МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

НКЦ заочної освіти

Кафедра інформаційних технологій

Бакалаврська кваліфікаційна робота

на тему: Розробка програмного забезпечення для класифікації зображень

Виконав студент 5 курсу групи КН-5
Спеціальність122 комп'ютерні науки
Шуманський Сергій Олександрович

Керівник д.т.н., проф.
Андрощук Олександр Степанович

Консультант _____
_____

Рецензент д.т.н., проф.
Мещеряков Володимир Іванович

Одеса 2020

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

БД – база даних

ІС – інформаційна система

ОС – операційна система

ПЗ – програмне забезпечення

ПП – програмний продукт

ПС – програмна система

РСКБД – реляційна система керування базами даних

СКБД – система керування базами даних

IDE – Integrated Development Environment – інтегроване середовище розробки

ВСТУП

У сучасному цифровому світі, з кожним днем зростає потреба в автоматизації різних процесів, наприклад сортування зображень. Цей процес має широкий спектр застосування, починаючи від аналізу змісту всіх зображень на конкретних сайтах, сортування виробів на конвеєрній стрічці, обробки даних відео спостереження і фіксації, і до впорядкувати на комп'ютері користувача.

На сьогоднішній день більша частина готових рішень є закритими і використовуються для потреб різних компаній, наприклад IBM, Intel, AMD і Google. При розробці подібних рішень, розробники часто стикаються з проблемою оптимізації швидкості роботи, точністю сортування, і вимогами до ресурсних потужностей. Активно застосовуються можливості нейронних мереж і штучного інтелекту, але не дивлячись на значні результати в даному напрямку, залишається проблема ресурсу-ємності і тривалого навчання. Так само слід зазначити, що конвеєри сортування використовують прості пристрої, і часто на цих пристроях використання глибокого навчання неможливо або недоцільно. Велика частина готових рішень, є спеціалізованими, і розробляються під потреби конкретних підприємств.

У простих системах обробки зазвичай потрібно отримати кількісну і якісну інформацію з візуальних даних (зображень): такі параметри, як розмір, колір, кількість, напрямок і характер руху, а також контрастні переходи в околицях пікселя зображення, з яких відбувається отримання характерних рис (т.зв. «фічі», від англ. Features). На їх основі проводиться аналіз зображення для отримання корисної інформації.

У системах обробки зображень використовуються такі методи, як машинне навчання (Machine Learning), системи глибокого навчання (Deep Learning) і нейронні мережі (Neural Networks). Ці методи імітують процес розпізнавання і аналізу, який проходить в мозку людини. Або алгоритми, що працюють з пікселями.

Основні підходи до вирішення завдань комп'ютерного зору:

– контурний аналіз;
– пошук за шаблоном (template matching);
– пошук поза шаблонів, зіставлення по ключових точках (feature detection, description matching);
– поєднання даних (data fusion).

Комп'ютерне зір не обмежується тільки цими основними методами, наприклад, можна виділити так звані генетичні алгоритми, які застосовуються, зокрема, для розпізнавання осіб.

Метою даної кваліфікаційної роботи є розробка програмного забезпечення для класифікації зображень.

Концепція програми, що розроблюється, ґрунтується на компактності і простоті використання. Його можна запустити практично на будь-який інтернет речі (IoT), в тому числі за рахунок використаних бібліотек. Воно дозволяє класифікувати зображення з інтернет ресурсів за певними ознаками і має можливістю перегляду результатів вибірки.

Для досягнення поставленої мети були сформульовані наступні завдання:

– провести аналіз предметної області та аналіз вимог до програмної системи, що розроблюється;
– обґрунтувати вибір програмних та апаратних засобів розробки системи;
– провести проектування програмної системи з використанням нотації UML;
– виконати реалізацію програмної систем;
– навести приклади роботи системи по аналізу зображень;
– виконати тестування програмної системи.

Передбачається, що система, що розроблюється, може бути використана для вирішення наступних завдань:

– пошук схожого контексту в інтернет ресурсах (дублікати);

– захист дітей – попередній аналіз зображення і обмеження доступу якщо контекст сумнівний;
– пошук зображень для творчої діяльності – автоматичний пошук зображень за критеріями (для художників, архітекторів і дизайнерів).

Структура дипломної роботи складається з вступу, трьох розділів, висновків, переліку посилань на 16 найменувань, додатків. Повний обсяг проекту становить 50 сторінок, містить 15 рисунків.

1 АНАЛІЗ ПОТОЧНОГО СТАНУ ПРОБЛЕМИ ТА ВИБІР ІНСТРУМЕНТАРІЮ ДЛЯ ОПТИМАЛЬНОГО РІШЕННЯ

1.1 Аналіз сфер застосування завдань класифікації зображень

Розвиток цифрових технологій і постійне зростання обсягів інформації, що надходить, потребують нових рішень обробки даних. Сьогодні інтернет-технології дозволяють доставити повідомлення майже в будь-який віддалений куточок планети і навіть в космос, що призводить до бурхливому розвитку інтернет речей. Будь-який сучасний автомобіль можна описати як ком'ютер на колесах, який постійно обмінюється сотнями і тисячами мегабайм в хвилину. Розумні ваги, холодильники, годинник і двері теж обмінюються даними. Серед загальної різноманітності даними можна виділити досить об'ємну групу медіа-даних – такі як зображення і відео. Відео реєстратори, вебкамери, камери відео спостереження постійно створюють колосальні обсяги інформації для аналізу яких може не вистачати людських ресурсів. Ринок постійно зростає і вимагає все нових і нових рішень.

Розпізнавання облич – практичне застосування теорії розпізнавання образів, в завдання якого входить автоматична локалізація обличчя на нерухомому або рухомому зображенні і, в разі необхідності, ідентифікація особистості за характерними параметрам особи. Розпізнавання облич людей та визначення особистості людини – одна з найуживаніших функцій VCA, яка використовується практично у всіх сучасних системах безпеки на базі інтелектуального CV.

Наведу кілька потенційних сфер застосування технології розпізнавання осіб.

Розпізнавання облич в соціальних мережах. Facebook замінив присвоєння тегів зображень вручну на пропозиції тегів, що автоматично генеруються для кожного зображення і завантажуються на платформу. Facebook вико-

ристовує простий алгоритм розпізнавання осіб для аналізу пікселів на зображенні і порівняння його з відповідними користувачами.

Розпізнавання осіб у сфері безпеки. Простий приклад використання технології розпізнавання осіб для захисту особистих даних – розблокування смартфона «по обличчю». Таку технологію можна впровадити і в пропускну систему: людина дивиться в камеру, а вона визначає дозволити йому увійти чи ні.

Розпізнавання облич для підрахунку кількості людей. Технологію розпізнавання осіб можна використовувати при підрахунку кількості людей, які відвідують будь-який захід (наприклад, конференцію або концерт). Замість того щоб вручну підраховувати учасників, ми встановлюємо камеру, яка може захоплювати зображення осіб учасників і видавати загальна кількість відвідувачів. Це допоможе автоматизувати процес і заощадити час.

Загальне формулювання проблеми розпізнавання осіб (в комп'ютерному зорі) може бути сформульована таким чином: для заданих нерухомих або відео зображень сцени ідентифікуйте або перевірте одного або декількох осіб в сцені, використовуючи збережену базу даних осіб.

Розпізнавання обличчя зазвичай включає дві стадії:

1) виявлення особи, де для пошуку особи виконується пошук фотографії, потім зображення обробляється для обрізки і вилучення особи людини для більш легкого розпізнавання;
2) розпізнавання осіб, де це виявлене і оброблене особа порівнюється з базою даних відомих осіб, щоб визначити, хто ця людина.

1.2 Перелік програмних та апаратних засобів розробки системи

Оскільки однією з проблем при розробці проекту є забезпечення максимальної сумісності при оптимальній продуктивності навіть на слабких пристроях, довелося ретельно підійти до вибору інструментарію, який так само має свої переваги і деякі недоліки. Самостійне написання відомих алго-

ритмів розпізнавання зображення занадто трудомістке завдання, оскільки вимагає тривалого процесу налагодження (тестування).

Так в реалізації цифрового зору на автомобілі використовуються ті ж алгоритми і що і в обраній бібліотеці. Але через обмеження в апаратної частини, використання універсальної бібліотеки не є доцільним. Оскільки апаратні характеристики накладають свої обмеження, що змушує використовувати власну реалізацію алгоритмів і адаптацію них під обладнання. Такий підхід складний, що вимагає довгого тестування, але дає хороший результат для обраної конфігурації обладнання.

У зв’язку з цим, на етапі планування було обрано такі компоненти і цільові напрямки:

– операційна система сімейства – Linux;
– архітектури процесора – x86/64, arm, aarch64;
– обладнання – Intel Edison, Rasberry Pi і Nvidia Jetson Nano;
– компілятор – g ++ / clang;
– система складання – Cmake/Make;
– бібліотека cURL – для роботи з http/https протоколом;
– бібліотека OpenCV – для роботи з зображенням;
– СКБД SQLite – для роботи з базою даних.

Нижче, надане докладне обґрунтування, використання кожного з них, а також опис.

1.3 Обґрунтування вибору операційної системи та архітектури процесору

Linux – є частиною сімейства Unix-подібних операційних систем на базі ядра Linux, що включають той чи інший набір утиліт і програм проекту GNU, і, можливо, інші компоненти. Як і ядро Linux, системи на його основі

як правило створюються і поширюються відповідно до моделі розробки вільного та відкритого програмного забезпечення [1-3][1)].

З'явившись як рішення навколо створеного на початку 1990-х років ядра, вже з початку 2000-х років системи Linux є основними для суперкомп'ютерів і серверів, розширюється застосування їх для вбудованих систем і мобільних пристроїв, деяке поширення системи отримали і для персональних комп'ютерів.

Традиційно системами Linux вважаються тільки ті, які включають в якості компонентів основні програми проекту GNU, такі як bash, gcc, glibc, coreutils, GNOME і ряд інших, в зв'язку з чим часто все сімейство іноді ідентифікується як GNU/Linux, притому існує суперечка про іменування GNU/ Linux. Існує проект стандартизації внутрішньої структури Linux-систем – Linux Standard Base, частина документів якого зареєстрована в якості стандартів ISO.

Нові програми можуть виконуватися на різних мовах (Perl, Pascal, Python, C / C ++, Rust, Java, Bash і ін.). Однак стандартною мовою програмування в середовищі ОС Linux є мова С, який останнім часом все більше замінюється на C++. Це пояснюється тим, що по-перше, сама система Linux написана на мові Сі, а по-друге, мова Сі є одним з найбільш якісно стандартизованих мов.

