НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА СИСТЕМНОГО ПРОГРАМУВАННЯ І СПЕЦІАЛІЗОВАНИХ КОМП'ЮТЕРНИХ СИСТЕМ

«На правах рукопису»
УДК 004.9

«До захисту допущено»
Завідувач кафедри СПСКС

_____
В.П.Тарасенко_
(підпис) (ініціали, прізвище)
"___"_____2018р.

Магістерська дисертація

на здобуття ступеня магістра

зі спеціальності 123 Комп'ютерна інженерія
Системне програмування

на тему: Методи визначення параметрів джерела акустико-оптичного випромінювання

Виконав: студент II курсу, групи _КВ-72мп_____
(шифр групи)

Віфлінзідер Віталій Володимирович_____ _____
(прізвище, ім'я, по батькові) (підпис)

Науковий керівник.проф. кафедри СПСКС, д.т.н, Терейковський І.А_____
_____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент_____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2018 року

РЕФЕРАТ

Актуальність теми. На сьогоднішній час комп'ютерний зір досить розвинена галузь інформаційних технологій, але напрацювання з цієї галузі рідко використовуються у промислових маштабах. У сучасному світі нас оточує велика кількість камер спостереження чи іншого призначення. Пошук об'єктів чи явищ на відео у реальному часі може зробити життя людини безпечнішим і комфортнішим. Актуальність даної роботи полягає у недостатній у оптимізації сучасних систем безпеки.

Об'єктом дослідження є джерела акустико-оптичного випромінювання.

Предметом дослідження є модель визначення параметрів джерела акустико-оптичного випромінювання.

Мета роботи: дослідити існуючі методи визначення параметрів джерела акустико-оптичного випромінювання та оптимізація одного з методів.

Методи дослідження. В роботі використовуються методи визначення параметрів акустико-оптичного випромінювання.

Наукова новизна роботи полягає в наступному:

1. Розроблено математичну модель визначення параметрів об'єкта, що є джерелом акустико-оптичного випромінювання.
2. Вперше запропоновано використовувати технологію комп'ютерного зору задля підвищення ефективності пожежної системи безпеки.
3. Розроблено пожежну систему нового типу, яка базується на пошуку полум'я в режимі реального часу на відео потоці.

Практична цінність отриманих в роботі результатів полягає в тому, що возроблена модель і метод були використані у пожежній системі нового типу. Пожежена система це тільки одним з можливих варіантів використання результатів.

Структура та обсяг роботи. Магістерська дисертація складається з вступу, чотирьох розділів, висновків та додатків.

У вступі надано загальну характеристику роботи, виконано оцінку сучасного стану проблеми, обґрунтовано актуальність напрямку досліджень, сформульовано мету і задачі досліджень, показано наукову новизну отриманих результатів і практичну цінність роботи, наведено відомості про апробацію результатів і їх впровадження.

У першому розділі розглянуто існуючі рішення та бібліотеки в галузі комп'ютерного зору, проведено детальний аналіз бібліотеки OpenCV і запропоновано галузі у яких може бути використано комп'ютерний зір.

У другому розділі проаналізовано алгоритми, методи і моделі для визначення параметрів акустико-оптичного випромінювання.

У третьому розділі описано розроблене програмне забезпечення на основі розробленого методу визначення параметрів джерела акустико-оптичних, UML діаграмів класів, використаних пакетів та бібліотек а також алгоритм роботи всієї системи.

У четвертому розділі оцінено отримані результати, зроблена порівняльна характеристика з існуючими аналогами, а також наведено подальші кроки удоскаоналення системи.

У висновках проаналізовано отримані результати роботи.

У додатках наведено алгоритм роботи розробленого програмного забезпечення, UML діаграма класів.

Робота виконана на 0 аркушах, містить 0 додатків та посилання на список використаних літературних джерел з 0 найменувань. У роботі наведено 0 рисунків та 0 таблиць.

Ключові слова: комп'ютерний зір, opencv, акустично-оптичні випромінювання, алгоритм Віоли-Джонса

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет прикладної математики
Кафедра системного програмування

Рівень вищої освіти – другий (магістерський) за освітньо-професійною програмою

Спеціальність (спеціалізація) – 123 «Комп'ютерна інженерія»

«Системне програмування»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ В.П. Тарасенко

«___»_____2017 р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Віфлініздеру Віталію Володимировичу

1. Тема дисертації «Методи визначення параметрів джерела акустико-оптичного випромінювання», науковий керівник дисертації Терейковський Ігор Анатолійович, д.т.н., доцент, затверджені наказом по університету від «30» жовтня 2018 р. №4030-с

2. Термін подання студентом дисертації «10» грудня 2018 р.

3. Об'єкт дослідження: методи визначення параметрів джерела акустико-оптичних випромінювань.

4. Предмет дослідження: моделі визначення параметрів у просторі об'єктів, які є джерелом акустико-оптичного випромінювання..

5. Перелік завдань, які потрібно розробити:
   - провести аналіз існуючих методів изначення параметрів джерела акустико-оптичного випромінювання;
   - дослідити способи цифрового аналізу відео файлів;
   - розробити та дослідити модель визначення параметрів у просторі об'єктів, які є джерелом акустико-оптичного випромінювання;
   - обґрунтувати вибір обраних інструментів;
   - розробити метод визначення параметрів джерела акустико-оптичних випромінювань;
   - розробити програмне забезпечення на основі розроблених методів та моделей;
   - провести тестування розробленого програмного забезпечення.

6. Орієнтовний перелік графічного (ілюстративного) матеріалу:
   - схема алгоритів методіввизначення параметрів джерела акустико-оптичного випромінювання;
   - схема алгоритму роботи розробленої системи на основі розроблених методів та моделей
   - UML діаграми розробленого програмного забезпечення;
   - порівняльна таблиця отриманих результатів і результатів аналогів;
   - презентація

7. Перелік публікацій:

- Тези доповіді “Пошук полум’я у потоковому відео в режимі рельного часу як основа пожежної системи ”
- Тези доповіді “Багатоканальна система пошуку пожежі у закритому приміщенні”

8. Дата видачі завдання «04» жовтня 2017 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1.	Ґрунтовне ознайомлення з предметною галуззю	17.10.2017	
2.	Визначення структури магістерської дисертації; вивчення літератури, пошук додаткової літератури, патентний пошук	04.12.2017	
3.	Робота над першим розділом магістерської дисертації; проведення наукового дослідження	15.02.2018	
4.	Проведення наукового дослідження; робота над другим розділом магістерської дисертації; розроблення програмного забезпечення	05.04.2018	
5.	Проведення наукового дослідження; робота над статтею за результатами наукового дослідження	15.05.2018	
6.	Проведення наукового дослідження; робота над третім розділом магістерської дисертації	15.06.2018	
7.	Завершення роботи над основною частиною магістерської дисертації; підготовка ілюстративного матеріалу; підготовка матеріалів доповіді на конференції ПМК-2018	05.11.2018	
8.	Оформлення текстової і графічної частини магістерської дисертації	04.12.2018	

Студент В.В. Віфлінзідер

Науковий керівник дисертації І. А.. Терейковський

ЗМІСТ

СПИСОК СКОРОЧЕНЬ І ПОЗНАЧЕНЬ

CPU – Central processing Unit, центральний процесор

API – Application Programming Interface, прикладний програмний інтерфейс

АУПС – автоматична установка пожежної сигналізації

IoT – Internet of Things, інтернет речей

GPIO – general-purpose input/output

CLI – command line interface

ВСТУП

Перші комп’ютери були створені виключно для математичних обчислень, а уже через менше ніж через століття у кожного у домі декілька пристроїв, які можуть виконувати мільйони операцій в секунду. За цей період інформаційні технології проникли у всі аспекти людського життя. Нас оточують пристрої, які генерують терабайти інформації і ми не завжди помічаємо ці пристрої. Наприклад, камери спостереження, датчики різних видів у офісах та підприємствах. Аналіз даних, отриманих з цих пристроїв, можуть зробити життя людини. Пошук об’єктів чи явищ за допомогою технології комп’ютерного зору на відео-даних – один з основних методів аналізу відео.

Досить довгий час завдання розпізнавання розглядалося людиною з боку біологічного і психологічного аспектів. При цьому вивченню піддавалися лише якісні характеристики, які не дозволяли точно описати механізм функціонування. Отримання функціональних залежностей було, як правило, пов'язане з дослідженням рецепторів органів слуху, дотику або зору. Однак принципи формування рішення залишалися загадкою. Вважається, що основною помилкою на початку дослідження була думка про те, що мозок функціонує за певними алгоритмами, а отже, з'ясувавши цю систему правил, можна її відтворити за допомогою постійно обчислювальних і технічних засобів, які постійно розвиваються.

Історично склалося так, що теорія розпізнавання образів розвивалася в двох напрямках: детерміністському і статистичному, хоча найчастіше однозначно розрізнити їх не вдається. Детерміністський підхід включає різні методи: емпіричні, евристичні, в основі яких лежать здоровий глузд, більш-менш вдале моделювання дій, які здійснюються мозком людини; математично формалізовані, наприклад, засновані на моделі породження об'єктів (реалізацій) того чи іншого образу. При цьому використовується

різний математичний апарат (математична логіка, теорія графів , топологія, математична лінгвістика, математичне програмування та ін.). Статистичний підхід опирається на фундаментальні результати математичної статистики (теорія оцінок, послідовний аналіз, стохастична апроксимація, теорія інформації).

Інтеграція систем на основі комп'ютерного зору у системи безпеки, суттєво покращить результат роботи останніх. Наприклад, пошук людини на камері спостереження допоможе нічному охоронцю складу швидше зреагувати на крадіжку чи напад, чи система пошуку полум'я у відео допоможе виявити пожежу у приміщенні до того як спрацюють сенсори диму.

РОЗДІЛ 1. ОГЛЯД ІСНУЮЧИХ РІШЕНЬ ТА БІБЛІОТЕК

1.1. Системи комп'ютерного зору

Комп'ютерним зором називають галузь знань, яка об'єднує методики обробки, аналізу та розпізнавання зображень, а також великих масивів даних з фізичного виміру з метою отримання певних результатів (інтерпретованих чисельних або символьних).

Основним призначенням комп'ютерного зору є можливість моделювати властивості людського зору, а саме розпізнавати образи на зображенні та отримувати з нього необхідну інформацію. Розпізнавання образів – це аналіз даних зображення, використовуючи запрограмовані моделі, що можуть базуватися на основах геометрії, статистики, фізики, математики та машинному навчанні.

Комп'ютерний зір також є науковою дисципліною, що базується на теорії створення та використання моделей та систем, які призначені для виділення інформації з зображень. Основою для дослідження є не лише статичні зображення, але й відеодані (послідовність зображень, зображення з камер, тривимірні дані тощо).

Дана дисципліна є досить практичною та технологічною, її теорії та методики активно застосовуються для створення систем комп'ютерного зору для автоматизованого розпізнавання зображень та відеоданих. Наведемо приклади таких систем:

- системи призначені для керування процесами (автопілоти, промислові роботи, автоматизовані системи керування, машинний зір);
- системи призначені для відеоспостереження;
- системи призначені для обробки та систематизації інформації;
- системи для моделювання об'єктів (в медицині: аналіз результатів мікроскопії, рентгенографії, томографії, ультразвукових досліджень тощо;

у військовій справі: аналіз топографічних даних, системи виявлення ворожих військових сил; системи виявлення лісових пожеж тощо)

- системи моделювання антропомашинної взаємодії.

Варто відзначити, що дана область є досить новою та швидко розвиваючою. Прийнято вважати, що комп'ютерний зір почав активно розвиватися як прикладна область в 70-х роках 20 століття, хоча певні напрацювання в цій сфері були і раніше. Відсутність реальних прикладів застосування раніше пояснюється відсутністю комп'ютерів, здатних обробляти великі масиви інформації. Проте дослідження у сфері комп'ютерного зору переважно проводилися в рамках інших наукових питань, тому важко сказати, що конкретна проблема дисципліни була чітко сформульована. Але існує багато точкових задач, для яких були сформовані рішення, хоча їх важко узагальнити до масштабів дисципліни комп'ютерного зору. Досить багато рішень наразі знаходяться на рівні теоретичних напрацювань, проводяться різноманітні експерименти тощо. Поряд з цим все більше розроблених технологій знайшли застосування в реальних комерційних продуктах і широко використовуються на практиці. В практичних кейсах ми можемо бачити, що використовуються вузькоспеціалізовані методи програмування комп'ютерів для кожної задачі, які проте гуртуються на досить узагальнених принципах комп'ютерного зору.

В рамках задач штучного інтелекту вагомою є область автоматичного прийняття рішень та планування (наприклад, автоматичний рух робота, автоматизація складського обладнання тощо). Для їх ефективної реалізації необхідно отримувати інформацію з зовнішнього середовища про об'єкт прийняття рішення штучним інтелектом. Комп'ютерний зір допомагає в цьому тим, що надає необхідну ввідну інформацію з відеосенсорів, спеціальних датчиків для подальшої роботи систем взаємодії з середовищем.

Також комп’ютерний зір використовується для розпізнавання образів та застосування навчання, що відносять до області штучного інтелекту. Виходячи з вище сказаного, можна сказати, що комп’ютерний зір можна віднести до предметної області штучного інтелекту.

Окрім комп’ютерних наук дана область тісно пов’язана також з фізикою. Багато підходів комп’ютерного зору ґрунтуються на розумінні фізичних явищ, зокрема це стосується електромагнітного випромінювання (видимого або інфрачервоного спектру), що відображаються об’єктами. Ці випромінювання можуть фіксуватися спеціальними датчиками і передавати необхідну інформацію про об’єкт. Ці процеси ґрунтуються на основі принципів фізики твердих тіл та безпосередньо оптики. Для роботи зі складними датчиками та правильного зчитування зображення необхідно розуміти принципи квантової механіки. І навпаки, багато задач фізики можуть бути вирішені за рахунок використання комп’ютерного зору. Як результат, комп’ютерний зір також може бути визначений як доповнення до фізики.

Не менш важливу роль відіграє нейробіологія в рамках вивчення комп’ютерного зору, а саме розуміння основ функціонування біологічного зору. Наразі існує багато досліджень людського та тваринного зору, очей різних ссавців, структури мозку та нейронів, що відповідають за розпізнання об’єктів, кольорів тощо. Таким чином сформовані принципи роботи біологічного зору лягли в основу розробки систем комп’ютерного зору, створені різноманітні системи, що наслідують роботу реальних біологічних систем на різних рівнях комплектності.

Також для комп’ютерного зору є важливим розуміння основ обробки сигналів. Широко використовуються аналоги до методів обробки одномірних сигналів в комп’ютерному зорі та розширюються на аналіз двовимірних та тривимірних сигналів. Проте існують методи обробки сигналів в рамках дисципліни комп’ютерного зору, що не мають аналогів в

стандартних методах обробки одномірних сигналів (наприклад, вони вирізняються своєю нелінійною природою, багатомірністю тощо). Тому обробку сигналів можна вважати підобластю комп’ютерного зору.

Не менш важливою є математика в рамках роботи з комп’ютерним зором. Багато його методів ґрунтуються на чисто математичних підходах, зокрема широко застосовуються методи статистики, оптимізації, геометрії тощо.

Окрім теоретичних основ, надзвичайно важливим є питання оптимізації роботи комп’ютера під час виконання задач комп’ютерного зору. Існують і створюються нові методи апаратної оптимізації, що покликані збільшити швидкодію комп’ютера та зменшити використання ним ресурсів, що є надзвичайно важливим для правильної та ефективної роботи досліджуваної дисципліни.

Загалом багато сфер пов’язані з комп’ютерним зором (обробка зображень, робототехніка, в рамках сприйняття роботом візуальної інформації, машинний зір тощо). Проте ще досі не було виділено всі ці сфери в окрему галузь, хоча всі вище перераховані області дуже тісно пов’язані і мають багато спільних рис. Наразі їх розрізняють за предметною областю, що кожна маю свою специфіку. Наведемо приклади.

Обробка зображень спеціалізуються на роботі з зображенням у двовимірному форматі. Основною його задачею є трансформація зображення, наприклад зменшення контрастності, корегування країв, різного роду перетворення, усунення шумів тощо. При цьому припускається, що виконання цих дій не залежить від змісту самого зображення.

Комп’ютерний зір працює в тривимірному просторі. Одна з основних задач, це відновлення повної картини по одному чи кільком зображенням. При цьому вміст зображень тут відіграє вагому роль.

Машинний зір має переважно промислову специфіку (наприклад, роботизація, системи оптичної перевірки та вимірювання тощо). Широко використовуються різноманітні датчики, інформація з відеоданих застосовуються автоматизованими системами та апаратними засобами в режимі реального часу.

