Вінницький національний технічний університет

(повне найменування вищого навчального закладу)

Факультет інформаційних технологій та комп'ютерної інженерії

(повне найменування інституту, назва факультету (відділення))

Кафедра програмного забезпечення

(повна назва кафедри (предметної, циклової комісії)

Пояснювальна записка

до магістерської кваліфікаційної роботи

магістр

(освітньо-кваліфікаційний рівень)

на тему Модифікація методів та розробка засобів для розпізнавання облич у реальному часі

Виконав:студент_2_курсу,групи1ПІ-18м

спеціальності

121 – Інженерія програмного забезпечення

(шифр і назва напряму підготовки, спеціальності)

Стеблюк Д.Є.

(прізвище та ініціали)

Керівник Коваленко О.О.

(прізвище та ініціали)

Вінниця − 2019 року

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Освітньо-кваліфікаційний рівень – магістр
Спеціальність 121 - «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О.Н.
“____”_____2019_р
оку

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ

СТУДЕНТУ

Стеблюку Дмитру Євгенійовичу

1. Тема роботи: «Модифікація методів та розробка засобів для розпізнавання облич у реальному часі»
керівник роботи: Коваленко Олена Олексіївна, к.т.н., доцент кафедри ПЗ, затверджені наказом вищого навчального закладу від “___”_____20__ року №___

2.Строк подання студентом роботи_____
3. Вихідні дані до роботи : Операційна система – Unix*
Мови програмування – Python
Технології – OpenCv

4. Зміст розрахунково-пояснювальної записки: вступ; огляд методів виявлення та розпізнавання облич; обробка кадрів відеопотоку; розробка системи розпізнавання облич у реальному часі; економічна частина; висновки; перелік посилань; додатки.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень): вступ; мета, об’єкт, предмет та задачі дослідження; огляд методів виявлення обличчя; розробка системи розпізнавання; загальна схема роботи додатку; презентація роботи додатку;

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-3	Коваленко О.О., к.т.н, доцент кафедри ПЗ		
4	Бальзан М.В., к.е.н., доц. каф. ЕПВМ		

7.Дата видачі завдання_____

8.КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз завдання і вибір методів його вирішення	03.09.19 – 27.09.19	Вик.
2	Аналіз існуючих аналогів та постановка задач дослідження	28.09.19 – 8.10.19	Вик.
3	Розробка методів розпізнавання	09.10.19 – 01.11.19	Вик.
4	Розробка структур і алгоритмів програмного продукту	02.11.19 – 10.11.19	Вик.
5	Розробка програмного забезпечення	11.11.19 – 21.11.19	Вик.
6	Економічна частина	26.11.19 – 30.11.19	Вик.
7	Оформлення матеріалів до захисту МКР	01.12.19 – 06.12.19	Вик.

Студент _____ **Стеблюк Д.Є.**
(підпис) (прізвище та ініціали)

Керівник магістерської кваліфікаційної роботи _____ **Коваленко О.О.**
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

У магістерській кваліфікаційній роботі «Модифікація методів та розробка засобів для розпізнавання облич у реальному часі» розроблено програмний додаток, який дозволяє розпізнавати обличчя у реальному часі. Дана програма може застосовуватися як додатковий пункт аутентифікації.

В ході виконання проаналізовано сучасні методи прогнозування та системи-аналоги. Обґрунтовано вибір засобів для розробки програмного забезпечення. До них належить середовище програмування PyCharm 2019, технології Python; фреймворки – OpenCv. Побудовано моделі та структуру системи, розроблено програмні модулі та протестовано коректну роботу програми.

ABSTRACT

Master's qualification work "Modification of methods and development of tools for face recognition in real time" made a modification of methods of face recognition and developed a software application that allows face recognition in real time. This program can be used as an additional authentication run.

In the course of implementation, modern forecasting methods and analog systems have been analyzed. The choice of software development tools is justified. These include PyCharm 2019 programming environment, Python technology; frameworks - OpenCv. Models and structure of the system were built, software modules were developed and the correct operation of the program was tested.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. Розпізнавання обличчя є надзвичайно важливим інструментом ведення бізнесу та взаємодії із споживачами, партнерами, працівниками. Розпізнавання обличь – служать не лише для розваг методом накладання веселих фільтрів. Не рідко його можна зустріти при екзаменації на електронних курсах, для того щоб за вас не здавав хтось інший, також при реєстрації на різні заходи, та й для захисту території [1].

Якщо говорити про сервіси, то сьогодні на ринку представлений широкий вибір послуг розпізнавання обличчя проте вони не забезпечуються розпізнавання облич в реальному часі, тому є сенс модифікувати існуючі методи розпізнавання облич, та основі їх розробити засоби для розпізнавання облич в реальному часі.

Серед відомих існуючих алгоритмів виявлення облич можна виділити декілька актуальних методів [2], серед яких:

Метод Віоли-Джонса, який демонструє високі результати при оброці зображень в реальному часі. Недоліки методу – необхідна велика навчальна вибірка і багато часу на навчання; обмеження на положення обличчя при знаходженні [3].

AdaBoost (Adaptive Boosting) – це алгоритм підсилення класифікаторів шляхом об’єднання їх в «комітет», який може використовуватись в поєднанні з декількома алгоритмами класифікації для покращення їх ефективності. Недоліки: алгоритм чутливий до шумів і викидам даних; потрібно багато часу на навчання, яке залежить від кількості класифікаторів і розміру навчальної вибірки.

SNoW (Sparse Network of Winnows) [5] – алгоритм виявлення облич, який представляє собою двошарову мережу, В якості ознак в даному алгоритмі використовуються SMQT (Successive Mean Quantization

Transform) ознаки, прапори рівності певним величинам середнього значення і дисперсії в кожному з прямокутних фрагментів зображення розміром 1x1, 2x2, 4x4 і 10x10 (усі зображення мають розмір 20x20 пікселів). Дані перетворення дозволяють вилучити з локальної області зображення складову, яка не залежить від освітленості. Воно полягає в квантуванні області зображення з порогом квантування рівним середньому значенню пікселів, які входять в цю область [6]. Також чутливий до шумів та викидів даних.

Нейромережеві методи включають в себе цілий клас алгоритмів. Їх недоліки: чутливість до шуму, необхідність в ретельному налаштуванні параметрів нейронної мережі для отримання хороших результатів.

Методи розпізнавання осіб можна розділити на дві підгрупи. Перша підгрупа – це методи, засновані на значеннях пікселів, і методи, засновані на характерних точках [7].

Короткий аналіз різноманітних методів для розпізнавання облич свідчить про те, що необхідно виконати більш детальний аналіз методів для виявлення найбільш оптимального та подальшої модифікації існуючих методів розпізнавання облич.

Зв'язок роботи з науковими програмами, планами, темами. Робота виконувалася згідно плану виконання наукових досліджень на кафедрі програмного забезпечення

Мета та завдання дослідження. Метою дослідження є модифікація існуючих методів розпізнавання облич для збільшення швидкодії розпізнавання облич в реальному часі.

Для досягнення поставленої мети необхідно розв'язати такі наступні завдання:

- розглянути існуючі методи та технології вирішення задач розпізнавання облич;
- модифікувати існуючі методи розпізнавання облич, для

забезпечення розпізнавання облич у реальному часі;

- розробити структуру та алгоритм роботи засобу для розпізнавання облич в реальному часі;
- розробити інформаційну технологію для забезпечення розпізнавання облич в реальному часі;
- виконати програмну реалізацію запропонованої інформаційної технології;
- провести тестування програмного продукту та виконати аналіз отриманих результатів.

Об'єкт дослідження – процес розпізнавання облич у реальному часі з використанням модифікованих методів розпізнавання облич та їх програмна реалізація.

Предмет дослідження – методи для розпізнавання облич в реальному часі.

Методи дослідження. У процесі дослідження застосовувались методи: системного аналізу структури інформаційної системи, розпізнавання облич; аналізу; алгоритмізації.

Наукова новизна отриманих результатів.

1. Отримав подальшого розвитку метод Віоли-Джонса, який відрізняється від відомого використанням локальних бінарних шаблонів, що дає підвищену швидкодію розпізнавання.

2. Запропоновано метод дослідження ефективності розпізнавання облич LBP перетворень: класичного, рівномірного та центрально-симетричного, яка на відміну від відомих дозволяє зробити кращий вибір методу для розпізнавання облич.

Практична цінність отриманих результатів. Практична цінність одержаних результатів полягає в тому, що на основі отриманих в магістерській кваліфікаційній роботі теоретичних положень розроблено програмний продукт розпізнавання облич у реальному часі для

комп’ютерних систем.

Особистий внесок здобувача. Усі наукові результати, викладені у магістерській кваліфікаційній роботі, отримані автором особисто. Зокрема, отримано подальшого розвитку метод Віоли-Джонса та Запропоновано методику дослідження ефективності розпізнавання облич LBP перетворень класичного, рівномірного та центрально-симетричного

Апробація матеріалів магістерської кваліфікаційної роботи. Основні положення магістерської кваліфікаційної роботи доповідалися та обговорювалися у розділі в колективній монографії «Інформаційні технології та автоматизація», а також тезах - «Розпізнавання виразу обличчя з використанням глибокого навчання»//Матеріали XLVII науково-технічної конференції підрозділів ВНТУ

Публікації. Основні результати досліджень опубліковано у 1 - статті в фаховому видавництві в Україні, 1 - у матеріалах науково технічної конференції.

Структура та обсяг роботи. Магістерська кваліфікаційна роботи складається зі вступу, чотирьох розділів, висновків, списку використаних джерел, що містить 36 найменувань, 3 додатки. Робота містить 36 ілюстрацій, 6 таблиць.

1 ОГЛЯД МЕТОДІВ ВИЯВЛЕННЯ ТА РОЗПІЗНАВАННЯ ОБЛИЧ

Задачу розпізнавання облич в кадрах відео потоку можна умовно розділити на два етапи. Перший етап – виявлення облич в кадрі. Другий етап – безпосередньо розпізнавання знайдених облич.

Розпізнавання облич – це віднесення вихідних даних до певного класу за допомогою виділення істотних ознак, що характеризують ці дані, із загальної маси несуттєвих даних [15].

Під чином будемо розуміти найменування області в просторі ознак, в якій відображається безліч об'єктів або явищ матеріального світу. Ознака - кількісний опис тієї чи іншої властивості досліджуваного предмета або явища [6].

Простір ознак - це N-мірний простір, певне для даної задачі розпізнавання, де N - фіксоване число вимірюваних ознак для будь-яких об'єктів. Вектор з простору ознак x, відповідний об'єкту завдання розпізнавання це N-мірний вектор з компонентами (x_1, x_2, ..., x_N), які є значеннями ознак для даного об'єкта.

Іншими словами, розпізнавання образів можна визначити, як віднесення вихідних даних до певного класу за допомогою виділення істотних ознак або властивостей, які характеризують ці дані, із загальної маси несуттєвих деталей [2].

Найчастіше вихідним матеріалом служить отримане з камери зображення. Завдання можна сформулювати як отримання векторів ознак для кожного класу на даному зображенні. Процес можна розглядати як процес кодування, що полягає в присвоєнні значення кожною ознакою з простору ознак для кожного класу [13].

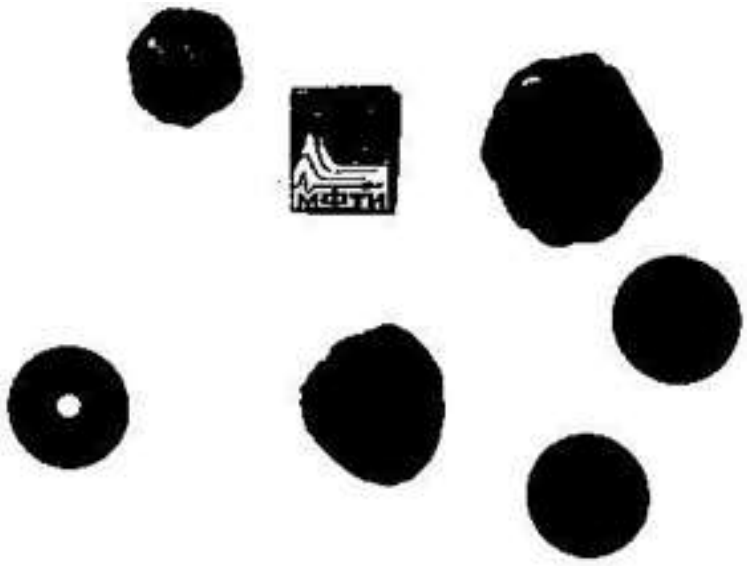
Нижче наведені деякі методи розпізнавання образів:

1.1. Фільтрація

Розглянемо методи, що дозволяють виділити на зображеннях цікавлять області, без їх аналізу. Велика частина цих методів застосовує якесь єдине перетворення на всі точки зображення.

Досить часто можна зустріти ідеальні завдання, в яких застосування такого перетворення виявиться досить. Припустимо, необхідно автоматично розпізнати предмети на білому аркуші паперу (Рисунок 1.1)

а)



б)

Рисунок 1.1 – Розпізнавання об'єктів, розташованих на чистому аркуші білого паперу: а) до розпізнавання б) після розпізнавання

У загальному випадку під фільтрацією зображення розуміють операцію, результатом якої буде зображення, яке отримали за заданими правилами з початкового зображення з таким же розміром. На рівні фільтрації аналіз зображення не проводиться, але точки, які проходять фільтрацію, можна розглядати як області з особливими характеристиками.

Оцінка якості фільтра проводиться по:

- Здібності фільтра видаляти перешкоди з зображення
- Здібності фільтра зберігати на зображенні все дрібні деталі і контури форм.
- Бінаризація по порогу, вибір області гістограми

Вибір порога, за яким відбувається бінаризація, як правило визначає процес бінаризації. В даному випадку, зображення було бінарізованими за середнім кольором. Зазвичай бінаризація здійснюється за допомогою алгоритму, який адаптивно вибирає поріг. Таким алгоритмом може бути вибір медіани. Або можемо орієнтуватися на максимальний пік гістограми. (рисунок 1.2)

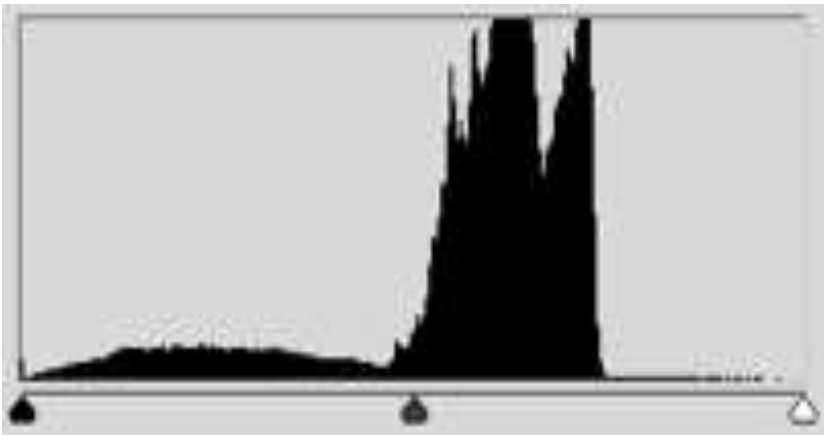


Рисунок 1.2 – Найбільший пік гістограми.

Можна наприклад, виділити цікавлять кольори. На цьому принципі можна побудувати як детектор мітки, так і детектор шкіри людини. (Рисунок 1.3)

а) б)

Рисунок 1.3 – Метод бинаризации по порогу застосований до фотографії людини: а) до розпізнавання б) після розпізнавання

Бінаризація дає дуже цікаві результати якщо працювати з гістограмами, особливо в ситуації, коли ми розпізнаємо НЕ КЗС зображення, а в HSB.

1.2 Частотна фільтрація: Фур'є-аналіз, Фільтр нижніх частот (ФНЧ), Фільтр верхніх частот (ФВЧ)

Такі методи фільтрації і обробки сигналів прекрасно застосовуються в безлічі завдань теорії розпізнавання образів. Для радіолокації класичним методом буде перетворення Фур'є (конкретніше – швидке перетворення Фур'є. В цілому метод частотної фільтрації ґрунтується на модифікації сигналу шляхом застосування до зображення двовимірного перетворення Фур'є, але про це далі. Суть перетворення Фур'є в тому, що воно дозволяє представити практично будь-який набір даних або ж функцію у вигляді комбінації тригонометричних функцій – синуса і косинуса, а це, в свою чергу, дозволяє виявити періодичні компоненти в даних і далі оцінити їх внесок в структуру вихідних форми функції або даних. Як правило існує три основні форми перетворення Фур'є: інтегральне перетворення Фур'є, ряди Фур'є і дискретне перетворення Фур’є. Інтегральне перетворення Фур'є переводить комплексну функцію в іншу або речову функцію в пару

дійсних функцій. Основним принципом частотної фільтрації буде фон і великорозмірні об'єкти, які відповідають низьким частотам. Але існує одне з небагатьох виняток, при яких використовується одновимірний перетворення Фур'є, - компресія зображень. Зазвичай для аналізу зображень одновимірного перетворення не вистачає, і в такому випадку необхідно використовувати більш ресурсне двовимірне перетворення. Формула дискретного перетворення Фур'е для двумірного масива чисел:

$$G_{uw} = \frac{1}{NM} \sum_{n=1}^{N-1} \sum_{m=1}^{M-1} x_{mn} e^{-2\pi j \left[\frac{mu}{M} + \frac{nu}{N}\right]}$$

Але в дійсності його мало хто розраховує, як правило, куди простіше і швидше використовувати згортку цікавить області з уже готовим фільтром, заточеним на високі (ФВЧ) або низькі (ФНЧ) частоти. Методи цей, звичайно, не дозволить зробити аналіз спектра, але в конкретному завданні відеоаналітики зазвичай потрібен не стільки сам аналіз, як його результат.