Архітектура ОС Linux – багаторівнева (рис.1.1). На нижньому рівні, безпосередньо над обладнанням, працює ядро операційної системи. Функції ядра доступні через інтерфейс системних викликів, що утворюють другий рівень. На наступному рівні працюють командні інтерпретатори, команди і утиліти системного адміністрування, комунікаційні драйвери і протоколи, – все те, що зазвичай відносять до системного програмного забезпечення. На-

---

[1)] [1] Колисниченко, Д.Н. Linux. Полное руководство / Д.Н. Колисниченко, Аллен, Питер: СПб: Наука и Техника, 2007.784 с.

[2] Linux – Вікіпедія. URL: https://uk.wikipedia.org/wiki/ Linux (дата звернення 04.05.2020)

[3] Маслинский К. Операционная система Linux. ИНТУИТ, 2005.

решті, зовнішній рівень утворюють прикладні програми користувача, мережеві і інші комунікаційні служби, СУБД і утиліти.

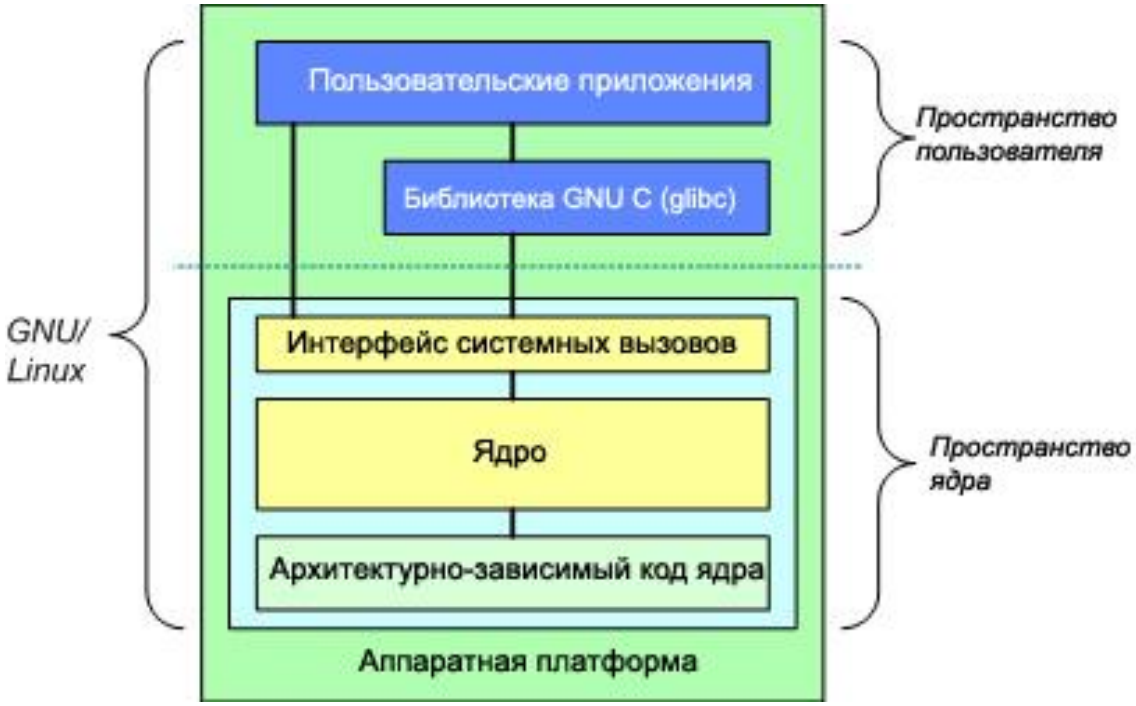

Рисунок 1.1 – Багаторівнева структура ОС Linux

Одним з основних переваг сімейства операційних систем типу Linux і підходу, що виник на їх основі до стандартизації інтерфейсів операційних систем (важлива частина загального підходу відкритих систем) є те, що вони забезпечують єдину операційну середу на комп'ютерах з різною архітектурою. Дана ОС, що є першою в історії мобільної ОС, що забезпечує надійне середовище розробки і використання мобільних прикладних систем, одночасно є практичне підґрунтя для побудови відкритих програмно-апаратних систем і комплексів. Завдяки використанню концепції відкритого коду можливе застосування вже готових рішень і, не меншою мірою завдяки наявності відкритих стандартів і стандартних бібліотек, досить просто здійснюється перенесення програмного забезпечення між різними апаратними платформами, що відповідає меті мого проекту, як і можливість розробки додатків під архітектури x86/64, arm і arm64 на SoC.

Архітектура x86-64 – 64-бітове розширення, набір команд для архітектури x86, розроблене компанією AMD і представлене в 2000 році, що дозволяє виконувати програми в 64-розрядному режимі. Дана архітектура являє собою розширення архітектури x86 з повною зворотною сумісністю. Існує безліч варіантів назви даної архітектури, що призводить до плутанини, хоча, по суті, всі ці назви означають одне і теж: x86-64, AA-64, Hammer Architecture, AMD64, Yamhill Technology, EM64T, IA-32e, Intel 64, x64.

64-розрядний процесор здатний обробляти за один такт 64 біта даних (8 байт) на відміну від 32-бітного, який займається обробкою тільки 32 біта (4 байта) за одиницю часу, тобто швидкість обробки даних в 2 рази вище. Відповідно, і програми для 64-бітових операційних систем працюють набагато швидше, ніж аналогічні, що працюють на 32-бітних ОС.

Підкреслимо основні переваги архітектури x86-64:

– 64-бітовий адресний простір;
– розширений набір регістрів;
– звичний для розробників набір команд;
– можливість запуску старих 32-бітних додатків в 64-бітної операційної системи;
– можливість використання 32-бітових операційних систем.

Архітектура ARM (від англ. Advanced RISC Machine – вдосконалена RISC-машина; іноді – Acorn RISC Machine) – сімейство ліцензованих 32 і 64 бітних мікропроцесорних ядер розробки компанії ARM Limited.

Процесори на даній архітектурі найбільш поширені серед мобільних пристроїв і IoT, а ОС UNIX дозволяє збирати проект, що розроблюється, під дану архітектуру з мінімальними ресурса-витратами.

1.4 Пристрої для тестування програми

На момент реалізації проекту, були доступні наступні пристрої:

– Intel Edison;

– Rasberry PI;
– Nvidia Jetson Nano.

Розглянемо характеристики пристрою Intel Edison ™ (рис.1.2) [4][1)]:

– процесор: Intel Atom (Silvermont) 2 Core 32-bit 500 MHz і Intel Quark micro-controller 32-bit 100 MHz;
– оперативна пам'ять: 1 GB LPDDR3 (PoP memory) - 2 channel 32bits @ 800MT/sec;
– пам'ять користувача: 4 GB eMMC (v4.51 spec) + роз'єм для micro SD карт;
– мережа: 802.11 a / b / g / n Wi-Fi (з підтримкою 5 ГГц, чіп Broadcom 43340) з вбудованою антеною або зовнішньої, і Bluetooth 4.0;
– плата розширення від Sparkfun: USB / OTA, UART, I2C, SPI, GPIO і PWM.

Використовувався для тестування програми під архітектуру x86 32bit.

Характеристики пристрою Raspberry Pi 2 Model B v1.1 (рис.1.3) [5][2)]:

– процесор: ARM Cortex-A7 4 Core 0,9G Hz;
– оперативна пам'ять: 1 GB;
– пам'ять користувача: 4 GB eMMC (v4.51 spec) + роз'єм для micro SD карт;
– мережа: 802.11 a/ b/g /n Wi-Fi;
– Інтерфейс: 4x USB / OTAб HDMI, UART, 1x Audio Jack, I2C, SPI, GPIO і PWM;
– додатково Camera interface (CSI), Display interface (DSI)

Використовувався для тестування програми під архітектуру arm 32bit.

[1)] [4] Intel Edison – Вікіпедія.. URL: https://uk.wikipedia.org/wiki/ Intel_Edison (дата звернення 04.05.2020)

[2)] [5] Raspberry Pi – Вікіпедія. URL: https://uk.wikipedia.org/wiki/ Raspberry_Pi (дата звернення 04.05.2020)

Рисунок 1.2 – Зовнішній вигляд пристрою Intel Edison

Рисунок 1.3 – Зовнішній вигляд пристрою Raspberry Pi 2 Model

Характеристики пристрою NVIDIA® Jetson Nano ™ (рис.1.4) [6][1)]:

– процесор: aarch64 A57 4 Core 64-bit 1,43 GHz і GPU Maxwell 128 Core;
– оперативна пам'ять: 4 GB LPDDR4, 64-bit 25,6 Гбіт/с;
– пам'ять користувача: micro SD карт:
– Інтерфейс: 4x USB 3.0, USB 2.0 Micro-B, GPIO, I2C, I2S, SPI, UART;

[1)] [6] Nvidia Jetson – Wikipedia. URL: https://en.wikipedia.org/wiki/ Nvidia_Jetson (дата звернення 04.05.2020)

– додатково Camera interface (CSI-2), MIPI, HDMI і DisplayPort.

Використовувався для тестування програми під архітектуру aarch64.

В наступних розділах розглянемо програмні засоби необхідні для створення програмної системи та обґрунтуємо їх вибір.

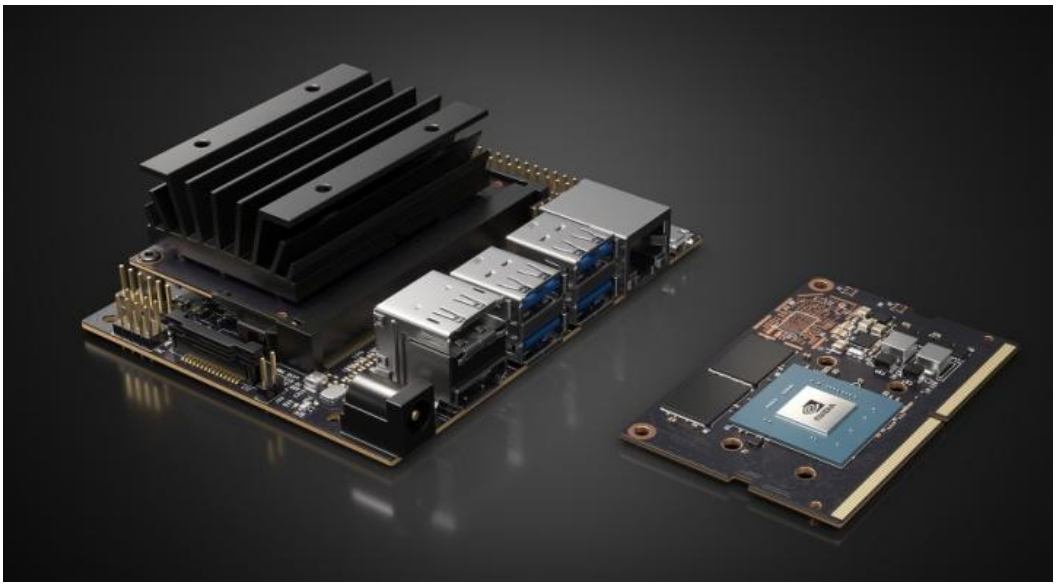

Рисунок 1.4 – Зовнішній вигляд пристрою NVIDIA Jetson Nano

1.5 Обґрунтування вибору мови програмування

Для реалізації проекту була обрана мова програмування C ++ та компіляторі g ++ / clang. C ++ це високо продуктивна мова програмування, яка дозволяє збирати додаток для більшості архітектур. На відміну, наприклад, від мови Java, що використовує Java Run Time Machine, встановлення якої потребує додаткового вільного місця на пристрої, мова C++ більш швидка [7][1)].

