**Луцький національний технічний університет  
Технологічний факультет  
Кафедра автоматизації та комп’ютерно-інтегровані технології**

### Пояснювальна записка до випускної кваліфікаційної роботи

\_\_\_\_\_\_\_\_\_магістр\_\_\_\_\_\_\_  
(освітньо-кваліфікаційний рівень)  
на тему: «Автоматизована система персоналізації та ведення обліку персоналу з використанням системи розпізнавання зображень»

Виконав: студент VI курсу,

групи \_АВм\_ спеціальності 151 Автоматизація та комп’ютерно – інтегровані технології  
Коблинський В.М.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
                                  (прізвище та ініціали)  
Керівник Лотиш В.В.\_\_\_\_\_\_\_\_\_\_\_\_\_  
                                  (прізвище та ініціали)  
Рецензент Заблоцький В.Ю\_\_\_\_\_\_\_\_  
                                  (прізвище та ініціали)

Засвідчую, що у цій науково-проектній роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис)

Луцьк – 2019

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Луцький НТУ\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
(повне найменування вищого навчального закладу)

Інститут, факультет, відділення технологічний**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Кафедра, циклова комісія автоматизації та комп’ютерно-інтегрованих технологій\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Освітньо-кваліфікаційний рівень магістра\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Спеціальність151Автоматизація та комп’ютерно-інтегровані технології**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**   
                                                                                                             (шифр і назва)

**ЗАТВЕРДЖУЮ**  
**Завідувач кафедри, голова циклової**  
**комісії \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  
**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  
"\_\_\_" \_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_ року

ЗАВДАННЯ  
НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Коблинському Владиславу Михайловичу \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
(прізвище, ім'я, по батькові)

1. Тема роботи «Автоматизована система персоналізації та ведення обліку персоналу з використанням системи розпізнавання зображень» **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  
керівник роботи **\_** Лотиш Володимир В’ячеславович, доцент, кандидат технічних наук**\_\_\_\_\_\_\_\_\_\_**,  
                                   (прізвище, ім'я, по батькові, науковий ступінь, вчене звання)  
затверджені наказом вищого навчального закладу від "29" жовтня 2019 року N 523-05-35

2. Строк подання студентом роботи \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3. Вихідні дані до роботи\_програмне забезпечення для ведення обліку персоналу\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1.Огляд технології розпізнавання 2.Вибір бази даних 3.Обгрунтування вибору мови програмування 4.Аналіз існуючих методів розпізнавання осіб 5.Бібліотеки для розпізнавання 6.Розробка програмного забезпечення для розпізнавання і запису даних в БД 7.Тестування створеного програмного забезпечення розпізнавання 8.Техніко-економічне обгрунтування

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)  
1.Схема взаємодії програмного комплексу – формат А1. 2.Схема ПЗ збереження та розпізнавання зображень – формат А1. 3.Схема реалізації графічного інтерфейсу формування звіту – формат А1. 4.Графіки тестування роботи ПЗ – формат А1

6. Консультанти розділів роботи

|  |  |  |  |
| --- | --- | --- | --- |
| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
| завдання видав | завдання прийняв |
| Техніко-економічне | Лотиш В.В. |  |  |
| обгрунтування |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

7. Дата видачі завдання \_\_\_\_\_\_\_\_\_**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| N з/п | Назва етапів випускної кваліфікаційної роботи | Строк виконання етапів випускної кваліфікаційної роботи | Примітка |
| 1 | Огляд технології розпізнавання | 20.09.2019 |  |
| 2 | Вибір бази даних | 15.10.2019 |  |
| 3 | Обгрунтування вибору мови програмування | 24.10.2019 |  |
| 4 | Аналіз існуючих методів розпізнавання осіб | 2.11.2019 |  |
| 5 | Бібліотеки для розпізнавання | 14.11.2019 |  |
| 6 | Розробка програмного забезпечення для розпізнавання і запису даних в БД | 26.11.2019 |  |
| 7 | Тестування створеного програмного забезпечення розпізнавання | 28.11.2019 |  |
| 8 | Техніко-економічне обгрунтування | 10.12.2019 |  |
|  |  |  |  |

|  |  |  |
| --- | --- | --- |
| **Студент** | **\_\_\_\_\_\_\_\_\_** (підпис) | **Коблинський В.М.\_\_\_\_\_\_\_\_\_\_\_\_\_\_** (прізвище та ініціали) |
| **Керівник роботи** | **\_\_\_\_\_\_\_\_\_** (підпис) | **Лотиш В.В.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** (прізвище та ініціали) |

