавати наявність кавової машина без нічого іншого на зображенні. З іншого боку, уявіть собі складність такого робота у виявлення машини на кухонній плиті, яка захаращена іншим посудом, гаджетами, інструментами тощо.

Процес пошуку або розпізнавання в такому сценарії дуже складний. Поки що ефективного рішення немає знайдено для цієї проблеми.

Дослідження з виявлення об'єктів є багатодисциплінарними і часто охоплює галузі обробки зображень, машинне навчання, лінійної алгебри, топології, статистика/ймовірність, оптимізація тощо. При проектуванні структурної архітектури програмної системи використовується ієрархічно-модульний підхід, оскільки даний підхід надає можливості максимально швидко проводити маніпуляції з окремими модулями системи: додавання, корегування, заміни та виділення обкремих структурних елементів програмної системи без страти її працездатності в цілому.

Більшість методів виявлення та розпізнавання об’єктів можна розділити на дві категорії на основі типу функції, який вони використовують у своїх методах. Дві категорії – це група методі на основі країв і група методі на основі міток. Примітно, що деякі дослідники використовували поєднання обох функції для виявлення об'єктів на основі країв і міток. І як показують результати

експериментів поєднання цих двох ознак у майбутньому стане все більш поширеним, тому що така схема дасть системі переваги обох типів ознак.

Методи, які використовують тип об’єкта на основі країв, витягують карту країв зображення та ідентифікують особливості об'єкта з точки зору ребер. Використання країв як функції є перевагою перед іншими функціями з різних причин. Вони значною мірою інваріантні до умов освітлення та варіацій кольорів і текстур об’єктів. Вони також добре відображають межі об’єкта та ефективно представляють дані у великому просторі обсяг зображень.

Іншим поширеним типом функції є тип функції на основі міток, який сприймає зовнішній вигляд як мнжину міток які характерні для певного класу. При розробці структури програмного додатку були враховані ці два підходи та обрано варіант отримання інформації через мітки, оскільки в даний момент необхідно просто фіксація процесу руху.

Спрощену структуру програмної системи наведено на рисунку 3.1.

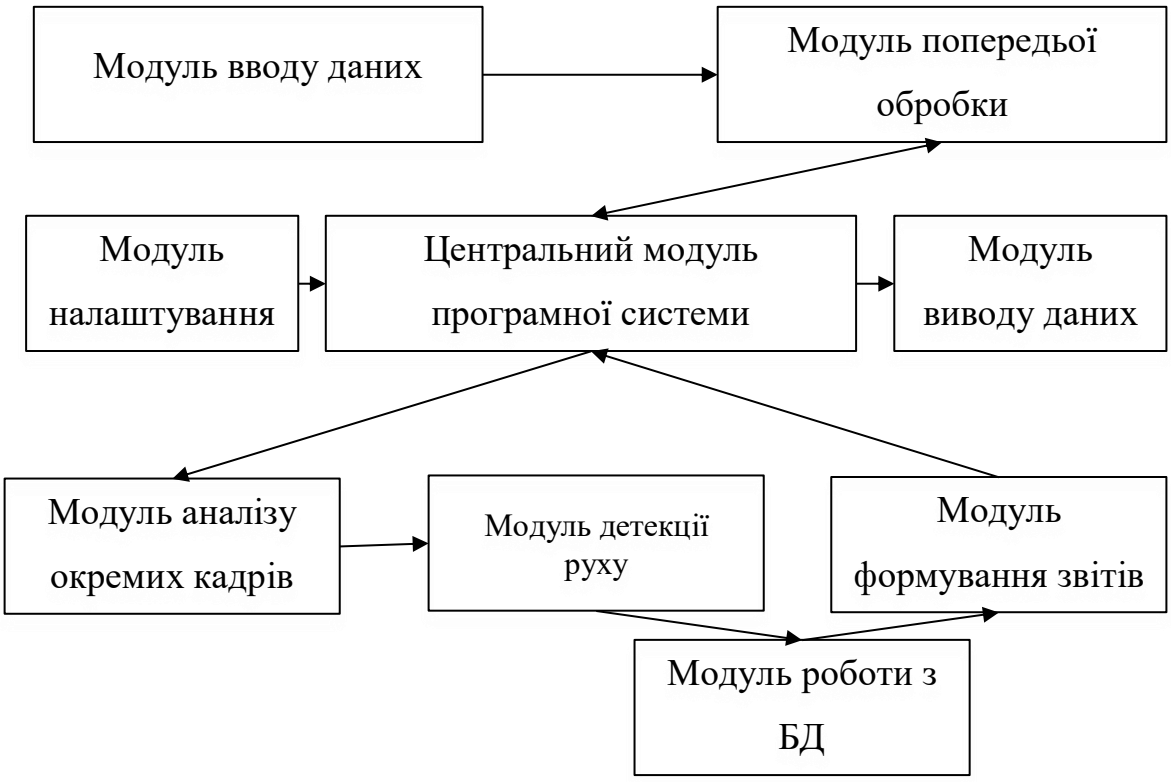

Рисунок 3.1 – Узагальнена структура програмного додатку детекції руху на відеопотоках

Розроблена структура програмного додатку є результатом об’єднання основних модулів різних програмних систем для ведення відеоспостереження, а також включенням структурних модулів призначених для реалізації запропонованого алгоритму детекції руху на відеофайлах. Серед головних структурних частин слід відмітити такі:

Центральний модуль програмної системи – це ядро програмного додатку, призначений для зберігання цілісності та підтримку роботи програмного додатку при різних ситуаціях (рисунок 3.2). В даному блоці реалізовані алгоритми перевірки цілісності програмного додатку, наявність поточних оновлень модулів та версії або від’єднання непотрібних або малоінформативних блоків програми, конвертація та перевірка коректності повідомлень в середині системи та при обміні з зовнішніми структурами. Вхідними даними даного модуля є вектор параметрів роботи програми. Вихідними – корекційні правки пакетів обміну даних, повідомленя для користувача при необхідності внести тих чи інших змін в роботу програмного додатку в цілому.