Також розглядалася потужна скриптова мова Python, з хорошою підтримкою від спільноті. Є готові рішення для нейронних мереж (TensorFlow) и роботи з мережевих протоколами. Але і для неї все одно буде потрібно підключати зовнішні бібліотеки написані на C і C ++.

[1)] [7] Кениг, Э. Эффективное программирование на C++. Практическое программирование на примерах. Т. 2 / Э. Кениг, Б.Э. Му. М.: Вильямс, 2016. 368 с.

Rust (від Fierfox), Go (від Google), Swift (від Aplle) – це перспективні мови програмування, але в них є проблема з сумісністю з деякими стандартними бібліотеками. Так Rust і Go, могли б мені підійти, оскільки без проблем працюватимуть на заявлених пристроях, але будуть потрібні зайві зусилля для інтеграції зовнішніх бібліотек.

Компілятор g ++ – поширюється на умовах ліцензії GNU, Фондом вільного програмного забезпечення (FSF), для nix-подібних ОС і є C++ компілятором, який управляється за допомогою командного рядка. Підтримує сучасні стандарти C++ 11/14 і C++ 17.

Компілятор Clang – це транслятор для C-подібних мов, створений спеціально для роботи на базі LLVM. Комбінація Clang і LLVM є повноцінним компілятор і надає набір інструментів, що дозволяють повністю замінити GCC. Завдяки архітектурі, заснованої на бібліотеках, Clang (як і LLVM) легко вбудовується в інші додатки [8][1)].

Однією з головних завдань Clang є підтримка інкрементной компіляції, що дозволяє більш тісно інтегрувати компілятор і графічний інтерфейс середовища розробки, на відміну від GCC, який був створений для роботи в класичному циклі «компіляція-лінковка-налагодження». На відміну від GCC, орієнтованого переважно на кодогенерацію, Clang прагне надати універсальний фреймворк для парсинга, індексації, статичного аналізу та компіляції мов сімейства Сі. Зокрема, Clang не виробляє спрощень вихідного коду на етапі парсинга (як це робить GCC), гарантуючи точне відтворення вихідного тексту в абстрактне синтаксичне дерево.

На відміну від GCC, Clang спочатку спроектований для максимального збереження інформації в ході процесу компіляції, в тому числі збереження «зовнішнього вигляду» вихідного коду. Ця особливість дозволяє Clang створювати розгорнуті контекстно-орієнтовані повідомлення про помилки, зрозумілі як для програмістів, так і для середовищ розробки. Модульний дизайн

[1)] [8] Секунов, Н. Программирование на C++ в Linux / Н. Секунов. СПб.: BHV, 2004. 368 с.

компілятора дозволяє використовувати його в складі середовища розробки для індексування коду, підсвічування синтаксису і переробки коду.

Clang підтримує більшість поширених опцій GCC.

1.6 Система автоматичної збірки CMake/Make

Cmake – це кроссплатформенная система автоматизації збирання програмного забезпечення з вихідного коду [9][1)].

Make – утиліта, яка автоматизує процес перетворення файлів з однієї форми в іншу. Найчастіше це компіляція вихідного коду в об'єктні файли і подальша компонування у виконувані файли або бібліотеки.

1.7 Бібліотека cURL

Бібліотека cURL – кросс-платформна (поширювана за ліцензією MIT), службова програма командного рядка, що дозволяє взаємодіяти з безліччю різних серверів по безлічі різних протоколів з синтаксисом URL [10][2)].

Дана бібліотека була обрана оскільки являє собою потужний інструмент який використовувався в браузерах на движку WEBkit, і до сих пір використовується в багатьох сучасних комерційних додатках за рахунок надійності, швидкодії і широкого функціоналу. Може парсити відповіді і генерувати запити. Бібліотека підтримує стиснення потоку, класичними алгоритмами gzip і deflite. Так само дана бібліотека дозволяє використовувати ssl і tls.

В якості альтернативи виступають закриті рішення, або рішення які вимагають значних зусиль з інтеграції через обмежений функціоналу. Наприклад, для роботи з безпечними з’єднаннями треба було б підключити одну з наступних бібліотек: wolfssl, openssl.

[1)] [9] CMake – Вікіпедія. URL: https://uk.wikipedia.org/wiki/CMake (дата звернення 04.05.2020)

[2)] [10] cURL – Вікіпедія. URL: https://uk.wikipedia.org/wiki/cURL (дата звернення 04.05.2020)

Deflate – це алгоритм стиснення без втрат, що використовує комбінацію алгоритмів LZ77 і Хаффмана. Спочатку був описаний Філом Кацем для другої версії його архіватора PKZIP, який згодом був визначений в RFC 1951 (1996 рік).

Deflate вважається вільним від всіх існуючих патентів, і поки залишався в силі патент на LZW (він застосовується в форматі GIF), це призвело до використання Deflate не тільки в форматі ZIP, для якого Кац спочатку його спроектував, але також в компресорі/ декомпресор gzip і в PNG-зображеннях.

GZIP забезпечує стиснення без втрат, іншими словами, вихідні дані можна повністю відновити при розпакуванні. Він заснований на алгоритмі DEFLATE, який використовує комбінацію алгоритму LZ77 і алгоритму Хаффмана.

Використовуючи цю бібліотеки ми можемо підключаться до ресурсів, які вимагають безпечне підключення і підтримують стиснення.

1.8 Бібліотека OpenCV

OpenCV (Open Source Computer Vision Library) – бібліотека алгоритмів комп'ютерного зору, обробки зображень та чисельних алгоритмів загального призначення. Реалізована на мові C / C ++, також розробляється для Python, Java, Ruby, Matlab, Lua та інших мов [11][1)].

OpenCV надає різні класифікатори, які можна використовувати для розпізнавання осіб, очей, автомобілів, і багатьох інших об'єктів. Ці класифікатори, однак, досить прості, вони не навчені з використанням технологій машинного навчання, тому, при розпізнаванні осіб можна розраховувати на точність приблизно в 80%.

[1)] [11] OpenCV – Вікіпедія. URL: https://uk.wikipedia.org/wiki/OpenCV (дата звернення 04.05.2020)

Функціональність OpenCV, яка буде використовуватися для розпізнавання осіб, міститься в декількох модулях. Нижче наводиться короткий опис ключових просторів імен [12,13][1)].

Простір імен CXCORE містить базові визначення типів даних, методи лінійної алгебри та статистики, функції сталості і обробники помилок. Дещо дивно, що тут також є графічні функції для малювання на зображеннях.

Простір імен CV містить методи обробки зображень і калібрування камери. Функції обчислювальної геометрії також знаходяться тут.

Простір імен CVAUX описано в документації OpenCV як такий, що містить застарілий і експериментальний код. Однак найпростіші інтерфейси для розпізнавання осіб знаходяться в цьому модулі. Код, що стоїть за ними, спеціалізується на розпізнаванні осіб, і вони широко використовуються для цієї мети.

Простір імен ML містить інтерфейси машинного навчання.

Простір імен HighGUI містить базові інтерфейси введення/виведення і можливості мультик-платформного управління вікнами.

Простір імен CVCAM містить інтерфейси для доступу до відео через DirectX на 32-бітних платформах Windows.

OpenCV містить в собі такі алгоритми: інтерпретація зображення, калібровка камери, усунення оптичних шумів, визначення подібності, аналіз преміщення об’єкта, сегментація зображення, аналіз жестів.

Основні модулі можна віднести до 4 груп:

1) Модулі core, highgui, які реалізують базову функціональність(базові структури, математичні функції, лінійна алгебра, ввід/вивід зображення)
2) Модулі imgproc, features2d для обробки зображення(фільтрації, геометричні перетворення, сегментація, пошук особливих точок)

[1)] [12] OpenCV Offical Webpage. URL: https://opencv.org/ (дата звернення 04.05.2020)
[13] Bradski G., Kaehler A. Learning OpenCV – Computer Vision with the OpenCV Library. O'Reilly Media. 2008. 580 с.

3) Модулі video, objdetect, calib3d (калібровка камери, аналіз руху, пошук положення в просторі, побудова карта глубин, оптичний потік)
4) Модуль ml, який реалізує алгоритми машинного навчання(метод ближніх сусідів, наївний байесівський класифікатор, машина опорних векторів, нейронні мережі)

1.9 Опис інтерфейсу GPIO

GPIO – інтерфейс для зв'язку між компонентами комп'ютерної системи, наприклад мікропроцесором і різними периферійними пристроями [14][1)]. Контакти GPIO можуть виступати як в ролі входу, так і в ролі виходу – це, як правило, конструюється. GPIO контакти часто групуються в порти.

I2C (IIC), Inter-Integrated Circuit послідовна асиметрична шина для зв'язку між інтегральними схемами всередині електронних приладів. Використовує двонаправлені лінії зв'язку (SDA і SCL), застосовується для з'єднання низько швидкісних периферійних компонентів з процесорами і микроконтроллерами (наприклад, на материнських платах, у вбудованих системах, в мобільних телефонах).

Широтно-імпульсна модуляція, pulse-width modulation (PWM)) – процес управління потужністю методом пульсуючого включення і виключення приладу. Розрізняють аналогову ШІМ і цифрову ШІМ, двійкову (дворівневу) ШІМ і трійкову (трьох рівневу) ШІМ.

GPIO використовуються: в пристроях з нестачею висновків (пинов, контактів):

– інтегральних схемах, таких як однокристальні системи (SoC), вбудованих і спеціальних системах (embedded і custom hardware) і програмованих логічних пристроях (наприклад FPGA);

– в багатофункціональних чіпах: керуючих живленням, аудіокодек і відеокартах.