Область візуалізації переважно стосується процесу створення зображення, проте обробка та аналіз зображень також частково покривається (наприклад, рентгенографія).

Досить близьким до комп'ютерного зору є розпізнавання образів. Ця область стосується можливості виділення інформації з відеоданих в основному на базі статистичних методів та підходів.

Комп'ютерний зір широко застосовується в багатьох сферах та інтегрований в багато сучасних процесів. Медицина є однією з ключових областей застосування комп'ютерного зору, де інформація отримана з зображень/відеоданих використовується з метою визначення діагнозу. Наприклад, виявлення злоякісних пухлин, переломів, атеросклерозу тощо. Комп'ютерний зір також може допомогти визначати розміри внутрішніх органів, правильність функціонування внутрішніх процесів. Отримана інформація можу бути застосована не лише для діагностування, але й в рамках проведення медичних досліджень, вивчення малодосліджених сфер функціонування людського організму.

Не менш важливим є застосування комп'ютерного зору в промисловості, що допомагає автоматизувати та оптимізувати багато процесів. Наприклад, досліджуваний підхід дуже ефективний в рамках контролю якості на виробництві, коли автоматично можливо перевірити кінцевий продукт, сировину чи напівфабрикат на наявність дефектів та браку. Комп'ютерний зір може бути застосований для вимірювання та орієнтації роботів, що переміщують предмети.

Однією з передових сфер застосування комп’ютерного зору є військова справа. Сучасні наробки широко використовуються для визначення дислокації ворожих військових сил, техніки, транспорту тощо. Комп’ютерний зір дуже ефективний при запуску ракет, автоматичного прицілювання, розвідки. Можливості даної сфери дозволяють отримати максимальну кількість корисної інформації про поле бою до початку бойових дій, що може бути використано для прийняття стратегічних військових рішень.

Однією з найсучасніших прикладних областей комп’ютерного зору є автопілотні транспортні засоби. При чому рівень автономності транспортного засобу може варіюватися від автоматизованого до повністю безпілотного. За допомогою комп’ютерного зору такі машини можуть визначати своє місце положення, потенційні перешкоди. Комп’ютерний зір може бути використаний для систем попереджувальної сигналізації.

Окрім вже описаного, комп’ютерний зір використовується для виявлення пожеж, створення спецефектів, систем спостереження тощо

Кожна сфера використання має свої специфічні задачі і вирішує їх з використанням специфічних методів. Проте деякі з них можна узагальнити та сформувати загальні задачі комп’ютерного зору. Наведемо приклади таких задач.

Основною задачею комп’ютерного зору є визначення, чи присутній певний об’єкт, особливість, дія чи процес на зображенні чи відео. Людський зір легко справляється з даною задачею, проте комп’ютер не завжди чітко може визначитися в окремих ситуаціях.

Розроблені методики визначення наявності шуканого об’єкту чи явища є досить обмеженими: вони можуть виявити лише прості предмети, обличчя, символи, машини тощо. Також перешкодою може бути наявність невідповідних умов: відсутність гарного освітлення, наявність розмитого фону, неправильне положення та інше.

Розпізнавання – це визначення об’єкта чи груп об’єктів при попередньому їх вивченні на двомірному зображенні чи в тривимірному просторі.

Ідентифікація – це вже визначення конкретного об’єкта, наприклад певного автомобіля, людини, відбитків, фігури.

Виявлення – це перевірка на наявність певних характеристик, наприклад, виявлення помилок чи невідповідностей. Виявлення ґрунтується на досить нескладних та швидких методиках, що є його перевагою. Виявлення використовуються часто для визначення певної області, що відповідає заданим вимогам, а для неї вже застосовуються більш складні та комплексні методи.

Виділимо стандартні приклади задач розпізнавання:

- Пошук зображень за вмістом: за певними ознаками з масиву виділяють конкретну одиницю, що має визначений вміст. Характер вмісту може мати різні інтерпретації та параметри вибору: параметри схожості, багатокритеріальний підхід в текстовому форматі тощо.
- Оцінка положення: визначення орієнтації об’єкта відповідно до положення датчика, камери чи іншого засобу зчитування інформації комп’ютерного зору. Наприклад, визначення роботом орієнтації продукту в складському приміщенні чи на конвеєрній лінії.
- Оптичне розпізнавання символів: визначення певних символів на зображенні та переведення їх в редагований текстовий формат.

Також популярною є задача характеристики руху. На послідовності зображень визначається швидкість руху певних точок, їх траєкторію. Вирішуються наступні задачі:

- спостереження за рухом камери;
- стеження за рухом об’єкта.

Окрім розпізнавання та виявлення об’єктів, комп’ютерний зір може відновлювати зображення.

Відновлення зображення може бути інтерпретовано як прибирання непотрібних шумів (результати дефектності датчика, розмитість тощо). Найпростіше та найбільш поширене вирішення даної проблеми – це застосування певних фільтрів різних частот. Існують також більш складні методи вирішення даної проблеми, що ґрунтуються на підході визначення «як має бути» і порівняння з «як є».

Більш складний процес видалення шумів базується за допомогою аналізу вхідних даних, визначення певних структурних елементів та потім вже використання більш вузькоспеціалізованих фільтрів на базі отриманої інформації.

Особливості практичного впровадження систем комп'ютерного зору сильно залежить від сфери застосування. Системи можуть бути окремими функціональними рішеннями, або входити до складу більш комплексної структури та виконувати допоміжну чи проміжну функцію. Використання даних систем також може варіюватися в залежності від того, чи задачі системи є визначеними наперед, чи можуть бути доповнені чи змінені під час її роботи.

Як можемо бачити, функції комп'ютерного зору досить різноманітні, проте можемо визначити певні спільні моменти для всіх систем.

Зображення отримуються з використанням різноманітних датчиків (світлочутливі камери, радари, ультразвукові камери тощо). В залежності від обраного технічного забезпечення отримуване зображення може бути двовимірним, тривимірним чи бути відеорядом (послідовністю зображень). Зображення складаються з пікселів, які як правило виражають інтенсивність світла, рідше інші характеристики (поглинання електромагнітних хвиль, відображення звукових хвиль, глибина, резонанс тощо).

Для того, щоб використовувати всі можливості методів комп'ютерного зору, необхідно спершу попередньо обробити зображення. Наприклад:

- коректування контрастності;
- виділення неточностей та шумів та їх ліквідація;
- перевірка координатної системи;
- масштабування.

У зображенні можна виділити певні деталі:

- лінії;
- межі;
- кути;
- точки;
- форми;
- рух;
- краплі.

На певному етапі виділяються ті деталі зображення, які будуть використовуватися надалі в роботі. Цей процес називають сегментацією. Приклади сегментації:

- виділення об’єкту;
- виділення набору точок;
- виділення контурів тощо.

Після попередньої обробки та сегментації переходимо до високорівневої обробки. На цьому етапі вже об’єкт дослідження є досить невеликим, тому можливо застосовувати більш складні підходи для його аналізу. Ми переконані, що в обраному проміжку є об’єкт, що нас цікавить, і на цьому етапі можемо визначати його параметри: розмір, об’єм, колір, положення та інше.

1.2 Бібліотека OpenCV

OpenCV — бібліотека функцій та алгоритмів з відкритим кодом, призначена для обробки зображень і чисельних алгоритмів загального призначення на основі комп'ютерного зору

Бібліотека була розроблена у 1999 році компанією Intel (Intel Research) з метою розвивати додатки, які навантажують CPU.

На початкових етапах розробники ставили перед собою такі задачі:

- забезпечити розвиток технологій у сфері комп'ютерного зору
- створити бібліотеку з добре оптимізованим і відкритим кодом
- створити умови для розвитку комерційних додатків на основі комп'ютерного зору

На сьогоднішній час підтримкою і подальшою розробкою займається Willow Garage та Itseez. Бібліотека реалізована на C / C++ і також доступні API для мов програмування Python, Java, Ruby, Matlab, Lua та інших. Спочатку OpenCV розроблялась C, тому досі є інтерфейс для цієї мови програмування.

На даний момент існують також обгортки для OpenCV для мов програмування C#, CH, Perl, Haskell і Ruby. Ці обгортки були розроблені для того, щоб збільшити аудиторію. Розвиток бібліотеки продовжується на мові програмування C++. Так як проект з відкритим кодом, то участь у розробці проекту може брати кожен охочий.

OpenCV підтримує такі операційні системи:

- Windows
- Linux
- MacOS
- FreeBSD
- NetBSD
- OpenBSD

- Android
- IOS
- Maemo
- BlackBerry

На сьогоднішній день основний напрямок розвитку бібліотеки — інтеграція машинного навчання, для покращення результатів роботи. Уже є підтримка таких фреймворків:

- TensorFlow (розроблений компанією Google)
- PyTorch (розробленийй компанією Facebook)
- Caffe (розроблений науковцями з Університету Каліфорнії)

1.3 Огляд сучасних систем визначення параметрів джерела акустико-оптичного випромінювання

На сьогоднішній час існує багато систем визначення параметрів джерела акустико-оптичного випромінювання і всі вони залежать від галузі. У своїй роботі ми будемо розглядати пожежні системи, як системи визначення параметрів акустико-оптичного випромінювання.

На даний час на ринку є чимало рішень у сфері пожежної безпеки. Найшвидшою та найнадійнішою системою для виявлення займання вважається автоматична установка пожежної сигналізації(АУПС). Існують схеми променевого та кільцевого з’єднання в АУПС. Розглянути детальніше ці схеми можна на рис. 1.1.

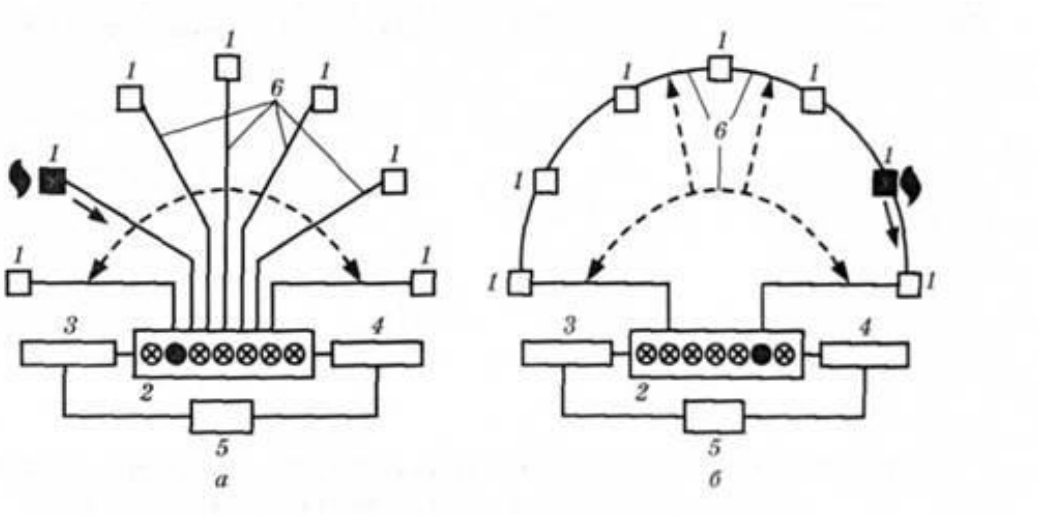


Рисунок 1.1 – Схеми променевого (а) та кільцевого (б) з'єднання в АУПС: 1 - сповіщувачі; 2 - приймально-контрольний прилад; 3 - блок живлення від електромережі; 4 - блок аварійного живлення; 5 - система перемикання живлення; 6 - з'єднувальні проводи

Також АУПС відрізняються між собою за типом сповіщувачів. Роль сповіщувача можуть виконувати сенсори тепла (рисунок. 1.2) і сенсори диму (рисунок 1.3) .

Рисунок 1.2 – Сенсор тепла

Теплові сенсори поділяються на у пожежних системах за принципом дії поділяються на максимальні, диференційні і максимально-диференційні.

Максимальні сенсори спрацьовують коли температура у приміщенні досягає максимальної поділки. Диференційні сенсори реагують на різку зміну температури, а максимально-диференційні реагують або на досягнення максимальної поділки або на різку зміну температури. Такі сенсори характеризуються хорошою точністю роботи, але поганою швидкістю реагування. Необхідно досить багато часу, щоб у приміщенні температура піднялась до максимальної чи щоб різниця температур була достатня для реакції сенсорів.

Сенсори диму у свої роботі використовують оптичний або радіоізотопний методи. Оптичний метод базується на ефекті Тиндаля. Ефект Тиндаля — світіння оптично неоднорідного середовища внаслідок розсіяння світла, яке через нього проходить. Є характерним для дисперсних систем (напр. гідрозолів, аерозолів) із низькою концентрацією частинок дисперсної фази, які мають показник заломлення, що відрізняється від показника заломлення дисперсійного середовища. Може спостерігатися у вигляді світлого конуса на темному фоні (конус Тиндаля) при розгляданні дисперсної системи під певним кутом (зазвичай 90°) до напрямку проходження через неї сфокусованого пучка світла [1].

Рисунок 1.3 – Сенсор диму

Швидкість реакції таких систем теж бажає кращого. Необхідно, щоб система спрацювала необхідно, щоб набралось достатньо диму. Цей показник можна покращити, збільшивши кількість сенсорів диму, але таке рішення дуже затратне.

Інтеграція елементів комп'ютерного зору у системи пожежної безпеки зменшить час реакції і збільшить точність роботи системи. Якщо у АУПС замінити сенсори диму на відеокамери, які будуть надавати дані для системи комп'ютерного зору, яка буде здійснювати пошук полум'я у потоковому відео у реальному режимі, то час реакції буде зменшений, оскільки пожежа буде виявлена на початковому етапі. Якщо ж інтегрувати вище описану систему разом з сенсорами диму, то буде збільшена завадостійкість пожежної системи, оскільки у системи з'являються два параметри для ідентифікації пожежі.

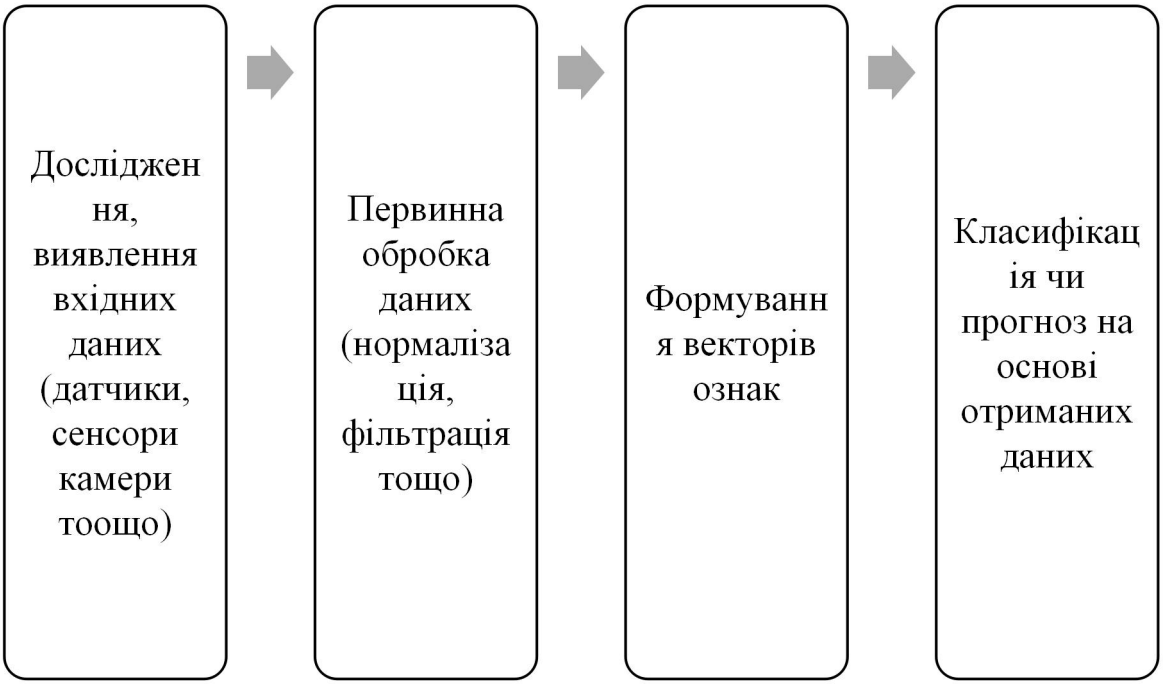
Комп'ютерний зір сильний і сучасний інструмент правильне використання якого збільшить показники роботи будь-якої системи, яка як дані може використовувати зображення або відео.