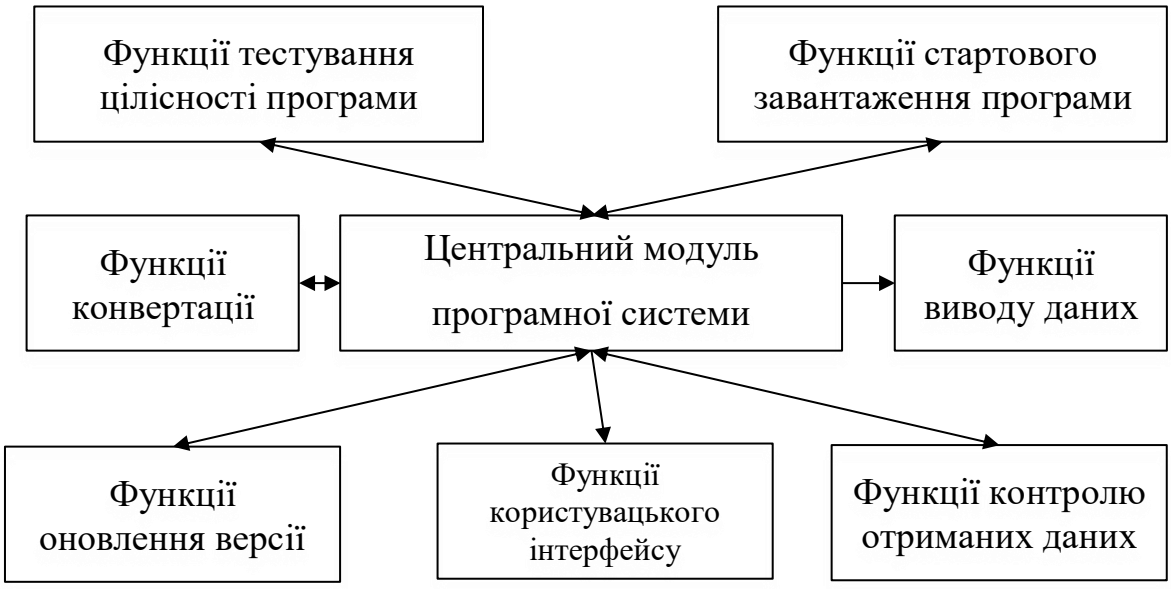

Рисунок 3.2 – Узагальнена структура центрального модуля програмної системи

Модуль завантаження відео – призначений для встаовленні інтерфейсу між апаратними засобами які генерують або зберігають відеопотік. Приорітет

важливості не високий, оскільки при відсутності зовнішніх джерал програмний добаток може опрацьовувати інформацію з внутрішнього сховища даних. Вхідні дані – набір параметрів для підєднання до зовніхніх джерел дани, вихідна – відеофайл для подальшого аналізу.

Модуль попередьої обробки – один з центральних елементів с втруктурі програмного додатку, бо саме від результатів первиннної обробки та аналізу вхідних даних в багато чому залежить подальша робота система, швидкість опрацювання файлів та коректність отриманих результатів. В модулі закладено функції підвищення якості зображень, попередня оцінка та підготовка даних для більш пізніх етапів аналізу та перетворення зображень. Більшість функцій були взяті з зовнішніх цифрових бібліотек, оскільки під час виконання індивідуального завдання перед виконавцем не ставились завдання розробки та проектування алгоримтів попередньої обробки зображень. В результаті обрацювання вхідного зображення даних модулемРезультатом роботи даного модуля є встановлення основних параметрів функціонування програмного додатку.

Модуль налаштування – невеликий допоміжний модуль для підвищення зручності роботи користувачів з системою, а також підбір стартових параметрів робти системи відносно досвіду та авдань користувача. Серед можливих значень, що можна змінити: тип та розмір шрифтів інтерфейсу та модулі виводу звітів, папаметрів підключення та роботи фото/веб/відеокамер, встановлення параметрів роботи агоритмів детекції руху об’єктів (розміри ковзного вікна та частота оновлення кадрів відеофайлу).

Модуль детекції руху – один з головних модулів програмного додатку в якому реалізовані запропоновані алгоритми детекції руху об’єктів у відеопотоці. Коректна робота даного модуля з однієї сторони напряму залежить від роботи всіх інших структурних одиниць, а з іншої в ньому зібрані функції які можуть самостійно провести аналіз та обробку вхідних дани. При цьому є велика ймовірність підвищення часу обробки та збільшення кількості хибних спрацювань. В результаті реалізації запропонованого алгоритму дани

програмний модуль отримує на вхід кадри з відеопотоку, проводить їх аналіз та фармує два масива даних: масив спостереження (вікна з попередніх етапів аналізу кадрів відео) та масиву рухомих цілей (вікна на яких визначено зміни). Окрім того формується повідомлення для користувачів при детекції руху на відео.

Модуль роботи з БД – допоміжний модуль який допомагає реалізувати програмний інтерфейс з сервером бази дани для забезпечення зберігання роботих масивів вікон. Результати його робти видаляютсья після закінченняроботи програми.

Модуль створення звітів – додатковий модуль який дозволяє організувати більш зручну роботи користувачів з системою. Від надає можливості користувачам обрати формати та способи виведення результатів роботи програми та інформування ористувачів про виявлення процесу руху на відеою Відеопослідовність формується на основі встановлених користувачем параметрів та може коригуватись під час виводу результатів опрацювання.

Розроблена та спроектована структура на основі ієрархічного підходу з використанням окремих модулів на кожному рівні ютакий підхід є гнучким та універсальним та дозволяє зберегти працездатність системи навіт при пошкодженні еяких блоків. Окрім того використаний модульний підхід дозволяє реагувати на зміни в робті програмної системи додаючи наві модулі чи модіфікуючи вже присутні, при цьому робота системи та цілісність не порушується.

Перед процесом реалізації розглянутих алгоритмів проведено ряд заходів для моделювання розробленого алгоритму та програмної структури на основі відомих програмних пакетів. Серед недоліків використання універсальної мови моделювання слід відмітити:

– часові затрати. Потрібно багато часу, щоб діаграма була розумною і синхронізована з фактичним кодом. Діаграми UML не запускаються, але вимагають багато часу. Тому вони корисні тільки в тому випадку, якщо розробник може ними керувати.