[1)] [14] GPIO – Вікіпедія. URL: https://uk.wikipedia.org/wiki/GPIO (дата звернення 04.05.2020)

– у вбудованих системах (наприклад, Arduino, BeagleBone, різні PsoC комплекти і Raspberry Pi) широко використовують GPIO для читання інформації від різних зовнішніх датчиків (ІК, відео, температура, орієнтації по 3 осях, прискорення), а також для управління двигунами постійного струму (використовуючи ШІМ), аудіо, ЖК-дисплеями, або світлодіодами для індикації стану.

Дані сфери використання, повністю задовольняють потреби даного проекту. У разі необхідності, проект може бути легко адаптований, до цих технологій. Наприклад, можна змінити джерело зображення на цифрову камеру, яка в свою чергу, може застосовуватися на конвеєрній лінії.

1.10 Вбудована крос-платформна БД SQLite

SQLite – компактна вбудована СКБД. Слово «вбудовується» (embedded) означає, що SQLite не використовує парадигму клієнт-сервер, тобто движок SQLite не є окремо працюючим процесом, з яким взаємодіє програма, а являє собою бібліотеку, з якої програма компонується, і движок стає складовою частиною програми. Таким чином, в якості протоколу обміну використовуються виклики функцій (API) бібліотеки SQLite. Такий підхід зменшує накладні витрати, час відгуку і спрощує програму.

SQLite зберігає всю базу даних (включаючи визначення, таблиці, індекси і дані) в єдиному стандартному файлі на тому комп'ютері, на якому виповнюється програма. Простота реалізації досягається за рахунок того, що перед початком виконання транзакції записи весь файл, який зберігає базу даних, блокується; ACID-функції досягаються в тому числі за рахунок створення файлу журналу [15][1)].

Не останнім пунктом на користь вибору того чи іншого движка для роботи з базами даних є його швидкість. На сайті розробників можна знайти порівняльні характеристики швидкодії SQLite, MySQL і PostgreSQL. За ре-

[1)] [15] Rick F. van der Lans. The SQL Guide to SQLite. First Lulu edition 2009. 524 p.

зультатами практично всіх тестів SQLite перевагає своїх клієнт-серверних суперників, розбіг у швидкості іноді вражає: деякі операції SQLite виконує в 2-3 рази швидше, ніж MySQL, а для PostgreSQL ця цифра доходить до 20 разів. Втім, це й не дивно, так як клієнт-серверні движки просто навіть за визначенням більш повільні.

Програє ж конкурентам SQLite тільки в двох "номінаціях": при виконанні операцій CREATE INDEX і DROP TABLE. Втім, розрив у цих тестах не настільки значний, а операції ці виконуються не так часто, щоб говорити про критичність цих відмінностей в швидкостях. В цілому, SQLite показує більш високу швидкість.

Таким чином, використання SQLite як вбудованого SQL-движка куди більш переважно, ніж використання в аналогічних цілях MySQL або особливо PostgreSQL. Останній, до того ж, дуже-дуже сильно відрізняється від SQLite за своїми розмірами, хоча, звичайно, значно виграє за можливостями.

SQL-движки з відкритим вихідним кодом отримують все більшу популярність серед розробників і користувачів. SQLite в цьому сенсі не виняток – його вихідний код відкритий, і не просто відкритий. Справа в тому, що SQLite поширюється по ліцензії public domain. Тобто зазвичай програмне забезпечення з відкритим вихідним кодом поширюється вільно, однак авторські права на сам продукт і на його вихідний код зберігаються за його розробниками. Для public domain-продуктів це правило не діє: і вихідний код, і продукт не захищені авторськими правами і можуть бути використані ким завгодно в яких завгодно цілях.

Крім відкритості (і частково внаслідок неї), перевагою SQLite є надійність: розробники стверджують, що протестовано на працездатність і стійкість понад 95% коду SQLite.

SQLite обраний для зберігання даних обробки даних, як рішення, що вимагає мінімальних обчислювальних ресурсів, так як бібліотека поставля-

ється з відкритим вихідним кодом. В якості альтернативи можна було б використовувати MySQl, але він не такий компактний як SQLite [16][1)].

[1)] [16] SQLite – Вікіпедія. URL: https://uk.wikipedia.org/wiki/SQLite (дата звернення 04.01.2020)

2 ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Вимоги до програмного забезпечення

Головна мета кваліфікаційної роботи – розробити програмне рішення для автоматичної класифікації зображень з інтернет ресурсів за заздалегідь визначеними властивостями.

Перелічимо основні вимоги до програмного забезпечення, що розроблюється:

1) Програма повинна працювати на вбудовуваних рішення (IoT).
2) Програма повинна працювати на різних архітектурах.
3) Програма повинна використовувати базу даних.
4) Програма повинна працювати з інтернет ресурсами (Web-сайт).
5) Програма повинна підключаться до інтернет ресурсів по протоколах HTTP і HTTPS.
6) Програма має вміти аналізувати HTML розмітку і пошуку тегів із зображенням.
7) Програма повинна вміти працювати з різними форматами зображення (jpeg, png, tiff).
8) Програма повинна класифікувати зображення за різними ознаками.
9) Програма може використовувати графічний інтерфейс для взаємодії з користувачем (user).
10) Програма може управляти пристроями підключеними через gpio, i2c і pwm.

Під базовими критеріями класифікації будемо розуміти:

– розмір зображення;
– співвідношення сторін (вертикальний / горизонтальний);
– кольоровий/монохромний;
– переважаючий колір зображення (більше 60%, чорний, білий, червоний, зелений, синій);
– наявність сигнатури (прямі лінії, кола, тощо).

2.2 Розробка структури бази даних

В робот була використана проста реляційна СКБД SQLite. Реляційна модель підтримує єдину логічну структуру, яка називається відношенням. Це двомірна структура даних, яка відповідає таблиці у фізичній БД. Атрибути відношення відповідають стовбцям в таблиці. Фактичні значення даних атрибутів відношення зберігаються в кортежах, або в строках таблиці. На структуру даних накладаються обмеження цільності – це по-перше, первинний ключ, коли кортеж залежить і може бути однозначно визначений від значення ключового атрибуту. Також, можна визначити унікальні ключі відношення. Це більш слабке обмеження, так як унікальний ключ може містить пусті значення, а первинний – ні. Зв'язок між відношеннями зазвичай організують за допомогою атрибута, загального для двох відношень. Цей атрибут зазвичай є первинним ключем однієї таблиці та зовнішнім ключем іншої. Правила посилальної цільності потребують, щоб значення зовнішнього ключа одного відношення посилалося на значення первинного ключа другого відношення.

Перелічимо операції які виконуються над реляційними структурами даних:

1) додавання нових кортежем (INSERT);
2) видалення кортежем, що існують (DELETE);
3) модифікація кортежем, що існують (UPDATE);
4) вибірка значень з відношення (SELECT).

Для вирішення поставленого завдання била використана досить проста, але при цьому ефективна структура бази даних. База даних містить дві основні таблиці і одну допоміжну, а так само два тригера для каскадного видалення даних.

Розроблена структура бази даних дозволяє зіставляти кілька властивостей одного зображення. Для додавання нового типу в базу даних не вимагає зміна її структури. Використання таблиці Type замість програмних індексів дозволить використовувати цю базу даних окремо від програмного рішення.

Таблиця File:

– поле id: ineger – унікальний ідентифікатор;

– поле url: text – містить адресу знайденого зображення на ресурсі;

– поле date: date – містить дату та час додавання запису в таблицю.

Таблиця Type:

– поле id: ineger – унікальний ідентифікатор;

– полк name: text – текстове позначення фільтра.

Таблиця Link:

– поле id: ineger – унікальний ідентифікатор;

– поле uri: ineger – індекс елемента з таблиця Uri;

– поле type: ineger – індекс елемента з таблиця Type.

Крім того проект містить тригер type_del для автоматичного видалення записів з допоміжної таблиці після видалення властивості зображення:

```
CREATE TRIGGER IF NOT EXISTS file_del AFTER DELETE ON file
FOR EACH ROW BEGIN
DELETE FROM link WHERE _file=OLD._id;
END;
```

Тригер type_del для автоматичного видалення записів з допоміжної таблиці після видалення властивості зображення:

```
CREATE TRIGGER IF NOT EXISTS type_del AFTER DELETE ON type
FOR EACH ROW BEGIN
DELETE FROM link WHERE _type = OLD._id;
END;
```

file_view – просте уявлення для компактного відображення інформації по зображеннях:

```
CREATE VIEW IF NOT EXISTS file_view AS –
SELECT file._url AS url, group_concat (type._name) AS type
FROM file, type, link
WHERE link._file = file._id AND link._type = type._id
GROUP BY file._id
ORDER BY file._url AND type._name;
```

type_view – просте уявлення для перегляду і пошуку за властивостями зображення:

```
CREATE VIEW IF NOT EXISTS type_view AS
SELECT file._url AS url, type._name AS name
```

```
FROM file, type, link
WHERE link._file = file._id AND link._type = type._id
ORDER BY file._url AND type._name;
```

Два уявлення file_view і type_view використовуються для спрощення конструкції sql-запиту (select). Структура бази даних програмної системи наведена на рис.2.1.

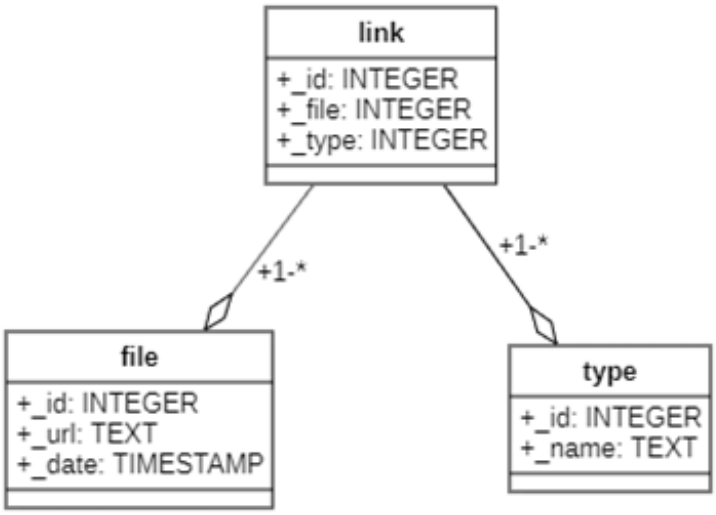

Рисунок 2.1 – Структура бази даних програмної системи

2.3 Архітектура програмного рішення

У програмі використовується багатопотокова обробка, заснована на класичній схемі черзі завдань. Ядром всієї програми є TaskManager – в його завдання входить створення і управління допоміжними потоками, а також розподіл завдань між ними.