РОЗДІЛ 2. ОГЛЯД МЕТОДІВ ДЛЯ ПОШУКУ ОБ'ЄКТІВ НА ВІДЕО І АУДІО

2.1 Метод контурного аналізу

Розпізнавання образів є досить широка сфера, вже виділено багато задач з цієї області. В першу чергу необхідно розуміти базові поняття даного процесу. Основні категорії:

- Клас – множина об'єктів, які мають спільні властивості.
- Класифікація – процес розподілу об'єктів за класами, відповідно до їх характеристик.
- Класифікатор – засіб, який на вхід отримує набір властивостей об'єкта, а на виході видає клас, до якого об'єкт має бути віднесений.



- Верифікація – процес співставлення певного об’єкта з моделлю об’єкта або з описом класу.
- Ознака – кількісний опис властивості досліджуваного об’єкта чи явища.

Тому можемо зробити висновок про те, що задача розпізнавання об’єктів зводиться до наступної послідовності дій:

Рисунок 2.1 – Етапи розпізнавання об’єктів

Для аналізу та розпізнавання зображень існує ряд методів та підходів, один з них – метод контурного аналізу. Основою даного методу є отримання зовнішнього контуру зображення об’єктів у вигляді замкненої кривої чи сукупності дуг. Точки, що представляють контур зображення записуються, і представляться за допомогою одного з трьох методів представлення границь об’єктів: апроксимації кривих, відслідковування контурів або зв’язування точок перепадів.

Даний метод ґрунтується на припущенні, що контур зображення містить максимум необхідної інформації про форму об’єкта. Важливо відмітити, що внутрішні точки об’єкта при цьому не аналізуються. З одного боку це обмежує функціональність методу та його гнучкість, але з іншого спрощує обрахування та алгоритмічний підхід.

Метод контурного аналізу дозволяє вирішувати основні задачі розпізнавання об’єктів: перенесення, поворот, зміна масштабів об’єкта тощо. Існує дослідження, що відзначає сприйняття людиною образів як сукупність формування образів контуру об’єкта та його внутрішньої частини. З цієї точки зору даний метод є досить вагомим та обґрунтованим.

Основною категорією даного підходу є поняття контуру. Контур – це сукупність точок (пікселів), які відділяють об’єкт від фону. При виділенні контуру можуть виникати ряд проблем:

- недостатня зміна яскравості призводить до розриву контуру;
- виділення зайвих контурів у зв’язку з наявністю шумів на зображенні;
- широкі лінії контурів через наявність шумів, неякісного зображення, розмитості тощо.

Алгоритм кодування контуру описаний на рис. 2.1:

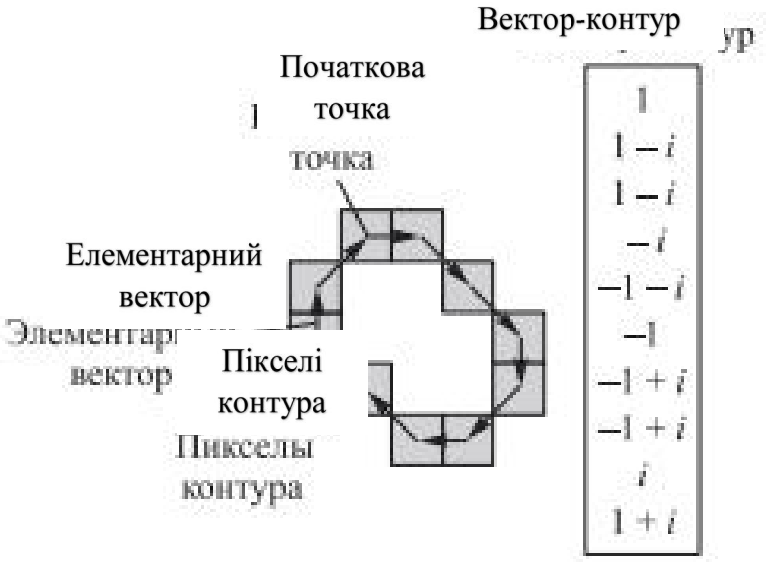


Рисунок 2.2 – Схема кодування контуру

Можуть використовуватися різні підходи до кодування контурів, найбільш відомі серед них: код Фрімена, двовимірне кодування, полігональне кодування. Кодування контуру під час розпізнавання зображень відбувається за допомогою послідовності комплексних чисел за наступною схемою:

- обирається початкова точка;
- контур описується елементарними векторами, що зміщуються, і відображаються за допомогою комплексного числа a+ib (a – зміщення точки по осі X, b – зміщення точки по осі Y)
- контур замикається, коли кінець останнього вектору співпадає з початковою точкою.

Виділяють декілька різновидів контурного аналізу:

- метод активних контурів (для визначення контурів використовуються криві мінімальної енергії);
- метод активних контурів без попереднього виділення границь (аналогічний до попереднього за винятком відсутності етапу попереднього виділення контурів та згладжування зображення);

- детектор границь Кенні (алгоритм складається з зглажування, пошуку градієнтів, подавлення немаксимумів, подвійну порогову фільтрацію, трасування областей невизначеності);
- відслідковування контурів (підхід «викреслювання» границі між об’єктом та фоном за допомогою точки-«жука»);
- кластеризація;
- локальна обробка (аналіз характеристик пікселів контура);
- аналіз за допомогою графів (представлення у вигляді графів та пошук шляхів з найменшою вартістю, що відповідають значими контурам).

Загалом процедуру розпізнавання зображення за допомогою контурного аналізу можна описати наступним чином:

- Попередня обробка зображення (згладжування, фільтрація, відділення шумів, підвищення контрасту тощо)
- Бінаризація зображення та виділення контурів об’єкта
- Початкова фільтрація контурів за периметром, площі, фрактальності, коефіцієнту форми тощо
- Зведення контурів до єдиної довжини, згладжування
- Перебирання всіх знайдених контурів, пошуки шаблонів, що максимально подібні до даного контуру.

Основною перевагою даного методу є його відносна простота. Можлива фільтрація по простим характеристикам, а також існує можливість оброблювати зображення в прогресивному режимі. Враховуючи все це, можемо зробити висновок, що даний алгоритм може бути використаний в режимі реального часу, що є безумовною перевагою в сучасному світі.

Проте метод контурного аналізу має ряд недоліків. Загалом можемо виділити дві основні групи проблем даної методики.

Перша група пов’язана з складністю визначення контуру при умові низької якості зображення, наявності значної кількості шумів, відсутності контрасту між фоном та контуром тощо. Таким чином іноді алгоритм не може точно визначити контур зображення, чи робить це неправильно. Отже з цього випливає, що метод контурного аналізу чутливий до недоліків зображення, не допускається наявності поганої видимості.

Інша група проблем пов’язана з принципами методу, а саме алгоритм не враховує можливість пересікання об’єктів, що не завжди можливо досягнути. Таким чином сфери його застосування можуть бути значно обмежені.

Загалом можемо стверджувати, що незважаючи на наявні недоліки, метод контурного аналізу є досить простим, швидким, а отже привабливим для використання в умовах нормальної якості зображення та чіткої визначеності контурів об’єктів.

2.2 Модель пошуку об’єктів на базі каскадного класифікатора Хаара

Розпізнавання об’єкта можна поділити на два змістовні етапи: фільтрація зображення та аналіз результатів фільтрації. Один з методів аналізу результатів фільтрації ми розглянули в попередньому пункті. Також виділяють алгоритми розпізнавання, що засновані на нейронних мережах. Одним з таких алгоритмів є каскад Хаара.

Основною категорією алгоритму є примітиви Хаара – це ознаки цифрового зображення, які використовуються для розпізнавання об’єктів та працюють на основі веєвлету Хаара. Примітиви є самі по собі розділені на декілька частин прямокутники, що позиціонують в зображенні. Обраховується інтенсивність пікселів кожної області, і знаходиться різниця між обрахованими сумами, що називається значенням даної ознаки Хаара. Математично значення ознаки Хаара можна описати наступним чином:

$$a_i = \sum_{j=y_{a_i}}^{y_{a_i}+h_{a_i}-1} \sum_{k=x_{a_i}}^{x_{a_i}+w_{a_i}-1} v_{jk}$$
$$b_i = \sum_{j=y_{b_i}}^{y_{b_i}+h_{b_i}-1} \sum_{k=x_{ab}}^{x_{b_i}+w_{b_i}-1} v_{jk} \qquad (2.1)$$
$$h(u) = \sum_{i=1}^{N_a} a_i - \sum_{i=1}^{N_b} b_i$$

де v_{jk} - яскравість пікселя з координатами [j,k], a_i - сума яскравостей пікселів в і-ій області першої групи, b_i - сума яскравостей пікселів в і-ій області другої групи, h - значення ознаки Хаара для даного зображення, h_{a_i}, w_{a_i}, h_{b_i}, w_{b_i} - висота та ширина і-их областей першої та другої групи відповідно, y_{a_i}, x_{a_i}, y_{b_i}, x_{b_i} - зміщення по осі у та х і-их областей першої та другої групи, N_a, N_b - кількість областей в першій та другій групах.

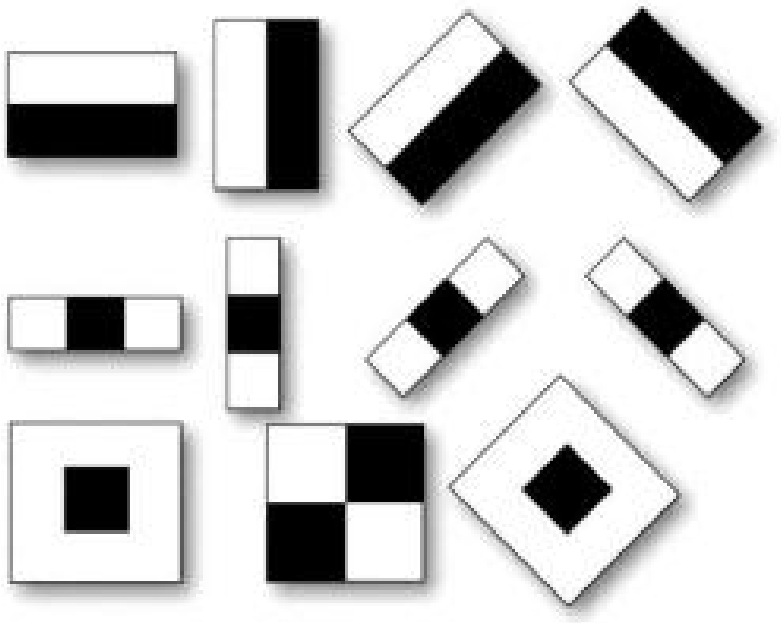


Рисунок 2.3 – Ознаки Хаара

Правильно навчений каскад Хаара допомагає визначити чи є шуканий об’єкт на зображенні чи ні (задача класифікації). При коректному

виконанні задачі, каскад Хаара має досить високу швидкість, стійкість до різних відхилень тощо.

Каскадна модель класифікаторів – це дерево прийняття рішень, де кожний вузол дерева побудований таким чином, щоб знаходити більшу кількість необхідних образів і відхиляти регіони, які не є образами. При цьому вузли розміщені за наступною логікою: чим ближче вузол знаходиться до кореня дерева, тим з меншої кількості примітивів він складається, а отже необхідно менше часу, щоб прийняти рішення.

Каскад Хаара формується за рахунок примітивів Хаара, розраховуючи значення ознак. Для того, щоб навчити класифікатор, необхідно на вхід подати набір правильних зображень, на яких заздалегідь виділена необхідна область. Потім перебираються примітиви і розраховується ознака, після чого розраховані значення зберігаються.

Для того, щоб вибрати ознаки, які найкраще описують зображення, яке досліджується, використовують специфічні алгоритми. Одним з таких алгоритмів є алгоритм AdaBoost (adaptive boosting algorithm). Даний алгоритм був запропонований вченими Йовавом Фройдом і Робертом Шапіром. Головною ідеєю даного алгоритму є те, що великої кількості простих класифікаторів можна створити більш складний та узагальнений класифікатор, який зможе виконувати роботу більш ефективно. Його ще називають алгоритмом посилення класифікаторів. Посилення в даному контексті означає такий ансамблевий алгоритм навчання, який поєднує декілька алгоритмів навчання в один.

Різниця між слабим та сильним класифікаторами в точності їх роботи, якщо точність оцінки за слабим класифікатором дуже низька, то сильний класифікатор дає досить високі шанси отримати коректну класифікацію.

Слабкий класифікатор – це функція, входом для якої є зображення чи ряд зображень, вони обраховує значення відповідного примітива Хаара. З

цією метою ввідні дані порівнюються з пороговим значенням, повертаючи 0 або 1.

$$h_i(x) = \begin{cases} 1, p_i f_i(x) < p_i \theta_i \\ 0 \end{cases}$$

(2.2)

де θ_i - порогове значення, x - вхідне зображення, $f_i(x)$ - значення відповідного примітива Хаара, x, p_i - направлення знаку нерівності, $h_i(x)$ - слабкий класифікатор.

Основна ідея алгоритму в тому, що він перебирає всі можливі слабкі класифікатори та обирає саме ті, які в результаті дають найменші помилки. Після вибору кожного слабкого класифікатора ваги перерозподіляються таким чином, що ті зображення, які були неправильно класифіковані починають сильніше впливти на значення помилки. Повна послідовність алгоритму наведена нижче.

Вхідні дані:

h_i - і-ий примітив Хаара;

E_i - і-ий приклад, що навчається;

y_i - 0, якщо і-ий приклад від'ємний, 1, якщо і-ий приклад позитивний;

n - кількість прикладів.

Змінні алгоритму:

ω_i - вага, яка відповідає певному і-ому прикладу;

θ_i і p_i - порогове значення та напрямок знаку нерівності для і-ого примітива Хаара, яке дає найменшу помилку;

$\varepsilon(h_i)$ - помилка і-ого слабкого класифікатора;

$\overline{h_i}$ - слабкий класифікатор, обраний на і-ій операції;

β_i - мінімальне значення помилки слабкого класифікатора на і-ій ітерації, який представлений в іншому форматі (для оптимізації обчислень).

Вихідні дані (результат):

1. Встановити ваги для прикладів: ω_i, h_i, β_i.

$$\begin{cases} \omega_i = \frac{1}{2m}, if\ y_i = 0 \\ \omega_i = \frac{1}{2l}, if\ y_i = 1 \end{cases}$$

(2.3)

де m – кількість негативних прикладів, l – кількість позитивних прикладів.

2. Для t=1…C:

- Нормалізувати ваги:

$$\omega_i = \frac{\omega_i}{\sum_{j=1}^{n} \omega_j}$$

(2.4)

- Знайти для кожного примітива Хаара такі параметри, щоб слабкий класифікатор видав найменшу помилку:

$$\overline{h}(x) = \begin{cases} 1, p_j h_j(x) < p_j \theta_j \\ 0 \end{cases} \qquad (2.5)$$

$$\varepsilon(\overline{h}) = \sum_{k=0}^{n} \omega_k \left| \overline{h}(E_k) - y_k \right| - minimum$$

- Знайти класифікатор з найменшою помилкою:

$$\overline{h_t} = \begin{cases} 1, p_n h_n(x) < p_n \theta_n \\ 0 \end{cases}; \varepsilon(\overline{h_t}) = min\ \varepsilon(h_j)$$

(2.6)

де n – номер слабкого класифікатора, який в результаті дає найменшу помилку.

- Оновити ваги прикладів:

$$\beta_t = \frac{\varepsilon(\overline{h_t})}{1-\varepsilon(\overline{h_t})}\omega_i = \omega_i \beta_t{}^{1-|\overline{h_t}(E_i)-y_i|}$$

(2.7)

В результаті отримуємо сильний класифікатор, який обчислюється за наступною формулою:

$$H(x) = \begin{cases} \sum_{t=1}^{C} \log(\frac{1}{\beta_t})\overline{h_t}(E_i) \geq \frac{1}{2}\sum_{t=1}^{T} \log(\frac{1}{\beta_t}) \\ 0 \end{cases}$$

(2.8)

Даний класифікатор може бути використаний для виконання задач класифікації, проте варто зауважити, що швидкість класифікації може бути досить низькою. Значення $a_i = \log(\frac{1}{\beta_i})$ надалі буде називатися коефіцієнтом і-ого слабкого класифікатора.

При використанні класифікатора Хаара постає питання швидкості класифікації. Перебрати всі примітиви можливо досить швидко, проте знайти найкраще порогове значення та визначити знак нерівності може бути досить довгою задачею, що призводить до певних труднощів застосування даного підходу на практиці.

Якщо зробити припущення про те, що ваги всіх прикладів, які ми навчаємо, є рівними, то порогове значення має розділяти позитивні та негативні приклади наступним чином: з однієї сторони нерівності найбільша кількість позитивних прикладів, а з іншої відповідно негативних. Розподіл прикладів відбувається відповідно до обраного знаку нерівності. Проте на практиці однакові ваги майже неможливі. Тому задача переформулюється таким чином, що сума вагів прикладів, які розподілені неправильно, має бути мінімальною.