– можливі втрати інформації. Не можливо представляти кожну умову в діаграмі послідовності. Тому діаграми стану повинні передавати основні факти, а не всі можливі результати.

– грошові витрати. Гарне програмне забезпечення UML коштує грошей, і для правильного освоєння потрібен якийсь час.

Моделювання на основі діаграми предецентів дозволить оцінити рівень взаємодії окремих функції та вирахувати відсотки їх важливості в загальній структурі проекту. Додатково розглядається можливості користувача при взаємодії з системою. Приклад діаграми прецеденів наведено на рисунку 3.3.

Рисунок 3.3 – Діаграма прецедентів програмної системи

Наведена діаграма ілюструє результати етапу аналізу рівня взіємодії окремих функції між мобою, а також показує можливості робобти користувача з відповідною програмною системою. Дії користувача обмежені тільки зобнішнім впливом не систему, а то й же час внісрішні механізми захищені від прямого доступу. Дане моделювання показало правильність обраної структури та набору реалізованих функцій. Проте, достатньо високий рівень автоматизації роботи програмного додатку забезпечує можливість використання її для роботи та

навчання користувачів без досвіду роботи. А сам процес опанування програмою є достатньо короткий та не ввимагає від нових кристувачів грибоких попередніх знань комп’ютерної техніки.

На наступному етапі моделювання аналізується коректність кроків які необхідно зробити для досягнення результатів. Аналізувалась взаємодія процесів та об’єктів, що будуть відбуватись під час роботи програми. Результат моделювання наведено на рисуноку 3.4.

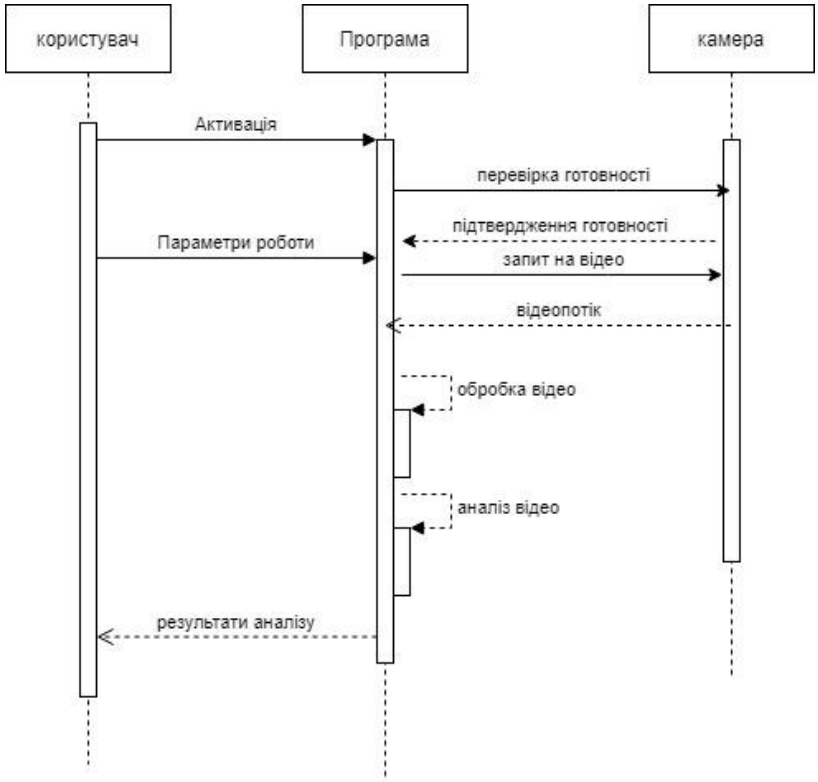

Рисунок 3.4 – Діаграма послідовності програмної системи

Навдена діаграма у повній мірі дозволяє проілюструвати можливості запропонованої структури програмної системи. Як показано усі процеси виконуються в чіткій послідовності, не викликають взіємної колізії та не переоплюють управніння ресурсами коли вони заняті іншими процесами. Лінійність виконання задах хоть і сповільнює рботу програми проте дозволяє уникнути проблем спільного доступу до ресурсів. А враховуючи, що програмний

додаток повністю виконує поставлені задачі за відведені часові межі, то даний фактор можна вважати не суттєвим.

На етапі моделювання було розглянуто основні фактори. Які можлуть впливати на працездатність системи, а також функціональні можливості які має виконати користувач для досягнення результату.

Іншим етапом проектування та розробки програмної системи є етам розробки користувацького інтерфейсу програмної системи. Під час проектування дизайну головного вікна програми були враховуні усі можливості для користувачів до швидкого доступу до окремих функцій та пунктів меню. А з іншої сторони вдалось уникнути значного завантаження головного вікна програми. При цьому основний простір на головному вікні було віддано областям для відображення вхідних зображень та область для відображення результатів роботои програми.

Графічний інтерфейс користувач було розроблено з врахуванням сучасних вимог до програмних інтерфейсів, а саме використано принцип мінімальзму при якому користувачу надається мінімальна кількість активних важелів впливу на роботу програмної системи, оскільки рівень знань та навичок користувача не завжди може бути достатнім для коректного налаштування та роботи з програмною системою. Для користувачі з досвідом роботи програмна система надає можливості більш грунтовного впливу на процес роботи програмної системи, проте і відповідальність за коректну роботу системи теж покладається на користувача.

Для роботи з програмою користувачеві необхідно виконати просту послідовність кроків. Даний набір чітко ілюструється підказками під час запуску програмного додатку, а також може бути проілюстрований в процесі робти програми у вигляді підказок які з’являються. Послідовність візуально відображено на рисунку 3.5:

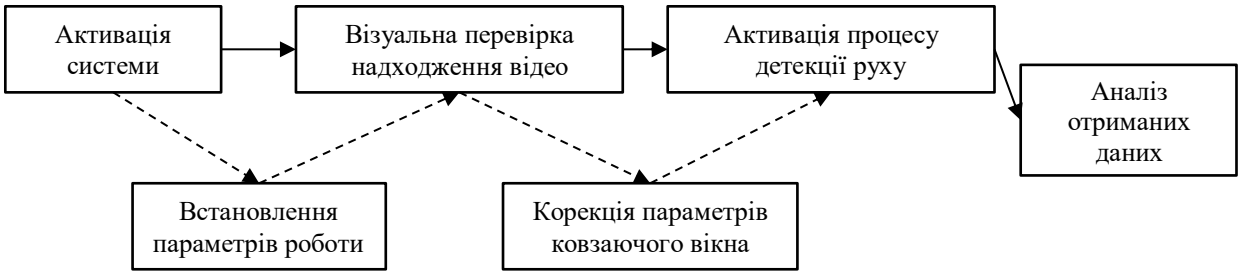

Рисунок 3.5 – Послідовність дій користувача при роботі з системою відеоспостереження