Механізм синхронізації менеджера заснований на зв'язці очікування і оповіщення. Так наприклад при ініціалізації менеджер створює дочірні потоки. У свою чергу потоки є циклами з умовним виходом і з механізмом очікування доступних завдань. Основний потік (з якого викликається додавання нових завдань для TaskManager) виконує додавання завдання в чергу і розсилку оповіщення. Перший вільний дочірній потік, що знаходиться в стані очікування, обробляє подію і забирає завдання з черги. По закінченню обробки

дочірній потік входить в стан очікування до наступної доступної задачі або ж команди завершення роботи (TaskManager :: stop ()). У поточному проекті використовується три різні реалізації завдань для обробки, що реалізують інтерфейс Task. Для взаємодії з TaskManager всі завдання реалізують метод do_work – це дозволяє менеджеру виконувати завдання абстрагуючись від фактичної реалізації. Кожне завдання виконує свою окрему роль:

а) завдання HttpParser:

1) підключення за протоколом HTTP або HTTPS;
2) пошук посилань на зображення в HTML розмітці;
3) додавання нових завдань (ImageLoader) в менеджер завдань;
4) якщо властивості даного завдання передбачає рекурсивний пошук, то буде створена нова задача HttpParser з вкладеною адресою.

б) завдання ImageLoader;

1) підключення до віддаленого ресурсу і завантаження контексту (зображення);
2) для вирішення проблем пов'язаних з колізією в найменуванні збережених файлів, файли зберігаються під новим ім'ям, в якості генератору для імен використовується хеш функція;
3) у випадку успіху – додавання нового завдання (ImageAnalyzer) в менеджер завдання.

в) завдання ImageAnalyzer – це основне завдання для роботи з вмістом зображення;

1) перевірка вмісту (є або вміст зображенням);
2) перевірка атрибутів (ширина, висота);
3) перевірка палітри (кольоровий або монохромний);
4) пошук переважаючого кольору на зображенні;
5) пошук фігур (лінії, кола тощо);
6) запис результату в базу даних.

На рис.2.2 представлена діаграма класів проекту.

Діаграма класів – структурна діаграма мови моделювання UML, що демонструє загальну структуру ієрархії класів системи, їх кооперацій, атрибутів (полів), методів, інтерфейсів і взаємозв'язків між ними. Широко застосовується не тільки для документування та візуалізації, але також для конструювання за допомогою прямого або зворотного проектування.

Метою створення діаграми класів є графічне представлення статичної структури декларативних елементів системи (класів, типів і т.п.) Вона містить в собі також деякі елементи поведінки (наприклад – операції), проте їх динаміка повинна бути відображена на діаграмах інших видів (діаграмах комунікації, діаграмах станів).

При поданні сутностей реального світу розробнику потрібно відобразити їх поточний стан, їх поведінку і їх взаємні відносини. На кожному етапі здійснюється абстрагування від незначних деталей і концепцій, які не належать до реальності (продуктивність, інкапсуляція, видимість і т. п.).

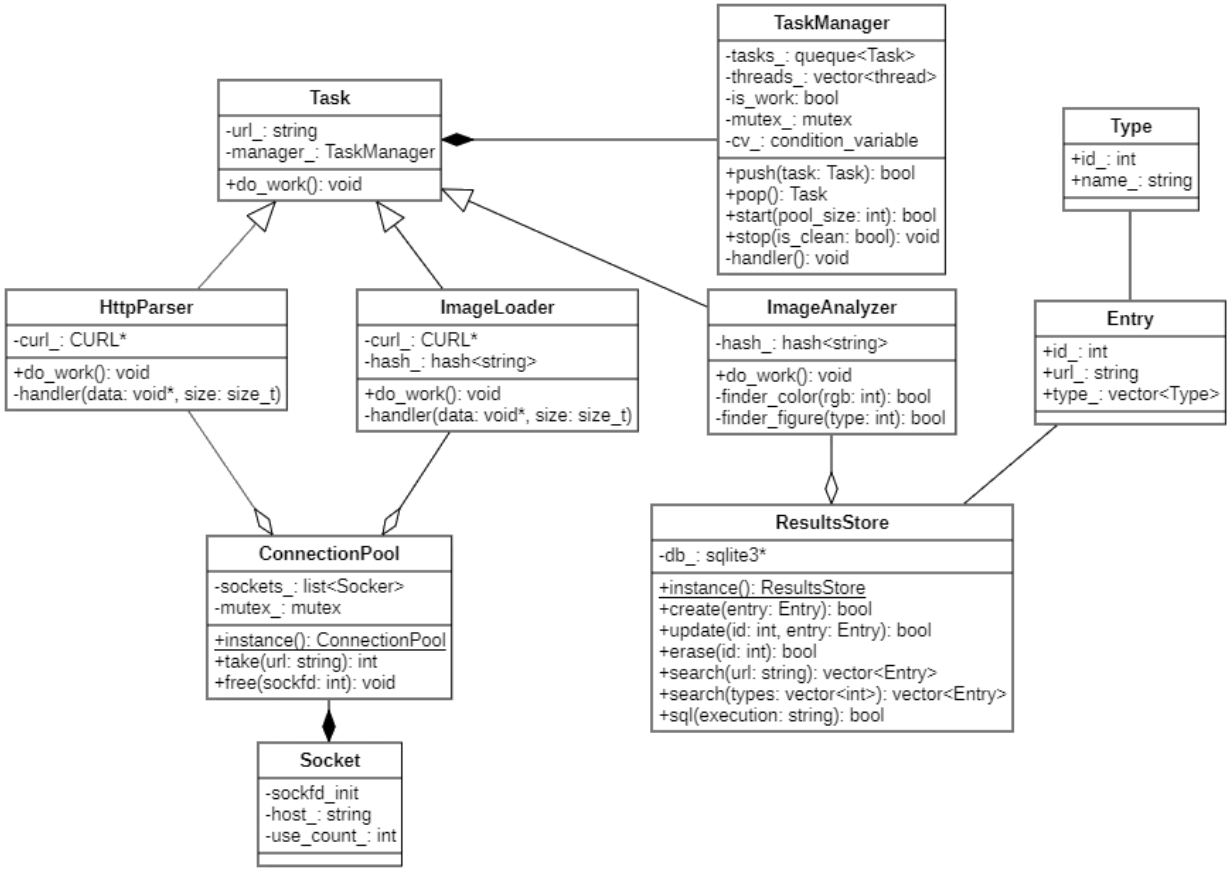

Рисунок 2.2 – Діаграма класів

Для роботи з базою даних використовувався ResultStore клас. Даний клас надає простий і безпечний спосіб роботи з sqlite і дозволяє виконувати запити не прив'язуючись до внутрішньої реалізації самої бази даних. Використання подібних класів-обгорток зручно тим, що при необхідності можна буде замінити реалізацію над змінюючи всієї архітектури. Наприклад, зміна sqlite3 на MySQL потребує тільки змінити бібліотеку, що підключається, та використовувані методи.

У програмі використовуються допоміжні класи такі як ConnectionPool. Роль даного класу в забезпеченні перевикористання ресурсів. На даний момент цей клас виконує прості завдання: створює сокет за запитом і закриває після використання. Деякі ресурси можуть мати обмеження на одночасне підключення і можуть блокувати наступні підключення, що може привести до неправильної роботи програми. Таким чином, через цей клас можна задати обмеження для одночасного підключення до одного ресурсу.

2.4 Послідовність роботи програми

Перелічимо в цьому розділі основні послідовності роботи програми. Діаграма послідовності наведена на рис.2.3.

Діаграми послідовностей використовуються для більш детального опису логіки сценаріїв використання. Це відмінний засіб документування проекту з точки зору сценаріїв використання.

Діаграми послідовностей зазвичай містять об'єкти, які взаємодіють в рамках сценарію, повідомлення, якими вони обмінюються, і які повертаються результати, пов'язані з повідомленнями. Втім, результати, що часто повертаються, позначають лише в тому випадку, якщо це не очевидно з контексту.

Об'єкти позначаються прямокутниками з підкресленими іменами (щоб відрізнити їх від класів).

Повідомлення (виклики методів) – лініями зі стрілками.

Результати, що повертаються – пунктирними лініями зі стрілками.

Прямокутники на вертикальних лініях під кожним з об'єктів показують "час життя" (фокус) об'єктів.