**АНОТАЦІЯ**

Завдання розпізнавання осіб є складним по наступних причинах: обличчя людини - це динамічний об'єкт, який має високу ступінь змінності в зовнішньому вигляді (наприклад, за формою і кольором шкіри); різні умови освітленості, що визначаються типом, напрямком і кількістю джерел світла; часткове перекриття осіб іншими об'єктами сцени; необхідність локалізації і розпізнавання 'осіб, що мають довільні положення в просторі. Однак існуючі системи локалізації і розпізнавання осіб не завжди враховують дані особливості, що не дозволяє досягти прийнятного рівня розпізнавання на зображеннях і відео.

Метою даної роботи є розробка програмного забезпечення ведення обліку персоналу із застосуванням сучасних технологій розпізнавання зображень.

Об’єктом дослідження є алгоритми, методи та сучасні технології комп’ютерного розпізнавання образів.

Предметом досліджень є методи та моделі, які застосовуються в процесі обробки зображень та в комп’ютерній системі розпізнавання образів.

Практична новизна. Запропоновані в даній роботі методи і алгоритми призначені для практичного застосування в системах відеоспостереження і контролю доступу. На основі досліджень та сучасних технологій розроблено програмне забезпечення для створення систем обробки, локалізації і розпізнаванню зображень обличь.

Об’єм графічної частини магістерської роботи складає 4 листи формату А1(841х594 мм). Обсяг пояснювальної записки становить 60 друкованих сторінок.

Магістерська робота містить 18 рисунків, 11 таблиць. Використано 16 літературних джерел та інформація з глобальної мережі Інтернет.

Ключові слова: програмне забезпечення, система розпізнавання, база даних, алгоритм.

**ABSTRACT**

The person recognition task is complicated for the following reasons: human face is a dynamical object with a high appearance changeability degree (for instance, by shape and skin color); different lighting conditions determined by type, direction and quantity of light sources; partial overlapping of persons with other objects in the scene; necessity to localize and recognize persons randomly positioned in space. Nevertheless, the existing person localization and recognition systems do not always consider these peculiarities making it impossible to reach the acceptable recognition on images and video.

The purpose of this research paper is to develop software for keeping records of personnel using modern image recognition technologies.

The research subject is algorithms, methods and modern technologies of computer image recognition.

The research scope is methods and models used in the image processing and computer image recognition system.

Practical novelty. Methods and algorithms proposed in this research paper are intended for practical application in the video surveillance and access control systems. Based on researches and modern technology, software for setting up the facial image processing, localization and recognition systems has been developed.

The volume of the graphic part of the master's work is 4 sheets of A1 format

(841x594 mm). The explanatory note is 60 printed pages.

The master's thesis contains 18 drawings, 11 tables. Used 16

literary sources and information from the Internet.

Keywords: software, recognition system, database, algorithm.

ЗМІСТ

[ВСТУП 7](#_Toc26872495)

[1 ОГЛЯД ТЕХНОЛОГІЇ РОЗПІЗНАВАННЯ 9](#_Toc26872496)

[1.1 Введення 9](#_Toc26872497)

[1.2 Методи розпізнавання осіб 10](#_Toc26872498)

[1.3 Області застосування технології 11](#_Toc26872499)

[1.3.1 Контроль доступу до об'єктів або систем 11](#_Toc26872500)

[1.3.2 Виявлення порушників 12](#_Toc26872501)

[1.3.3 Визначення портрета покупця 12](#_Toc26872502)

[1.3.4 Ідентифікація в банківському секторі 13](#_Toc26872503)

[1.3.5 Управління робочим часом 13](#_Toc26872504)