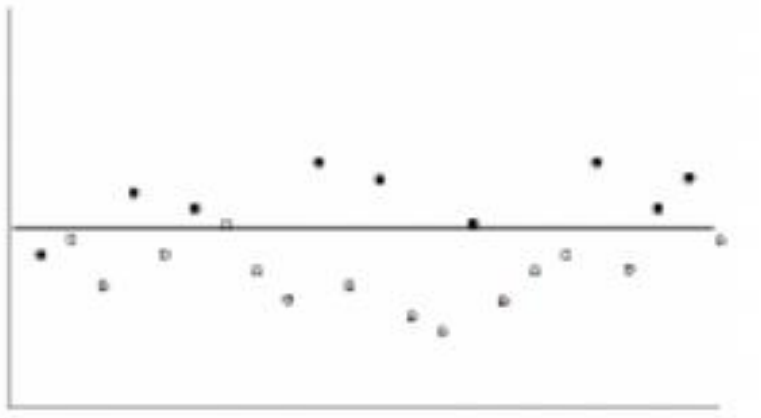


Рисунок 2.4 – Значення ознак Хаара

На рис. 2.3 ми можемо бачити значення однієї ознаки Хаара для різних прикладів. На горизонталі показані номери прикладів, а на вертикалі – значення ознаки Хаара для відповідних прикладів. Позитивні приклади позначені заповненими точками, а негативні – незаповненими. Горизонтальна лінія на графіку позначає можливе порогове значення.

Відсортувавши приклади за їх значенням ознаки Хаара отримуємо наступний графік:

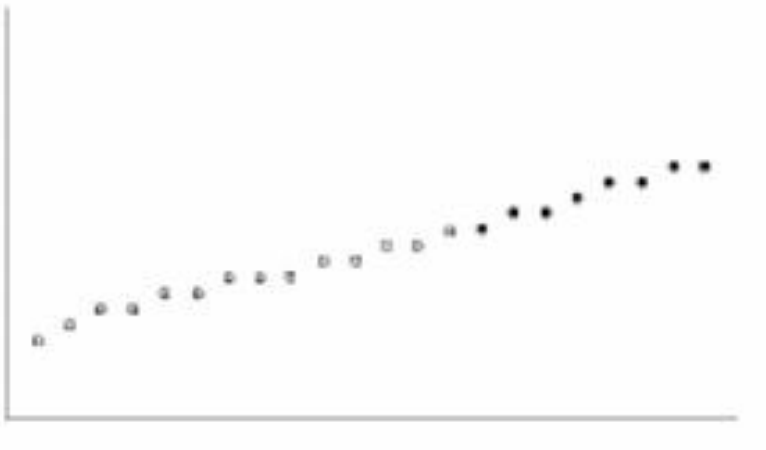


Рисунок 2.5 – Відсортовані значення ознак Хаара

З рис. 2.4 можемо зробити висновок, що всі порогові значення, які знаходяться між двома сусідніми значеннями на відсортованому графіку дають однакові результати класифікації. Відповідно, щоб отримати найменшу похибку, необхідно перевірити тільки n-1порогових значень, де

n – кількість прикладів. Можна обрати будь-яке значення, наприклад середнє між двома сусідніми відсортованими ознаками Хаара.

Проте це не вирішує повністю проблему з часом: перебрати всі ознаки все одно займає досить багато часу. Ключова причина цього – повторюване обрахування помилки слабкого класифікатора, яка потребує велику кількість операцій сумування.

На сьогодні існує рішення, яке нівелює вплив великої кількості сумувань. Наведемо послідовність необхідних дій.

h - ознака Хаара, для якої шукаємо порогове значення;

E_i – і-ий приклад;

y_i – 0, якщо і-ий приклад негативний, 1, якщо і-ий приклад позитивний;

n – кількість прикладів;

ω_i – вага відповідного і-ого прикладу;

ω^+, ω^- - додаткові масиви;

$\overline{\theta}$ – порогове значення, що перевіряється на даному етапі;

$\overline{\varepsilon}$ – помилка даного етапу;

ε – найменша помилка, яка спочатку дорівнює 1;

θ і p – порогове значення і напрямок знаку нерівності, за яких була отримана найменша помилка.

1. Відсортувати приклади. Ваги і елементи масиву y відсортувати відповідно до відсортованих прикладів, таким чином, щоб кожному прикладу до і після сортування відповідав один і той самий елемент і ваги з y.
2. Створити два додаткові масиви: в одному з них і-ий елемент містить суму вагів позитивних прикладів до і-ого значення, а в другому – негативних:

$$\omega_i^+ = \sum_{j=1}^{i} \begin{cases} \omega_i, if\ y_j = 1 \\ 0, if\ y_j = 0 \end{cases}$$

(2.9)

$$\omega_i^- = \sum_{j=1}^{i} \begin{cases} \omega_j, if\ y_j = 1 \\ 0, if\ y_j = 0 \end{cases}$$

3. Для всіх *i* від 2 до n:
    a. Обрахувати значення порогового значення, яке ми перевіряємо:

$$\overline{\theta} = \frac{h(E_{i-1}) + h(E_i)}{2}$$

(2.10)

    b. Порахувати помилку для p=1:

$$\overline{\varepsilon} = \omega_{i-1}^- + \omega_n^+ - \omega_{i-1}^+$$

(2.11)

    c. Якщо помилка порогового значення, що наразі перевіряється, менше поточної помилки, то запам’ятати це порогове значення і напрямок знаку нерівності:

$$if\ \overline{\varepsilon} < \varepsilon\ than\ \varepsilon = \overline{\varepsilon}, \theta = \overline{\theta}, p = 1$$

(2.12)

    d. Порахувати помилку для p=-1:

$$\overline{\varepsilon} = \omega_{i-1}^+ + \omega_n^- - \omega_{i-1}^-$$

(2.13)

    e. Якщо помилка порогового значення, яке наразі перевіряється менше за поточну помилку, то запам’ятати це порогове значення і напрямок знаку нерівності:

$$if\ \overline{\varepsilon} < \varepsilon\ than\ \varepsilon = \overline{\varepsilon}, \theta = \overline{\theta}, p = -1$$

(2.14)

2.3 Методи розпізнавання об’єктів

Загалом можна виділити 3 групи методів розпізнавання об’єктів за інструментарієм:

- Порівняння зі зразком: до цієї групи відносять методи і методики, які використовують категорію відстані та приближення (класифікації з найближчим середнім, відстань до найближчого сусіднього значення тощо).
- Статистичні методи: засновані на розрахунку ймовірності (наприклад, байєсівський метод).
- Нейронні мережі: використання навчання нейронних мереж. Дані методи розкривають можливість розпізнавання під час навчання мережі.

Наведемо також іншу альтернативну класифікацію методів за способом подання:

- Методи зіставлення цілком. Зіставлення об’єктів відбувається в цілому, не відокремлюючи окремих ознак. Перевагами є те, що мінімум важливої інформації може бути втрачено. Але недоліком є те, що дані методи є достатньо витратними, оскільки необхідно обробити велику кількість інформації, тому вони не завжди бувають продуктивними.
- Методи зіставлення за ознаками або структурні методи. Зіставлення об’єктів відбувається за певними характерними ознаками, особливостями. Головна перевага таких методів в тому, що вони стійкі до змін позиції, освітлення, розміру тощо. Але недоліком є те, що досить складно автоматично виділити оптимальний набір ознак для класифікації, а також висока складність обчислень.

- Гібридні методи. Найбільш за суттю схожі на принципи людського зору: об’єкт оцінюється як загалом, так і окремі його характерні риси.

На сьогодні одним з найпопулярніших методів розпізнавання об’єктів є метод Віоли-Джонса. Метод базується на декількох базових принципах:

- Інтегральне подання зображень за примітивами Хаара
- Робота з класифікатором на базі адаптивного бустинга
- Метод комбінування класифікаторів у каскад.

Ці підходи дозволяють методу Віоли-Джонса бути ефективним та швидкодіючим.

Розглянемо детальніше каскад класифікаторів. Якщо зменшувати порогове значення сильного класифікатору, то таким чином зменшується кількість неправильних негативних класифікацій, але збільшується кількість неправильних позитивних класифікацій. Отже, якщо сильний класифікатор складається з малої кількості слабких класифікаторів, знижуючи порогове значення і збільшуючи таким чином кількість неправильних позитивних класифікацій, маємо можливість мінімізувати кількість неправильних негативних класифікацій.

Базуючись на цій властивості, слабкі класифікатори можуть формувати каскади, який є набором сильних класифікаторів (стадій), через які послідовно проходить зображення, що проходить перевірку.

Наведемо алгоритм, який був описаний Віолою та Джонсом.

C - кількість наявних слабких класифікаторів;

C_i - кількість слабких класифікаторів на стадії *i;*

θ_i - порогове значення на стадії *i;*

s - порядковий номер ітерації;

ϕ - задане наперед значення;

P_i, N_i - частка неправильних позитивних та негативних класифікацій на стадії *i;*

a_i - коефіцієнт і-ого слабкого класифікатора.

1. Встановити першу стадію, коли кількість слабких класифікаторів дорівнює одиниці:

$$s = 1, C_s = 1$$

(2.15)

2. Встановити для даної стадії таке порогове значення, щоб усі позитивні приклади класифікувалися правильно:

$$set\,\theta_n\, to\, reach\, N_s = 0$$

(2.16)

3. Якщо частка неправильно класифікованих позитивних приклаідв на даній ітерації більше за ϕ, збільшити кількість слабких класифікаторів, якщо ні, то переходимо до наступної ітерації, встановлюючи кількість слабких класифікаторів рівну кількості слабких класифікаторів на попередній стадії збільшеному на одиницю:

$$if\ P_s > \phi, than\, C_s = C_s + 1,$$
$$else\, s = s + 1,\ C_s = C_{s-1} + 1$$

(2.17)

4. Якщо кількість слабких класифікаторів на даній ітерації менше доступної кількості слабких класифікаторів або частка неправдивих позитивних прикладів дорівнює нулю, то перейти до другого пункту:

$$if\ C_s < C\, or\, P_s = 0, go\, to\, the\, stage\, 2$$

(2.18)

5. Повернути стандартне порогове значення останньої ітерації, якщо з циклу вийшли по причині того, що закінчилися доступні слабкі класифікатори:

$$if\ P_s > 0, than\, \theta_n = \frac{1}{2}\sum_{i=1}^{C_s} a_i$$

(2.19)

Можемо бачити з другого пункту алгоритму, що шукание порогове значення знаходиться шляхом перебирання. Як відомо, на практиці даний підхід не є ефективним.

Щоб мінімізувати затрати часу для побудови каскаду класифікаторів, можна використовувати оптимізаційний підхід знаходження порогових значень слабких класифікаторів.

Для того, щоб застосувати даний метод, потрібно змінити умови, коли дане порогове значення вважається найкращим. Також необхідно взяти до уваги, що у порогового значення сильного класифікатора фіксується напрямок знаку нерівності (>).

Для слабкого класифікатора порогове значення вважається найкращим, якщо маємо мінімальну суму вагів неправильно класифікованих прикладів. Для сильних класифікаторів найкращим вважається таке порогове значення, при якому правильно класифікуються всі позитивні приклади, і отримуємо якомога більшу кількість негативних правильно розподілених прикладів.

Значення, яке порівнюється з пороговим значенням сильного класифікатора, є сумою коефіцієнтів тих класифікаторів, які без помилок розпізнали зображення, що подавалося на вхід:

$$\sum_{i=1}^{C} a_i \overline{h_i}(E)$$

(2.20)

де C - кількість доступних слабких класифікаторів, a_i - коефіцієнт і-ого слабкого класифікатору, $\overline{h_i}$ -і-ий слабкий класифікатор, E - зображення, що подається на вхід.

Отже, для того, щоб всі позитивні приклади були розподілені вірно, порогове значення має бути найменшим серед всіх таких сум для хороших прикладів.

Враховуючи, що нижче порогове значення призводить до більшої кількості поганих прикладів, які будуть класифіковані невірно, умову необхідно подавати в наступному форматі: порогове значення має бути найменшим серед усіх таких сум для позитивних прикладів на найменше можливе значення.

При такому розкладі, немає необхідності в сортуванні та перевірці обраного порогового значення, якщо порівнювати з пороговим значенням слабких класифікаторів. Таким чином ми можемо зробити обрахування сум для позитивних прикладів більш ефективним. Тому варто зосередитися на тому, що на кожній новій ітерації кількість слабких класифікаторів збільшується на один. Замість того, щоб щоразу перераховувати суми, можемо створити додатковий масив (для кожного позитивного прикладу окремий елемент масиву) і з кожною ітерацією додавати додаткові елементи.

Наведемо алгоритм, який враховує вище вказані оптимізаційні елементи.

C - кількість наявних слабких класифікаторів;

C_i - кількість слабких класифікаторів на стадії *i;*

θ_i - порогове значення на стадії *i;*

s - порядковий номер ітерації;

ϕ - задане наперед значення;

P_i, N_i - частка неправильних позитивних та негативних класифікацій на стадії *i;*

a_i - коефіцієнт і-ого слабкого класифікатора;

g_i - і-ий позитивний приклад;

S_i - і-ий елемент додаткового масиву, який відповідає і-ому позитивному прикладу;

δ - мінімальне можливе число, яке можливо відділити від зменшуваного.

1. Встановити кожному елементу додаткового масиву нульове значення:

$$S_i = 0$$

(2.21)

2. Встановити першу ітерацію, в якій кількість слабких класифікаторів дорівнює одиниці:

$$s = 1$$
$$C_s = 1$$

(2.22)

3. Для всіх позитивних прикладів: у випадку, якщо останній класифікатор розподілив приклад правильно, то необхідно додати коефіцієнт цього класифікатора до значення у відповідному елементі масиву, що прив'язаний до даного прикладу:

$$S_i = S_i + \overline{h_{c_s}}(g_i)$$

(2.23)

4. Визначити для порогового значення даної ітерації найменше з найменшого елементу додаткового масиву на мінімально можливе число:

$$\theta_s = \min(S_i) - \delta$$

(2.24)

5. Якщо частка неправильних позитивних прикладів даної ітерації більше за ϕ, необхідно збільшити кількість слабких класифікаторів, а інакше перейти до наступної ітерації, встановивши в ній таку кількість слабких класифікаторів, щоб дорівнювала кількості слабких класифікаторів на попередній стадії, збільшивши на одиницю:

$$if\ P_s > \phi, then\ C_s = C_s + 1$$
$$else\ s = s + 1, C_s = C_{s-1} + 1$$

(2.25)

6. Якщо кількість слабких класифікаторів на даній ітерації менше доступної кількості слабких класифікаторів чи частка неправильних визначень позитивних прикладів дорівнює нулю, перейти до другого пункту.

7. Повернути стандартне порогове значення останньої ітерації, якщо з циклу вийшли по причині того, що закінчилися доступні слабкі класифікатори:

$$if\ P_s > 0, than\ \theta_s = \frac{1}{2}\sum_{i=1}^{C_s} a_i$$

(2.26)

Описаний вище алгоритм майже точно реалізований в бібліотеці OpenCV і має досить непогану продуктивність. Проте існує ряд недоліків даного підходу.

Перший недолік застосування каскадів Хаара в OpenCV є непрозорість реалізації задачі. За рахунок великої кількості складних розрахунків в алгоритмі, навіть за умови відкритого коду, важко вирішити потенційні проблеми, оскільки практично неможливо визначити їх перебіг та потенційні рішення. Можливі проблеми зі стабільністю алгоритму. Як приклад, можлива ситуація, коли може виникнути безкінечний цикл.

2.4 Методи розпізнавання аудіообразів

В основі розпізнавання аудіообразів лежить, як правило, спектральний аналіз. Цифровий спектральний аналіз- це сукупність методів цифрової обробки сигналів, які дозволяють оцінити частотний склад (спектр) сингала, що досліджується.

Даний аналіз є базовим для вирішення ряду задач, зокрема, в сейсмології (визначення землетрусів, зсувів чи інших сейсмологічних явищ), геофізиці (пошук корисних копалин), в системах зображень, компресії мови тощо.

Існують різноманітні способи для отримання спектру. Розглянемо один з найбільш поширених з них – алгоритм швидкого перетворення Фур'є.