Згортка функцій – це операція, що застосовується в функціональному аналізі. За визначенням, згортка – це математична операція, що використовується до двох функцій f і g, що породжує третю функцію, яка іноді може розглядатися як модифікована версія однієї з початкових. По суті, цю операцію можна розглядати як особливий вид інтегрального перетворення.

Найпростішими прикладами, що реалізують вибір низьких частот, є фільтр Гауса, а для високих частот - Фільтр Габора.

Фільтр Гауса – електронний фільтр, чиєю імпульсної перехідної функцією є функція Гауса. Фільтр Гауса він спроектований таким чином, щоб не мати перерегулювання у перехідній функції та максимізувати

постійну часу. Така поведінка тісно пов'язане з тим, що фільтр Гауса має мінімально можливу групову затримку.

При реалізації ФНЧ і ФВЧ до отримання зображень, відображені на рисунку 1.4:

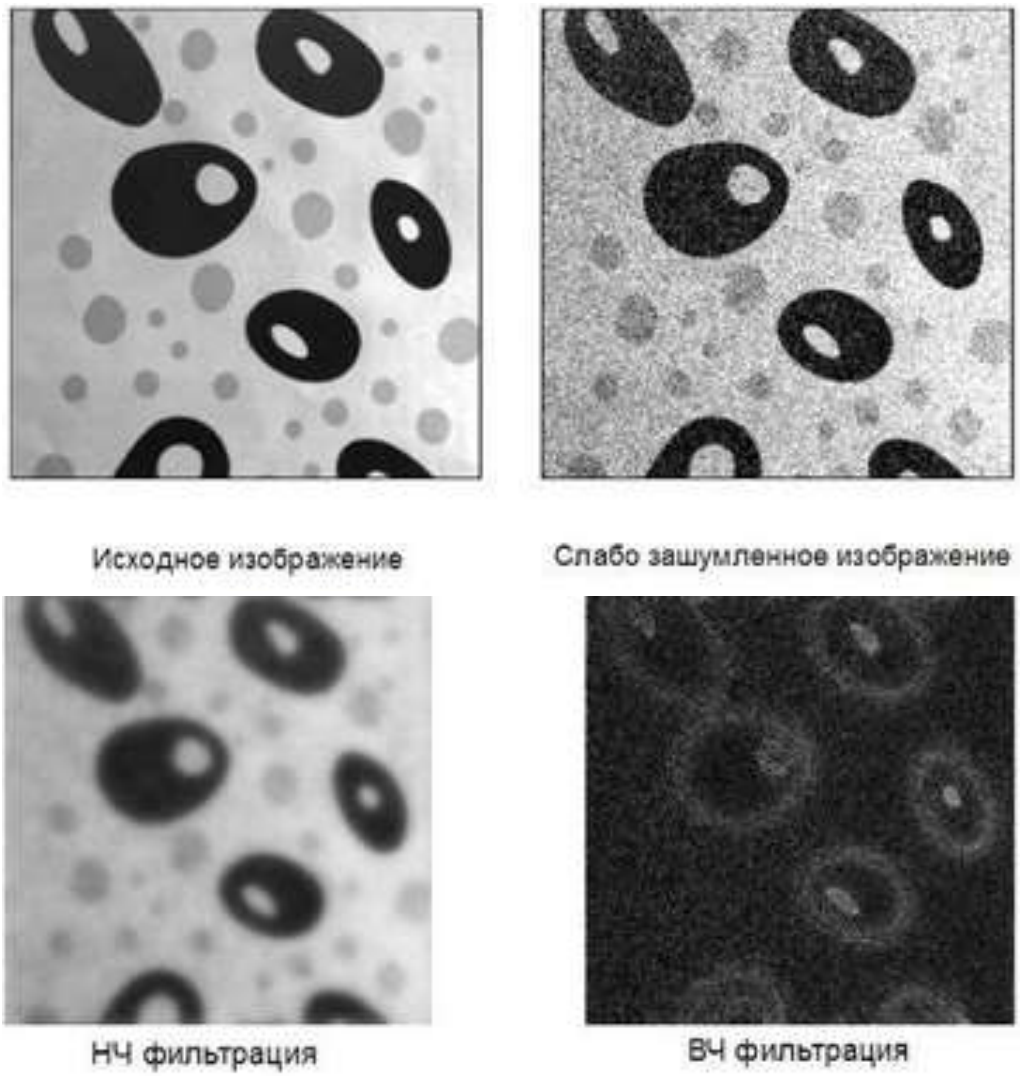


Рисунок 1.4 - Зображення, що виходять при реалізації методів ФНІ і ФВЧ.

Фільтр Гауса як правило використовується тільки в цифровому вигляді для обробки двовимірних зображень з метою зниження рівня перешкод. Однак при зміні частоти дискретного сигналу він дає сильне розмиття зображення.

Фільтр Габора – лінійний електронний фільтр, імпульсна перехідна характеристика якого визначається у вигляді гармонійної функції. При

цифровій обробці зображень цей фільтр застосовується для розпізнавання меж об'єктів.

Через властивості відповідності згортки в частотної області множенню у часовій області, перетворення Фур'є імпульсної передавальної характеристики фільтра Габора є згортка перетворень Фур'є гармонійної функції і Гауса.

Для кожної точки зображення вибирається вікно і перемножується з фільтром того ж розміру. Результатом такої згортки є нове значення точки.

1.3 Фільтри, засновані на порядкових статистиках

Такі фільтри як правило представляють собою просторові фільтри, для обчислення результату яких потрібне попереднє упорядкування значення пікселів, яке знаходяться всередині області обробки.

Одними з найпопулярніших фільтрів є:

- медіанний фільтр;
- фільтр, заснований на обчисленні максимумів і мінімумів;
- фільтр середньої точки;
- фільтр усередненого середнього.

Розповісти про дані фільтрах найпростіше на прикладі медіанного, так як він є досить популярним в практичній діяльності.

Суть даного фільтра полягає в заміні значення в точці зображення на медіану значень яскравості у вікні фільтрації цієї точки в розмірі вікна, m * n.

$$j\,x, y = mbd_{x-0,5\,m-1\,<k<x+0,5\,m-1\,_1}\,f\,k, i\ .$$

Решта фільтри, в основу яких входять порядкові статистики, працюють за подібним принципом, але замість медіани беруть інші статистичні особливості елементів фільтрації.

Наприклад, фільтри, засновані на обчисленні максимуму або мінімуму, про які розповідалося вище, визначають, як зрозуміло з назви, максимальне або мінімальне значення яскравості елемента вікна фільтрації, після чого привласнюють це значення центральному елементу, а це, в свою чергу дозволяє позбавиться від чорних – для максимуму і білих – для мінімуму перешкод, що мають максимальну або мінімальну яскравість.

1.4 Вейвлет перетворення

Вейвлет-перетворення (англ. Wavelet transform) – інтегральне перетворення, яке представляє собою згортку вейвлет-функції з сигналом. Вейвлет-перетворення переводить сигнал з тимчасового уявлення в частотно-тимчасове.

4 приклади класичних вейвлетов показані на рисунку 1.6.

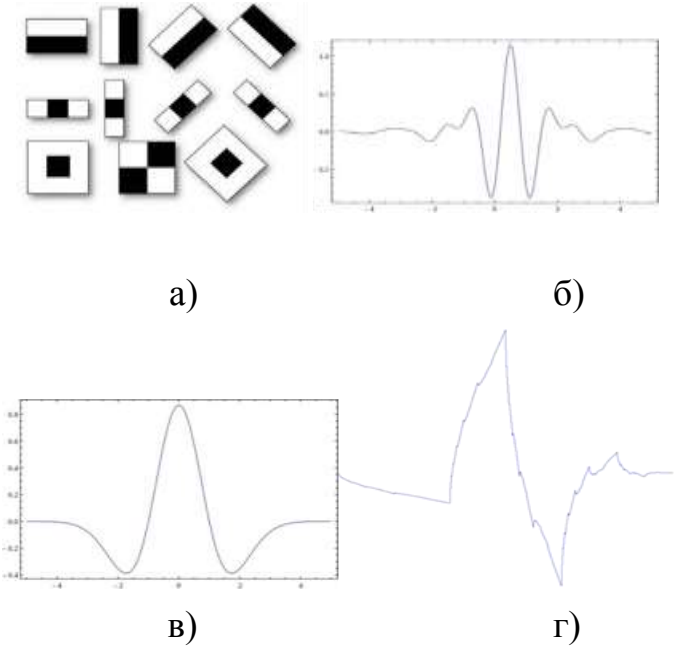


Рисунок 1.6 –а) Х-мерный вейвлет Хаара, б) 2х-мерные вейвлет Мейера, в) вейвлет Мексиканская Шляпа, г) вейвлет Добеши.

Даний метод заснований на перетворенні функції або сигналу в форму, деякі вилікувані вихідні сигнали якого, стають більш піддаються

вивченню, або ж дозволяє стиснути вихідний набір даних. Вейвлетного перетворення сигналів є узагальненням спектрального аналізу. Термін (англ. Wavelet) в перекладі з англійської означає

«Маленька хвиля». Вейвлети – це сімейство функцій, які локальні в часі і по частоті («маленькі»), і в яких всі функції виходять з однієї з її допомогою зрушень і розтягувань по осі часу (так що вони «йдуть один за одним»).

Хорошим прикладом використання розширеної трактування вейвлетов є завдання пошуку відблиску в оці, для якої вейвлетом є сам відблиск, показаний на рисунку 1.7.

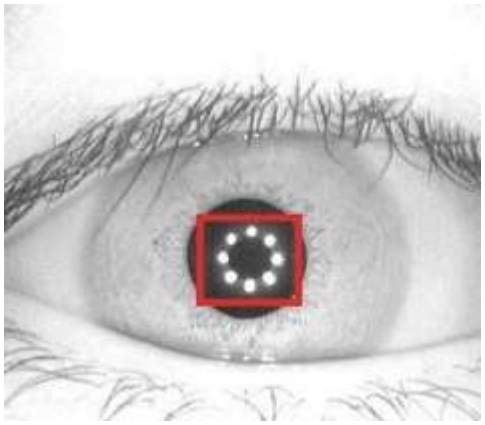


Рисунок 1.7 – Задача пошуку бліку в оці з допомогою вейвлет-аналізу.

Дане визначення вейлветов хоч і не є коректним, але традиційно склалося, що під велйвет-аналізом розуміють саме пошук довільного шаблонна або зразка на зображенні за допомогою згортки з моделлю цього зразка. Існує кілька класичних функцій, використовуваних в вейвлет-аналізі. До них відносяться вейвлет Хаара, вейвлет Морлі, вейвлет мексиканський капелюх тощо.

1.5 Кореляція цифрових зображень

Для фільтрації зображень незамінний інструмент – це кореляція цифрових зображень. Кореляція цифрових зображень (англ. Digital Image Correlation and Tracking, (DIC / DDIT)) – оптичний метод, який використовується в техніках відстеження та ідентифікації зображення для точних плоских і об'ємних вимірів змін на зображенні. Цей метод часто використовується не тільки для вимірювання деформацій, полів переміщень і оптичних потоків, а й широко використовується в багатьох областях науки і інженерного ремесла. Одне з найбільш широко відомих застосувань даного методу – ідентифікація переміщень оптичної мишки. (Рисунок 1.8)

Рисунок 1.8 - Приклад кореляції цифрового зображення

Класичне застосування – кореляція відеопотоку для знаходження зрушень або оптичних потоків. Найпростіший детектор зсуву – теж в якомусь сенсі різницевий корелятор.

1.6 Фільтрації функцій

Одним з найцікавіших класів фільтрів є фільтрація функцій. Суть в тому, що ці фільтри є математичними, і дозволяють виявити найпростішу математичну функцію на області розпізнавання, таку як, наприклад, пряма,

парабола або коло. Принцип цього фільтра полягає в побудові акумулює зображення, в якому для кожної точки вихідного зображення промальовується безліч функцій, які її утворили. Найбільш класичним перетворенням є перетворення Хафа для прямих. Перетворення Хафа (Hough Transform) – алгоритм, чисельний метод, застосовуваний для вилучення елементів з зображення (патент 1962 р Поля Хафа). Використовується в аналізі зображень, цифровій обробці зображень і комп'ютерному зорі. Призначений для пошуку об'єктів, що належать класу фігур, з використанням процедури голосування. Процедура голосування застосовується до простору параметрів, з якого і виходять об'єкти класу фігур по локального максимуму в так званому накопичувальному просторі (accumulator space), яке будується при обчисленні трансформації Хафа. Класичний алгоритм перетворення Хафа пов'язаний з ідентифікацією прямих в зображенні[9].

У цьому перетворенні для кожної точки (x; y) промальовується безліч точок (a; b) прямий y = ax + b, для яких вірно рівність. (Рисунок 1.9)

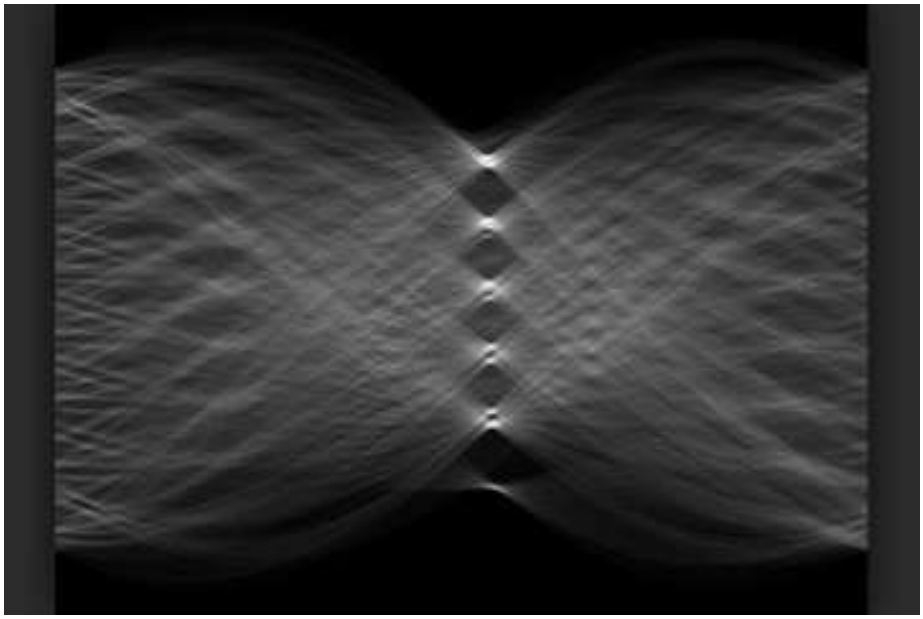


Рисунок 1.9 - Перетворення Хафа.

Перетворення Хафа дозволяє знаходити будь-які функції, засновані на параметрах. Наприклад, окружності. Так само існує і модифіковане перетворення, що дозволяє шукати абсолютно будь-які фігури. Дане перетворення дуже полюбилося математикам. Однак при обробці зображень, воно, на жаль, працює далеко не завжди, в зв'язку з властивою йому вкрай повільною швидкістю роботи і дуже високою чутливістю до якості бінерізації. Навіть в ідеальних ситуаціях переважно використовувати іншими методами.

Аналогом перетворення Хафа для прямих є перетворення Радону. Перетворення Радону розраховується по швидкому перетворенню Фур'є, що дозволяє отримати виграш в продуктивності при ситуації, коли точок дуже багато. До того ж його можна використовувати до небінаризованих зображень.

Перетворення Радону - інтегральне перетворення функції багатьох змінних, родинне перетворенню Фур'є. Вперше введено в роботі австрійського математика Йоганна Радону 1917-го року. Найважливіша властивість перетворення Радону – оборотність, тобто можливість відновлювати вихідну функцію по еѐ перетворенню Радону.