[1.3.6 Оплата послуг 14](#_Toc26872505)

[1.3.7 Розумне місто 14](#_Toc26872506)

[1.3.8 Доступ до іспитів 15](#_Toc26872507)

[1.4 Популяризація технології 15](#_Toc26872508)

[Висновок 16](#_Toc26872509)

[2 ВИБІР БАЗИ ДАНИХ 17](#_Toc26872510)

[2.1 Огляд існуючих популярних БД 17](#_Toc26872511)

[2.1.1 MySQL 17](#_Toc26872512)

[2.1.2 PostgreSQL 18](#_Toc26872513)

[2.1.3 Mongo DB 19](#_Toc26872514)

[2.1.4 SQLite 20](#_Toc26872515)

[3 ОБГРУНТУВАННЯ ВИБОРУ МОВИ ПРОГРАМУВАННЯ 23](#_Toc26872516)

[3.1 Огляд мови програмування Python 23](#_Toc26872517)

[3.1.1 Основні інструменти і можливості 23](#_Toc26872518)

[3.1.2 Вирази 25](#_Toc26872519)

[3.1.3 Типи даних 26](#_Toc26872520)

[4 АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ РОЗПІЗНАВАНН ОСІБ 27](#_Toc26872521)

[4.1 Метод гнучкого порівняння на графах (Elastic graph matching) 27](#_Toc26872522)

[4.2 Нейронні мережі 28](#_Toc26872523)

[4.3 Приховані Марковські моделі (СММ, HMM) 29](#_Toc26872524)

[4.4 Метод головних компонент або principal component analysis (PCA) 30](#_Toc26872525)

[5 БІБЛІОТЕКИ ДЛЯ РОЗПІЗНАВАННЯ 32](#_Toc26872526)

[5.1 Алгоритм розпізнавання 32](#_Toc26872527)

[5.2 Огляд бібліотеки OpenCV 34](#_Toc26872528)

[5.3 Алгоритми розпізнавання об’єктів haarcacade і local binary pattern 35](#_Toc26872529)

[5.4 Бібліотека для розпізнавання face\_recognition 39](#_Toc26872530)

[6 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ РОЗПІЗНАВАННЯ І ЗАПИСУ ДАНИХ В БД 40](#_Toc26872531)

[6.1 Алгоритм розробки та роботи програмного забезпечення 40](#_Toc26872532)

[6.2 Створення структури БД 41](#_Toc26872533)

[6.3 Створення зображень для подальшого «навчання» системи розпізнавання 42](#_Toc26872534)

[6.4 «Навчання» системи розпізнавання 42](#_Toc26872535)

[6.5 Програмне забезпечення розпізнавання і запису даних в базу 43](#_Toc26872536)

[6.7 Створення графічного інтерфейсу 46](#_Toc26872537)

[6.8 Компіляція ПЗ 49](#_Toc26872538)

[7 ТЕСТУВАННЯ СТВОРЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ РОЗПІЗНАВАННЯ 51](#_Toc26872539)

[7.1 Тестування програмного забезпечення 51](#_Toc26872540)

[7.2 Залежність розпізнавання від контрастності 51](#_Toc26872541)

[7.3 Залежність розпізнавання від яскравості 53](#_Toc26872542)

[7.4 Залежність розпізнавання від кількості зображень 54](#_Toc26872543)

[8 ТЕХНІКО-ЕКОНОМІЧНЕ ОБГРУНТУВАННЯ 56](#_Toc26872544)

[8.1 Розрахунок капітальних затрат на розробку програмного продукту 56](#_Toc26872545)

[8.2 Розрахунок часу на створення програмного продукту 56](#_Toc26872546)

[8.3 Розрахунок кошторису капітальних витрат на розробку програмного продукту 61](#_Toc26872547)

[8.4 Розрахунок виробничої площі робочих місць програмістів та вартість основних фондів 61](#_Toc26872548)

[8.5 Визначення розміру заробітної плати розробників проекту 62](#_Toc26872549)

[8.6 Розрахунок витрат на електроенергію 63](#_Toc26872550)