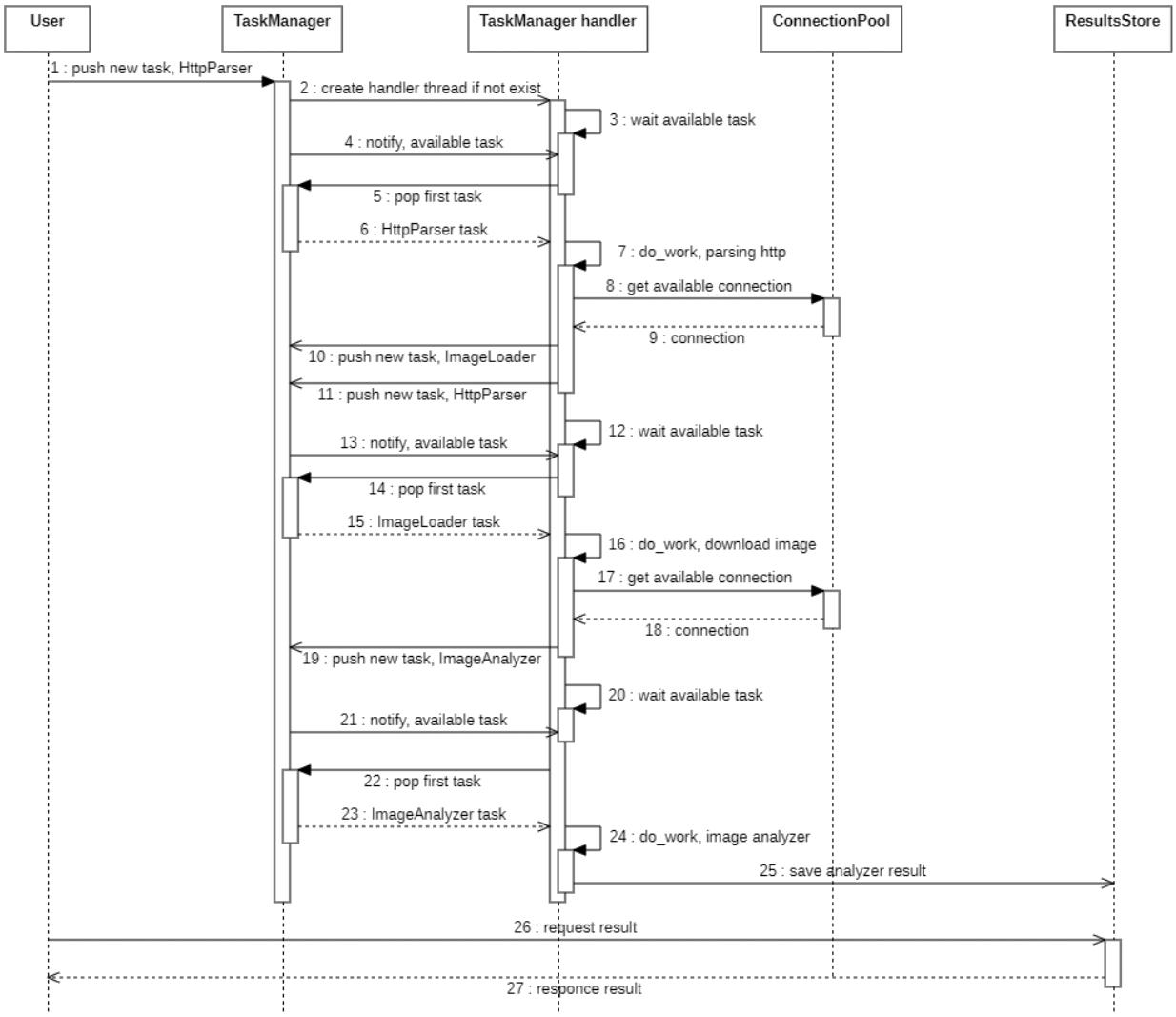

Рисунок 2.3 – Діаграма послідовності

Аналіз інтернет ресурсу:

1) додавання нового завдання (аналізу ресурсу) в чергу завдань;
2) витяг завдання з основної черги і початок обробки в дочірньому процесі;
3) підключення до ресурсу і завантаження вмісту (буфера);
4) пошук в буфері сигнатур (посилань на зображення);

5) додавання нових завдань в чергу (завантаження зображення);
6) витяг завдання з основної черги і початокобробки в дочірньому процесі;
7) підключення до ресурсу та завантаження вмісту через файлову систему;
8) додавання нових завдань в чергу (аналіз зображення);
9) витяг завдання з основної черги і початок обробки в дочірньому процесі;
10) аналіз зображення і запис результату в базу даних.

Перегляд результатів:

1) виконання запиту в базу даних;
2) перегляд результату.

Для складання програми використовувався CMake. Це дозволило використовувати зв'язку cross-compiler-c ++ & cmake & (toolchain) – можна збирати програми для цільового пристрою на високо продуктивних серверах, а після встановлювати або копіювати готову програму на пристрій.

Використання C ++ старше 11 стандарту (в моєму випадки 14), дозволило використовувати multithreading з STL бібліотеки замість класичного підходу з POSIX pthread для Unix. Таке підхід розширює список операційних систем для використання даного коду. Наприклад OS Windows – так як компілятор Visual Studio підтримує STL.

Для роботи з багатопоточностью було використано:

– std :: thread – клас для створення і управління потоками;
– std :: mutex і std :: lock_guard – клас для управління мютексом, для синхронізації використання загальних ресурсів;
– std :: condition_variable – клас є примітивом синхронізації, який можна використовувати для блокування потоку або декількох потоків одночасно, поки інший потік не змінить загальну змінну (умова) і не повідомить його.

Аналіз зображення. Стороння бібліотека OpenCV володіє великим набором різного функціоналу для обробки зображення і відео. Для цього проекту я скористався наступними:

1) читання атрибутів зображення такі як ширина і висота, а так же формат зображення;
2) маніпуляції із зображенням: зміна палітри, розмиття, контрастність, пошуку кордонів, обертання зображення і робота з масками.

Нижче розглянемо приклади використаних в проекті запитів. Використання бази даних SQLite:

1) для використання необхідно підключити sqlite3.h;
2) для відкриття або створення бази даних використовувати наступний код:

```
if (SQLITE_OK! = sqlite3_open (name, & db_)) {
return false;
}
```

де name – ім'я/шлях до файлу, а db_ – покажчик на структуру sqlite3, яку заповнюють;

3) для запобігання витоку пам'яті і втрати даних необхідно закривати використовуваний базу після того як була завершена робота з нею:

```
sqlite3_close (db_);
```

Приклад виконання запиту:

```
if (SQLITE_OK! = sqlite3_exec (db_, sql, sql_callback,
0, & message)) {
std :: cerr << "SQL error:" << message << std :: endl;
sqlite3_free (message);
return false;
}
```

де sql – рядок, що містить текст запиту, а db_ – покажчик на структуру sqlite3, sql_callback – покажчик на функцію (її можна використовувати для налагодження або для перевірки введених даних, а так само для обробки записів для вибірок) і message – це покажчик на рядок куди у випадки помилки буде записаний текст помилки. Знову ж для запобігання витоку пам'яті необхідно виконувати очищення – sqlite3_free.

Так як сама база даних містить механізм самоочищення від невикористовуваних зв'язків (використання тригерів при створенні) – для поставленого завдання цих методів повинно вистачати.

Приклад HTTP-запитів/відповідей (curl):

1) для використання необхідно підключити curl/curl.h;
2) для ініціалізації підключення використовувати наступний код:

```
CURL * curl_ = curl_easy_init ();
if (CURLE_OK! = curl_easy_setopt (curl_, option_name,
option_value)) {
return false;}
```

де option_name і option_value – це параметри для настройки з'єднання. Для настройки з'єднання були застосовані наступні параметри:

```
{CURLOPT_VERBOSE, 0L}//Display verbose information
{CURLOPT_NOPROGRESS, 1L}//Shut off the progress meter
{CURLOPT_NOSIGNAL, 1L}//Do not install signal handlers
{CURLOPT_TIMEOUT, 20L}//Timeout for the entire request
{CURLOPT_FOLLOWLOCATION, 1L}//Follow HTTP redirects
{CURLOPT_AUTOREFERER, 1L}//Automatically set Referer:
header
{CURLOPT_MAXREDIRS, 3L}//Maximum number of redirects to
follow
{CURLOPT_REDIR_PROTOCOLS, CURLPROTO_HTTP |
CURLPROTO_HTTPS}//Protocols to allow redirects to
```

А також кілька опцій для параметризації окремих завдань:

– CURLOPT_USERAGENT – рядок містить кодифікатор браузера (деякі ресурси можуть обмежити доступ у випадку відсутності даного поля в запиті);
– CURLOPT_WRITEFUNCTION – покажчик на функцію який буде викликаний після обробки заголовка і початку передачі контексту (це може бути текст сторінки або бінарні дані зображення);
– CURLOPT_ACCEPT_ENCODING з даними "gzip, deflate" – дозвіл використання стиснення переданих даних – цей параметр має рекомендаційний характер, в залежності від настройки сервера – дані

можуть стискатися, це збільшує навантаження на процесор, але при цьому зменшує загальний обсяг трафіку передається через мережу;

– CURLOPT_URL – адреса ресурсу до якого виконується запит;
– CURLOPT_WRITEDATA – покажчик на дані (покажчик на завдання).

3) для виконання запиту:

```
if (CURLE_OK! = ccurl_easy_perform (curl_)) {
return false;
}
```

4) для перевірки статусу відповіді:

```
char * effective_url = nullptr;
if (CURLE_OK == code) {
code = curl_easy_getinfo (curl_, CURLINFO_EFFECTIVE_URL,
& effective_url);
}
long response_code = 0;
if (CURLE_OK == code) {
code = curl_easy_getinfo (curl_, CURLINFO_RESPONSE_CODE,
& response_code);
}
char * content_type = nullptr;
if (CURLE_OK == code) {
code = curl_easy_getinfo (curl_, CURLINFO_CONTENT_TYPE,
& content_type);
}
```

Так як сайти можуть використовувати перенаправлення – необхідно перевірити який ресурс був завантажений або оброблений.

Так само необхідно перевірити код відповіді. Так наприклад код 2xx – запит пройшов успішно. 3xx – попередження, в більшості випадків можна знехтувати. А ось 4xx або 5xx – помилка.

Так само необхідно перевірити тип завантаженого контенту.

Для генерації імені тимчасового файлу використовувалася бібліотека string або STL.

std :: hash <std :: string> – це проста хеш функція дозволяє вирішити колізія імен. Ця функція не вирішує всі проблеми збіг хешів при різний вхідних

даних. При необхідності її можна замінити sha256sum з openssl – це дозволить скоротити ймовірність появи колізій.

3 АНАЛІЗ ЗОБРАЖЕННЬ ЗА ДОПОМОГОЮ ПРОГРАМНОЇ СИСТЕМИ

Будь-яке зображення можна представити у вигляді двомірного масиву.

У бібліотеці OpenCV для цього використовується Mat клас (від слова матриця). Елементом такої матриці виступає піксель, який представляє собою колір. OpenCV вміє працювати з різними форматами кольорів, наприклад, такі як RGB (red, green, blue), HVS (hue, saturation, value), монохромний (градація сірого).

Для детектування переважаючого кольору на зображенні зручніше використовувати HSV (рис.3.1), тому що колір в цій системі представляється як відтінок, інтенсивність і насиченість. Це дозволяє простим і зручним способом задавати необхідний діапазон для пошуку кольорів. Шляхом обертання hue ми задаємо відтінок або діапазон відтінків. На відміну від технічних креслень або зображень створених в простих редакторах, в фотографіях майже не зустрічаються пік селі, що мають одне значення. Наприклад для пошуку всіх пікселів підпадають під критерій жовтий колір, в системі HSV досить вказати hue – 40-45, saturation і value – 30-80. Для подібного пошуку в системі RGB цього результату можна досягти тільки для певних випадків.

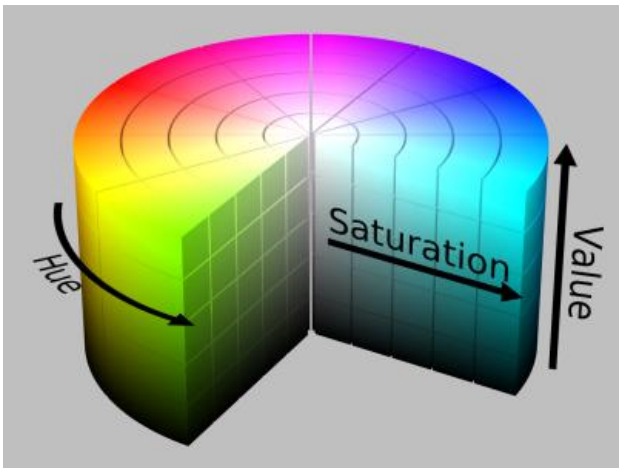

Рисунок 3.1 – Кольорова модель HSV

Нижче наводяться приклади використання для зображення, наведеного на рис.3.1:

– читання зображення з файлу:

```
cv :: Mat image = cv :: imread (name, cv :: IMREAD_COLOR);
```

– конвертація колірної схеми:

```
cv :: cvtColor (image, hsv, cv :: COLOR_BGR2HSV);
```

– отримання результату пошуку:

```
cv :: inRange (hsv, cv :: Scalar (30, 30, 40), cv :: Scalar (80, 80, 45), thresholded);
```

Результатом даного виклику буде нове зображення, а точніше чорно-білий фільтр (рис.3.2). Аналіз співвідношення світлих точок до загальної кількості відповідатиме процентному заповненню зображення цього відтінку.