1.7 Фільтрація контурів

Окремий клас фільтрів - це фільтрація кордонів і контуров.Фільтрація контурів можемо бути дуже корисна, еслінужно перейти від роботи з самим зображенням до роботи з об'єктами на цьому зображенні. У випадках, коли об'єкт складний, але добре виділяється, дуже частолучшім методом роботи з ним є саме виділення його контурів. Існує цілий ряд алгоритмів, які вирішують задачу фільтрації контурів:

оператор Кенні;

оператор Собеля;

оператор Лапласа;

оператор Прюітт;

оператор Робертса.

Вище наведені фільтри, які можу вирішити більшу частину завдань по розпізнаванню кордонів. Однак крім них існує безліч більш рідкісних фільтрів, які використовують в більш специфічних задачахпоетомув описі кожного з них просто немає сенсу. Однак найчастіше застосовують саме оператор Кенні, який добре працює, і реалізація якого є в OpenCV тому про нього розповісти варто.

Оператор Кенні (детектор кордонів Кенні, алгоритм Кенні) в дисципліні комп'ютерного зору - оператор виявлення кордонів зображення. Був розроблений в 1986 році Джоном Кенні (англ. John F. Canny) і використовує багатоступінчастий алгоритм для виявлення широкого спектра кордонів в зображеннях.

Кенні вивчив математичну проблему отримання фільтра, оптимального за критеріями виділення, локалізації та мінімізації декількох відгуків одного краю. Він показав, що шуканий фільтр є сумою четирèх експонент. Він також показав, що цей фільтр може бути добре наближений першої похідної Гауссіана. Кенні ввèл поняття придушення НЕ максимумів (англ. Non-Maximum Suppression), яке означає, що пікселями кордонів оголошуються пікселі, в яких досягається локальний максимум градієнта в напрямку вектора градієнта.

Алгоритм складається з п'яти окремих кроків:

Згладжування. Розумієте зображення для видалення шуму.

Пошук градієнтів. Межі відзначаються там, де градієнт зображення набуває максимальне значення.

Придушення НЕ-максимумів. Тільки локальні максимуми відзначаються як кордони.

Подвійна порогова фільтрація. Потенційні межі визначаються порогами.

Трасування області неоднозначності. Підсумкові межі визначаються путѐм придушення всіх краѐв, незв'язаних з певними (сильними) межами.

Цікавими є ітераційні фільтри (наприклад, активна модель зовнішнього вигляду), а так само ріджлет і курвлет перетворення, є сплавом класичної вейвлет фільтрації і аналізом в поле радон-перетворення. Бімлет-перетворення красиво працює на кордоні вейвлет перетворення і логічного аналізу, дозволяючи виділити контури. Але ці перетворення дуже специфічні і заточені під рідкісні завдання.

1.8 Логічна обробка результатів фільтрації

Фільтрація дає набір придатних для обробки даних. Але найчастіше можна просто взяти і використовувати ці дані без їх обробки. У цьому розділі буде кілька класичних методів, що дозволяють перейти від зображення до властивостей об'єктів, або до самих об'єктів.

У бінарної морфології двоичное зображення, представлене у вигляді упорядкованого набору (впорядкованої множини) чѐрно-білих точок (пікселів), або 0 і 1. Під областю зображення зазвичай розуміється деяка підмножина точок зображення. Кожна операція двійковій морфології є деяким перетворенням цього безлічі. В якості вихідних даних приймаються двоичное зображення B і деякий структурний елемент S. Результатом операції також є двійкове зображення. Класичний опис операцій бінарної математичної морфології було дано в термінах теорії множин, що оперує такими поняттями, як об'єднання множин, перетин множин і ставлення включення. При цьому бінарні зображення розглядаються безпосередньо як безлічі пікселів, тому відповідні теоретико-множинні операції мають очевидну наочну інтерпретацію в дусі "кіл Ейлера" (Рисунок 1.9).

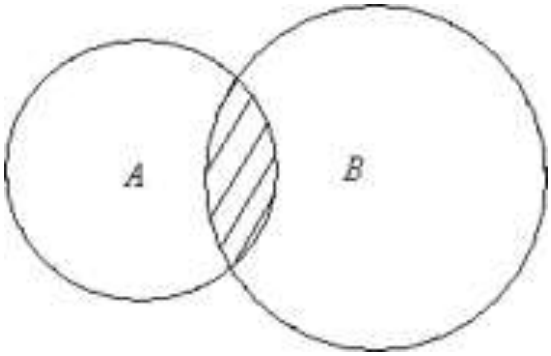
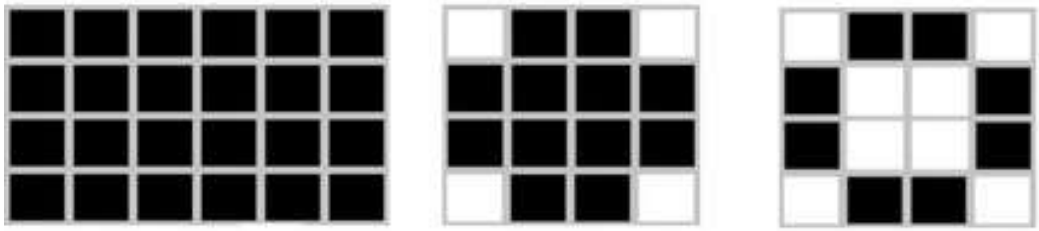


Рисунок 1.9 – Загальне уявлення про колах (діаграмі) Ейлера.

В першу чергу математична морфологія використовується для вилучення деяких властивостей зображення, корисних для його уявлення і опису. Наприклад, контурів, кістяків, опуклих оболонок. Також інтерес представляють морфологічні методи, що застосовуються на етапах попередньої та підсумкової обробки зображень. Наприклад, морфологічна фільтрація, потовщення або зменшення. Вхідними даними для апарату математичної морфології є два зображення: обробляється і спеціальне, залежне від виду операції та розв'язуваної задачі. Таке спеціальне зображення прийнято називати примітивом чи структурним елементом. Як правило, структурний елемент багато менше оброблюваного зображення. Структурний елемент можна вважати опис області з деякою формою. Зрозуміло, що форма може бути будь-який, головне, щоб еè можна було уявити у вигляді бінарного зображення заданого розміру. У багатьох пакетах обробки зображень найбільш поширені структурні елементи мають спеціальні назви: BOX [H, W] - прямокутник заданого розміру, DISK [R] – диск заданого розміру, RING [R] - кільце заданого розміру. (Рисунок 1.10)



а) б) в)

Рисунок 1.10 – Структурні елементи а) BOX б) DISK в) RING

Результат морфологічної обробки залежить як від розміру і конфігурації вихідного зображення, так і від структурного примітиву.

Розмір структурного елементу як правило дорівнює 3 * 3, 4 * 4 або 5 * 5 пікселів. Це обумовлено головною ідеєю морфологічної обробки, в процесі якої відшукуються характерні деталі зображення. Шукана деталь описується примітивом, і в результаті морфологічної обробки можна підкреслити або видалити такі деталі на всьому зображенні.

Одне з основних переваг морфологічної обробки – її простота: як на вході, так і на виході процедури обробки ми отримуємо бінаризованими зображення. Інші методи, як правило, з вихідного зображення спочатку отримують півтонове, яке потім приводиться до бінарним за допомогою порогової функції.

Контур є унікальною характеристикою об'єкта. Часто це дозволяє ідентифікувати об'єкт по контуру. Існує потужний математичний апарат, що дозволяє це зробити. Апарат називається контурним аналізом

Особливі точки - це унікальні характеристики об'єкта, які дозволяють зіставляти об'єкт сам з собою або зі схожими класами об'єктів. Існує кілька десятків способів, що дозволяють виділити такі точки. Деякі способи виділяють особливі точки в сусідніх кадрах, деякі через великий проміжок часу і при зміні освітлення, деякі дозволяють знайти особливі точки, які залишаються такими навіть при поворотах об'єкта. Почнемо з методів, що

дозволяють знайти особливі точки, що не такі стабільні, зате швидко розраховуються, а потім продовжимо по зростанню складності:

Перший клас. Особливі точки, які є стабільними протягом секунд. Такі точки служать для того, щоб вести об'єкт між сусідніми кадрами відео, або для відомості зображення з сусідніх камер. До таких точок можна віднести локальні максимуми зображення, кути на зображенні (кращий з детекторів, мабуть, детектор Харіса), точки в яких досягається максимуми дисперсії, определѐнние градієнти і.т.д.

Другий клас. Особливі точки, які є стабільними при зміні освітлення і невеликих рухах об'єкта. Такі точки служать в першу чергу для навчання і подальшої класифікації типів об'єктів. Наприклад, класифікатор пішохода або класифікатор особи - це продукт системи, побудованої саме на таких точках. Деякі з раніше згаданих вейвлетов можуть є базою для таких точок. Наприклад, примітиви Хаара, пошук відблисків, пошук інших специфічних функцій. До таких точок відносяться точки, знайдені методом гістограм спрямованих градієнтів (HOG).

Третя частина присвячена методам, які не працюють безпосередньо з зображенням, але які дозволяють приймати рішення. В основному це різні методи машинного навчання і прийняття рішень.

У 80% ситуацій суть навчання в задачі розпізнавання в наступному:

Є тестова вибірка, на якій є кілька класів об'єктів. Нехай це буде наявність / відсутність людини на фотографії. Для кожного зображення є набір ознак, які були виділені яким-небудь ознакою, будь то Хара, HOG, SURF або який-небудь вейвлет. Алгоритм навчання повинен побудувати таку модель, за якою він зможе проаналізувати нове зображення і прийняти рішення, який з об'єктів є на зображенні.

Кожне з тестових зображень - це точка в просторі ознак. Еѐ координати це вага кожного з ознак на зображенні. Нехай нашими ознаками

будуть: «Наявність очей», «Наявність носа», «Наявність двох рук», «Наявність вух»,...

Всі ці ознаки ми виділимо існуючими у нас детекторами, які навчені на частини тіла, схожі на людські. Для людини в такому просторі буде коректною точка [1; 1; 1; 1; ..]. Для мавпи точка [1; 0; 1; 0 ...] для коня [1; 0; 0; 0 ...]. Класифікатор навчається за вибіркою прикладів. Але не на всіх фотографіях виділилися руки, на інших немає очей, а на третій у мавпи через помилку класифікатора з'явився людський ніс. Той, якого навчають класифікатор людини автоматично розбиває простір ознак таким чином, щоб сказати: якщо перша ознака лежить в діапазоні 0.5 <x <1, другий 0.7 <y <1, і.т.д., тоді це людина.

По суті, мета класифікатора - промальовувати в просторі ознак області, характеристичні для об'єктів класифікації. Ось так буде виглядати послідовне наближення до відповіді для одного з класифікаторів (AdaBoost) в двовимірному просторі (Рисунок 1.11) [14].

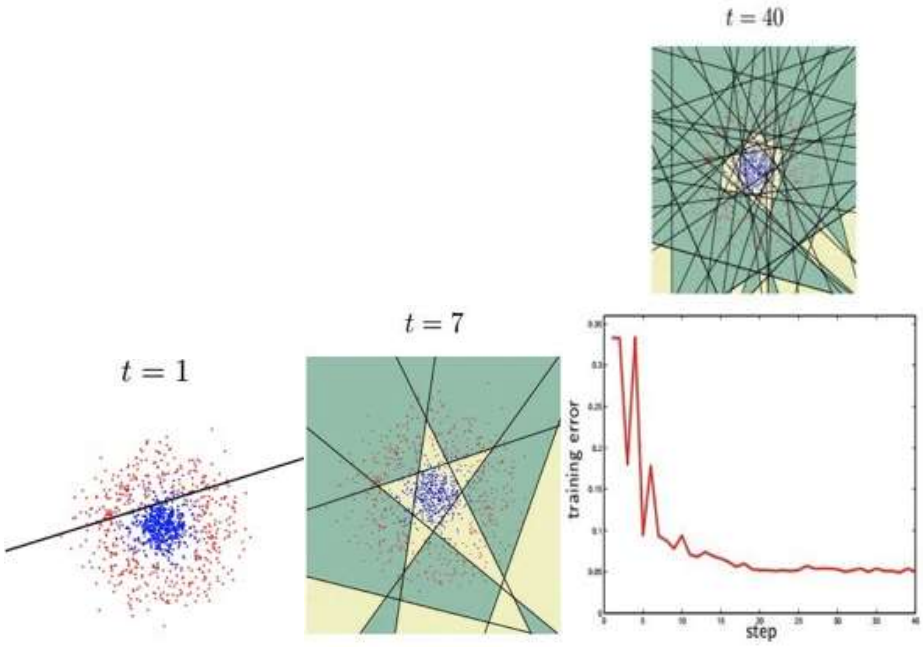


Рисунок 1. 11 – Послідовне наближення до відповіді класифікатора AdaBoost

1.9 Висновок

У даному розділі було проведено дослідження стану предметної області та виконано аналіз методів виявлення та розпізнавання облич. Зважаючи на розвиток технологій та зростання кількості різних комунікаційних та розважальних ресурсів було прийнято рішення модифікувати методи та розробити нові засоби для розпізнавання облич у реальному часі.

2. ОБРОБКА КАДРІВ ВІДЕОПОТОКУ У РЕАЛЬНОМУ ЧАСІ

У розділі будуть детально описані етапи обробки кадрів відео потоку у реальному часі, а також алгоритми та методи, які будуть використані.

2.1. Ідентифікація облич методом Віоли-Джонса

Основні принципи, на яких базується метод, такі:

- використовуються зображення в інтегральному уявленні, що дозволяє обчислювати швидко необхідні об'єкти;
- використовуються ознаки Хаара, за допомогою яких відбувається пошук потрібного об'єкта (в даному контексті, особи і його рис);
- використовується бустінг (від англ. boost - поліпшення, посилення) для вибору найбільш підходящих ознак для шуканого об'єкта на даній частині зображення;
- всі ознаки надходять на вхід класифікатора, який дає результат «вірно» або «брехня»;
- використовуються каскади ознак для швидкого відкидання вікон, де не знайдено особа.

Навчання класифікаторів йде дуже повільно, але результати пошуку особи дуже швидкі, саме тому був обраний даний метод розпізнавання осіб на зображенні. Віола-Джонс є одним з кращих по співвідношенню показників ефективність розпізнавання / швидкість роботи. Також цей детектор має вкрай низькою ймовірністю помилкового виявлення особи. Алгоритм навіть добре працює і розпізнає риси обличчя під невеликим кутом, приблизно до 30 градусів. При куті нахилу більше 30 градусів відсоток виявлень різко падає. І це не дозволяє в стандартній реалізації детектувати повернене обличчя людини під довільним кутом, що в значній

мірі ускладнює або робить неможливим використання алгоритму в сучасних виробничих системах з урахуванням їх зростаючих потреб [3].

Потрібен докладний розбір принципів, на яких базується алгоритм Віоли-Джонса. Даний метод в загальному вигляді шукає особи і риси обличчя за загальним принципом скануючого вікна.

Принцип скануючого вікна:

У загальному вигляді, завдання виявлення особи та рис обличчя людини на цифровому зображенні виглядає саме так:

- є зображення, на якому є шукані об'єкти. Воно представлено двовимірної матрицею пікселів розміром w * h, в якій кожен піксель має значення:

  - від 0 до 255, якщо це чорно-біле зображення;

  - від 0 до 2553, якщо це кольорове зображення (компоненти R, G, B).

- в результаті своєї роботи, алгоритм повинен визначити особи і їх риси і позначити їх - пошук здійснюється в активній області зображення прямокутними ознаками, за допомогою яких і описується знайдене особа і його риси:

  rectanglei = {x, y, w, h, a}, (2.1)

  де x, y - координати центру i-го прямокутника, w - ширина, h - висота, a - кут нахилу прямокутника до вертикальної осі зображення.

Іншими словами, стосовно до малюнків і фотографій використовується підхід на основі скануючого вікна (scanning window): сканується зображення вікном пошуку (так зване, вікно сканування), а потім застосовується класифікатор до кожного положення. Система навчання і вибору найбільш значущих ознак повністю автоматизована і не вимагає втручання людини, а тому цей підхід працює швидко.

Завдання пошуку і знаходження осіб на зображенні за допомогою даного принципу часто буває черговим кроком на шляху до розпізнавання

характерних рис, наприклад, верифікації людини по розпізнаного особі або розпізнавання міміки обличчя.

Інтегральне представлення зображень:

Для того, щоб проводити будь-які дії з даними, використовується інтегральне представлення зображень [3] в методі Віоли-Джонса. Таке уявлення використовується часто і в інших методах, наприклад, в вейвлет-перетвореннях, SURF і багатьох інших розібраних алгоритмах. Інтегральне уявлення дозволяє швидко розраховувати сумарну яскравість довільного прямокутника на даному зображенні, причому який би прямокутник ні, час розрахунку незмінно.