Швидке перетворення Фур'є або БПФ – це різновид дискретного перетворення Фур'є (ДПФ), яке обраховується за формулою:

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk}, 0 \le k \le N-1$$

(2.27)

де $X(k)$ - k-а комплексна амплітуда (складова) спектра; $x(n)$ - відрахунки дискретного сигналу(періодичного з періодом N чи кінцевої довжини N); W_N^{nk} - повернутий множник (чи ядро перетворення), який дорівнює:

$$W_N^{nk} = e^{-j\frac{2\pi}{N}nk}$$

(2.28)

Пряме вирахування за формулою 2.27 для великих N (при обробці звукових сигналів довжина звукового сигналу може досягати $2^{10} = 1024$) є неефективним, велика кількість операційне дає можливість працювати в режимі реального часу. Для обрахування N-точкового перетворення необхідно провести $(N-1)^2$ комплексних множень і N (N-1) комплексних сумувань, а отже об'єм розрахунків має порядок N^2 операцій додавання та множення комплексних чисел.

Для зменшення розрахункових витрат часу були розроблені алгоритми БПФ, які засновані на періодичності ядра перетворення W_N^{nk} . Ідея БПФ в тому, щоб розділити N-точкову послідовність на дві, із ДПФ яких можна отримати ДПФ вихідної послідовності, і продовжувати таке ділення

кожної нової послідовності до того часу, поки не залишаться послідовності, які складені тільки з двох елементів.

Розрізняють БПФ з проріджуванням по часі і частоті.

В алгоритмі БПФ з проріджуванням по часі N-точкова послідовність ділиться на дві N\2-точкові послідовності, одна з яких X_1 містить відрахування з непарними номерами, а інша X_2 – з парними номерами. Тоді N-точкове ДПФ вихідної послідовності перетворюється в два N\2-точкових ДПФ:

$$X(k)=\begin{cases}X_1(k)+W_N^k X_2(k)\\ X_1(k)+W_N^k X_2(k)\end{cases},0\le k\le N-1$$

(2.29)

Далі аналогічним чином N\2-точкові ДПФ замінюють двома N\4-точковими і так далі. Таке сортування продовжується до тих пір, поки не буде створена N\2 послідовностей по два елементи в кожному. В результаті N-точкове ДПФ зводиться до $m=\log_2 N$ етапам, на кожному з яких необхідно обчислити N коефіцієнтів.

Формула 2.29 показує, що на кожному етапі необхідно N комплексних множень з $N^2\frac{3}{2}N\log_2 N$, що є суттєвою економією обрахункових і часових ресурсів.

В алгоритмах БПФ з проріджуванням по частоті вхідна послідовність ділиться навпіл на N\2 перших і N\2 останніх відрахувань до того часу, поки не буде сформовано N\2 двоелементних послідовностей, що призводить до подібного скорочення обрахункових операцій. Розрахунки відбуваються за наступною формулою:

$$X(k)=\begin{cases}X_1(k)+X_2(k)\\ (X_1(k)-X_2(k))W_N^k\end{cases},0\le k\le N-1$$

(2.30)

Завдяки своїм перевагам, алгоритми БПФ є дуже поширеними в сучасних технологічних рішеннях, в рамках яких необхідно працювати зі спектром, аналізувати чи перетворювати його. Його широко використовують в навчальних ПО, наприклад в Matlab чи Arduino. Наявність достатньо кількості бібліотек спрощують роботу з цим алгоритмом і дозволяють використовувати його в системах без необхідності детального вивчення принципів його роботи.

РОЗДІЛ 3 СИСТЕМА ВИЗНАЧЕННЯ ПАРАМЕТРІВ ДЖЕРЕЛА АКУСТИКО-ОПТИЧНОГО ВИРОМІНЮВАННЯ

3.1 Характеристика мінікомп'ютера Raspberry PI

Raspberry Pi – це одноплатовий мінікомп'ютер (Рисунок 3.1), створений фондом Raspberry Pi Foundation.

Рисунок 3.1 – Мінікомп'ютер Raspberry Pi

Перша версія Raspberry Pi була випущена у 2013 році. Основною задачею при створенні мінікомп'ютера було створення пристрою для кращого вивчення комп'юрених наук у школах. Але мінікомп'ютер почав набирати популярність серед інтузіастів, які хотіли створити щось нове у галузі Internet of Things (IoT).

Популярність мінікомп'ютер набув в першу чергу через свою вартість, яка в залежності від комплектації варіюється від 5 до 35 доларів США.

У нашій системі було використано одну з перших моделей, а саме Raspberry Pi Model B+. Така версія мікрокомп'ютера володіє наступними властивостями.:

- SoC Broadcom BCM2835 (CPU, GPU, DSP и SDRAM)
- CPU: 700 МГц ARM1176JZF-S core (сімейство ARM11)
- GPU: Broadcom VideoCore IV, OpenGL ES 2.0, 1080p30 h.264/MPEG-4 AVC high-profile decoder
- Пам'ять (SDRAM): 512 Мб

Замість твердотільного накопичувача, використовується microSD картка, у нашому випадку це microSD від компанії Samsung об'ємом на 32Гб. Такого об'єму більш ніж достатньо для операційної системи, програмного забезпечення, яке йде разом з операційною системою і власних напрацювань.

Для мінікомп'ютера Raspberry Pi розроблено декілька операціних систем, а саме:

- Raspbian – модифікація операційної системи Debian. Є рекомендованою для початківців.
- Pidora – модифікація операційної системи Fedora
- Windows 10 з версії 2B.

Мінікомп'ютер Raspberry Pi Model B+ має такі інтерфейси підключення зовнішніх пристроїв:

- Gigabit Ethernet (Microchip LAN7515, швидкість до 300Mbps через USB 2.0 шину)
- Слот для карт microSD
- USB 2.0 x 4
- Повнорозмірний HDMI
- Композитний 3.5 jack для виводу звуку/відео
- 40 пінів GPIO
- Інтерфейс для підключення камери CSI

- Інтерфейс для підключення дисплея DSI
- Живлення 5V 2.5A через microUSB.

На рисунку 3.2 показана схема розташування інтерфейсів на платі.

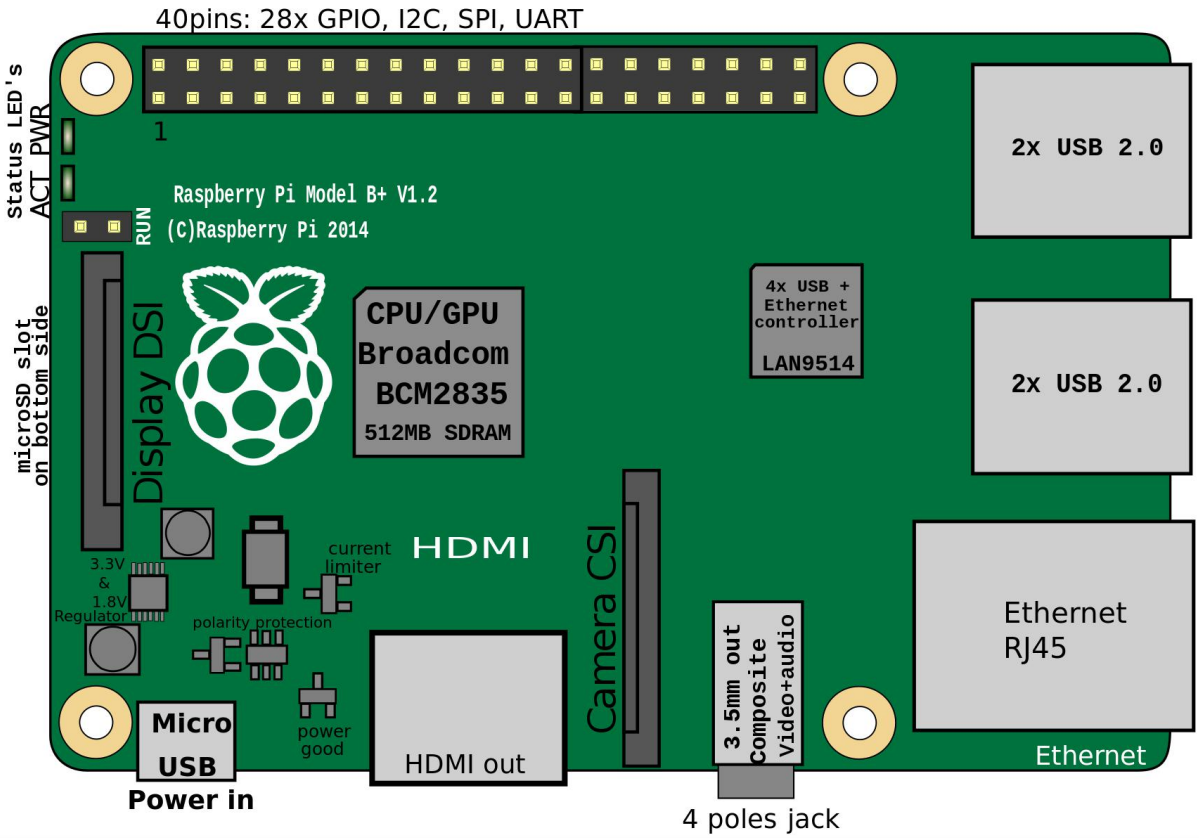


Рисунок 3.2 – Схема розташування інтерфейсів на Raspberry Pi Model B+

Піни GPIO на Raspberry Pi бувають трьох типів (рис 3.3):

1. Живлення - 5V(позначені червоним на схемі) і 3.3V(позначені оранжевим)
2. Заземлення (позначені чорним)
3. Інформаційні

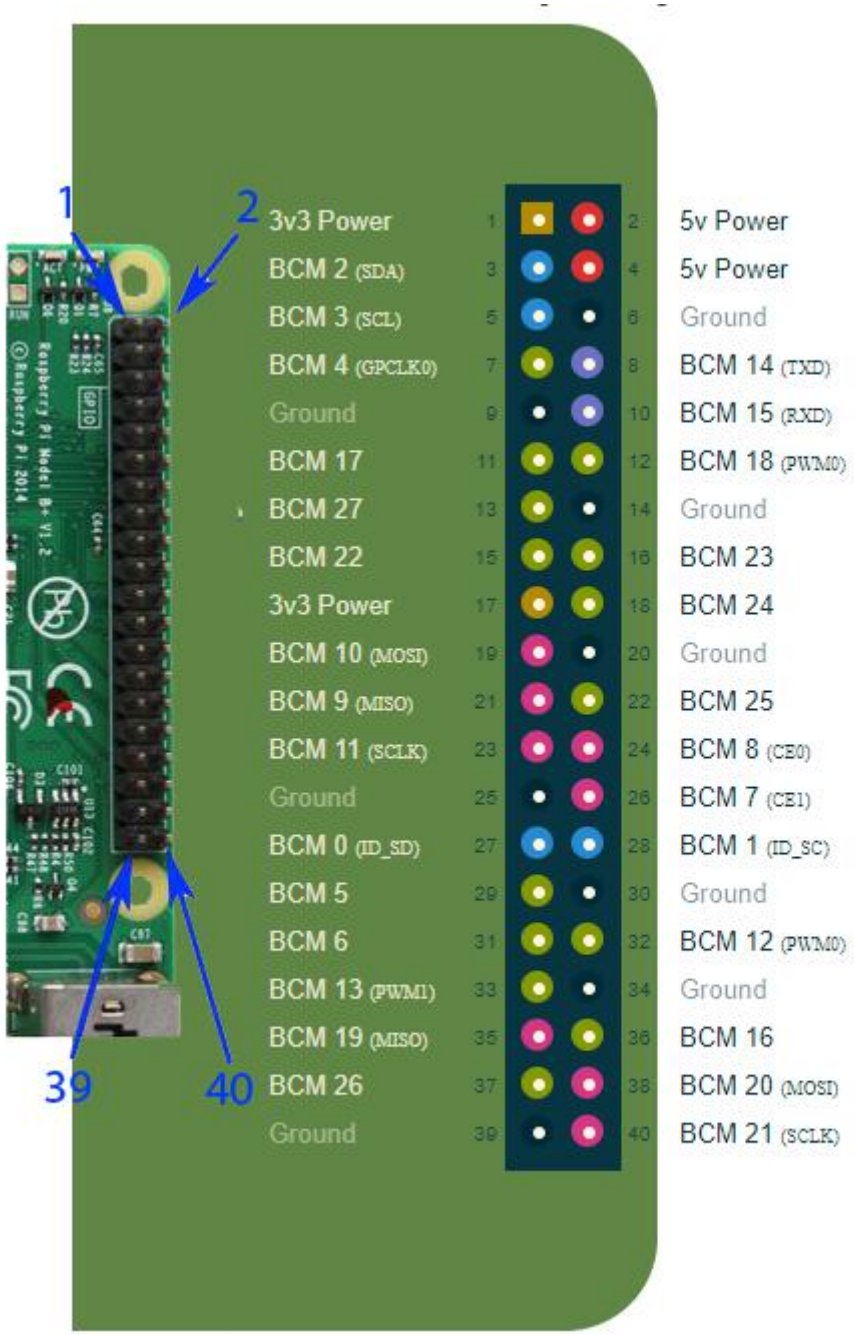


Рисунок 3.3 – Схема пінів GPIO на Raspberry Pi

Для створення системи визначення параметрів джерела акустико-оптичного випромінювання до Raspberry Pi Model B+ необхідно підключити наступні модуль:

- Камера Waveshare RPi G, кут огляду якого 160° для опрацювання оптичного випроміювання

Рисунок 3.4 – Камера Waveshare RPi G

- Сенсор звуку Waveshare для опрацювання акустичного випромінювання

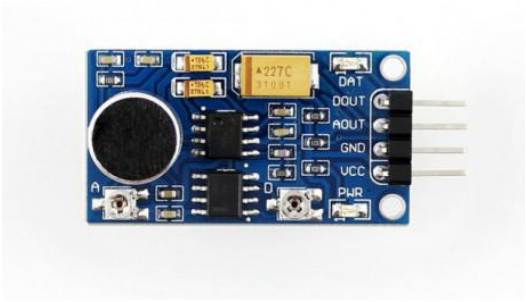


Рисунок 3.5 – Сенсор звуку Waveshare

- Wi-Fi адаптер TP-LINK TL-WN725N для зв’язку з персональним комп’ютером

Отже, ми маємо пристрій повністю готову для поставленої задачі. Після підключення всіх модулів пристрій виглядає так:

Рисунок 3.6. – Зібраний пристрій на основі Raspberry Pi Model B+

3.2 Архітектура розробленої системи

Для розробки пожежної системи ми взяли променеву схема АУПС як показано на рис 1.1. Кожен сенсор є відокремленим від інших і зв’язується тільки з комутатором. Raspberry PI з усіма піключеними модулями, як було описано вище виконує у розробленій системі роль сенсора.

На відмінну від всіх існуючих АУПС у нашй пожежній системі сенсори не тільки збирають дані з навколишнього середовища, а й проводять обчислення на основі отриманих даних.

Для розробки програмуного забезпечення ми вибирали з мов програмування, які підтримуються біблотекою OpenCV нативно або через обгортки.

Бібліотека OpenCV нативно підтримує такі мови програмування:

- C / C++
- Python
- Java
- Ruby
- Matlab
- Lua

На даний момент існують також обгортки для OpenCV написані для мов програмування:

- C / C++
- C#
- Perl
- Haskell
- Javascript

Серед усіх вище перечислених мов програмування найкращу пітримку мають C/C++ і Python. Звичайно найкращого результати можна досягти використовуючи C/C++ тому що С починалась розробка бібліотеки і вона досі підтримує API для С, а на C++ зараз продовжується розробка OpenCV. Python дуже вподобала спільнота користувачів бібліотеки OpenCV і тому розробники бібліотеки підтримують API для мови програмування Python підтримують на такому ж рівні, що й API для мови C++.

Варто також врахувати, що програмне забезпечення ми будемо писати на Raspberry PI, розробники якої рекомендують використовувати у Python.

Серед переваг Python варто відмітити також:

1. Велику кількість бібліотек і пакетів з якою може поконкурувати хіба тільки кількість пакетів для мови програмування Javascript
2. Лаконічний і чистий код

3. Простота освоєння мови
4. Велика стандартна бібліокека
5. Python підтримується усіма існуючими операційними системами

Серед недоліків можна звернути увагу на:

1. Низька швидкодія. Це зв’язано в першу чергу з тим що Python це інтерпретована, нетипізована мова програмування з
2. Відсутність типізації. Це насправді як і перевага так і недолік. Все залежить від задачі. В більшості випадків відстутність статичної типізації рахують як недолік, тому що присутність статичної типізації довзволяє знаходити помилки які зв’язані з типами ще на етапі написання коду. Але відсутність статичної типізації дозволяє писати код не переживаючи про типи, що можна рахувати за плюс. Але знову ж таки це залежить від задачі, яка стоїть перед розробником

Незважаючи на явні недоілки ми все ж вибрали мову програмування Python для розробки нашого програмного забезпечення.На це нас підштовхнули наступні моменти:

1. Пітримка спільноти розробників OpenCV
2. Пітримка спільноти розробників Raspberry
3. Велика кількість пакетів
4. Велика стандартна бібліотека

Отже, з мовою програмування ми оприділились. Залишилось вибрати платформу на якій будемо розробляти. Серед потеційних варіантів були операційні системи сімейства Linux і Windows.