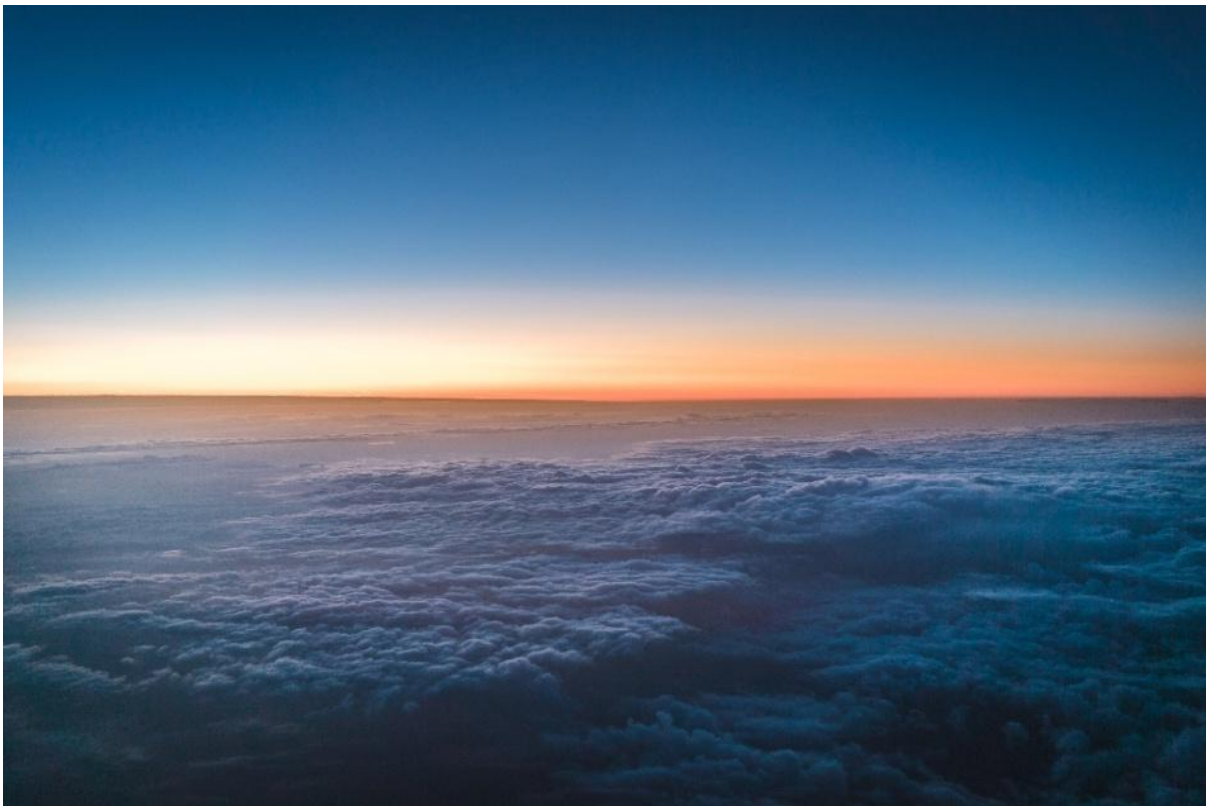

Рисунок 3.1 – Оригінальне зображення для пошуку максимального заповнення відтінків

Рисунок 3.2 – Результат пошуку синього відтінку

Розглянемо як відбувається пошук прямих ліній на зображенні (рис.3.3).

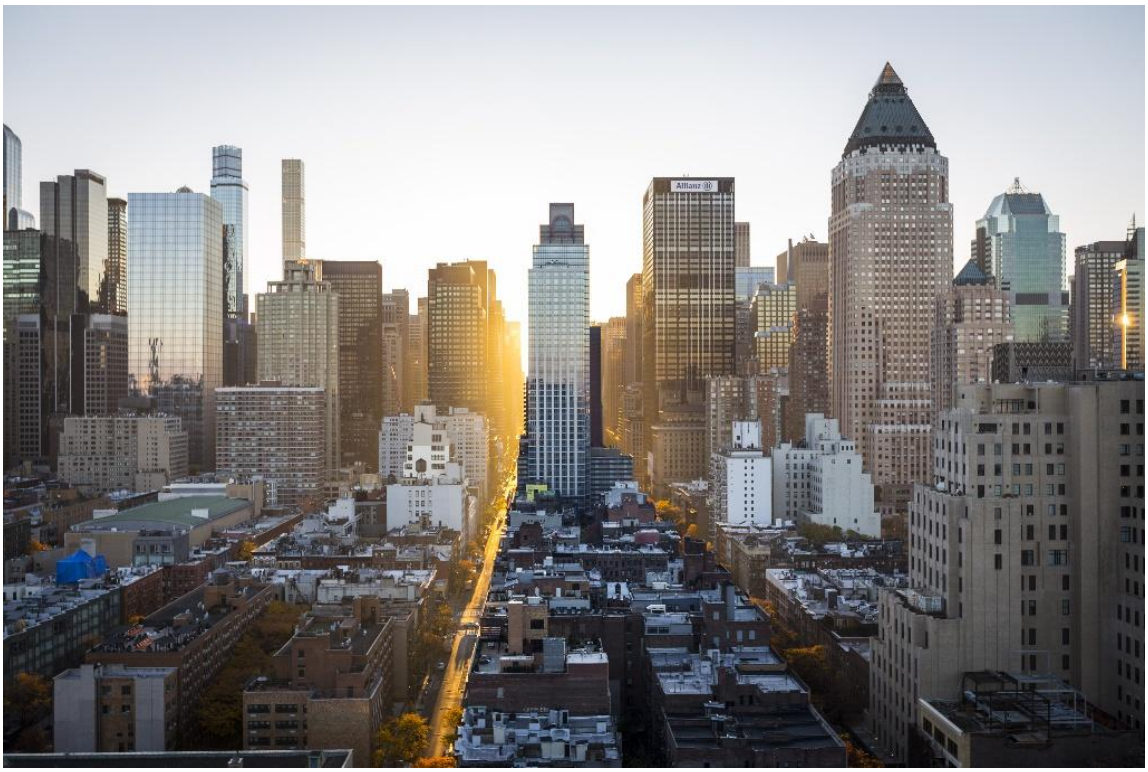

Рисунок 3.3 – Оригінал зображення для пошуку прямих ліній

Для пошуку зображень потрібна попередня підготовка. Для простоти роботи кольорове зображення слід перетворити в монохромне (рис.3.4). Так як суть пошуку заснована на пошуку контрастних фрагментів, а у різних кольорів може бути однакова насиченість в монохромному поданні. Це може привести до помилкового спрацьовування при пошуку кордонів.

Рисунок 3.4 – Зображення після перетворення до монохромного

Для перетворення формату колірної схеми використовується наступна функція:

```
cv :: cvtColor (src, dst, cv :: COLOR_BGR2GRAY);
```

Для прискорення роботи деяких алгоритмів, а так само для ігнорування незначних деталей, зображення можна зменшити. Це знизить загальний час обробки, а так само загальний обсяг споживаної пам'яті.

```
cv :: resize (image, image, cv :: Size (image.cols * ratio, image.rows * ratio), 0, 0, cv :: INTER_LINEAR);
```

У цьому рядку коду відбувається зміна розміру зображення image з подальшим записом результату в image. Для перерахунку кольору кожного пікселя використовується лінійний алгоритм (LINEAR).

Для зменшення «шумів» на фотографіях необхідно застосувати розмиття (рис.3.5). Розмиття позбавить зображення від зайвої деталізації, а так само прибере незначний перехід кольорів між рядом пікселями, що знаходяться поруч.

Рисунок 3.5 – Зображення після застосування розмиття

```
cv :: GaussianBlur (src, dst, cv :: Size (5, 5), 1.5);
```

Ця функція виконує згладжування поруч розташованих кольорів. Ступінь розмитості задається розміром size. Чим більше розмір тим більше ефект розмиття, для роботи цього алгоритму розмір повинен обчислюватися за формулою 2 * i + 1, наприклад 3, 5, 7, 9 тощо.

Для пошуку меж використовується наступна функція:

```
cv :: Canny (src, dst, 50, 200, 3);
```

Ця функція виконує пошук кордону по зазначеним критеріям.

Для безпосереднього пошуку прямих на зображенні потрібні всі вище згадані дії для відсіювання помилкових спрацьовувань і для корегування визначення. Функція виклику має вигляд:

```
std :: vector <cv :: Vec4i> lines;
HoughLinesP (edge, lines, 1, CV_PI / 180, 20, 20, 5);
```

Результатом роботи цієї функції буде масив відрізків (рис.3.6, рис.3.7). Мінімальна довжина відрізка, а так само і товщина задається в аргументах. Кількість елементів в масиві в моїй програмі використовується як критерій для фільтра зображень.