Інтегральне представлення зображення - це матриця, що збігається за розмірами з вихідним зображенням. У кожному елементі її зберігається сума інтенсивностей всіх пікселів, що знаходяться лівіше і вище даного елемента. Елементи матриці розраховуються за такою формулою(2.2):

$$L(x,y) = \sum_{i=0,j=0}^{i \le x, j \le y} * I(i,j), \quad (2.2)$$

де I (i, j) - яскравість пікселя вихідного зображення.

Кожен елемент матриці L [x, y] являє собою суму пікселів в прямокутнику від (0,0) до (x, y), тобто значення кожного пікселя (x, y) дорівнює сумі значень усіх пікселів лівіше і вище даного пікселя (x, y). Розрахунок матриці займає лінійний час, пропорційне числу пікселів в зображенні, тому інтегральне зображення прораховується за один прохід.

Розрахунок матриці можливий за формулою 2.3:

L (x, y) = I (x, y) - L (x-1, y-1) + L (x, y-1) + L (x-1, y) (2.3)

За такою інтегральною матриці можна дуже швидко вирахувати суму пікселів довільного прямокутника, довільної площі.

Нехай в прямокутнику ABCD є цікавий для нас об'єкт D(рис 2.1):

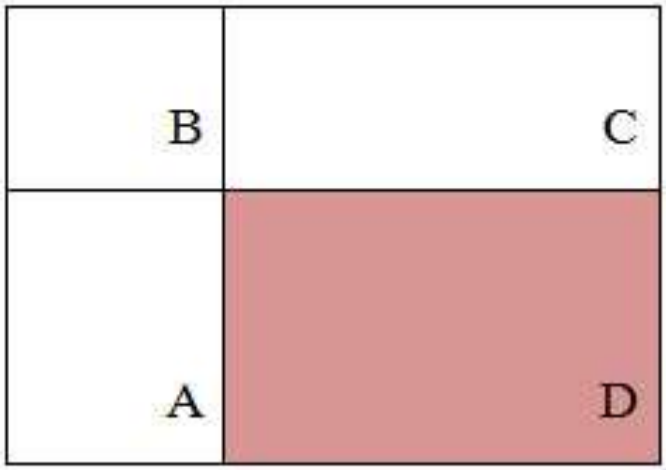


Рисунок 2.1 - Прямокутник обєктів

З рисунка зрозуміло, що суму всередині прямокутника можна виразити через суми і різниці суміжних прямокутників за такою формулою(2.4):

S (ABCD) = L (A) + L (C) - L (B) - L (D) (2.4)

Приблизний розрахунок показаний на рисунку 2.2.

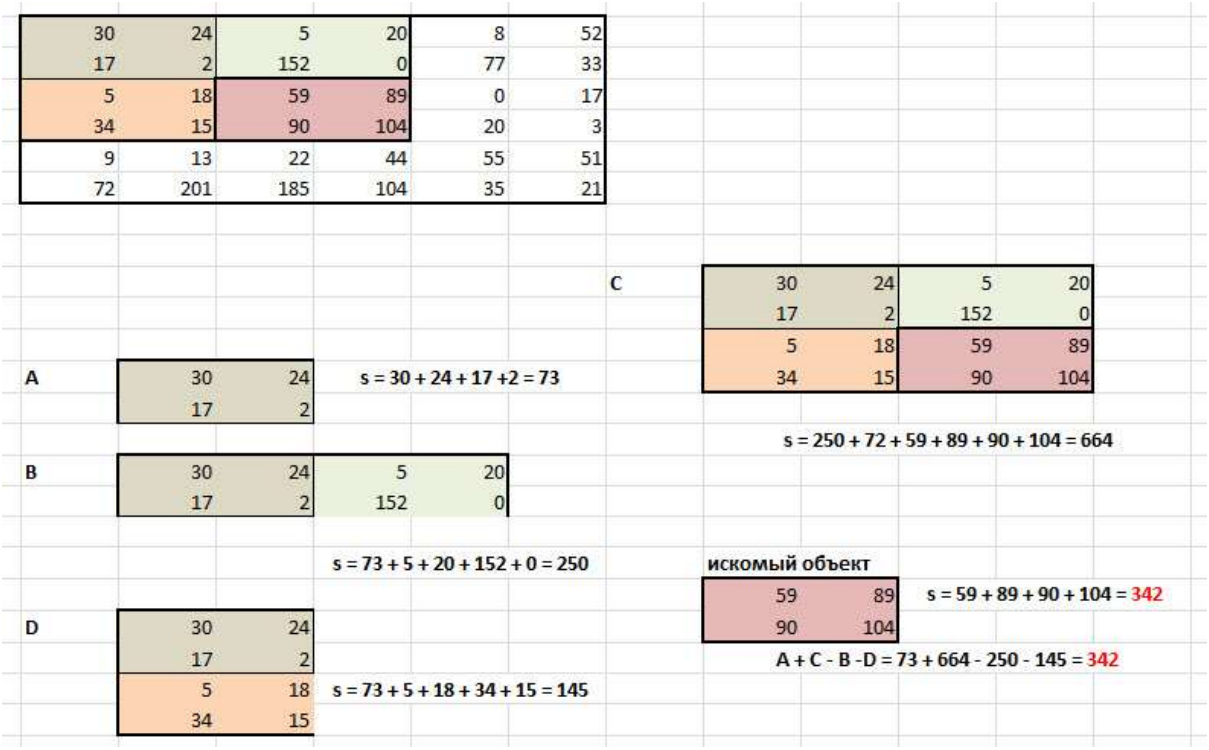


Рисунок 2.2 - розрахунок суми всередині прямокутника через суми і різниці суміжних прямокутників

2.2 Використання ознак Хаара

Ознака - відображення f: X => Df, де Df - безліч допустимих значень ознаки. Якщо задані ознаки f1, ..., fn, то вектор ознак x = (f1 (x), ..., fn (x)) називається признаковая описом об'єкта x ∈ X. просторі ознак опису допустимо ототожнювати з самими об'єктами. При цьому безліч X = Df1 * ... * Dfn називають просторі ознак [1].

Ознаки поділяються на такі типи в залежності від безлічі Df:

- бінарний ознака, Df = {0,1};
- номінальний ознака: Df - кінцеве безліч;
- порядковий ознака: Df - кінцеве впорядкована множина;
- кількісний ознака: Df - безліч дійсних чисел.

Природно, бувають прикладні завдання з різнотипними ознаками, для їх вирішення підходять далеко не всі методи. У стандартному методі Віоли - Джонса використовуються прямокутні ознаки, зображені на малюнку нижче, вони називаються примітивами Хаара (рис 2.3).

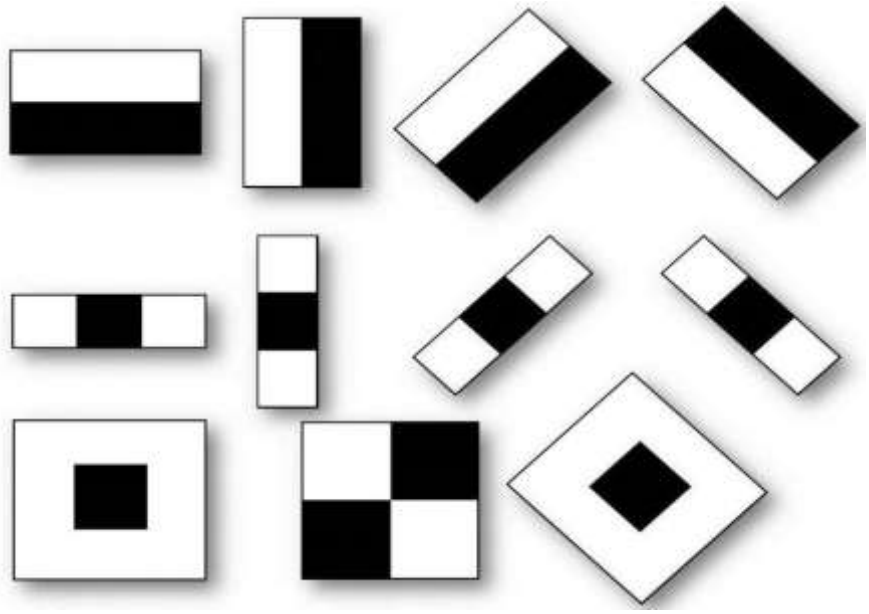


Рисунок 2.3 - прямокутні ознаки Хаара

У розширеному методі Віоли-Джонса, що використовується в бібліотеці OpenCV використовуються додаткові ознаки(рис 2.4).

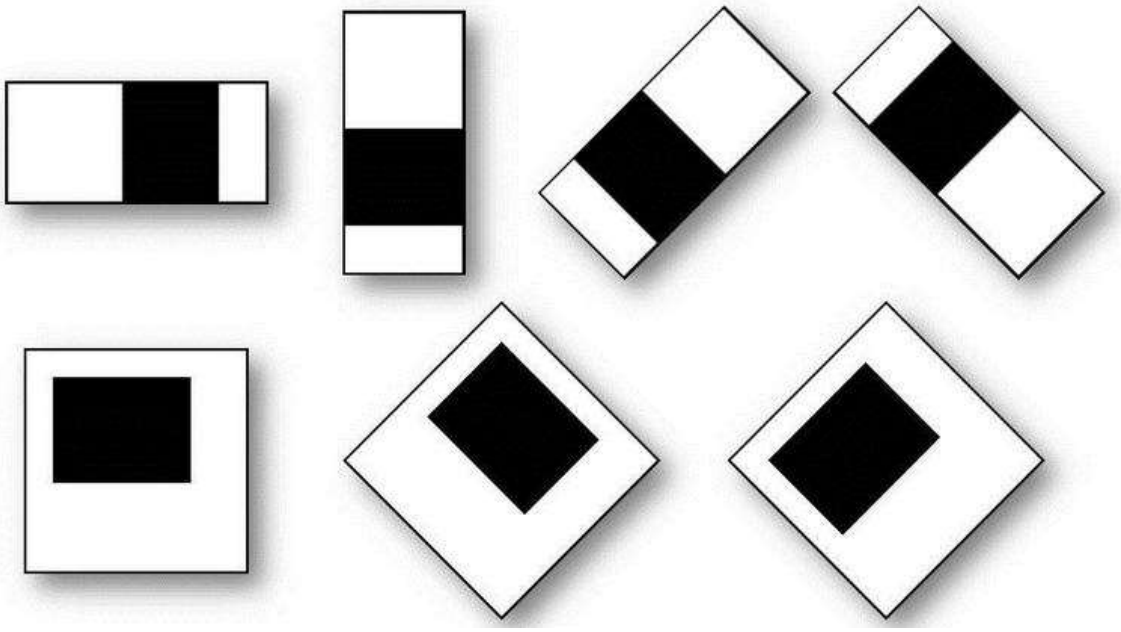


Рисунок 2.4 - розширені ознаки Хаара

Обчислюваним значенням такого показника буде

F = X-Y, (2.5)

де X - сума значень яскравості точок закриваються світлою частиною ознаки, а Y - сума значень яскравості точок закриваються темної частиною ознаки. Для їх обчислення використовується поняття інтегрального зображення, розглянуте вище.

Ознаки Хаара дають точкове значення перепаду яскравості по осі X і Y відповідно.

Пошук облич виконується за допомогою так званого скануючого вікна, розміри якого в оригінальному алгоритмі складають 24*24 пікселя. Вікно рухається по зображенню з кроком в 1 піксель і для кожного його положення обчислюються ознаки Хаара з різним масштабом і положенням у вікні. При цьому саме сканування проводиться так само і для різних масштабів скануючого вікна. Знайдені ознаки передаються класифікатору, який визначають по їх значенням, чи являється область зображенням, яка відноситься до вікну, обличчям чи ні.

2.3. Усунення шумів фільтром Гауса

Завдяки даному фільтру можна зменшити з відстанню вплив пікселів один на одного. Ядро даного фільтра можна виразити формулою(2.5)

$$F_{gauss}(i,j) = \frac{1}{2\pi\sigma^2}\exp(-\frac{i^2+j^2}{2\sigma^2}) \quad (2.5)$$

Де i, j - координати пікселя зображення; f - сигнал, а σ - шум, що знаходиться на оригінальному документі.Використовуючи фільтр Гаусса, можна розмивати шум, піддаючи змістовні контури зображення розмиття в малому ступені. Наприклад, якщо на оригінальному документі потрібно розмити дрібні деталі, що не вимагають відділення від фону, а питання, що цікавлять нас великі об'єкти будемо виділяти в подальшому за допомогою бинаризації.

Приклад роботи розмиття по Гаусу для одномірного масиву(рис 2.5)

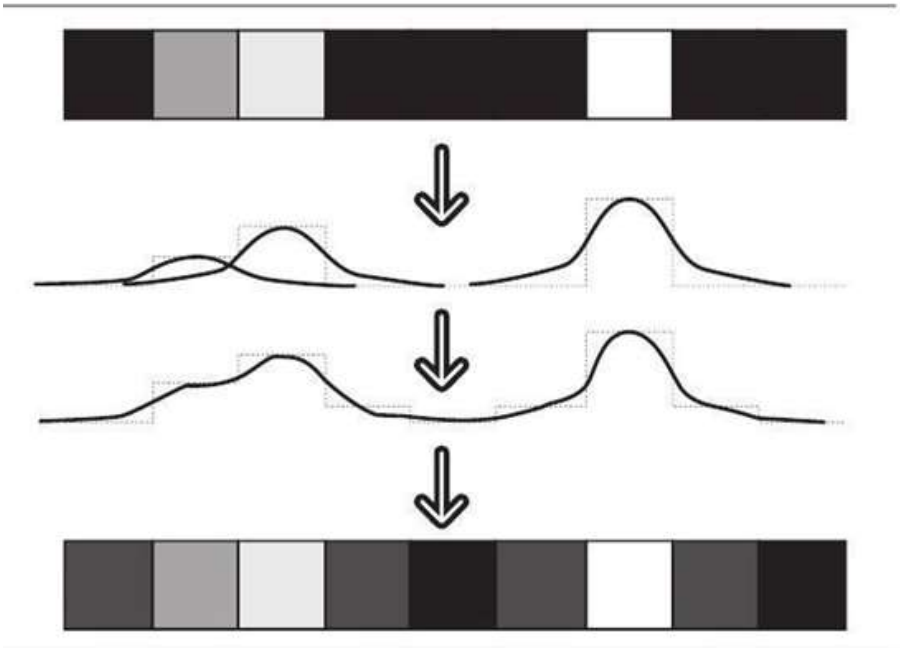


Рисунок 2.5 – Розмиття по Гаусу на одномірному масиві

Розмиття по Гаусу дає нам можливість позбутися від шумів на зображеннях, що зводить до мінімуму їх вплив при подальшій класифікації облич. Результат застосування фільтра Гауса до цілого зображення продемонстрована на (рис. 2.6).

Рисунок 2.6 – Результат застосування фільтра Гауса.

2.4. LBP перетворення

LBP - простий та ефективний оператор перетворення зображень, вперше запропонований у 1996 році для класифікації текстур . Пізніше він знайшов застосування в розпізнаванні облич.

Цей оператор аналізує яскравість кожного пікселя на зображенні та призначає значення кожному пікселю за допомогою функції. Потім отримане зображення ділиться на дві частини, кожна з яких має гістограму. Гістограми поєднуються та порівнюються за допомогою методів машинного навчання. У класичній версії використовується метод найближчого сусіда.

Перевагою цього методу є легка реалізація та швидка швидкість роботи, які можуть бути збільшені різними модифікаціями алгоритму. У цьому випадку алгоритм добре працює з розпізнаванням обличчя і

витримує монотонні зміни освітлення. Все це робить його ідеальним для розпізнавання обличчя в системах обробки в режимі реального часу.

Існує три алгоритми перетворення LBP:

- Класичний алгоритм - ядро оператора полягає в застосуванні піксельного порогового перетворення зображення, де значення яскравості оброблюваного пікселя порівнюється зі значеннями яскравості пікселів навколо нього;
- Уніфікований алгоритм LBP зменшує розмір гістограми, пояснюючи, що істотна інформація про форму об'єктів зображення містить лише частину локальних бінарних моделей;
- Центральне для симетричного LBP - Суть модифікації полягає в тому, що порогове значення для кожного пікселя носія не розглядається як значення яскравості середнього пікселя середовища, а швидше значення яскравості навпроти центру піксельного середовища.

Усі три алгоритми обчислення гістограми LBP були протестовані на двох різних наборах даних. Як алгоритм класифікації використовувався метод найближчого сусіда. Метод найближчого сусіда, розроблений для дослідження реалізації трансформацій LBP, пізніше був використаний для розробки системи розпізнавання обличчя та тестування швидкості цієї системи з використанням різних варіантів варіанту LBP.