Так як Raspberry PI поки погано пітримується операційною системою Windows, то для Raspberry PI вибір буде очевидний, а саме Raspbian, модифікація Debian.

Так як у нас уже є одна операційна система сімейства Linux, то логічно було б використовувати її всюди. Крім того у нашій роботі є

робота з сокетами, а на оперційни системі Windows немає можливості працювати з сокетами.

Тому розробка буде проводитись на операційних системах сімейства Linux на мові програмування Python.

Дальше розглянемо архітектурне рішення, яке ми використали в роботі.

Як було сказано к попередньому підрозділі мінікомп'ютер Raspberry Pi оснащений камерою і сенсором звуку. Ці два модулі збиратють дані для нашої системи. Аналіз отриманих даних проводиться на мінікомп'ютері, а на комутатор подається тільки результат роботи сисиеми для кінцевого користувача

Для зв'язку сенсорів і комутатора ми використали клієнт-серверну архітектуру. Архітектура типу клієнт-сервер – один з найбільш популярних архітекутрних рішень у сфері розподілених мережних застосунків.

У системі яка використовує таке архітектурне рішення є такі три компоненти:

1. Сервер
2. Клієнт
3. Мережа

Клієнти працюють паралельно один від одного паралельно і не знаю про існування інших. Дані через мережу передаються з клієнтів на сервер. Принцип роботи таких систем можна розглянути на рис. 3.7.

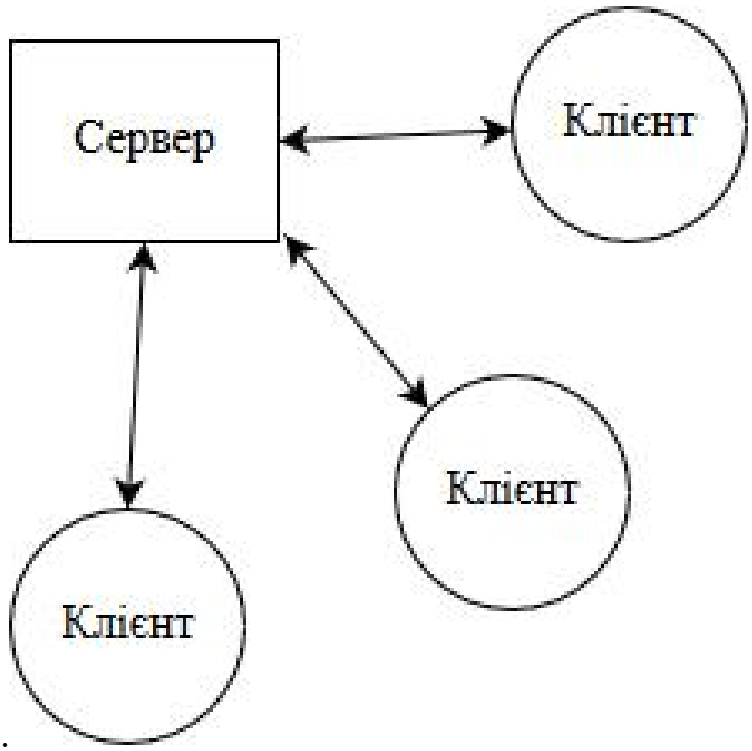


Рисунок 3.7 – Клієнт-серверна архітектура

Роль клієнта у нашій системі буде виконувати сенсор, тобто мінікомп'ютер Rspberry PI з усіма підключеними модулями.

Роль сервера виконує комутатор. Комутатором у нашій ПК з встановленим програмним забезпечення.

Роль мережі у системі виконує мережа на основі сокетів.

Отже, ми маємо наступну систему роботи сенсори отримують дані і передають через мережу на основі сокетів до комутатора, де вже інформацію отримує користувач системи.

Варто розглянути, що з себе представляє сокет перед тим як продовжувати і чому саме їх ми вибрали для зв'язку.

Сокет – це прогрманий інтерфейс для зв'язку і передачі даних між процесами. Процеси можуть бути виконуватись як на одному пристрої так і на різних.

Сокети бувають двох типів:

1. Клієнтский, які нагадують апаратами телефонної мережі

2. Серверні, як можна порівняти з комутаторами тієї ж телфеонної мережі.

Для коректної роботи мережі на основі сокетів необхідно, щоб було у мережі був хоч один сокет кожного типу. Як було сказано раніше клієнтський сокет може бути не на тому ж пристрої, що й серверний

Для отримання інформації створюється серверний сокет, який прив’язується до певного порту операційної системи. Всі клієнти, які хочуть надати інформацію серверу, надсилають на фізицний адрес сервера в мережі (локальне чи підключення до мережі Інтернет) і відповідний на відповідний порт серверу. На сьогоднішній день такий зв’язок доступний тільки на операційних системах сімейства UNIX.

При з’єднанні клієнтського сокет і серверного, останній запам’ятаовує параметри першого і доки зв’язок не перерветься сокети можуть обмінюватись інформацією. На відмінну від систем, де зв’язок відбувається через HTTP, зв’язок не перериваєтся після отрмання клієнтом відповіді з сервера.

Тобто, якщо підключити декілька клієнтів до одного сервера, то сервер буде знати про існування всіх клієнтів і при потребі може сам зв’язатись з клієнтом або розсилати дані одночасно всім клієнтам.

Приклад роботи мережна на основі сокетів можна побачити на рис 3.8.

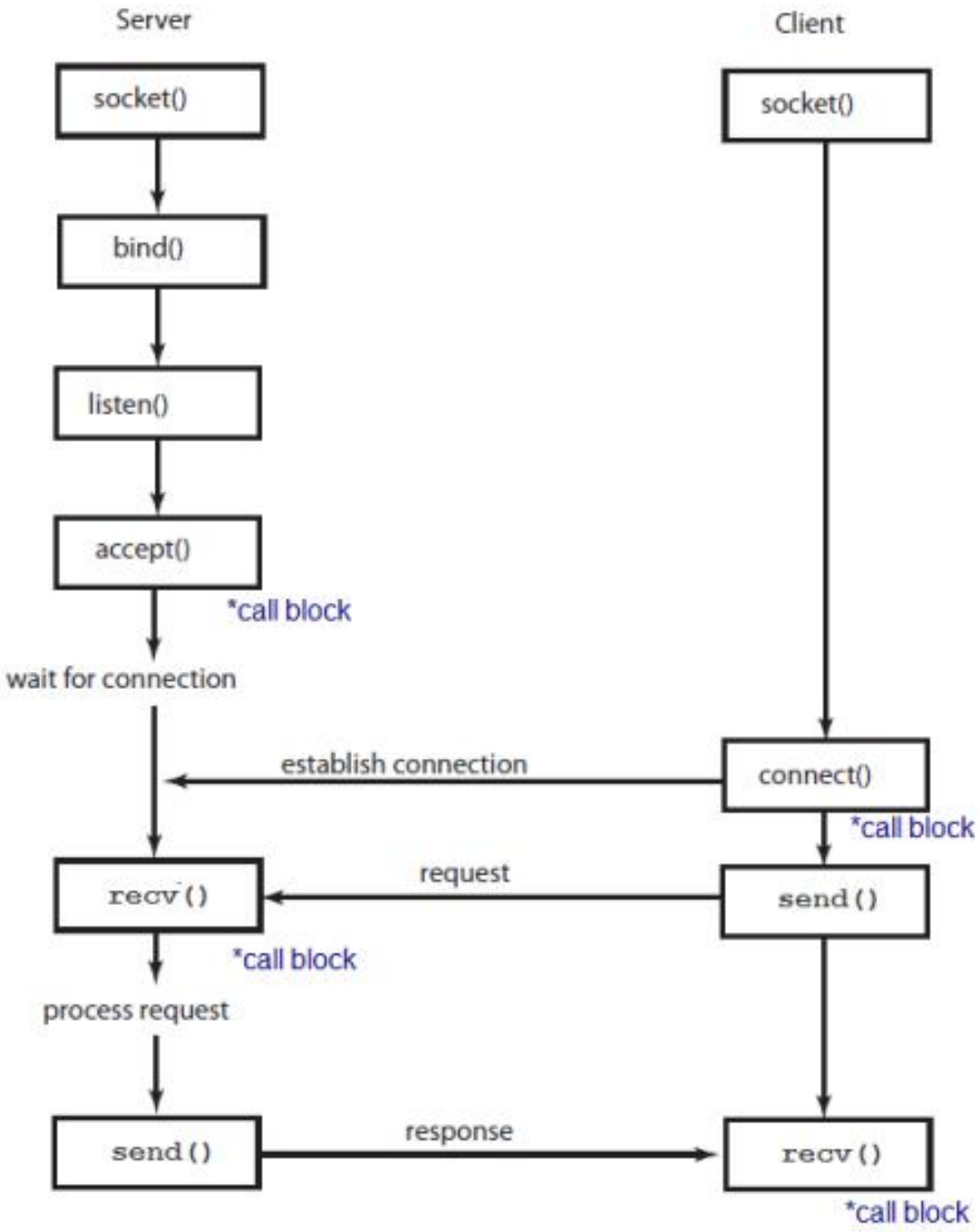


Рисунок 3.8 – Принцип роботи мережі на основі з’єднання через сокети

Отже, як архітектурне рішення було використано клієнт-серверну архітектуру зі з’єднанням на основі сокетів. Система складається з 3 основних компонентів:

а) Клієнт – програмне забезпечення на укоплектованому камерою та сенсором звуку мінікомп’ютері Raspberry PI

b) Сервер – програмне забезпечення на персональному комп’ютері
c) Мережа – програмне забезпечення для зв’язку клієнта і сервера через сокети.

Розглянемо детальніше принцип роботи і алгоритми по яких працює кожен компонент.

Клієнт поділяється на 7 логіних блоки:

1. Блок, який відповідає за зчитування даних з камери
2. Блок, який відповідає за зчитування даних з сенсора звуку
3. Блок, який опрацьовує дані отримані з камери
4. Блок, який опрацьовує дані отримані з сенсора звуку
5. Блок, зберігання результатів отриманих з блоків 3 і 4
6. Блок аналізу результатів
7. Блок передачі даних через мережу

Блоки 1, 3 ,5 праюють по алгоритму на рис. 3.9.

Камера записує відео і покадрово передає на опрацювання. Під час опрацювання, за допомогою алгоритму Віоли-Джонса, відбувається пошук полум’я на кожному отриманому кадрі. У разі знаходження полум’я кадр передається в блок зберігання даних і там архівається у формат pickle.

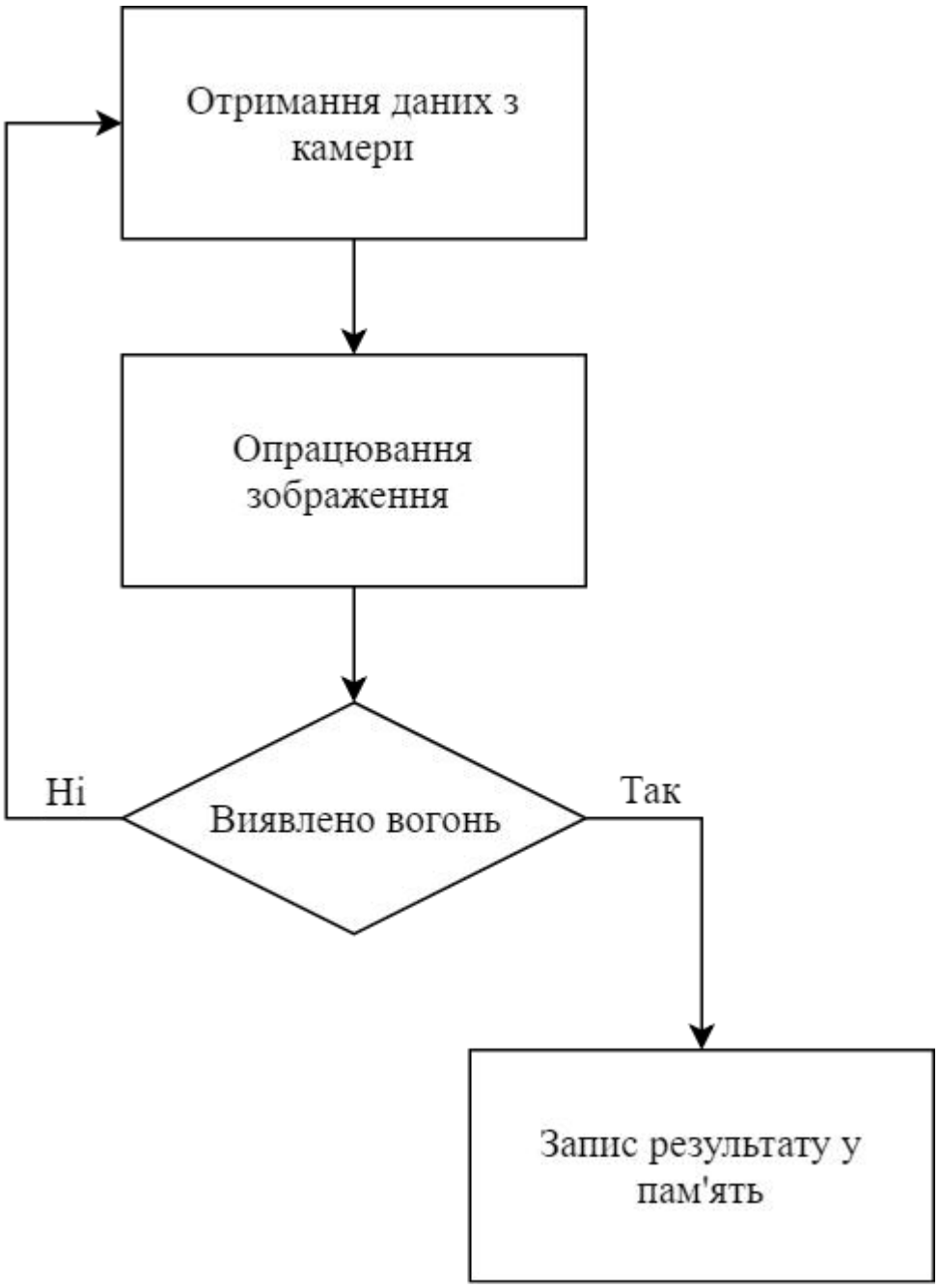


Рисунок 3.9 – Блок схема роботи блоків 1,3 і 5.

По схожому принципу працюють блоки 2, 4, 5 (Рис. 3.10).

Сенсор звуку записує аудіо і передає на опрацювання. Під час опрацювання, за допомогою аналізу частот, відбувається пошук звуку загорання. У разі знаходження в блок зберігання даних передається команда створити запис у структуру даних інформацію, по те що звук виявлено..

Рисунок 3.10 – Блок схема роботи блоків 2,4 і 5.

Залишилось розглянути як зв’язані між собою блоки 5, 6 і 7. Блок-схема для цих блоків можна розглянути на рис. 3.11.

Як було сказано вище, дані після опрацювання блоками 3 і4 потрапляють на блок зберігання даних. Зображення з виявленим полум’ям перетворюється у формат pickle, а команда з блока 2 записує у пам’ять значення, що звук був виявленим. Перетворення зображення у формат

pickle необхідне для передачі даних через мережу. Формат pickle найкраще підходить для зберігання даних у програмах написаних на Python.