Нижче наведено код для пошуку ліній в зображенні:

```
cv::Mat image = cv::imread(name, cv::IMREAD_COLOR);
if (image.empty()) {
  return false;
}
constexpr double kMaxSize = 1024.0;
double ratio = 1.0;
if (image.cols > kMaxSize) {
  ratio = kMaxWidth / image.cols;
} else if (image.rows > kMaxSize) {
  ratio = kMaxWidth / image.rows;
}
cv::Size new_size(image.cols * ratio, image.rows * ratio);
cv::resize(image, image, new_size, 0, 0, cv::INTER_LINEAR);
cv::Mat gray;
cv::cvtColor(image, gray, cv::COLOR_BGR2GRAY);
cv::Mat blur;
cv::Size blur_size(11, 11);
cv::GaussianBlur(gray, blur, blur_size, 1.5);
cv::Mat edge;
cv::Canny(blur, edge, 50, 200, 3);
std::vector<cv::Vec4i> lines;
HoughLinesP(edge, lines, 1, CV_PI / 180, 20, 20, 5);
return lines.size();
```

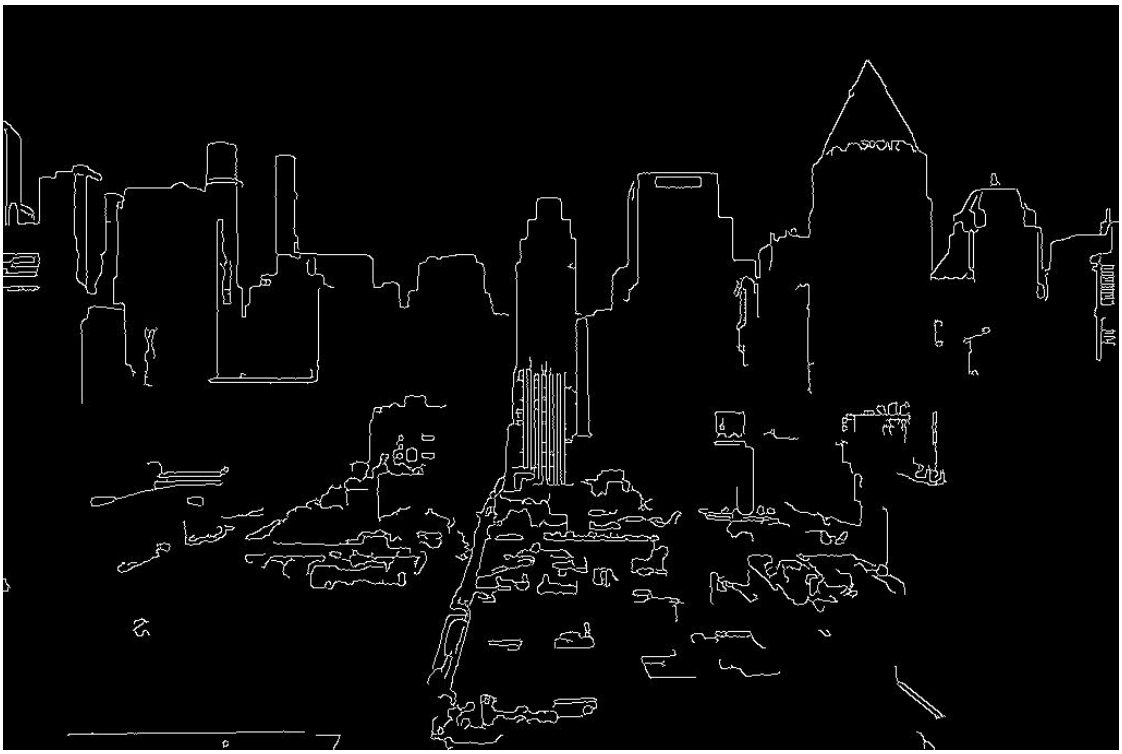

Рисунок 3.6 – Результат пошуку кордонів

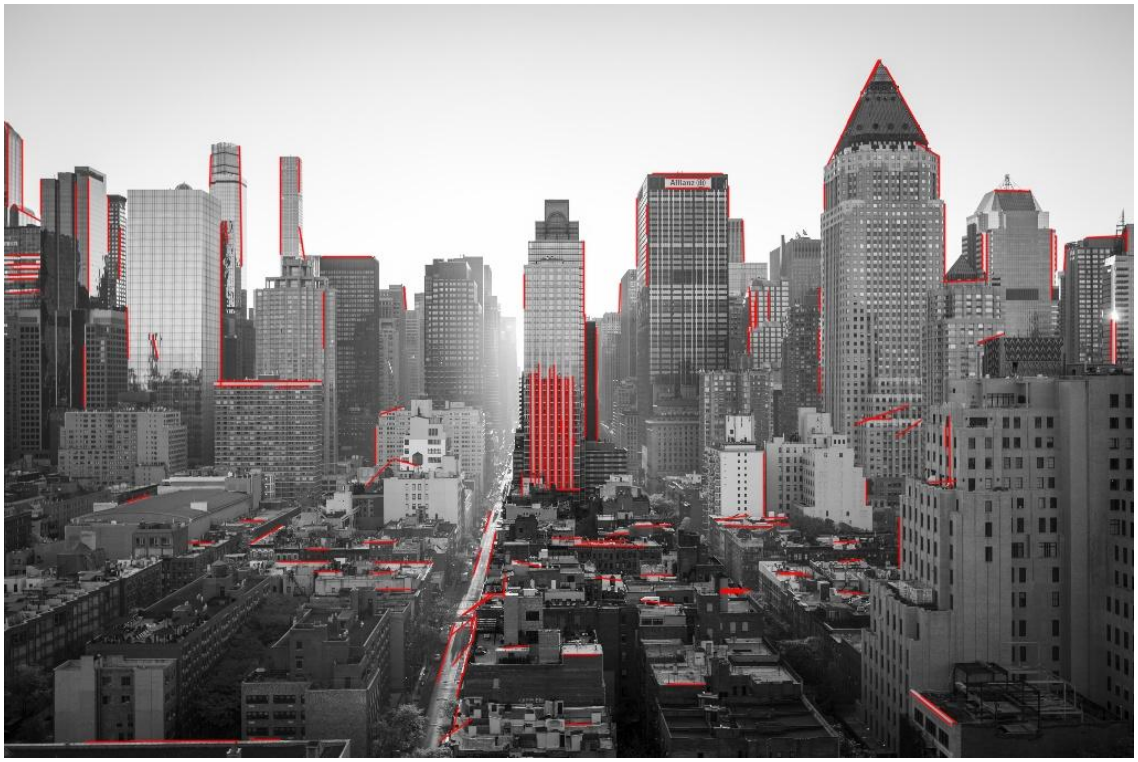

Рисунок 3.7 – Візуальне відображення результату пошуку ліній на зображенні

Підсумки використання алгоритмів. Всі критерії пошуку знаходилися експериментальним шляхом, що в свою чергу позначається на результаті деяких зображень. Так для зображення (рис. 3.3) і результату пошуку прямих (рис. 3.7). видно більше ліній ніж визначає алгоритм. Але така чутливість достатня, щоб не реагувати на кілька пік селів, що стоять вряд, що не є прямою на інших фотографіях. Вибір такого підходу підбору аргументу обумовлений хорошою швидкістю і незначним часом виконання.

Альтернативний спосіб – це виконувати кілька ітерація починаючи від розмиття монохромного зображення і закінчуючи пошуком прямих для кожного вхідного зображення. При цьому кожній ітерації буде відповідати свій набір аргументів. Після виконання пошуку необхідно буде порівняти отримані результати для одного зображення і вибрати найкращий. В цілому це підвищить точність пошуку і негативно позначиться на швидкості.

ВИСНОВКИ

В результаті виконання бакалаврської роботи було розробдено програмне забезпечення для класифікації зображень. Проведено аналіз вимог до програмної системи. Здійснено обґрунтований вибір програмних засобів розробки. На етапі проектування програмної системи було створено діаграми послідовності та класів. За допомогою них мета створення програмного забезпечення стала більш прозора, та його розробка стала легшою.

Типова задача для застосування запропонованого програмного забезпечення:

– це пошук схожого контексту в інтернет ресурсах (дублікати) – інтернет магазини, дошки оголошення або форуми;
– захист дітей попередній аналіз зображення і обмеження доступу якщо контекст сумнівний;
– пошук зображень для творчої діяльності – автоматичний пошук зображень за критеріями (для художників, архітекторів і дизайнерів).

Це програмне рішення не ідеальне, але воно дозволило на практиці познайомитися з типовими проблемами аналізу зображень. Використання модульної системи збирання дозволяє збирати програму майже для будь-якої архітектури, від компактного комп'ютера, що поміщається на долоні, до високопродуктивних серверів, що займають цілі поверхи. Закладена підтримка багатопоточности призводить до значного скорочення загального часу аналізу, тим самим підвищує продуктивність і дозволяє використовувати доступні обчислювальні можливості на максимум.

Поділ програми на логічні модулі дозволило додати універсальності. Це дозволяє замінювати різні компоненти, модифікувати і навіть додавати нові без необхідності повністю переписувати архітектуру. Наприклад поточне завдання можна змінити і додати нові можливості: робота з камерою як з джерелом, а для виведення використовувати GPIO або PWM для керування роботизованим маніпулятором.

Використання мови програмування C ++ підвищує планку входження і розуміння вихідного коду (для багатьох це досить складно), але це дозволяє отримати компактний і бінарний код після компіляції. Що в свою чергу позначається на продуктивності.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Колисниченко, Д.Н. Linux. Полное руководство / Д.Н. Колисниченко, Аллен, Питер: СПб: Наука и Техника, 2007.784 с.
2. Linux – Вікіпедія. URL: https://uk.wikipedia.org/wiki/ Linux (дата звернення 04.05.2020)
3. Маслинский К. Операционная система Linux. ИНТУИТ, 2005.
4. Intel Edison – Вікіпедія.. URL: https://uk.wikipedia.org/wiki/ Intel_Edison (дата звернення 04.05.2020)
5. Raspberry Pi – Вікіпедія. URL: https://uk.wikipedia.org/wiki/ Raspberry_Pi (дата звернення 04.05.2020)
6. Nvidia Jetson – Wikipedia. URL: https://en.wikipedia.org/wiki/ Nvidia_Jetson (дата звернення 04.05.2020)
7. Кениг, Э. Эффективное программирование на C++. Практическое программирование на примерах. Т. 2 / Э. Кениг, Б.Э. Му. М.: Вильямс, 2016. 368 с.
8. Секунов, Н. Программирование на C++ в Linux / Н. Секунов. СПб.: BHV, 2004. 368 с.
9. CMake – Вікіпедія. URL: https://uk.wikipedia.org/wiki/CMake (дата звернення 04.05.2020)
10. cURL – Вікіпедія. URL: https://uk.wikipedia.org/wiki/cURL (дата звернення 04.05.2020)
11. OpenCV – Вікіпедія. URL: https://uk.wikipedia.org/wiki/OpenCV (дата звернення 04.05.2020)
12. OpenCV Offical Webpage. URL: https://opencv.org/ (дата звернення 04.05.2020)
13. Bradski G., Kaehler A. Learning OpenCV – Computer Vision with the OpenCV Library. O'Reilly Media. 2008. 580 с.
14. GPIO – Вікіпедія. URL: https://uk.wikipedia.org/wiki/GPIO (дата звернення 04.05.2020)

15. Rick F. van der Lans. The SQL Guide to SQLite. First Lulu edition 2009. 524 p.
16. SQLite – Вікіпедія. URL: https://uk.wikipedia.org/wiki/SQLite (дата звернення 04.01.2020)