Як показано зі схеми дій користувача, при роботі користувач повинен виконати три основі дії з програмною системою, що включають початок роботи шляхом активації джерел передачі відео, перевірки коректності надходження відеопотоку та перегляду отриманих результатів в процесі перетворення зображень. Окрім того користувач може виконати ще дві додаткові дії, при необхідності внесення змін при негативному або не достатьо хорошому результаті роботи програми.

3.2 Програмні модулі системи перетворення елементів зображення

ImageAI – це бібліотека виявлення об’єктів, вбудована на python. За допомогою функцій ImageAI створюють програми та системи з автономним глибоким навчанням і можливостями комп’ютерного зору, використовуючи кілька рядків прямого коду. Бібліотека ImageAI має можливість реалізації майже всіх найсучасніших алгоритмів глибокого навчання, таких як YOLOv3 і TinyYOLOv3.

ImageAI використовує кілька API, які працюють в автономному режимі – API працюють на виявлення об’єктів, розташування об’єктів, виявлення відео та відстеження об’єктів, які можна викликати без доступу до Інтернету. ImageAI в

основному працює на попередньо навченій моделі і може бути легко налаштована. За допомогою ImageAI, можна виявляти та розпізнавати 80 різних видів звичайних повсякденних об’єктів.

Серед нових можливостей даної розробки слід відмітити:

Додано SqueezeNet, ResNet50, InceptionV3 і DenseNet121. Модель виконує визначені користувачем навчання прогнозування зображень.

Додана спеціальна модель навчання та файл json для імпорту та експорту користувацьких зображень.

Попередній перегляд: додано визначення відеооб'єкта та користувацьке відео виявлення об'єктів (відстеження об'єктів).

Додано файли для всіх завдань прогнозування зображень і виявлення об’єктів, типів масивів numpy і потоків.

Додано файли та тип виводу масиву numpy, Використовується для виявлення об’єктів у зображеннях та користувацьких об’єктів

Розроблено чотири режими швидкості (звичайний, швидкий, швидший і найшвидший), в залежності від типу зображення, обирається найшвидшим. У швидкісному режимі час передбачення буде скорочено на 50%, зберігаючи точність передбачення.

Введена частота виявлення кадрів, Дозволяє розробникам регулювати інтервал виявлення у відео frame_detection_interval, Допомагає досягти певного ефекту

Як видно з наведених вище переліку можливостей бібліотека ImageAI максимально ефективно підходить як базовийінфструмент для реалізації запропонованої програмної системи. Використання розроблених функцій дозволять значно зменшити час розробки системи при цьому будемо використовувати функції для виконання попередніх підготовчих операцій для виконання основного завдання по розробці програмного забезпечення на основі ковзного вікна.

Для успішної роботи програмної розробки необхідно встановити стартові параметри її роботи та підключити необхідні модулі в яких знаходяться описи

відповідних моделей та функцій необхідних для функціонування та коректної роботи розробки. Приклад лістингу підключення відповідних моділів наведено на рисуноку 3.6.

```
import cv2, os
import numpy as np
import matplotlib.pyplot as plt

OBJ_THRESH = .6
P_THRESH = .6
NMS_THRESH = .5

np.random.seed(999)
```

Рисунок 3.6 – Приклад встановлення початкових параметрів роботи програми

В першу чергу необхідно підключти модуль для моливості роботи з функціями бібліотеки OpenCV дана бібліотека містить ряд широкоповсюджених функцій для обробки цифрових зображень та відео. В даному випадку підключення відбуваєтсья командою import cv2. OpenCV це величезна бібліотека з відкритим кодом для комп’ютерного зору, машинного навчання та обробки зображень. OpenCV підтримує широкий спектр мов програмування, таких як Python, C++, Java тощо. Він може обробляти зображення та відео, щоб ідентифікувати об'єкти, обличчя або навіть почерк людини. Коли він інтегрований з різними бібліотеками, такими як Numpy, яка є високооптимізованою бібліотекою для чисельних операцій, то кількість зброї у вашому арсеналі збільшується, тобто будь-які операції, які можна виконувати в Numpy, можна об’єднати з OpenCV. Усі пакунки містять каскадні файли Haar. cv2.data.haarcascades можна використовувати як ярлик до папки даних.

Наступним модулем для коректної роботи програми є модуль os. Модуль os у Python надає функції для взаємодії з операційною системою. OS належить до стандартних допоміжних модулів Python. Цей модуль забезпечує портативний спосіб використання функцій, що залежать від операційної системи. Модулі *os* і *os.path* містять багато функцій для взаємодії з файловою системою. Розглянемо поточний робочий каталог (CWD) як папку, в якій працює Python.

Кожного разу, коли файли викликаються лише за їхнім іменем, Python припускає, що він починається в CWD, що означає, що посилання лише на ім’я буде успішним, лише якщо файл знаходиться в CWD Python.

Оскільки розроблювальний програмний додаток буде опрацьовувати великі масиви числової інформації, то доцільно підйлючити відповідний модуль на NumPy. NumPy, що розшифровується як Numerical Python – це бібліотека, що складається з багатовимірних об’єктів масиву та набору підпрограм для обробки цих масивів. За допомогою NumPy можна виконувати математичні та логічні операції над масивами.

Matplotlib є одним з найпопулярніших пакетів Python, що використовуються для візуалізації даних. Це кросплатформна бібліотека для створення двовимірних графіків із даних у масивах. Matplotlib написаний на Python і використовує NumPy, розширення чисельної математики Python. Він надає об’єктно-орієнтований API, який допомагає вбудовувати графіки в програми з використанням інструментів GUI Python, таких як PyQt, WxPythonotTkinter. Його також можна використовувати в оболонках Python і IPython, ноутбуках Jupyter і серверах веб-додатків.