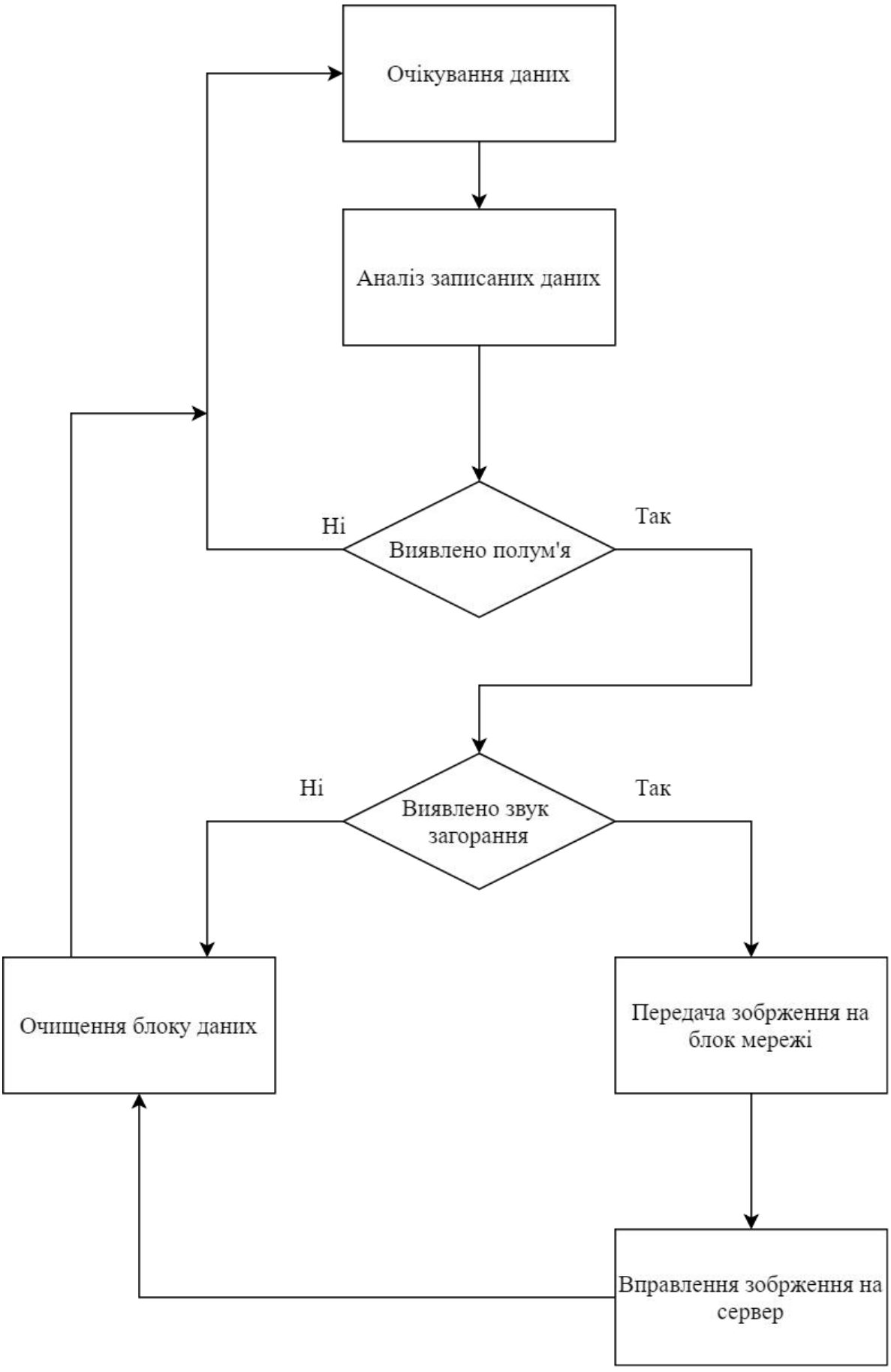

Очікування даних
Аналіз записаних даних
Виявлено полум'я
Ні
Так
Виявлено звук загорання
Ні
Так
Очищення блоку даних
Передача зобрження на блок мережі
Вправлення зобрження на сервер

Рисунок 3.11 – Блок схема роботи блоків 5, 6 і 7.

Отже ми отримали дані з аналізатора зображень про виявлення полум’я і записали їх в систему. Далі система чекає на дані з сенсора звуку. Якщо через 3 секунди таких даних немає, то зображення видаляється. Якщо через 3 секунди такі дані надходять, то система передає зображенння на блок мережі. Блок мережі упаковує зображення у пакет і відправляє на сервер. Після чого всі дані з блоку пам’яті видаляються.

Як бачимо всі обчислення і аналіз відбуваються на клієнті, а на сервер передається уже підготовлена інформація.

Тепер розглянемо алгоритм роботи мережі на основі сокетів. З блок-схемою можна ознайомитись на рис.3.12.

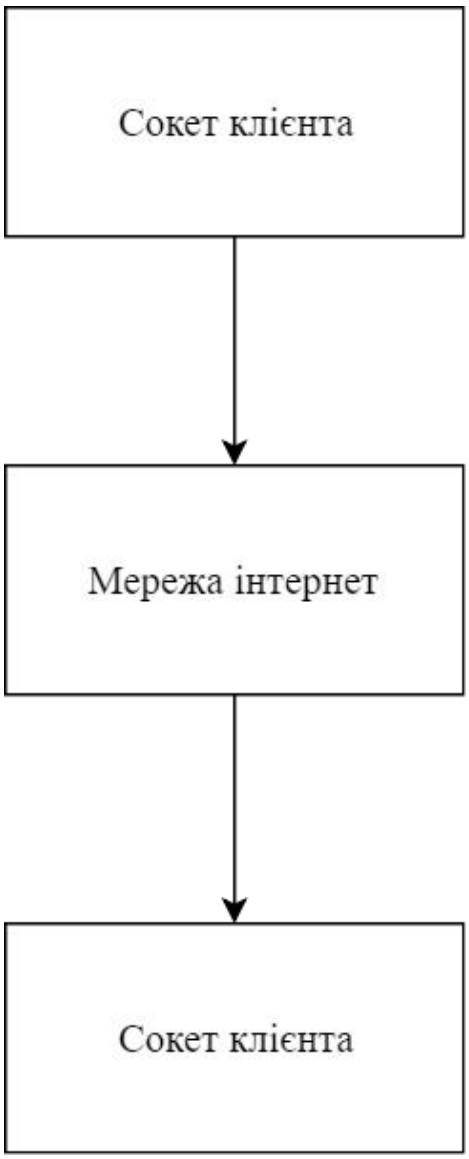


Рисунок 3.12 – Блок схема роботи мережі на основі сокетів

Як бачимо, дані з сокета клієнта через мережу інтернет передаються на сокет сервера.

Роль сервера у нашій системі досить проста. З блок-схемою роботи серевера можна ознайомитись на рис.3.13.

Після отримання даних, сервер перетворює зображення з формату pickle у jpg. Після перетворення сервер відкриває зображення на екрані користувача і лунає звук сирени.

Так як всі розрахунки відбуваються на клієнті, то алгоритм роботи сервера досить примінивний.

Рисунок 3.13 – Блок схема роботи сервера.

3.3 UML діаграми розроблених класів

Наше програмне забезпечення складається з 2 пакетів (рис. 3.14)

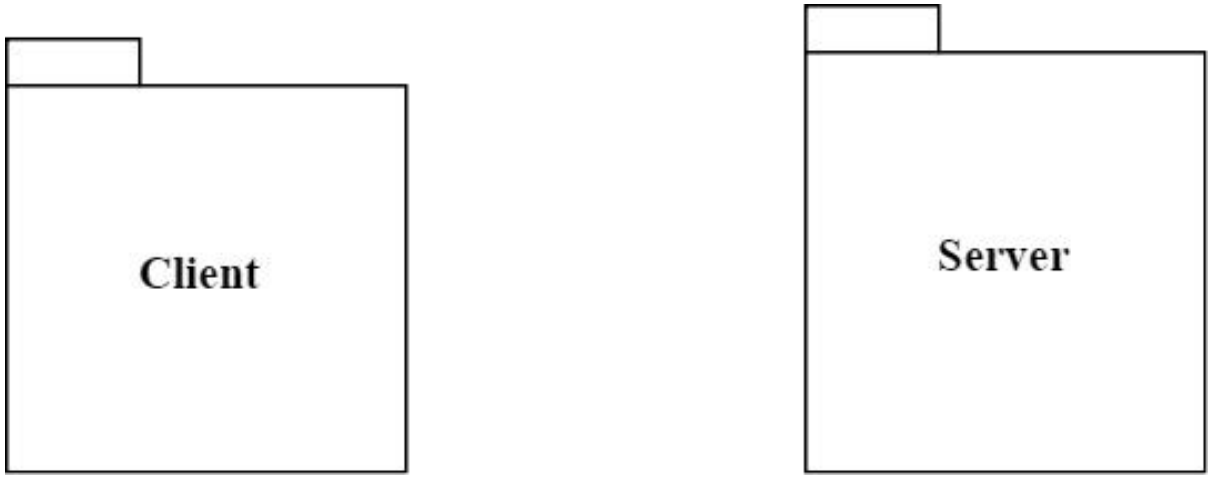


Рисунок 3.13 – UML діаграма пакетів прогрмного забезпечення

У пакеті Client знаходяться всі класи, які відповідають за:

1. Запис відео
2. Запис аудіо
3. Опрацювання зображення
4. Опрацювання аудіо
5. Перетворення даних
6. Зберігання даних
7. Видалення непотрібних даних
8. Аналіз даних
9. Створення зв’язку з сервером
10. Передачу даних на сервер

Пакет Client складається з таких класів:

1. SocketClient
2. Detector

Клас SocketClient (рис. 3.14) необхідний для зв’язку з сервером через сокети.

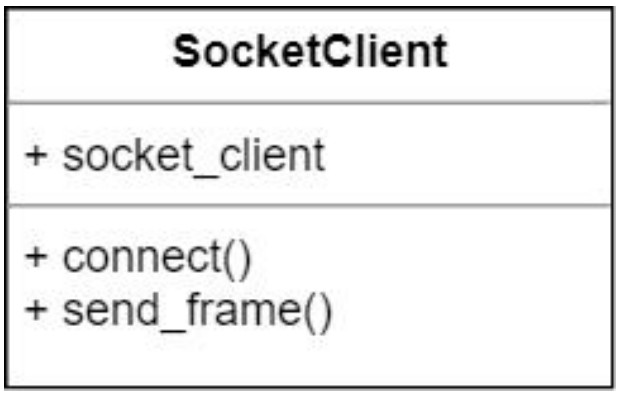


Рисунок 3.14 – UML діаграма класу Client

Розглянемо детальніше клас Client.

Як бачимо клас Client має атрибут socket_client і методів connect та send_frame.

Метод socket_client відповідає за створення підключення клієнта до сервера.

Передача зображення на сервер відбувається за допомогою методу send_frame. Перед відправлення зображення на сервер, зображення перетворюють у формат pickle. Під час створення програмного забезпечення ми розглядали декілька варіантів, перетворення зображення, серед яких:

- Перетворення зображення у інший формат
- Архівування зображення
- Упакування у структуру даних

Зупинилися ми на останньому методі, оскільки він дозволяє непогано зберегти якість зображення і без проблем опрацювати програмним забепеченням.

Серед варіантів у який формат файлу серіалізувати зображення ми розглядали:

- JSON
- YAML
- Pickle

Вибір впав на останній, оскільки цей тип серіалізації нативно підтримується мовою прогрмування Python.

Клас Detector це головний мозок нашого програмного забезпечення. Тут відбуваються:

1. Запис відео
2. Запис аудіо
3. Опрацювання зображення
4. Опрацювання аудіо
5. Перетворення даних
6. Зберігання даних
7. Видалення непотрібних даних
8. Аналіз даних

З UML діаграмою класу Detector можна ознайомитись на рис. 3.15.

Рисунок 3.15 – UML діаграма класу Detector

Розглянемо детальніше атрибути класу Detector.

1. video_stream – сюди записуються дані отримані з камери.
2. fire_cascade – тут зберігаєься каскадний класифікатор Хаара підготовлений для виявлення полум’я і який попередньо зберігається у файлі формату XML
3. host – фізична адреса сервера з яким створюється зв’язок
4. port – порт сервера з яким створюється зв’язок
5. timeCheck – змінна потрібна для визначення часу який необхідного для роботи програмного забезпечення на стороні клієнта
6. socket_client – об’єкт класу Client, якому ми передаємо результат роботи на відправлення.
7. sound_detected – змінна типу boolean у яку записується значення True, якщо на аудіо було виявлено звук загорання. У іншому випадку значення залишається False.

Клас Detector має наступні методи:

1. start() – запускає роботу програмного забезпечення
2. detect_flame_on_video() – цей метод отримує зображення на аналіз. Під час виконання методу відбувається аналіз зображення по методу Віоли-Джонса, з використанням примітивів Хаара. Як було сказано вище, каскадний класифікатор Хаара зберігається у змінній fire_cascade. Якщо полум’я знайдено, то полум’я виділяється на зображенні і передається дальше.
3. detect_flame_on_audio() – цей метод отримує на аналіз аудіо. Тут відбувається аналіз аудіо на основі перетворення Фур’є . Після аналізу, дані передаються на порівняння.
4. save() – цей метод зберігає дані у об’єкт. Для зображення це просто файл, для аудіо це глобальна змінна типу boolean,

sound_detected, яка означає, що у аудіо було знайдено звуки полум’я.

5. check_data() – цей метод після аналізу зображення, перевіряє чи є в періоді 5 секунд у системі одночасно зображення, яке появилось після аналізу відео і значення змінної sound_detected дорівнює True. Якщо обидві умови виконуються, то запускається функція send_frame().
6. stop() – цей метод зупиняє прогрмане забезпечення по нажиманюю клавіші “q”.
7. send_frame() – цей метод викликає, метод класу Client і передає йому зображення, на якому виділено вогонь червном прямокутником. І далі зображення передається на сервер.

Тепер розглянемо пакет Client. У пакеті Client у нас є тільки клас SocketClient. UML діаграма цього класу можна розглянути на рис.3.16.

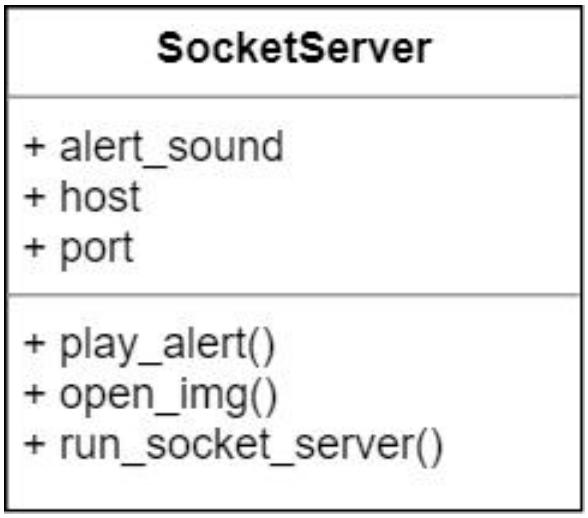


Рисунок 3.15 – UML діаграма класу SocketServer

Розглянемо детальніше атрибути класу Detector.

1. alert_sound – у цю змінну загружається звук сирени, який лунає при отриманні даних з клієнта.
2. host – фізична адреса сервера на який будуть проходити дані клієнта
3. port - порт сервера на який будуть проходити дані клієнта

Клас SocketClient має наступні методи:

1. play_alert() – відтвоює звук тривоги, який попередньо загружений у змінну alert_sound.
2. run_socket_server() – запускає сокетний сервер, на якій і будуть дані з клієнта.
3. open_img() – перетворює зображення з формату pickle у формат jpg,, створює вікно і на ньому показує зображення. Запис зображення у файл потрібний на випадок якщо користувач випадково закриє вікно.

3.4 Користувацький інтерфейсу

Користувацький інтерфейс програмного забезпечення досить примітианий.

Запуск програмного забезпечення на клієнті відбувається за допомогою CLI(command line interface). На стороні клієнта виводяться настпуні команди:

1. Program started – повідомляє про запуск програмного забезпечення
2. Fire detected on picture – повідомляє про виявлення полум’я на зображенні
3. Fire detected on audio – повідомляє про виявлення звуку загорання на аудіо.
4. Sended to server – зображення відправлене на сервер.

Процес роботи клієнта можна побачити на рис. 3.16

Рисунок 3.16 – Процес роботи клієнта

Запуск програмного забезпечення на сервері відбувається також за допомогою CLI. У консоль виводять на ступні компанди:

1. Program started – повідомляє про запуск програмного забезпечення
2. Image from “host” – повідомляє про отримання зображення з клієнта і виводить дані клієнта з якого прийшло повідомлення.

Процес роботи сервера можна побачити на рис. 3.17

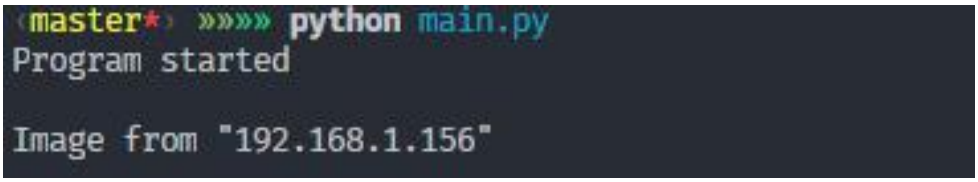


Рисунок 3.17 – Процес роботи сервера

На сервері також на екран виводиться зображення на якому позначено виявлений вогонь (рис. 3.18).