Перший набір даних, використаний для тестування, - це база даних зображень Університету Кембриджу. Він містить зображення 40 облич, 10 зображень у кожному. Яскравість цих зображень не змінюється, але при зйомці можуть виникнути відмінності в положенні обличчя. Зображення однієї поверхні цієї основи показані на рисунку. 2.7.

Рисунок 2.7 - Фотографії з бази даних Кембриджського університету

П'ять знімків обличчя використовували для тренування та відбору проб. 128x128 пікселів перед обробкою. В результаті було класифіковано 400 зображень.

Друга - база даних зображень обличчя університету Єльського. [4] Ця база даних містить зображення 38 облич, 65 зображень кожного, включаючи різні режими освітлення. З них було обрано 10 зображень для тестування кожного обличчя. Приклад зображень з іншої бази даних обличчя показаний на малюнку 2.8.

Рисунок 2.8 – Фотографії з бази даних Єльського університету

Тестування проводилося так само, як у першій серії. Всього було засекречено 380 картин.

Ефективність виявлення кожного з трьох операторів LBP при тестуванні в першому наборі даних випробувань показана в таблиці. 2.1.

Таблиця 2.1 – Ефективність розпізнавання для кожного з трьох LBP на даних Кембриджського університету

Метод	Блоки							
	1x1	2x2	3x3	4x4	5x5	6x6	7x7	8x8
LBP	82,5%	91%	94%	95,5%	94%	93,25%	92,5%	89,3%
Uniform LBP	81%	93,8%	97%	94,5%	92%	92%	92%	89,5%
CS-LBP	67,8%	92,3%	95%	94,8%	94,3%	93,3%	90,3%	90,3%

Ефективність виявлення кожного з трьох операторів LBP при тестуванні з другим набором тестових даних показана в таблиці. 2.2.

Таблиця 2.2 - Ефективність розпізнавання для кожного з трьох LBP на даних Єльського університету

Метод	Блоки							
	1x1	2x2	3x3	4x4	5x5	6x6	7x7	8x8
LBP	41,8%	71,6%	88,2%	91,8%	92,6%	93,2%	95,8%	96,1%
Uniform LBP	41,3%	75,26 %	91,8%	91,8%	92,9%	92,1%	95%	94,2%

Результати показують, що Classic LBP і Uniform LBP працюють з однаковою точністю. Для досягнення роздільної здатності 90% або вище,

рекомендується використовувати роздільну здатність 4x4. Однак слід зазначити, що перший набір даних виявився найкращим розділенням 3x3.

Центрально-симетричний LBP, коли ділить зображення на невелику кількість блоків, нижче, ніж інші локальні двійкові моделі. Але коли використовується більше одного підмножини, точність його класифікації в середньому менше 3% для інших LBP. Під час перевірки першого набору даних CS-LBP він не перетинає жодного набору шаблонів розділів.

Як результат, центрально-симетричні локальні бінарні моделі популярні для розпізнавання обличчя через їх швидкість та точність і майже поступаються іншим LBP.

Оптимальна роздільна здатність і витрата пам'яті шляхом поділу зображення на підмножини за допомогою CS-LBP - це розділ 4x4, який використовується в системах.

2.5 Модифікація методу Віоли-Джонса

Як було зазначено раніше, обробка кадрів відеопотоку розроблюваною системою повинна включати два основних етапи. Перший етап - виявлення осіб методом Віоли-Джонса.

Другий етап - розпізнавання знайдених осіб за допомогою гістограм локальних бінарних шаблонів. Але продуктивність алгоритмів залежить від таких факторів як положення особи, освітлення, і т. д.. Тому, потрібно відразу описати умови використання системи, що розробляється, в яких може бути забезпечена її коректна робота:

- Допустимо тільки монотонну зміна освітлення. Навчальна та тестова вибірка повинна зніматися в однакових умовах освітлення.
- Використовується фронтальне, або близькі до нього положення осіб. Нейтральний вираз облич в зображеннях.
- Особи не перекриваються іншими об'єктами.

Узагальнена блок-схема алгоритму обробки кадрів розробляється системою представлена на (рис. 2.9).

Рисунок 2.9 – Узагальнена блок-схема алгоритму обробки кадрів

Також доцільно відразу описати необхідний функціонал, що розробляється:

- Обробка відеопотоку з підключеною до комп'ютера камери в реальному часі.
- Можливість налаштування параметрів роботи використовуваних для виявлення і розпізнавання алгоритмів.

- Виведення інформації про обличчя, що розпізнається, що включає міру належності до певного класу, графічне відображення гістограми і LBP представлення обличчя, що відслідковується.
- Можливість навчання і додавання класів осіб з використанням камери через інтерфейс програми.

Крім етапів виявлення і розпізнавання доцільно використовувати проміжні етапи обробки знайдених осіб. Застосування фільтра Гауса після 38 виявлення осіб допоможе знизити вплив шумів при розпізнаванні. Також є сенс застосувати маску значущих областей до локалізованим і перетворенням оператором LBP зображенням осіб, яка дозволить прибрати вплив при розпізнаванні кутових областей зображення, що містять задній план.

2.6. Висновок

В результаті загальний алгоритм розпізнавання повинен містити наступні кроки: виявлення осіб в кадрі, обробка знайдених осіб фільтром Гауса, застосування LBP трансформації до знайденим особам з подальшим застосуванням маски значущих областей, розрахунок гістограм знайдених осіб, класифікація осіб по гістограмі методом найближчого сусіда. В результаті буде отримано список що відслідкованих облич з їх характеристиками і координатами прямокутних областей кадру, в яких вони знаходяться.

Процедура навчання проводиться аналогічним чином. Знайдене в кадрі обличчя послідовно обробляється відповідно до описаного алгоритмом, розраховані гістограми осіб навчальної вибірки кожного класу зберігаються. Розпізнавання проводиться на основі пошуку мінімальної відстані між гістограмою вхідного зображення особи і гістограм що вже збережені.

3 РОЗРОБКА ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Варіативний аналіз і обґрунтування вибору засобів реалізації

3.1.1 Обґрунтування вибору мови програмування

При виборі засобів для розробки програмного проекту необхідно врахувати надзвичайно велику кількість різноманітних аспектів, найбільш важливим із яких є мова програмування, тому що вона в значній мірі визначає інші доступні засоби. На приклад, для розробки користувацького графічного інтерфейсу розробникам необхідна сумісна з мовою програмування GUI-бібліотека, яка надає готові елементи інтерфейсу, такі, як кнопки и меню.

При виборі мови програмування основними критеріями були продуктивність програмування, продуктивність роботи додатку та ефективність використання пам’яті. Програма складається з трьох частин, дві з яких є прикладними додаткам, а третя – веб-додатком. Тому важливим чинником для вибору мови, в даному випадку є її універсальність, оскільки використання багатьох мов в одному проекті може негативно вплинути на його якість та ускладнити підтримку.

Для розробки програмних систем такого рівня існує три мови, які відповідають вимозі універсальності: Java, C# та Python.

Java – об'єктно-орієнтована мова програмування, випущена компанією Sun Microsystems у 1995 році як основний компонент платформи Java. Зараз мовою займається компанія Oracle, яка придбала Sun Microsystems у 2009 році. Синтаксис мови багато в чому походить від C та C++. У офіційній реалізації, Java програми компілюються у байткод, який при виконанні

інтерпретується віртуальною машиною для конкретної платформи. Oracle надає компілятор Java та віртуальну машину Java, які задовольняють специфікації Java Community Process, під ліцезією GNU General Public License.

C# (вимовляється Сі-шарп) — об'єктно-орієнтована мова програмування з безпечною системою типізації для платформи .NET. Розроблена Андерсом Гейлсбергом, Скотом Вілтамутом та Пітером Гольде під егідою Microsoft Research (при фірмі Microsoft).

Синтаксис C# близький до C++ і Java. Мова має строгу статичну типізацію, підтримує поліморфізм, перевантаження операторів, вказівники на функції-члени класів, атрибути, події, властивості, винятки, коментарі у форматі XML. Перейнявши багато що від своїх попередників — мов C++, Delphi, Модула і Smalltalk — C#, спираючись на практику їхнього використання, виключає деякі моделі, що зарекомендували себе як проблематичні при розробці програмних систем, наприклад множинне спадкування класів (на відміну від C++).

Python (рекомендоване прочитання — «Па́йтон) — інтерпретована об'єктно-орієнтована мова програмування високого рівня з динамічною семантикою. Розроблена в 1990 році Гвідо ван Россумом. Структури даних високого рівня разом із динамічною семантикою та динамічним зв'язуванням роблять її привабливою для швидкої розробки програм, а також як засіб поєднання існуючих компонентів. Python підтримує модулі та пакети модулів, що сприяє модульності та повторному використанню коду. Інтерпретатор Python та стандартні бібліотеки доступні як у скомпільованій так і у вихідній формі на всіх основних платформах. В мові програмування Python підтримується декілька парадигм програмування, зокрема: об'єктно-орієнтована, процедурна, функціональна та аспектно-орієнтована.

Основним недоліком мови C# перед Java та Python є її проприєтарна ліцензія та можливість використання усіх її можливостей в повному обсязі лише на ОС Windows. Оскільки розроблювана система є серверною на повинна витримувати досить високі навантаження бажаною є повна підтримка UNIX-подібних операційних систем, зокрема GNU/Linux. Тому реальним є вибір між Java та Python.

Швидкість розробки програмного забезпечення мовою Python є на порядок вищою у порівнянні із Java завдяки його динамічній типізації. Крім того Python є інтерпретованою мовою, що полегшує внесення змін та налаштування програмної системи на етапі впровадження. Тому основною мовою програмування обрано мову Python.

Для реалізації клієнтської частини веб-інтерфейсу обрано мову програмування JavaScript, мову розмітки HTML та мову задання стилів CSS по причні відсутності у них аналогів.

3.1.2 Обґрунтування вибору бібліотек

Бібліотека (від англ. Library) - збірка підпрограм або об'єктів, що використовуються для розробки програмного забезпечення (ПО). У деяких мовах програмування (наприклад, в Python) те ж, що модуль, в деяких – кілька модулів. З точки зору операційної системи (ОС) і прикладного ПО бібліотеки поділяються на динамічні та статичні.

В сучасних умовах розробка великого проекту без використання, готових бібліотек є, практично, не можливою. Бібліотеки дають можливість програмісту зосередитись на вирішенні проставлених перед ним задач, а не побудови низькорівневої інфраструктури для їх вирішення. Такий підхід дає змогу значно скоротити час розробки та тестування програмного продукту, спростити його підтримку, а відповідно – зменшити його вартість.

Для прискорення розробки ми використали бібліотеку компютерного зору OpenCV(Open Computer Vision) - це бібліотека функцій програмування, головним чином спрямованих на комп'ютерний зір у режимі реального часу. Спочатку розроблений Intel, він згодом був підтриманий Willow Garage, а потім Itseez (який згодом був придбаний Intel). Бібліотека є платформою і є безкоштовною для використання за ліцензією BSD з відкритим кодом.

OpenCV підтримує рамки глибокого навчання TensorFlow, Torch / PyTorch та Caffe

Офіційно запущений у 1999 році, проект OpenCV спочатку був ініціативою Intel Research для просування додатків, що вимагають процесора, частина серії проектів, включаючи трасування променів у режимі реального часу та 3D-стіни дисплея. Основні учасники проекту включали низку експертів з оптимізації в Intel Russia, а також команду Intel Performance Performance Library. У перші дні OpenCV цілі проекту були описані як:

- Попередні дослідження зору, забезпечуючи не тільки відкритий, але й оптимізований код для базової інфраструктури зору. Більше не винаходити колесо.
- Поширювати знання про бачення, надаючи загальну інфраструктуру, яку розробники могли б побудувати, щоб цей код був легше читабельним та готовим до передачі.
- Заздалегідь розвиваючи комерційні програми на основі бачення, роблячи безкоштовний доступний портативний, оптимізований для роботи код - з ліцензією, яка не вимагала відкриття або вільного коду.

Перша альфа-версія OpenCV була оприлюднена на конференції IEEE з питань комп’ютерного бачення та розпізнавання образів у 2000 році, а п'ять бета-версій були випущені між 2001 та 2005 рр. Перша версія 1.0 була

випущена в 2006 році. Попередня версія 1.1 " "було випущено в жовтні 2008 року.

Другий головний реліз OpenCV відбувся у жовтні 2009 року. OpenCV 2 включає значні зміни в інтерфейсі C ++, спрямовані на простіші, безпечніші для типових моделей, нові функції та кращі реалізації для існуючих з точки зору продуктивності (особливо на багато- основні системи). Офіційні випуски зараз відбуваються кожні півроку [6], а розробку зараз здійснює незалежна російська команда за підтримки комерційних корпорацій.

У серпні 2012 року підтримку OpenCV взяла на себе некомерційний фонд OpenCV.org, який підтримує розробників [7] та користувацький сайт. [8]

У травні 2016 року Intel підписала угоду про придбання Itseez, провідного розробника OpenCV.

3.1.3 Обґрунтування вибору середовища розробки

Інтегроване середовище розробки, ICP (англ. IDE, Integrated development environment) - система програмних засобів, використовувана програмістами для розробки програмного забезпечення (ПО) [7].

Зазвичай, середа розробки включає в себе:

- текстовий редактор,
- компілятор і / або інтерпретатор,
- засоби автоматизації збирання,
- відладчик.

Іноді містить також засоби для інтеграції з системами управління версіями і різноманітні інструменти для спрощення конструювання графічного інтерфейсу користувача. Багато сучасних середовища розробки також включають браузер класів, інспектор об'єктів і діаграму ієрархії

класів - для використання при об'єктно-орієнтованої розробки ПЗ. Хоча й існують ICP, призначені для декількох мов програмування – такі як Eclipse, Embarcadero RAD Studio, Qt Creator, останні версії NetBeans, Xcode або Microsoft Visual Studio, але зазвичай ICP призначається для одного певної мови програмування - як, наприклад, Visual Basic, Delphi, Dev-C++.

Інтегровані середовища розробки були створені для того, щоб максимізувати продуктивність програміста завдяки тісно пов'язаним компонентам з простими користувача інтерфейсами. Це дозволяє розробнику зробити менше дій для перемикання різних режимів, на відміну від дискретних програм розробки.

ICP, зазвичай, являє собою єдину програму, в якій проводилася вся розробка. Вона, зазвичай, містить багато функцій для створення, зміни, компілювання, розгортання і налагодження програмного забезпечення. Мета середовища розробки полягає в тому, щоб абстрагувати конфігурацію, необхідну, щоб об'єднати утиліти командного рядка в одному модулі, який дозволить зменшити час, щоб вивчити мову, і підвищити продуктивність розробника. Також вважається, що важка інтеграція завдань розробки може далі підвищити продуктивність. Наприклад, ICP дозволяє проаналізувати код і тим самим забезпечити миттєвий зворотний зв'язок і повідомити про синтаксичні помилки.

Сьогодні існує досить велика кількість ICP для Python: Boa Constructor, Eclipse, PyDev , Eric, IDLE, Komodo, PyCharm, PyScripter, SPE, Wing IDE. Проте їхні можливості поки що не можуть задовольнити усіх потреб розробників програмного забезпечення. Винятками є Eclipse та PyCharm. Основним середовищем для розробки сервісу поштових розсилок було обрано PyCharm.

PyCharm - інтегрована середовище розробки для мови програмування Python. Надає засоби для аналізу коду, графічний відладчик, інструмент для

запуску юніт-тестів і підтримує веб-розробку на Django та Flask. PyCharm розроблена чеською компанією JetBrains на основі IntelliJ IDEA.

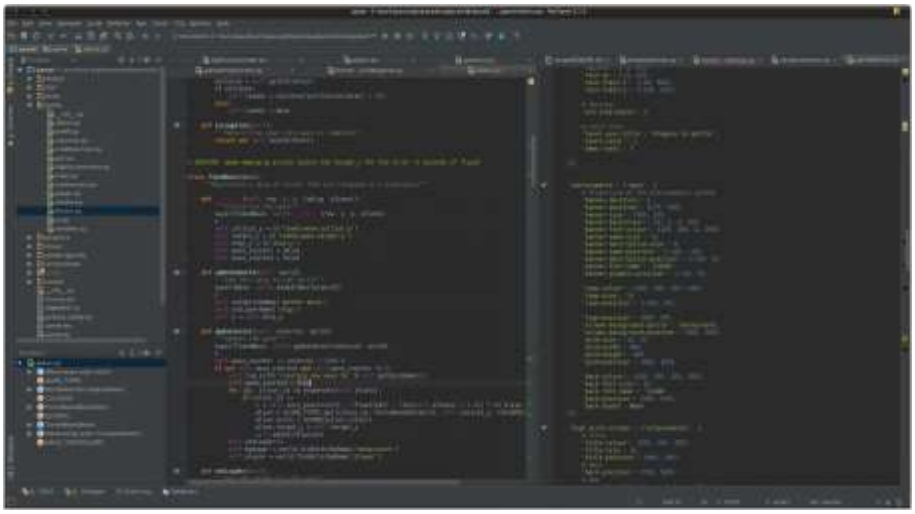


Рисунок 3.1 – Інтерфейс PyCharm

PyCharm працює під операційними системами Windows, Mac OS X і Linux.