Matplotlib має процедурний інтерфейс під назвою Pylab, який нагадує MATLAB, власну мову програмування, розроблену MathWorks. Matplotlib разом із NumPy можна розглядати як еквівалент MATLAB з відкритим кодом.

Окрім встановлення стартових параметрів роботи програми додатково були реалізовані методи попредньої обробки, які дозволять зменшити час робти програми в загальному.

Оскільки запропонований алгоритм повинен працювати швидко при цьому він повинен тільки відрізняти чи відбулись зміни в певній області, то для зменшення кількості необхідних операцій проводиться процедура перетворення зображенні з кольорового в монохромне (рисунок 3.7).

```
import cv2

image = cv2.imread('C:\\test.jpg')
cv2.imshow('Original', image)
cv2.waitKey(0)

gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

cv2.imshow('Grayscale', gray_image)
cv2.waitKey(0)

cv2.destroyAllWindows()
```

Рисунок 3.7 – Програмний код перекодування зображення з кольорового в монохромне

Приклад результатів роботи фуфнкції переконвертації зображення з кольорового в монохромне наведено на рисунку 3.8.

а) Кольорове зображення б) Монохромне

Рисунок 3.8 – Програмний код перекодування зображення з кольорового в монохромне

Іншою важливою задачею для роботи розроблювальної системи є модуль отримання кадрів з відео. Оскільки відеопотік можна розглядати як послідовність кодрів то з однієї сторони це здається простою задачею, проте з іншої сторони слід врахувати, під час передачі відое інформація піддаєтьскодуванню. Тому перед початком робти з рпограмною системою виникає необхідність відновити іформацію та отримати кадр з мінімальною втрадою даних. Для цього було реалізовано програмну функцію з використанням віоповідних технічних засобів (рисунок 3.9).

В даній реалізації використовуються можливості бібліотеки OpenCV для роботи з цифровими зображеннями та відеопотоками. При цьому можна одразу

підрахувати кількіст кадрів з яких складається відповідне відео. Також в функції реалізовано додаткову перевірку на можливість відкриття відеофайлу, щоб зменшити шанси отримати помилку під час роботи програми.

```
import cv2
import os

cam = cv2.VideoCapture("C:\\Test_video.mp4")
try:
    if not os.path.exists('data'):
        os.makedirs('data')
except OSError:
    print ('Error: Creating directory of data')
currentframe = 0
 while(True):
    ret,frame = cam.read()
    if ret:
        name = './data/frame' + str(currentframe) + '.jpg'
        cv2.imwrite(name, frame)
        currentframe += 1
    else:
        break
cam.release()
cv2.destroyAllWindows()
```

Рисунок 3.9 – Програмний код розбиття відео на окремі кадри

Окрім можливості обробки статичного відео в програмі передбачено моливість отримання відео режимі реального часу, для цього використано функцію для отримання відео з активного пристрою вводу відео інформації. В загальному вигляді це буде веб-камера яка підєднана до компютера та проводить відеотрансляцію. Пі час отримання вхідного відеопотоку відбувається одночасне розбиття вхідного потоку на окремі кадри які в подільшуму передаються на детектор руху. Програмний код функції отримання відео наведено на рисунку 3.10.

```
import cv2

cap = cv2.VideoCapture(0)
i = 0
while(cap.isOpened()):
    ret, frame = cap.read()
    if ret == False:
        break
    cv2.imwrite('Frame'+str(i)+'.jpg', frame)
    i += 1
cap.release()
cv2.destroyAllWindows()
```

Рисунок 3.10 – Отримання відео з камери в режимі онлайн

Ще однією важливою функцією роботи програмного додатку є можливість збереження отриманого відеопотоку (рисунок 3.11). При цьому для зменшення об’ємів відповідних файлів та скорішої подальшої обробки відеофайл одразу конвертуєтсья в монохромний формат. При цьому, якість зображення та всі інформативні ознаки не пропадаю, оскільки основна інформація, а саме зміна параметра окремих пікселів буде відбуватись. Якщо дві точки матимут однакові параметри до переконвертуваня то і після переконвертування одним і тим же алгоритмом вони будуть мати однакові значення.

```
import numpy as np
import cv2

cap = cv2.VideoCapture(0)
fourcc = cv2.VideoWriter_fourcc(*'XVID')
out = cv2.VideoWriter('temp.avi', fourcc, 20.0, (640, 480))
while(True):
    ret, frame = cap.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    out.write(gray)
    if cv2.waitKey(1) & 0xFF == ord('a'):
        break
cap.release()
cv2.destroyAllWindows()
```

Рисунок 3.11 – Отримання відео з камери в режимі онлайн, конвертацыя та збереження на жорсткий носій даних

Ще однією важливою функцією роботи програмного додатку є елемент порівняння значень двох вікон з відповідними координатами. Дане порівняння відбуваєтьс япопіксельно, для того щоб можна було отримати відсоткове значення подібності двох зображень. На основі даної подібності і буде відбуватись подальше детектування наявності руху на відеопотоці. При цьому рівень похибки можна виставляти на необхідному для користувача рівні. Даний алгоритм порівняння можна прискорити якщо брату до уваги не кожну точку зображення, а виділяти точки зображення з деяким кроком. Оскільки рухомий об’єкт не буде займати один піксель, то відповідно відпадає необхінність превірки усіх пікселів вікна. Проте якщо відбуваєть додаткова процедура класифікації, то для уникнення ситуації втрати важливої інформації необхідно все ж таки

враховувати кожен пуксель. Це призведе до збільшення кількості одчислень, проте дозволить збільшити кількість іфнормативних ознак які будуть приймати участь в подальшому опрацюованні. Приклад порівняння двох вікон наведено на рисунку 3.12.

```
import cv2

new_image = cv2.imread("c:\\new_image.jpg")
test_image = cv2.imread("c:\\test_image.jpg")

    difference = cv2.subtract(original, duplicate)
    b, g, r = cv2.split(difference)
    if cv2.countNonZero(b) == 0 and cv2.countNonZero(g) == 0 and cv2.countNonZero(r) == 0:
        print("The color is equal")
    else:
        print('The color of image is different')

```

Рисунок 3.12 – Програмний код порівняння двох зображень

В загальному випадку програмний код розроблювальної системи буде умовно розбитий на окремі структурні модулі, які в подальшому будуть реалізовувати окремі підзадачі. При чьому такий підході дозволить значно простіше проводити додаткові налаштування та модифікації в середині самої програмної системи, оскільки виникатиме неодхідність модивікації не всієї програмної частини а окремих модулів, які при потребі модна буде повністю замінити на інші.