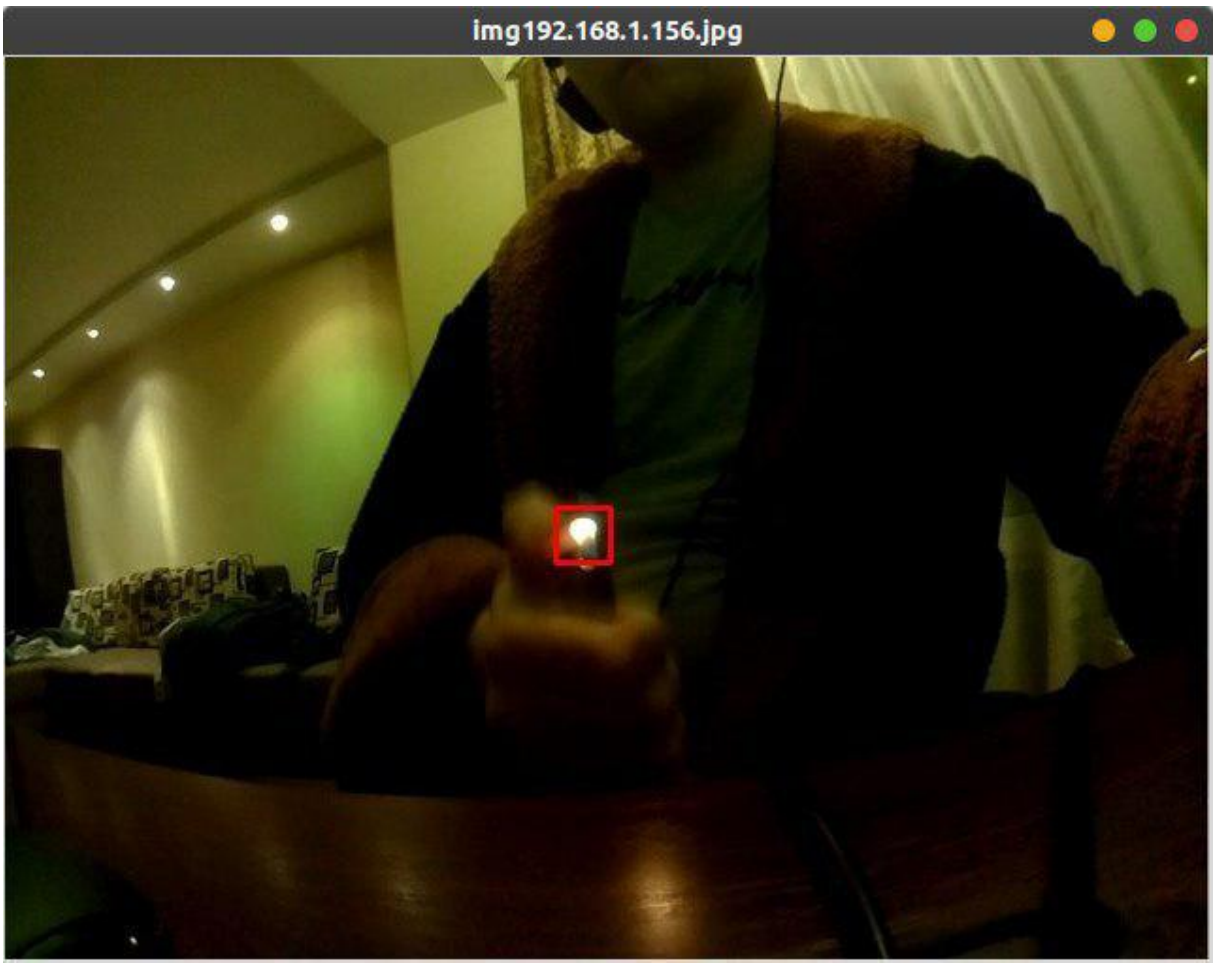


Рисунок 3.18 – Результат роботи прогрманого забезпечення

РОЗДІЛ 4 ТЕСТУВАННЯ І АНАЛІЗ РОЗРОБЛЕНОЇ СИСТЕМИ

4.1 Оцінка результатів системи пошуку полум’я у відео

Для оцінки роботи системи пошуку полум’я ми будемо використовувати наступні параметри:

1. Точність системи
2. Швидкість роботи системи
3. Завадостійкість

Точність системи визначається тим наскільки ймовірно, що полум’я буде виявлене на відео. Спочатку були певні проблеми. Тоді ймовірність виявлення складала приблизно 0.75. Тобто полум’я знаходилося у 3 з 4 тестів. Це досить непогана точність, але не така на яку ми розраховували

Для підвищення точності ми змінили характеристи по яких каскадний класифікатор Хаара визначав чи є полум’я на зображенні чи ні. Може виникнути питання чому пошук відбувається на зображенні, а не на відео як написано в назві підрозділу. Це зв’язано з тим що відео це послідовність зображень(кадрів).

Швидкість роботи системи визначається періодом який необхідний для виявлення полум’я на зображенні. У першому випадку, коли була більша похибка, швидкість системи була більша. Системі в середньому вистачало 0.7-1.0 секунди для знаходження полум’я. Варто також врахувати, що розрахунки проводились на мінікомп’ютера Raspberry PI Model B+ обчислювальна потужність якого досить мала. На ПК час був би набагато меншим.

Після модифікації характеристик, яка була націлена на зібльшення точності, час необхідний для виявлення полум’я збільшився. Це в першу чергу зв’язано з тим, що зросла складність очислень і відповідно потрібно більше часу на їх виконання. Після модифікації час пошуку зріс на приблизно 0.2 секунди.

Завадостійкість системи визначається ймовірністю, що системи сприйме як полум’я якийсь інший об’єкт. З такими проблемами ми не стикались. Як можна побачити на рис. 3.18 система не реагує на світіння лампочок, хоча по розмірах вони майже ідентичні з полум’я. Така ситуація можлива завдяки тому, що класифікатор Хаара шукає полум’я по певних характеристиках, а світіння ламп володіє не такими характеристиками. Різниця у :

1. Яскравості
2. Кольоровій гаммі
3. Інтенсивності світла

Як бачимо після модифікацій параметрів нам вдалось покращити точність системи і збільшити завадостійкість, але прийшлось заплатити зібльшенням часу, потрібного на обчислення.

4.2 Оцінка роботи всієї системи

Оцінку роботи всієї системи будемо проводити по наступних параметрах:

1. Точність системи
2. Час реагування
3. Завадостійкість
4. Вартість

Порівнювати будемо з системами АУПС, які використовують сенсори тепла та диму.

Точність нашої системи залежить в першу чергу від точності системи пошуку полум’я у відео. Система аналізу аудіо виконую роль більш корегуючого фактора.

У всіх системах включно з нашою точність роботи залежить від фактора по якому оприділяється екстренна ситуація. Для сенсорів диму це кількість диму в приміщенні, для сенсорів тепла це досягнення верхньої планки

темпаратури або різка зміна температури в приміщенні, а у нашій системі це полум’я, яке попадає в об’єктив камери. Нагадаю, що у нашій системі кут огляду камери складає 160 градусів. Всі три системи мають достатньо хорошу точність роботи. А от решта параметрів сильно різняться.

Час реагування визначається періодом часу, який пройшов з початку займайння і до включення пожежної сигналізації.

Для систем на основі теплових сенсорів цей показник досить поганий. В залежності від типу системи час реакції складає від 60 до 120 секунд. Що насправді дуже багато. За такий довгий період пожежа може набрати таких маштабів, що важко буде її зупинити.

Системи на основі сенсорів диму у цьому тесті показують себе набагато краще. Знову ж таки в залежності від типу системи час реакції складає від 5 до 25 секунд. Мабуть, тому системи на основі сенсорів диму майже витіснили з ринку системи на основі теплових сенсорів.

Наша система показує наступний результат.

1. Час на розпізнавання полум’я на зображенні – 1-1.5 секунди
2. Час на розпізнавання звуку загорання – 0.5-1 секунда
3. Час на передачу даних – 1-2 секунди

Сумарно получається 2.5 - 4.5 секунди. Як бачимо результат роботи нашої системи у найгіршому випадку кращий за результат роботи системи на основі сенсорів диму у найкращому випадку.

Всі три системи досить легко збити з толку, якщо поставити перед собою таку ціль. Сенсор тепла можна нагріти, під сенсором диму можна запалити димову шашку, а наша система може зреагувати на будь-яке полум’я. Навіть якщо це полум’я запальнички. Але знову ж таки це стається кщо поставити перед собою таку мету. У звичайних умовах такі проблеми виникають рідко. Тому завадостійкість всіх систем оцінюємо як середню.

Вартість системи – це ще один пункт у якому наша система може дати фору іншим. Мінімальна вартість системи на основі теплових сенсорів складає около 10 тисяч гривень. У цю вартість входить тільки сенсори і система оповіщення.

Для систем на основі сенсорів диму вартість трохи більша і складає від 12 тисяч гривень.

Вартість нашої системи можна порухвати наступним чином:

1. Вартість Raspberry PI – 1300 гривень
2. Вартість модуля камери – 800 гривень
3. Вартість сенсора звуку – 100 гривень.

Для якісної роботи системи необхідно два мінікомп'ютера оснащених всіма модулями. І того варість нашої системи дорівнює (1300 + 800 + 100)*2 = 4400 гривень.

Як бачимо наша система виглядає досить конкурентноздатною. З порівняльною характеристика можна ознайомитись у таблиці 3.1.

Таблиця 3.1 – Порівняння пожежних систем.

	Система на основі сенсорів тепла	Система на основі сенсорів диму	Розроблена система
Точність	Висока	Висока	Висока
Час реагування, с	60-120	5-25	2.5-4.5
Завадостійкість	Середня	Середня	Середня
Варість, грн	>10 000	>12 000	4400

4.3 Рекомендації щодо подальшого вдосконалення системи

Наша система показує досить непогані результати, як можна побачити з попереднього підрозділу, але досконалості немає меж.

Удосконалення системи можна розбити на декілька пунктів.

1. Покращення якості підключених до мінікомп'ютера Raspberry PI модулів. Модулі які зараз підключені непогані, але модулі кращі.
2. Використання новішого мінікомп'ютера Raspberry PI. У цій роботі було використано Raspberry PI Model B+, яка є моделлю 2014 року. На даний момент випущена Raspberry PI 3 Model B, обчислювальні можливості якої більші в рази 3.
3. Інтеграція нашої системи з іншими розглянутими системами. Наприклад, інтреграція з системою на основі сенсорів диму. Це допоможе збільшити завадостійкість систем, оскільки зібльшиться кількість параметрів по яким опридіяється пожежа.
4. Підлючення до системи гасіння пожежі. На даний момент наша система тільки може виявити пожежу. У подальшому можна дозволити також цій системі включати систему гасіння пожежі.

ВИСНОВОК

Метою магістерської дисертації було дослідити існуючі методи визначення параметрів джерела акустико-оптичного випромінювання та оптимізація одного з методів

Проаналізувавши інструментарій та мови програмування, для розробки програмного продукту було обрано мову програмування Python. Як бібліотека комп'ютерного зору було обрано OpenCV.

В процесі розробки дипломного проекту було вивчено та порівняно декілька алгоритмів визначення параметрів джерела акустико-оптичного випромінювання. Для реалізації обрано метод який найкраще відповідає вимогам та поставленим задачам.

Результатом дипломного проекту став програмний продукт, який дозволяє обробляти кадри з відеокамери та шукати на них полум'я. Для визначення полум'я на кадрах був використаний методі Віоли-Джонса, який базується на використанням каскадного класифікатора Хаара. Для визначення параметрів джерела акустичного випромінювання використано метод спектрального аналізу.

Розроблене програмне забезпечення складається з семи модулів. Перші два модулі відповідають за роботу з камерою і сенсором звуку, два наступних за опрацювання зображень отриманих з камери і відповідно аудіо отриманого із сенсора звуку. П'ятий модуль це клієнт мережі на основі сокетів, яку ми використовували для передачі даних, а шостий відповідно сервер. І сьомий модуль це модуль у якому реалізовано логіка виведення кінцевих результатів на екран.

Всі очислення відбувались на мінікомп'ютері Raspberry PI Model B+, а результат роботи передавався на програмне забезпечення, яке виконується на ПК.

В ході тестування програмного заберезпечення було виявлено певні недоліки роботи методу визначення параметрів джерела

акустично-оптичного випромінювання. Серед недоліків можна виділити точніть роботи системи, яка дорівнювала 0.75. Після модифікації параметрів по яких відбувається пошук полум’я на зображенні, точність зросла до 0.98, але це повпливало на швидкість роботи алгоритму, яка зросла на 0.3 секунди. Але загальний час роботи системи залишився на допустимому рівні, а саме 1-1.5 секунди.

На основі розробленого програмного забезпечення для пошуку полум’я було реалізовано пожежна система. Розроблена пожежна система відрізняється від своїх аналогів тим що основним параметром виявлення пожежі є полум’я яке потрапило в об’єктив камери (у нашій роботі ми використали камеру з кутом огляду 160 градусів) і звук зарогання.

Було сладено порівняльну характеристику з існуючими аналогами. Порівняння відбувалось по таких критеріях: точність, час реакції, завадостійкість і вартість системи.

Для покращення роботи було запропоновано наступні рішення: оновлення апартної частини продукту (як окремих модулів так і самого мінікомп’ютера Raspberry PI) та інграція з існуючими рішеннями чи з системою гасіння пожежі.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Зимон А.Д., Лещенко Н.Ф. Коллоидная химия. — М., 2001; Шиц Л.А. Тиндаля эффект. В кн.: Физическая энциклопедия. — М., 1998. — Т. 5.

2. Солонина А.И.Алгоритмы и процессоры цифровой обработки сигналов / А.И. Солонина, Д.А. Улахович, Л.А. Яковлев // СПб.: БХВ-Петербург, 2001. – 484 с.

3. Выдрин Д.Ф Платформа Ардуино: преимущества / Д.Ф. Выдрин, А.О.Махнёва, А.Р. Мавлютов // Academy. 2017. № 1 (16). С. 9–12.

4. Щелбанин А. В. Алгоритмы преобразования Фурье и их применение при анализе звуковой информации / А. В. Щелбанин, Л. А. Зинченко // Молодой ученый. — 2016. — №20. — С. 29-34. [Електронний ресурс] — Режим доступу: https://moluch.ru/archive/124/34105/
5. Рабинер Л. Теория и применение цифровой обработки сигналов. / Л. Рабинер, Б. Гоулд // М: Мир, 1978.
6. Алфимцев А. Н. Каскадный детектор характерных признаков для распознавания пользователя в информационной системе / А. Н. Алфимцев // Вестник МГТУ им. Н. Э. Баумана. Сер. Приборостроение. - 2011. - Спец.вып. Технические средства. - С. 164-170.
7. Рахметов М.С. АНАЛИЗ АЛГОРИТМОВ РАСПОЗНАВАНИЯ ОБЪЕКТОВ HAAR CASCADE И LOCAL BINARY PATTERN / М. С. Рахметов // Научное сообщество студентов: МЕЖДИСЦИПЛИНАРНЫЕ ИССЛЕДОВАНИЯ: сб. Ст. По мат. XXXVI междунар. Студ. Науч.-практ. конф. № 1(36). [Електронний ресурс] — Режим доступу: https://sibac.info/archive/meghdis/1(36).pdf
8. Сакович И. О. Обзор основных методов контурного анализа для выделения контуров движущихся объектов. / И. О. Сакович, Ю. С. Белов // Инженерный журнал: наука и инновации, 2014, вып. 12. [Електронний ресурс] — Режим доступу: http://engjournal.ru/catalog/it/hidden/1280.html.

9. Зенин А. В. Анализ методов распознавания образов / А. В. Зенин // Молодой ученый. — 2017. — №16. — С. 125-130. — [Електронний ресурс] — Режим доступу: https://moluch.ru/archive/150/42393/
10. Белых Е. А. Обучение каскадов Хаара / Е. А. Белых // Вестник Сыктывкарского университета. Сер. 1: Математика. Механика. Информатика. 2017. Вып. 1 (22). С. 41-53. [Електронний ресурс] — Режим доступу: https://cyberleninka.ru/article/n/obuchenie-kaskadov-haara
11. Великий Я. О. Анализ принципа распознавания объектов на изображении методом Виолы–Джонса / Я. О. Великий // Открытые информационные и компьютерные интегрированные технологии. - 2015. - Вып. 68. - С. 162-166. [Електронний ресурс] — Режим доступу: http://nbuv.gov.ua/UJRN/vikt_2015_68_22
12. Юр Т. В. Аналіз методів розпізнавання облич на зображеннях / Т. В. Юр // Наукові праці Донецького національного технічного університету. Серія : Інформатика, кібернетика та обчислювальна техніка. - 2015. - Вип. 2. - С. 42-46. [Електронний ресурс] — Режим доступу: http://nbuv.gov.ua/UJRN/Npdntu_inf_2015_2_8