- Статичний аналіз коду, підсвічування синтаксису і помилок
- Навігація за проектом і вихідного коду: відображення файлової структури проекту, швидкий перехід між файлами, класами, методами і використаннями методів
- Рефакторинг: перейменування, витяг методу , введення змінної, введення константи, підйом і спуск методу і т. д.
- Вбудований відладчик для Python
- Вбудовані інструменти для юніт-тестування
- Підтримка систем контролю версій: загальний користувальницький інтерфейс для Mercurial, Git, Subversion, Perforce і CVS з підтримкою списків змін та злиття
- Велика кількість плагінів, що дозволяють значно розширити можливості ICP, зокрема плагіни інтеграції із MongoDB

3.2 Тестування розробленої програмної системи

Тестування програмного забезпечення — це процес, що використовується для виміру якості розроблюваного програмного забезпечення. Зазвичай, поняття якості обмежується такими поняттями, як коректність, повнота, безпечність, але може містити більше технічних вимог, які описані в стандарті ISO 9126. Тестування — це процес технічного дослідження, який виконується на вимогу замовників, і призначений для вияву інформації про якість продукту відносно контексту, в якому він має використовуватись. До цього процесу входить виконання програми з метою знайдення помилок [29].

Якість не є абсолютною, це суб'єктивне поняттям. Тому тестування не може повністю забезпечити коректність програмного забезпечення. Воно тільки порівнює стан і поведінку продукту зі специфікацією. При цьому треба розрізняти тестування програмного забезпечення і забезпечення якості програмного забезпечення, до якого належать усі складові ділового процесу, а не тільки тестування.

Існує багато підходів до тестування програмного забезпечення, але ефективне тестування складних продуктів — це по суті дослідницький процес, а не тільки створення і виконання рутинної процедури.

У термінології професіоналів тестування (програмного й деякого апаратного забезпечення), фрази «тестування білого ящика» і «тестування чорного ящика» відносяться до того, чи має розробник тестів доступ до вихідного коду ПЗ, що тестується, або ж тестування виконується через інтерфейс користувача або прикладний програмний інтерфейс, наданий модулем, що тестується. При тестуванні білого ящика (англ. white-box testing, також говорять — прозорого ящика), розробник тесту має доступ до вихідного коду й може писати код, що пов'язаний з бібліотеками ПЗ, що тестується. Це типово для юніт-тестування (unit testing), при якому

тестуються тільки окремі частини системи. Воно забезпечує те, що компоненти конструкції — працездатні й стійкі до певного ступеня [30].

При тестуванні чорного ящика (англ. black-box testing), тестер має доступ до ПЗ тільки через ті ж інтерфейси, що й замовник або користувач, або через зовнішні інтерфейси, що дозволяють іншому комп'ютеру або іншому процесу підключитися до системи для тестування. Наприклад, модуль, що тестується, може віртуально натискати клавіші або кнопки миші в програмі, що тестується, за допомогою механізму взаємодії процесів, із упевненістю в тім, що ці події викликають той же відгук, що й реальні натискання клавіш і кнопок миші [31].

Тестування ПЗ дає можливість виявити переважну більшість критичних місць розроблювальної системи та перевірити всі механізми, що повинні забезпечити цілісність та конфіденційність даних. Воно проводиться не лише під час розробки системи та схеми ПЗ, але також на етапі супроводу, що пояснюється зміною вимог до системи зі сторони замовника або виявленням помилок в роботі тощо.

Для того, щоб ефективно провести тестування ПЗ, розробляється план тестування, який містить в собі опис стратегій, які будуть використовуватись для проведення тестування, а також сформовано тестовий набір даних. Все це відбувається на етапі проектування ПЗ. Велику увагу при цьому приділено формуванню тестового набору даних, оскільки заповнення ПЗ даними може тягнути за собою додаткові фінансові та людські ресурси. Для автоматизації цього процесу використано спеціалізоване програмне забезпечення.

Для програмного забезпечення даних було обрано TFD метод (Test-First Development). Цей метод передбачає підготовку тестів, невдале завершення яких буде вказувати на наявність помилок, ще до початку роботи по розробці основного функціоналу. На рис. 3.2 приведено схему TFD-тестування.

Це тестування складається з декількох етапів, опис яких подано нижче:

1. Прискорене створення тестів з метою невдалого завершення тестування на момент створення через відсутність потрібного функціоналу.

2. Виконання підготовлених тестів для перевірки того, що новий тест закінчується провалом.

3. Оновлення функціоналу таким чином, щоб він проходив тест.

4. Повторення виконання всіх тестів. Якщо тест не проходить, то відбувається повернення до кроку 3, а в іншому випадку – до початку циклу.

Перевагами такого методу при тестуванні як системи в цілому, так і окремих її модулів, яка є її частиною, в тому, що завжди можна бути впевненим в наявності тестів, за допомогою яких можна виконати перевірку якості виконаної роботи; а також вона дає впевненість в тому, що подальший розвиток системи можливий. В кінцевому випадку це тестування дає регресійний набір тестів для ПЗ, який можна виконувати над неї при внесення будь-яких змін.

Рисунок 3.2 – Схема TFD.

Роботу додатку на роботоздатність було перевірено за наступними критеріями :

- виявлення відомого обличчя (рис 3.3)

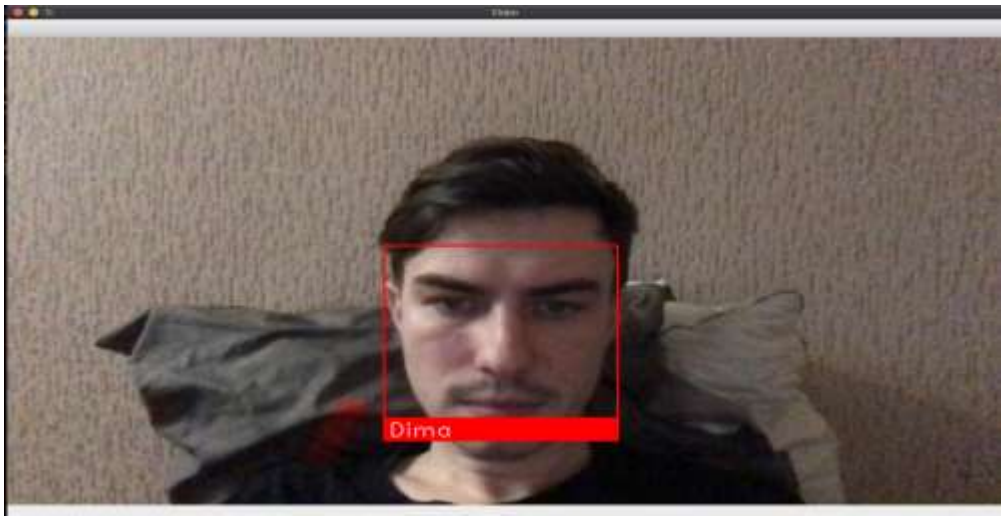


Рисунок 3.3 – Виявлення відомого обличчя

- виявлення невідомого обличчя (рис 3.4)

Рисунок 3.4 – Виявлення невідомого обличчя

- виявлення двох відомих облич (рис 3.5)

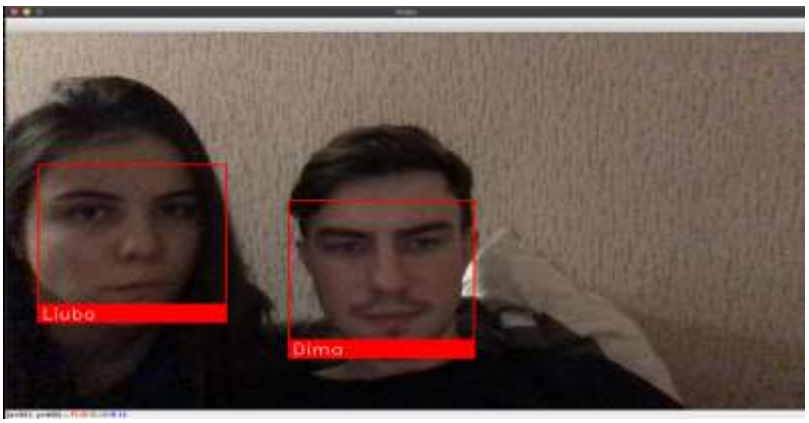


Рисунок 3.5 – Виявлення двох відомих облич

- виявлення відомого і невідомого облич (рис 3.6)

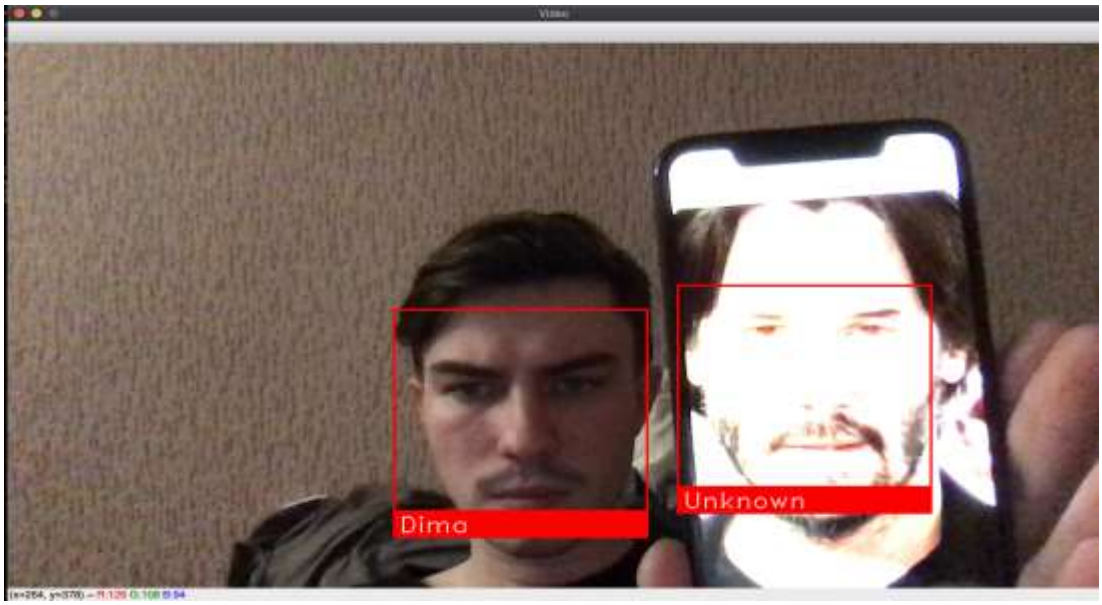


Рисунок 3.6 – Виявлення відомого і невідомого облич

- виявлення відомого обличчя з екрану телефону (рис 3.7)

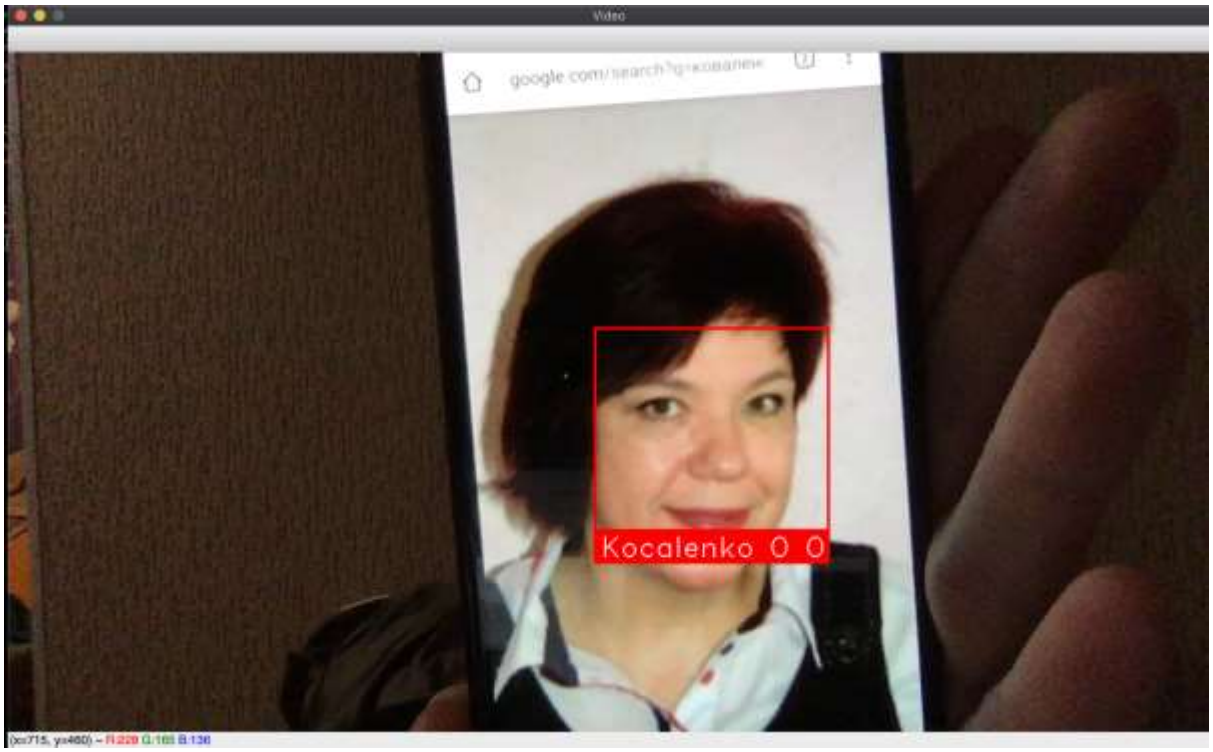


Рисунок 3.7 – Виявлення відомого обличчя з екрану телефону

Результати тестування свідчать про відповідність параметрів програми розпізнавання.

3.5 Аналіз результатів роботи програми

Розроблений програмний продукт було перевірено на повноту, безпечність, відповідність технічним вимогам, коректність роботи у різних програмних та апаратних середовищах шляхом проведення тестування білого та чорного ящиків, перевірки вихідного коду спеціалізованим програмним забезпеченням, з метою виявлення та усунення недоліків допущених при його розробці.

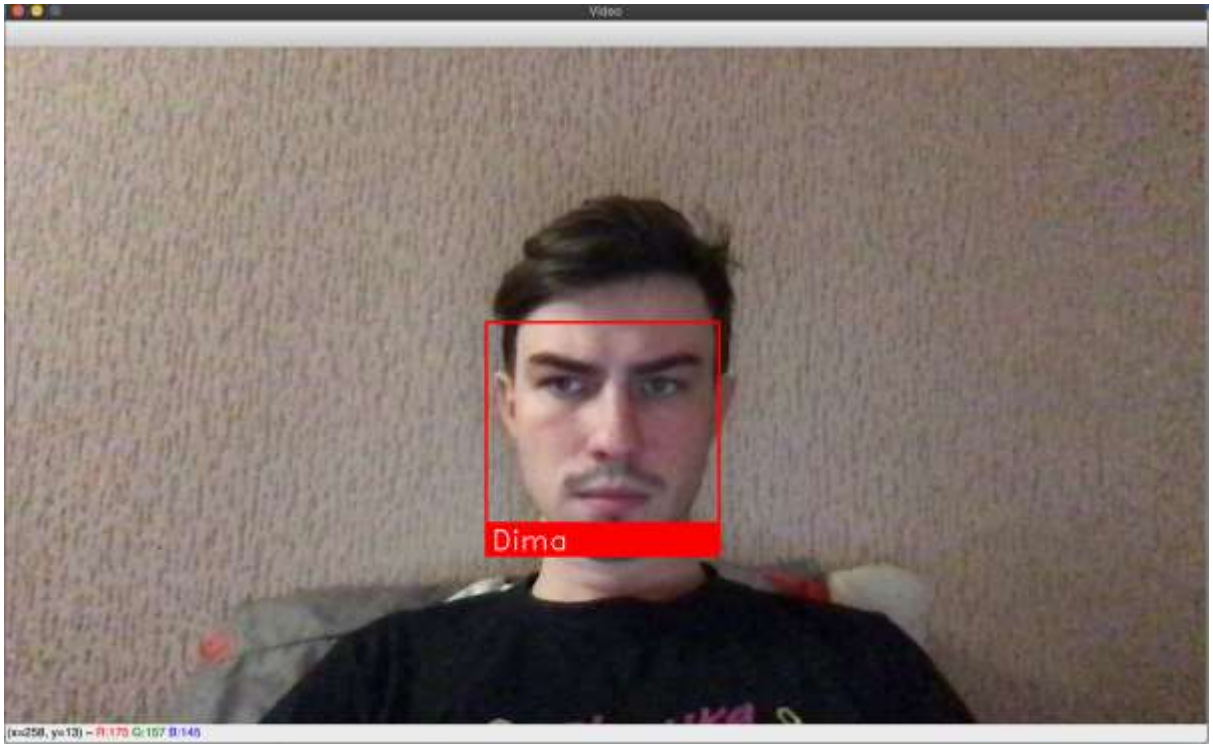


Рисунок 3.4 – Вигляд вікна розробленої програми

В ході досліджень програма показала себе, як стабільний продукт що відповідає усім поставленим вимогам. Система яскраво демонструє переваги розробленої архітектури, з точки зору продуктивності, відмово стійкості та масштабованості.