Перший умовний блок програмної розробки – це блок запуску програми та встановлення параметрів роботи системи, окрім того буде проводитись вибір джерела отримання відеофайла або веб-камера (відео в режимі реального часу) або завантажуватиметься попередньо збережений відеофайл (оффлайн режим робти програмного додатку). На рисуноку 3.13 наведено приклад програмного коду встановлення параметрів роботи програми та відкриття попередньо збереженого відео. При цьому встановлюється частота вибору кудрів, в даному прикладі було обрано кожний 60 кадр. Експерементально було встановлено, що при нормальній швидкості руху об’єктів дана частота кадрів є цілком

прийнятною для прийняття правильних рішень та своєчасному виявленні рухів у відеопотоці.

```
15 count = 0
16 cv2.namedWindow('Video window')
17 while(vid.isOpened()):
18 perform detection = count % 60 == 0
```

Рисунок 3.13 – Встановлення початкових параметрів роботи програми

Після встановлення стартових параметрів роботи програми необхідно отримати перший кадр для створення стартового масиву зображень та подальшого його аналізу. Для цього за відеопотоку за допомгою функції blobFromImage () виділяємо перше зображення. Перший параметр даної функції відповідає даним зображення, другий – коефіцієнт масштабування, третій – розмір масштабованого зображення, четвертий – середнє значення, п'ятий – прапор каналу колірного обміну, а шостий – виконання операції обрізки.

```
cv2.dnn.blobFromImage(frame, 1 / 255, (416, 416), [0,0,0], 1, crop=False)
```

Після чого дана інформація передається на подальше опрацювання з метою отримання інформаційних ознак вікна.

Використання саме цієї функції дозволяє швидко отримати необхідну інформацію та при цьому не витрачаючи значні часові та обчислювальні ресурси.

Після завершення процедури створення початкового масиву вікон, продовжується процес аналізу наступних кадрів у відеопослідовності на основі порівняння відповідних областей зображення з метоюдерекції спроб руху або появи нових об’єктів на відповідних зображеннях. На рисунку 3.14 наведено приклад проведення маркування відповідних вікон на вихідному відеоряді з метою демонстрації процесу дедекції рухі об’єктів.

```
mtracker.add(cv2.TrackerMedianFlow_create(), frame, (x, y, w, h))
count += 1
else:
 cv2.putText(frame, 'Detection failed', (20, 80),
 cv2.FONT_HERSHEY_SIMPLEX, 0.75, (0,0,255), 2)
 else:
is_tracking, bboxes = mtracker.update(frame)
 if is_tracking:
for i, bbox in enumerate(bboxes):
 x, y, w, h = [int(val) for val in bbox]
 class_id = classes[class_ids[idxs[i][0]]]
col = [int(c) for c in colors[class_ids[idxs[i][0]], :]]
cv2.rectangle(frame, (x, y), (x+w, y+h), col, 2)
 cv2.putText(frame, class_id, (x, y - 15),
 cv2.FONT HERSHEY SIMPLEX, 0.5, col, 2)
```

Рисунок 3.14 – Приклад результатів детекції та нанесення міток на виділений об’єкт на зображенні

Наведени перелік програмний функцій у повній мірі дозволяє оцінити результати реалізації окремих програмних модулів системи, а також ілюструє рівень роботи окремих модулів та обраних програмних рішень. Обрана мова програмування та набір цифрових бібліотек є актуальними на сьогоднішній день та широко використовуються під час проектування та реалізації програмних засобів обробки та аналізу цифрових зображень та відеоінформації. Тому реалізований проект має усі переваги які надають бібліотеки OpenCV та ImageAI, що позитивно впливає на результати роботи в цілому.

3.3 Тестування та аналіз реалізованої системи

Для здійснення тестуванн було обрано графічна станція середнього рівня, що підходить для вирішення завдань із проектування простих об'єктів та 2D моделей. Може використовуватися як робоче місце конструктора, дизайнера або менеджера. Робоча станція має такі технічні характеристики:

– Процесор: Intel Core i3-7100 1 шт.

– Чіпсет: Intel® B250.

– ОЗУ: 4GB (1x4GB) DIMM DDR4 (частота шини 2400MHz) Crucial, об'єм пам'яті (макс.) 32GB, слотів пам'яті 2шт.

– Диски: 500GB (1x500GB) SATA 6Gb/s 3.5" Seagate.

– Відеокарта: ASUS, серія: GeForce GT 730, об'єм відео-пам'яті 2GB, GT730, встановлено 1 шт.

– Мережа: на мат. платі: 1Gb/s 1шт.

– Блок живлення: 1шт. 450W.

– Виробник: Gigabyte.

Технічні характеристики пристрою відображення зображень (DELL P2018H 19.45″):

– матриця: TN;

– роздільна здатність: 1600×900 (16:9);

– частота поновлення кадрів: 75 Гц;

– яскравість: 250 кд/м²;

– підсвічування: WLED;

– входи: HDMI 1.4, DisplayPort 1.2, VGA (D-Sub);

– розміри: 570 * 446 * 226 мм.

Технічні параметри комп’ютера є достатніми для проведення тестування розробленої програми та дозволить отримати результат для проведення аналізу коректності роботи програмної системи.

Для провдення тестування та для об’єктивної оцінки запропонованого алгоритму було обрано ряд відеофайлів, які відрізнялись за наступними критеріями:

– тривалість – у всіх відео приблизно однакова +-30с. Даний інтервал часу дозволяє оцінити поведінку програмної системи;

– освітленість – в даному випадку були одрані файли які не містять у собі значні перепади освітленості за короткий проміжок часу, тобто відсутні різкі перепади, наприклад при включенні освітлення в темну пору доби;

– кількість рухомих об’єктів – даний критерій повинен проілюструвати як відреагує система на ведику кількість рухомих об’єктів на відео. При тестуванні були обрані відео в яких кількість рухомих обєктів була від 1 до 5 цілей, від 10 до 20, понад 20 одночасних цілей;

– середній розмір об’єктів на відео;

– розмір ковзного вікна.

При проведенні тестування всі експерименти з відео були виконані по 10 разів для а результати тестування були усереднені. Даний підхід забезпечив більш об’єктивний результат оскільки дозволоив зменшити вплив сторонніх похибок під час проведення експериментів. Іншим важливим чинником який впливає на результати тестування є правильність обрання розмірів ковзного вікна відносно розмірів рухомих об’єктів. Як показали експерименти, якщо рухомих об’єкт більший від розмірів вікна на 30% хоча б за однією з осей, то даний об’єкт не буде детектовано, тому для коректної робти програми розміни ковзноговікна необхідно обирати експерементально.

При тестування відео на яких було присутні невелика кількість рухомих об’єктів (від 0 до 5) віуально не було помітно ніяких проблем. Детекція здійснювалась миттєво, навіть якщо об’єкт не повністю пребував в полізору камери, супровід рухомих об’єктів здійснювався без явних проблем. Підрахуно кількості вівся безпомилково. Проте алгоритм показував хибні результати при умові якщо два об’єкти дотикались один до одного в такому випадку алгритм вважав, що на відео знаходиться один об’єкт. Проте як тільки два обєкти розходились на відстань яка дозволяла провести між ними візуальну границю, алгоритм повертав вірну кількість об’єктів.

При тестуванні групи відео на яких пребувала середня кількість людей результати були подібними до попередньо проведених експериментів. Проте зізбільшенням кількості об’єктів на зображенні збільшилась і кількість об’єктів які дотикаються в процесі руху, що негативно впливало на підрахунок кількості рухомих об’єктів. Проте детекція самого процесу руху відбувалась без збоїв.

При тестуванні відео з великою кількістю об’єктів детекція самого процесу руху відбувалась з високою точністю. Підрахунок рухомих об’єктів був коректний при умові що між окремими обєктами можна провести умовну лінію розмежевання. При дотиканні великої кількості об’єктів на зображенні формувався одина суцільна область великого формату яка ситуативно ставала більшою за розміри ковзаючого вікна і це ускладнювало процес підрахунку кількості рухомих цілейн на відео, проте тільки цілі розходились алгоритм продовжував коректно підраховувати їх кількість.

Як видно з наведених результатів тестування, розроблена програмна система успішно пройшла етап тестуваннята показала високі результати роботи при різних вхідних відео. Робота програми відбувалась в автоматичному режимі, а результати роботи зберігались у текстовому файлі.

В результаті проведених експерементів та на основі аналізу отриманих даних була сформована підсумкова таблиця де відображаються отримані результати у згрупованому вигляді. Результати наведено в таблиці 3.1

Таблиця 3.1 – Узагальнена таблиця результатів тестування

Параметри вхідних відео	Кількість цілей від 0 до 5	Кількість цілей від 5 до 10	Кількість цілей від 10 до 15
Детекція руху	відмінна	відмінна	відмінна
Правильність підрахунку кількості рухомих цілей	висока	висока/ поодинокі похибки	висока/ поодинокі похибки
Вплив освітлення	відсутній	відсутній	відсутній
Вплив швидкості руху об’єктів (занадто швидко або занадто повільно)	відсутня детекція	відсутня детекція	відсутня детекція
Похибка підрахунків при дотиканні сусідніх об’єктів	+	+	+
Загальна швидкість роботи алгоритму	висока	висока	висока

Отримані результати підтвердили припущення, які були зроблені на етапі моделювання, що розроблений програмний код детекції має високу швидкодію, та пороводить обробку відео при різних вхідних параметрах, при цьому забезпечуючи відповідну якість результатів роботи.

Отримані результати в процесі тестування показали правильність обраного шляху розробки програмного додатку. Програний додаток успішно пройшов тестування та показав відмінні результати при різних вхідних даних.

3.4 Висновки до розділу

Розроблено та проведено теоретичне моделювання програмного додатку детектування рухомих об’єктів у відеопотоці, що надало можливість програмно реалізувати систему відеоспостереження.

Здійснено тестові випробування розробленої програмної системидетекції рухомих об’єктів у відеопотоці, на основі ковзного вікна при різних вхідних даних, що підтвердило правильність у виборі механізмір реалізації програмного додатку та розробленого алгоритму.

ВИСНОВКИ

На основі аналізу сучасних програмних систем відеоспостереження з елементами детекції руху та аналізу алгоритмів виділення об’єктів на основі аналізу послідовності кадрів у відеопотоці можна зробити такі висновки:

1. Проведено аналіз та класифікацію завдань комп’ютерного зору, що надало можливість окреслити основні завдання та напрямки роботи алгоритмів комп’ютерного зору при обробці цифрових зображень.

2. Проаналізовано методи кодування та передачі сигналів у відеопотоках, що дозволило визначити оптимальні алгоритми кодування для створення прорамної системи з елементами детекції руху.

3. Проведено аналіз наявних програмних засобів для виділеня та класифікації об’єктів на зображеннях який дозволив виділити основні структурні модулі, та встановити інтерфейси обміну даними між окремими модулями.

4. Проведно аналітичний огляд алгоритмі виділення та класифікації обєєктів на цифрових зображень, що дозволило в подальшому розробити алгоритм виділення рухомих об’єктів у відеопотоці.

5. Розроблено алгоритм детекції рухомих обєктів у відеопотоці, на основі використання принципів ковзного вікна на границі зображення, щозволило провести структурне проектування та реалізації програмної системи детекції рухомих об’єктів на зображеннях відеопотоку.