Графічний користувацький інтерфейс також цілком відповідає поставленим вимогам. Програма має зрозумілий і цікавий інтерфейс, приємні для сприйняття кольори.

3.6 Висновок

1. Розроблено алгоритми роботи основних модулів програмної системи
2. Обґрунтовано вибір мови програмування Python, середовища розробки PyCharm та необхідних бібліотек.
3. Розроблено програмне забезпечення для розпізнавання облич у реальному часі за допомогою модифікованого методу Віоли-Джонса.
4. Здійснено тестування розробленого програмного забезпечення на наявність помилок та відповідність поставленим вимогам.
5. В ході досліджень програма показала себе, як стабільний продукт що відповідає усім поставленим вимогам.

4 ЕКОНОМІЧНА ЧАСТИНА

4.1 Оцінювання комерційного потенціалу розробки

Метою проведення технологічного аудиту є оцінювання комерційного потенціалу розробки. Для проведення технологічного аудиту було залучено 2-х незалежних експертів. Здійснюємо оцінювання комерційного потенціалу розробки за 12-ма критеріями за 5-ти бальною шкалою.

Результати оцінювання комерційного потенціалу розробки наведено в таблиці 4.1.

Таблиця 4.1 – Результати оцінювання комерційного потенціалу розробки

Критерії	Прізвище, ініціали, посада експерта	
	1. Експерт 1	2. Експерт 2
	Бали, виставлені експертами:	
1	4	4
2	4	3
3	3	4
4	4	3
5	3	4
6	4	4
7	3	3
8	4	4
9	4	4
10	4	3
11	3	4
12	3	4
Сума балів	$СБ_1 = 44$	$СБ_2 = 44$
Середньоарифметична сума балів $\overline{СБ}$	$\overline{СБ} = \frac{\sum_{1}^{3} СБ_i}{2} = 44$	

Отже, з отриманих даних таблиці 4.1 видно, що нова розробка має високий рівень комерційного потенціалу.

4.2 Прогнозування витрат на виконання науково-дослідної роботи та конструкторсько–технологічної роботи.

Для розробки нового програмного продукту необхідні такі витрати.

Основна заробітна плата для розробників визначається за формулою (4.1):

$$З_{□} = \frac{□}{□_p} \cdot □, \quad (4.1)$$

де М- місячний посадовий оклад конкретного розробника;

T_p - кількість робочих днів у місяці, T_p = 22 дні;

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

Розрахунки заробітних плат для керівника і програміста наведені в таблиці 4.2.

Таблиця 4.2 – Розрахунки основної заробітної плати

Працівник	Оклад М, грн.	Оплата за робочий день, грн.	Число днів роботи, t	Витрати на оплату праці, грн.
Науковий керівник	5500	250	6	1500
Інженер-програміст	4000	181,81	45	8181,45
Всього:				9681,45

Розрахуємо додаткову заробітну плату:

$$Здод = 0,1 \cdot 9681,45 = 968,14 \ (грн.)$$

Нарахування на заробітну плату операторів НЗП розраховується як 37,5…40% від суми їхньої основної та додаткової заробітної плати:

$$\square зп = (Зо + Зр) \cdot \frac{\square}{100}, \qquad (4.2)$$

$$\square зп = (9681{,}45 + 968{,}14) \cdot \frac{36{,}3}{100} = 3865{,}80 \text{ (грн.)}.$$

Розрахунок амортизаційних витрат для програмного забезпечення виконується за такою формулою:

$$A = \frac{Ц \cdot H_a}{100} \cdot \frac{T}{12}, \qquad (4.3)$$

де Ц – балансова вартість обладнання, грн;

H_a – річна норма амортизаційних відрахувань % (для програмного забезпечення 25%);

Т – Термін використання (Т=3 міс.).

Таблиця 4.3 – Розрахунок амортизаційних відрахувань

Найменування програмного забезпечення	Балансова вартість, грн.	Норма амортизації, %	Термін використання, міс.	Величина амортизаційних відрахувань,грн
Персональний комп'ютер	10000	25	3	625
Всього:				625

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

$$K = \sum_1^n \quad Н_i \cdot Ц_i \cdot К_i, \qquad (4.4)$$

де n – кількість комплектуючих;

Ні - кількість комплектуючих і-го виду;

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

Кі – коефіцієнт транспортних витрат (приймемо Кі = 1,1).

Таблиця 4.4 - Витрати на комплектуючі, що були використані для розробки ПЗ.

Найменування матеріалу	Одиниці виміру	Ціна, грн.	Витрачено	Вартість витрачених матеріалів, грн.
Флешка	шт.	180	1	180
Пачка паперу	уп.	115	1	115
Ручка	шт.	5	1	5
Всього з урахуванням транспортних витрат				330

Витрати на силову електроенергію розраховуються за формулою:

$$B_e = B \cdot П \cdot Ф \cdot K_n ; \qquad (4.5)$$

де В – вартість 1кВт-години електроенергії (В=1,7 грн/кВт);

П – установлена потужність комп'ютера (П=0,6кВт);

Ф – фактична кількість годин роботи комп'ютера (Ф=200 год.);

$К_п$ – коефіцієнт використання потужності ($К_п < 1$, $К_п = 0,7$).

$$Ве = 1,7 \cdot 0,6 \cdot 200 \cdot 0,7 = 142,8 \ (грн.)$$

Розрахуємо інші витрати $В_{ін}$.

Інші витрати $І_в$ можна прийняти як (100…300)% від суми основної заробітної плати розробників та робітників, які були виконували дану роботу, тобто:

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

Отже, розрахуємо інші витрати:

$$В_{ін} = 1 * (9681,45 + 968,14) = 10649,59 \ (грн).$$

Сума всіх попередніх статей витрат дає витрати на виконання даної частини роботи:

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

В = 9681,45+968,14+3865,80+625+330+142,8+10649,59= 26262,78 (грн.)

Розрахуємо загальну вартість наукової роботи $B_{заг}$ за формулою:

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

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

$$В_{заг} = \frac{26262,78}{1} = 26262,78$$

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

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

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

Отже, розрахуємо загальні витрати:

$$ЗВ = \frac{26262,78}{0,9} = 29180,86 \text{ (грн.)}$$

4.3 Прогнозування комерційних ефектів від реалізації результатів розробки.

Спрогнозуємо отримання прибутку від реалізації результатів нашої розробки. Зростання чистого прибутку можна оцінити у теперішній вартості грошей. Це забезпечить підприємству (організації) надходження додаткових коштів, які дозволять покращити фінансові результати діяльності .

Оцінка зростання чистого прибутку підприємства від впровадження результатів наукової розробки. У цьому випадку збільшення чистого прибутку підприємства $\Delta П_i$ для кожного із років, протягом яких очікується отримання позитивних результатів від впровадження розробки, розраховується за формулою:

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

де $\Delta П_я$ – покращення основного якісного показника від впровадження результатів розробки у даному році;

N – основний кількісний показник, який визначає діяльність підприємства у даному році до впровадження результатів наукової розробки;

ΔN – покращення основного кількісного показника діяльності підприємства від впровадження результатів розробки;

$П_я$ – основний якісний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки;

n – кількість років, протягом яких очікується отримання позитивних результатів від впровадження розробки.

В результаті впровадження результатів наукової розробки витрати на виготовлення програмного продукту зменшиться на 20 грн (що автоматично спричинить збільшення чистого прибутку підприємства на 20 грн), а кількість користувачів, які будуть користуватись збільшиться: протягом першого року – на 200 користувачів, протягом другого року – на 175 користувачів, протягом третього року – 125 користувачів. Реалізація

програмного продукту до впровадження результатів наукової розробки складала 600 користувачів, а прибуток, що отримував розробник до впровадження результатів наукової розробки – 300 грн.

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

Отже, збільшення чистого продукту $\Delta П_1$ протягом першого року складатиме:

$$\Delta П_1 = 20 \cdot 600 + (300 + 20) \cdot 200 = 76000 \text{ грн.}$$

Протягом другого року:

$$\Delta П_2 = 20 \cdot 600 + (300 + 20) \cdot (200 + 175) = 132000 \text{ грн.}$$

Протягом третього року:

$$\Delta П_3 = 20 \cdot 600 + (300 + 20) \cdot (200 + 175 + 125) = 172000 \text{ грн.}$$

4.4 Розрахунок ефективності вкладених інвестицій та період їх окупності

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

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

$$E_{абс} = (ПП - PV),$$

(4.10)

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДДКР, грн;

т – період часу, протягом якого виявляються результати впровадженої НДДКР, 3 роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник знаходиться на рівні 0,1;

t – період часу (в роках) від моменту отримання чистого прибутку до точки 2, 3,4.

Рисунок, що характеризує рух платежів (інвестицій та додаткових прибутків) буде мати вигляд, рисунок 4.1.

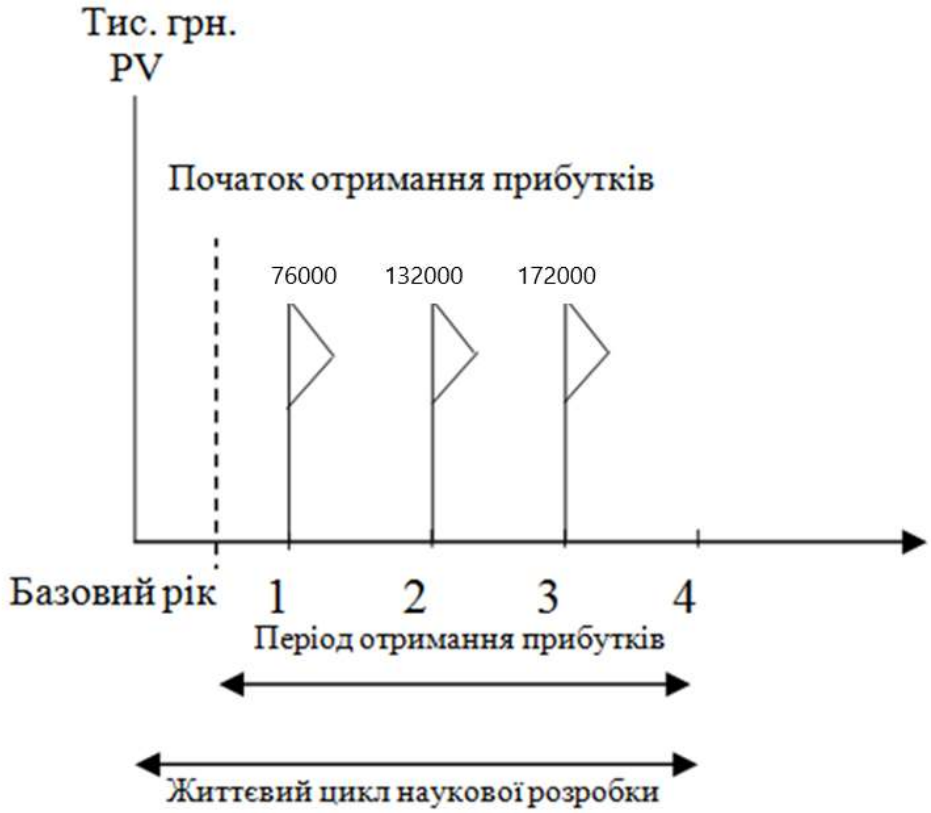


Рисунок 4.1 – Вісь часу з фіксацією платежів, що мають місце під час розробки та впровадження результатів НДДКР

Розрахуємо вартість чистих прибутків за формулою:

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

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДДКР, грн;

т – період часу, протягом якого виявляються результати впровадженої НДДКР, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник знаходиться на рівні 0,1;

t – період часу (в роках) від моменту отримання чистого прибутку до точки.

Отже, розрахуємо вартість чистого прибутку:

$$ПП = \frac{29180,86}{(1+0,1)^0} + \frac{76000}{(1+0,1)^2} + \frac{132000}{(1+0,1)^3} + \frac{172000}{(1+0,1)^4} = 308642,63 \text{ (грн.)}$$

Тоді розрахуємо $E_{абс}$:

$$E_{абс} = 308642,63 - 29180,86 = 279461,77 \text{ грн.}$$

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

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $E_в$ за формулою:

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

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

PV – теперішня вартість інвестицій PV = ЗВ, грн;

$T_ж$ – життєвий цикл наукової розробки, роки.

Тоді будемо мати:

$$E_в = \sqrt[3]{1 + \frac{279466,71}{29180,86}} - 1 = 1,19 \text{ або } 119\ \%$$

Далі, розраховану величина $E_в$ порівнюємо з мінімальною (бар'єрною) ставкою дисконтування $\tau_{мін}$, яка визначає ту мінімальну дохідність, нижче

за яку інвестиції вкладатися не будуть. У загальному вигляді мінімальна (бар'єрна) ставка дисконтування $\tau_{мін}$ визначається за формулою:

$$\tau = d + f,$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2019 році в Україні d = 0,2;

f – показник, що характеризує ризикованість вкладень, величина f = 0,1.

$$\tau = 0{,}2 + 0{,}1 = 0{,}3$$

Оскільки $E_в = 119\% > \tau_{мін} = 0{,}3 = 30\%$, то у інвестор буде зацікавлений вкладати гроші в дану наукову розробку.

Термін окупності вкладених у реалізацію наукового проекту інвестицій. Термін окупності вкладених у реалізацію наукового проекту інвестицій $T_{ок}$ розраховується за формулою:

$$T_{ок} = \frac{1}{E_в}$$

$$T_{ок} = \frac{1}{1{,}19} = 0{,}84 \text{ року}$$

Обрахувавши термін окупності даної наукової розробки, можна зробити висновок, що фінансування даної наукової розробки буде доцільним.

ВИСНОВКИ

У ході виконання магістерської кваліфікаційної роботи було розроблено програмний продукт для розпізнавання облич у реальному часі.

Основні результати, отримані при виконанні дипломної роботи наступні:

- Виконано аналіз сучасного стану питання, на базі якого було визначено доцільність розробки нового продукту.
- Виконано аналіз існуючих методів розпізнавання облич.
- Розроблено модифікований метод Віоли-Джонса з використанням локальних бінарних шаблонів.

Розроблено програмний продукт для розпізнавання облич у реальному часі та протестовано коректність його роботи, в результаті чого було виявлено що програмний продукт працює коректно.

Для розробки було обрано мову програмування Python та технологія OpenCV2.

Наукові результати магістерської кваліфікаційної роботи:

1. Отримав подальшого розвитку метод Віоли-Джонса, який відрізняється від відомого використанням локальних бінарних шаблонів, що дає підвищену швидкодію розпізнавання.

2. Запропоновано метод дослідження ефективності розпізнавання облич LBP перетворень: класичного, рівномірного та центрально-симетричного, яка на відміну від відомих дозволяє зробити кращий вибір методу для розпізнавання облич.

Практична цінність одержаних результатів полягає в тому, що на основі отриманих в магістерській кваліфікаційній роботі теоретичних положень розроблено програмний продукт розпізнавання облич у реальному часі для комп’ютерних систем.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Video Analytics Market worth $3,971.2 Million by 2020 // Markets and Markets.URL:http://www.marketsandmarkets.com/PressReleases/iva.asp.
2. V. Gupta, D. Sharma. A Study of Various Face Detection Methods // International Journal of Advanced Research in Computer and Communication Engineering Vol. 3, May 2014, №5. С. 6694–6697.
3. P. Viola, M. Jones. Rapid object detection using a boosted cascade of simple features/ 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Vol. 1. 8–14 December 2001 / The Institute of Electrical and Electronics Engineers, Inc. С. 511–518.
4. Y. Freund, R. E. Schapire. A Short Introduction to Boosting // Journal of Japanese Society for Artificial Intillegence. 1999, №14(5), С. 771-780.
5. D. Roth, The SNoW Learning Architecture // Technical Report UIUCDCS-R-99-2102 . UIUC Computer Science Department, 1999.
6. M. Nilsson, M. Dahl, I. Claesson. The successive mean quantization transform /Proceedings of IEEE Int. Conf.ICASSP 2005, Vol. 4 / The Institute of Electrical and Electronics Engineers Signal Processing Society С. 429 – 432.
7. H. A. Rowley, S. Baluja, .T. Kanade. Neural Network-Based Face Detection / PAMI, January 1998.8. E. Osuna, R. Freund, F. Girosi. Training support vector machines:an application toface detection // In Proceedings of Computer Vision and pattern Recognition 1997,C. 130-136.
8. Machine learning methods // Graphicon. URL: http://www.graphicon.ru/oldgr/ru/publications/text/gc2006avezh.pdf.
9. L.Juwei, N. P.Konstantinos, A. Venetsanopoulos, “Face recognition using kernel direct discriminant analysis algorithms”, IEEE Transactions On Neural Networks, vol.14, no. 1, pp.117–126, January 2003.