6. Здійснено тестові випробування розробленої програмної системидетекції рухомих об’єктів у відеопотоці, на основі ковзного вікна при різних вхідних даних, що підтвердило правильність у виборі механізмір реалізації програмного додатку та розробленого алгоритму.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Bradski G. Learning OpenCV - Computer Vision with the OpenCV Library / G. Bradski.,2018 – 580 с.
2. Гонсалес Р. Цифровая обработка изображений в среде MATLAB / Р. Гонсалес, Р. Вудс, С. Эддинс. - М.: Техносфера, 2016. – 616 с.
3. Сафонов В. О. Параметризованные типы данных. История, теория, реализация и применение / В.О. Сафонов. – М.: Издательство Санкт-Петербургского университета, 2013. – 116 с.
4. Секунов Н.Ю. Самоучитель Visual C++ 6.0 / Н.Ю. Секунов. – М.: СПб: BHV, 2014. - 960 с.
5. Страуструп Б. Язык программирования C++ / Б. Страуструп. – М.: Радио и связь,2019. – 350 с.
6. Страуструп Бьерн Дизайн и эволюция C++ / Бьерн Страуструп. – М.: ДМК Пресс, 2016. – 446 с.
7. Сойфер В.А. Методы компьютерной обработки изображений / В.А. Сойфер. – М.: Физматлит,2013. – 784 с.
8. Оппенгейм А. Цифровая обработка сигналов / А. Оппенгейм, Р. Шафер. – М.: Техносфера,2006. – 856 с.
9. Прэтт У. Цифровая обработка изображений. В 2-х книгах: пер. с англ. / У. Прэтт. – М.: Мир,2009. – 792 с.
10. Грузман И.С. Цифровая обработка изображений в информационных системах: учеб, пособие / И.С. Грузман, В.С. Киричук., 2002. – 352 с.
11. Яне Б. Цифровая обработка изображений / Б. Яне. - М.: Техносфера, 2007. – 584 с.
12. Селянкин В.В. Компьютерное зрение. Анализ и обработка изображений: учебное пособие / В. В. Селянкин.,2019 – 152с.
13. Форсайт Д. Компьютерное зрение. Современный подход / Д. Форсайт.,2004. – 928 с

14. Поляков А. Методы и алгоритмы компьютерной графики / А. Поляков.,2003. – 545 с.

15. Безрядин С. Н. Преобразование яркости в программном обеспечении / С. Н. Безрядин.,2006 – 211с.

16. Баженова И. Ю. Язык программирования С++ / И. Ю. Баженова., 2017. – 366 с.

17. Бартлетт Н. Программирование на Delphi Путеводитель / Н. Бартлетт., 2016. – 116с.

18. Вебер Дж. Технология С++в подлиннике / Дж. Веберю.,2017. –256с.

19. Волш А. И. Основы программирования на С++для World Wide Web / А. И. Волш.,2016. – 458с.

20. Абрамов С. А. Задачи по программированию / С. А. Абрамов., 2018. – 256 с.

21. Березин Б.И. Начальній курс Delphi / Б. И. Березин., 2016. – 331с.

22. Бондарев В.М. Основі программирования / В. М. Бондарев., 2017. – 446с.

23. Вирт Н. Алгоритмі и структурі данніх / Н. Вирт, 2019. – 345с.

24. Гладков В. П. Задачи по информатике на вступительном экзамене в вуз и их решения: Учебное пособие / В. П. Гладков., 2014. – 516с.

25. Грогоно П. Программирование на язіке Delphi / П. Грогоно.,2012. –216с.

26. Дагене В.А. 100 задач по программированию / В. А. Дагене.,2013. – 106с.

27. Джамса К. Библиотека программиста Java / К. Джамса., 2016. – 656с.

28. 2D Image Features Detector And Descriptor Selection Expert System / I.Merino, J. Azpiazu, A. Remazeilles, A. Sierra, 2019. – С. 51–61.

29. Заварикин В.М. Основі информатики и вычислительной техники / В. М. Заварикин., 2019. – 556с.

30. Кен А. Язык программирования С++ / А. Кен, 2017. – 378с.

31. Керниган Б. Язык программирования Delphi. / Б. Керниган., 2012. – 391с.

32. Ляхович В.Ф. Руководство к решению задач по основам информатики и вычислительной техники. / В.Ф. Ляхович., 2014. – 127с.

33. Мейнджер Дж. С++ Основі программирования / Дж. Мейнджер., 2017. – 346с.

34. Миков А. И. Информатика. Введение в компьютерніе науки / А. И. Миков., 2018. – 442с.

35. Могилев А. В. Информатика: Учеб. пособие для студ. пед. Вузов / А. В. Могилев., 2019. – 629с.

36. Нотон П. JAVA:Справ.руководство / П. Нотон., 2016. – 447с.

37. Нотон П. Полный справочник по Java / П. Нотон.,2017. – 556с.

38. Ренеган Э.Дж. 1001 адрес WEB для программистов :Новейший путеводитель программиста по ресурсам World Wide Web / Э. Дж. Ренеган., 2017. – 512с.

39. Родли Дж. Создание Java-апплетов / Родли Дж., 2016. – 466с.

40. Томас М. Секреты программирования для Internet на Java / М. Томас., 2017. – 396с.

41. Семакина И. Г. Информатика. Задачник-практикум: В 2 томах / И. Г. Семакина., 2019. – 476с.

42. Сокольский М.В. Все об Intranet и Internet / М. В. Сокольский.,2018. – 254с.

43. Тассел Д. Стиль, разработка, эффективность, отладка и испытание программ / Д. Тассел,2011. – 56с.

44. Тюрин Ю.Н. Анализ данных на компьютере / Ю. Н. Тюрин., 2015. – 384с.

45. Флэнэген Д. Java in a Nutshell / Д. Флэнэген, 2018. – 473с.

46. Чен М.С. Программирование на С++:1001 совет:Наиболее полное руководство по С++ и Visual C++ / М. С. Чен., 2017. – 640с.

47. Эферган М. С++: справочник. QUE Corporation / М. Эферган.,1998. – 256с.

48. Sinha P. Perceiving and Recognizing threedimensional forms / P. Sinha., 2016. – 278p.

49. Pablo F. Fast Explicit Diffusion for Accelerated Features in Nonlinear Scale Spaces / F. Pablo.,2013. – 187c.

50. Методичні рекомендації до виконання дипломної роботи з освітньо-кваліфікаційного рівня “Магістр”. Спеціальність „Комп’ютерні системи та мережі”/О.М. Березький, Л.О. Дубчак, Г.М. Мельник /Під ред. О.М. Березького Тернопіль: ТНЕУ, 2018. 41с.