10. M. Lades, J. Vorbruggen, J. Buhmann, “Distortion invariant object recognition in the dynamic link architecture”, IEEE Transactions on computers, 1993, vol. 42, no. 3, pp. 300 -310, March 1993.
11. P. Viola, “Robust realtime face detection”, International Journal of Computer Vision, 2004, vol. 57, no. 2, pp. 137-154, 2004 .
12. А.М.Лисенко, “Застосування біометричних систем для ідентифікації особи”, Вісник Київського нац. ун.-ту ім. Т.Шевченка, Юридичні науки, 2004, №60/62, с. 87-91 .
13. Метод Виолы-Джонса (Viola-Jones) как основа для распознавания лиц, Электронный ресурс, Режим доступа:https://habrahabr.ru/post/133826/
14. P. Viola, “Rapid object detection using a boosted cascade of simple features”, IEEE Conf. on Computer Vision and Pattern Recognition, V. 1, Kauai, Hawaii, USA., pp. 511–518, 2001.
15. S.Lawrence, C.L. Giles., C. Tsoiта, “Face Recognition: A Convolutional Neural Network Approach”, IEEE Transactions on Neural Networks, Special Issue on Neural Networks and Pattern Recognition, vol. 8, no 1, pp.98–113, 1997.
16. Y.Taigman, M.Yang, M.Ranzato, “DeepFace: Closing the Gap to human-Level Performance in Face Verification”, [Online]. Available at: https://www.cs.toronto.edu/~ranzato/publications/taigman_cvpr14.pdf
17. Joo Er Meng, W.Chen, Wu Shiqian, “High-speed face recognition based on discrete cosine transform and RBF neural networks”, IEEE Transactions on Neural Networks, vol. 16, no. 3, pp. 679 – 691,2005.
18. H.A. Rowley, S. Baluja, T.Kanade (1998), “Rotation invariant neural network-based face detection”, Computer Vision and Pattern Recognition, Proceedings. IEEE Computer Society Conference, [Online]. Available at : http://ieeexplore.ieee.org/abstract/document/698585/

19. T. Mäenpää (2003), “The local binary pattern approach to texture analysis – extensions and applications”, Finland ,Oulu University Press.
20. T.Ahonen, A. Hadid, M.Pietikainen, “Face Description with Local Binary Patterns: Application to Face Recognition”,IEEE Transactions on Pattern Analysis and Machine Intelligence,2006, vol. 28, no 12, pp. 2037 – 2041.
21. Ю. Лифшиц, Методы распознавания лиц, Електронний ресурс,Режим доступу: http://yury.name/modern/08modernnote.pdf
22. T. F.Cootes, G. J. Edwards, C. J. Taylor, “Active appearance models”, Computer Vision ECCV’98, vol.. 14, no. 7 of the series Lecture Notes in Computer Science, pp. 484-498, 2006.
23. М. В.Давидов, Ю. В. Нікольський, С. М. Тиханський, “Алгоритм визначення форми губ під час артикуляції для української жестової мови”, Вісник Національного університету "Львівська політехніка", no. 673 : Інформаційні системи та мережі, сс. 267-273, 2010.
24. T. F. Cootes, G. J. Edwards, C. J. Taylor, “Active appearance models”, IEEE Trans. on Pattern Recognition and Machine Intelligence, vol. 23, no.6 , pp. 681–685, 2001.
25. Алан Шаллоуей, Джеймс Р. Тротт. Шаблоны проектирования. Новый подход к объектно-ориентированному анализу и проектированию = Design Patterns Explained: A New Perspective on Object-Oriented Design. — М. : «Вильямс», 2002. — 288 с. — ISBN 0-201-71594-5.
26. Буров Є. В. Комп'ютерні мережі: підручник / Євген Вікторович Буров. — Львів: «Магнолія 2006», 2010. — 262 с. ISBN 966-8340-69-8
27. Гото Келлі, Котлер Емілі. Веб-редизайн, 2-е видання. — СПб. : «Символ-Плюс», 2006. — С. 416. — ISBN 5-93286-082-0.
28. Джек Грінфілд, Кіт Шорт, Стів Кук, Стюарт Кент, Джон Крупи Фабрики розробки програм (Software Factories): потокова збірка типових додатків, моделювання, структури та інструменти = Software Factories: Assembling Applications with Patterns, Models, Frameworks,

and Tools. — М.: «Діалектика», 2006. — С. 592. — ISBN 978-5-8459-1181-0

29. Калбертсон Роберт, Браун Крис, Кобб Гэри. Быстрое тестирование. — М. : «Вильямс», 2002. — 374 с. — ISBN 5-8459-0336-X.

30. Синицын С. В., Налютин Н. Ю. Верификация программного обеспечения. — М. : БИНОМ, 2008. — 368 с. — ISBN 978-5-94774-825-3.

31. Бейзер Б. Тестирование чёрного ящика. Технологии функционального тестирования программного обеспечения и систем. — СПб. : Питер, 2004. — 320 с. — ISBN 5-94723-698-2.

ДОДАТКИ

ДОДАТОК А

Технічне завдання на Магістерську кваліфікаційну роботу.

Технічне завдання на магістерську кваліфікаційну роботу

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О.Н.
"____"_____2019року

Технічне завдання
на магістерську роботу за спеціальністю
Інженерія програмного забезпечення

Керівник бакалаврської роботи:
к.т.н., доцент
О.О. Коваленко
"____"_____2019 р.

Виконав:
студент гр.1ПІ-18м
Д.Є. Стеблюк
"____"_____2017 р.

Вінниця ВНТУ 2019

1. Галузь застосування – комп’ютерні системи.

2. Підстава для розробки.
Підставою для розробки магістерської кваліфікаційної роботи є рішення засідання кафедри програмного забезпечення (протокол №__ від «__»_____ 2017 року).

3. Мета та призначення розробки.
Мета виконання магістерської кваліфікаційної роботи – підвищення швидкості розпізнавання обличчя, за рахунок модифікації методів та розробки засобів розпізнавання на основі модифікованого методу.
Призначення роботи – модифікації методів та розробки засобів розпізнавання облич у реальному часі.

4. Джерела розробки:
- завдання на магістерську кваліфікаційну роботу.

5. Технічні вимоги:

- Операційна система – Unix*.
- Мови програмування – Python.
- Технології – OpenCV.

6. Кліматичні умови.
Температурний діапазон +5ºС +40ºС, відносна вологість повітря не більше 75% та тиск – 720-740 мм. рт.ст.

7. Конструктивні вимоги.
Графічна та текстова документація повинна відповідати всім діючим стандартам України.
8. Перелік технічної документації, що пред’являється по закінченню робіт:

- пояснювальна записка до МКР;
- лістинги програми.

9. Стадії та етапи розробки:

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз завдання і вибір методів його вирішення	03.09.19 – 27.09.19	Вик.
2	Аналіз існуючих аналогів та постановка задач дослідження	28.09.19 – 8.10.19	Вик.
3	Розробка методів розпізнавання	09.10.19 – 01.11.19	Вик.
4	Розробка структур і алгоритмів програмного продукту	02.11.19 – 10.11.19	Вик.
5	Розробка програмного забезпечення	11.11.19 – 21.11.19	Вик.
6	Економічна частина	26.11.19 – 30.11.19	Вик.
7	Оформлення матеріалів до захисту МКР	01.12.19 – 06.12.19	Вик.

10. Порядок контролю та прийняття.

Виконання етапів магістерської кваліфікаційної роботи контролюється керівником згідно з графіком виконання роботи.

Прийняття магістерської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком захисту.

Корегування технічного завдання допускається з дозволу керівника магістерської кваліфікаційної роботи.

ДОДАТОК Б

Лістинг програми

```
import face_recognition.api
import cv2
import numpy as np

video_capture = cv2.VideoCapture(0)

dima_image = face_recognition.load_image_file("Dima.jpg")
dima_face_encoding = face_recognition.face_encodings(dima_image)[0]

liuba_image = face_recognition.load_image_file("liuba.jpg")
liuba_face_encoding = face_recognition.face_encodings(liuba_image)[0]

romaniuk_image = face_recognition.load_image_file("romaniuk.jpg")
romaniuk_face_encoding =
face_recognition.face_encodings(romaniuk_image)[0]

kovalenko_image = face_recognition.load_image_file("kovalenko.jpg")
kovalenko_face_encoding =
face_recognition.face_encodings(kovalenko_image)[0]


known_face_encodings = [
    dima_face_encoding,
    liuba_face_encoding,
    romaniuk_face_encoding,
    kovalenko_face_encoding
]
known_face_names = [
    "Dima",
    "Liuba",
    "Romaniuk O N",
    "Kocalenko O O"
]

face_locations = []
face_encodings = []
face_names = []
```

```
process_this_frame = True

while True:
    ret, frame = video_capture.read()

    # Resize frame of video to 1/4 size for faster face recognition processing
    small_frame = cv2.resize(frame, (0, 0), fx=0.25, fy=0.25)

    rgb_small_frame = small_frame[:, :, ::-1]

    if process_this_frame:
        face_locations = face_recognition.face_locations(rgb_small_frame)
        face_encodings = face_recognition.face_encodings(rgb_small_frame,
face_locations)

        face_names = []
        for face_encoding in face_encodings:
            matches = face_recognition.compare_faces(known_face_encodings,
face_encoding)
            name = "Unknown"



            face_distances =
face_recognition.face_distance(known_face_encodings, face_encoding)
            best_match_index = np.argmin(face_distances)
            if matches[best_match_index]:
                name = known_face_names[best_match_index]

            face_names.append(name)

    process_this_frame = not process_this_frame



    for (top, right, bottom, left), name in zip(face_locations, face_names):
        top *= 4
        right *= 4
        bottom *= 4
```

```
        left *= 4

        cv2.rectangle(frame, (left, top), (right, bottom), (0, 0, 255), 2)

        cv2.rectangle(frame, (left, bottom - 35), (right, bottom), (0, 0, 255),
cv2.FILLED)
        font = cv2.FONT_HERSHEY_DUPLEX
        cv2.putText(frame, name, (left + 6, bottom - 6), font, 1.0, (255, 255, 255),
1)

    cv2.imshow('Video', frame)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

video_capture.release()
cv2.destroyAllWindows()
```

api.py

```
# -*- coding: utf-8 -*-

import PIL.Image
import dlib
import numpy as np
from PIL import ImageFile

try:
    import face_recognition_models
except Exception:
    print("Please install `face_recognition_models` with this command before
using `face_recognition`:\n")
    print("pip install git+https://github.com/ageitgey/face_recognition_models")
    quit()

ImageFile.LOAD_TRUNCATED_IMAGES = True
```

```
face_detector = dlib.get_frontal_face_detector()

predictor_68_point_model =
face_recognition_models.pose_predictor_model_location()
pose_predictor_68_point = dlib.shape_predictor(predictor_68_point_model)

predictor_5_point_model =
face_recognition_models.pose_predictor_five_point_model_location()
pose_predictor_5_point = dlib.shape_predictor(predictor_5_point_model)

cnn_face_detection_model =
face_recognition_models.cnn_face_detector_model_location()
cnn_face_detector =
dlib.cnn_face_detection_model_v1(cnn_face_detection_model)

face_recognition_model =
face_recognition_models.face_recognition_model_location()
face_encoder = dlib.face_recognition_model_v1(face_recognition_model)


def _rect_to_css(rect):
    """
    Convert a dlib 'rect' object to a plain tuple in (top, right, bottom, left) order

    :param rect: a dlib 'rect' object
    :return: a plain tuple representation of the rect in (top, right, bottom, left)
order
    """
    return rect.top(), rect.right(), rect.bottom(), rect.left()


def _css_to_rect(css):

    return dlib.rectangle(css[3], css[0], css[1], css[2])


def _trim_css_to_bounds(css, image_shape):
```

```
    return max(css[0], 0), min(css[1], image_shape[1]), min(css[2],
image_shape[0]), max(css[3], 0)


def face_distance(face_encodings, face_to_compare):

    if len(face_encodings) == 0:
        return np.empty((0))

    return np.linalg.norm(face_encodings - face_to_compare, axis=1)


def load_image_file(file, mode='RGB'):

    im = PIL.Image.open(file)
    if mode:
        im = im.convert(mode)
    return np.array(im)


def _raw_face_locations(img, number_of_times_to_upsample=1,
model="hog"):

    if model == "cnn":
        return cnn_face_detector(img, number_of_times_to_upsample)
    else:
        return face_detector(img, number_of_times_to_upsample)


def face_locations(img, number_of_times_to_upsample=1, model="hog"):

    if model == "cnn":
        return [_trim_css_to_bounds(_rect_to_css(face.rect), img.shape) for face in
_raw_face_locations(img, number_of_times_to_upsample, "cnn")]
    else:
```

```
    return [_trim_css_to_bounds(_rect_to_css(face), img.shape) for face in
_raw_face_locations(img, number_of_times_to_upsample, model)]


def _raw_face_locations_batched(images, number_of_times_to_upsample=1,
batch_size=128):

    return cnn_face_detector(images, number_of_times_to_upsample,
batch_size=batch_size)


def batch_face_locations(images, number_of_times_to_upsample=1,
batch_size=128):

    def convert_cnn_detections_to_css(detections):
        return [_trim_css_to_bounds(_rect_to_css(face.rect), images[0].shape) for
face in detections]

    raw_detections_batched = _raw_face_locations_batched(images,
number_of_times_to_upsample, batch_size)

    return list(map(convert_cnn_detections_to_css, raw_detections_batched))


def _raw_face_landmarks(face_image, face_locations=None, model="large"):
    if face_locations is None:
        face_locations = _raw_face_locations(face_image)
    else:
        face_locations = [_css_to_rect(face_location) for face_location in
face_locations]

    pose_predictor = pose_predictor_68_point

    if model == "small":
        pose_predictor = pose_predictor_5_point
```

```
    return [pose_predictor(face_image, face_location) for face_location in
face_locations]


def face_landmarks(face_image, face_locations=None, model="large"):

    landmarks = _raw_face_landmarks(face_image, face_locations, model)
    landmarks_as_tuples = [[(p.x, p.y) for p in landmark.parts()] for landmark in
landmarks]

    # For a definition of each point index, see https://cdn-images-
1.medium.com/max/1600/1*AbEg31EgkbXSQehuNJBlWg.png
    if model == 'large':
        return [{
            "chin": points[0:17],
            "left_eyebrow": points[17:22],
            "right_eyebrow": points[22:27],
            "nose_bridge": points[27:31],
            "nose_tip": points[31:36],
            "left_eye": points[36:42],
            "right_eye": points[42:48],
            "top_lip": points[48:55] + [points[64]] + [points[63]] + [points[62]] +
[points[61]] + [points[60]],
            "bottom_lip": points[54:60] + [points[48]] + [points[60]] + [points[67]]
+ [points[66]] + [points[65]] + [points[64]]
        } for points in landmarks_as_tuples]
    elif model == 'small':
        return [{
            "nose_tip": [points[4]],
            "left_eye": points[2:4],
            "right_eye": points[0:2],
        } for points in landmarks_as_tuples]
    else:
        raise ValueError("Invalid landmarks model type. Supported models are
['small', 'large'].")
```

```
def face_encodings(face_image, known_face_locations=None, num_jitters=1):

    raw_landmarks = _raw_face_landmarks(face_image, known_face_locations,
model="small")
    return [np.array(face_encoder.compute_face_descriptor(face_image,
raw_landmark_set, num_jitters)) for raw_landmark_set in raw_landmarks]


def compare_faces(known_face_encodings, face_encoding_to_check,
tolerance=0.6):

    return list(face_distance(known_face_encodings, face_encoding_to_check)
<= tolerance)
```

ДОДАТОК В

Ілюстративний матеріал

Ілюстративний матеріал до захисту магістерської кваліфікаційної роботи

Завідувач кафедри ПЗ, д. т. н., професор _____ О.Н. Романюк

Науковий керівник, к.т.н., доцент _____ О.О. Коваленко

Рецензент, к.ф.-м.н., професор кафедри КН _____Т.О. Савчук

Нормоконтроль, к.т.н., доцент _____О.О. Коваленко

Виконавець, студент гр.1ПІ-18м _____ Д.Є Стеблюк