[8.7 Розрахунок собівартості програмного продукту 64](#_Toc26872551)

[ЗАГАЛЬНІ ВИСНОВКИ 66](#_Toc26872552)

[СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ 67](#_Toc26872553)

[ДОДАТКИ 69](#_Toc26872554)

**ВСТУП**

Сучасному світу знайомі інновації, спрямовані на зниження рівня злочинності. Вченим вдалося знайти рішення для складної математичної задачі - комп'ютерного розпізнавання осіб. Тепер такі технології є в різних країнах, адже з їх допомогою можна вирішити безліч проблем в різних сферах, в тому числі в сфері охорони, криміналістики, фейс-контролю. Ця тема близька до однієї з цілей сталого розвитку ООН: індустріалізація, інновації та інфраструктура. Як працюють системи розпізнавання осіб, які вони бувають, де і для чого використовуються? І чи можна обійти систему?

Як це працює

Технологія розпізнавання осіб працює за схожим принципом з людським мозком. Адже ми спочатку бачимо зображення, в даному випадку людини, звертаємо увагу на риси його обличчя і обробляємо їх у себе в голові. Так само і з технологіями: система повинна шукати обличчя на зображенні і виділяти потрібну область.

Для цього використовуються різні алгоритми. Іноді система визначає схожість пропорцій, виділяє контури на зображенні і зіставляє їх з контурами осіб або виділяє симетрії за допомогою нейромереж. Фахівці вважають метод Віоли - Джонса найбільш ефективним, адже його використовують в режимі реального часу. Така система може розпізнати обличчя людини навіть при повороті на 30 градусів. Даний метод працює на ознаках Хаара: він являє собою чорно-білі прямокутні маски різної форми, які накладаються на зображення. Тоді алгоритм складає яскравості пікселів знімка, які знаходяться під пікселями маски, і розраховує різницю значень. Система порівнює результати з завантаженими даними і продовжує відстежувати особу на зображенні (при цьому підбираючи потрібний ракурс і регулюючи якість знімка).

Коли система вибрала вдалий знімок, вона починає розпізнавати обличчя і порівнювати його зі своєю базою. Програма шукає опорні точки на обличчі, з яких складаються риси обличчя. Вона виділяє близько 100 таких точок (найважливіші з них - відстань між очима, ширина ніздрів, довжина носа, форма скул, чола, підборіддя).

У громадських місцях зазвичай використовуються 3D-зображення, так як вони дають можливість краще розпізнавати об'єкт. Кілька камер фотографують зображення з різних ракурсів, на основі чого створюється його тривимірна модель. На ній система визначає контрольні точки і порівнює їх зі своєю базою даних. Якщо параметри збігаються, система ідентифікує людину.

Але 3D-моделями сучасні компанії вже не здивувати. Ось компанія Identix, наприклад, створила високоточну біометричну технологію. Вона аналізує текстуру шкіри людини (лінії, шрами, пори). Самі розробники повідомляють, що це збільшує точність роботи системи на 25%.

Розробники стартапу NTechLab вважають, що традиційне розпізнавання осіб не завжди безпечно, тому вони працюють над технологією Life Detection. Вона може визначити, чи живий чоловік знаходиться перед камерою. Для цього його можуть попросити посміхнутися, моргнути або щось сказати. Таку систему буде набагато складніше обдурити маскою або надрукованою фотографією.

**1 ОГЛЯД ТЕХНОЛОГІЇ РОЗПІЗНАВАННЯ**

* 1. **Введення**

Сьогодні розпізнавання особи - це зручна і практична функція ідентифікації без пароля. Сама технологія відноситься до області застосування теорії розпізнавання образів, яка виникла значно раніше сучасних комп'ютерних систем. Розпізнавання образів - невід'ємна частина діяльності мозку. Тому в спектрі комп'ютерних дисциплін завдання розпізнавання відносяться до проблематики штучного інтелекту.

Алгоритм роботи технології розпізнавання осіб складається з двох етапів: ідентифікація (хто ця людина?) і верифікація (а чи це та людина, за яку він себе видає?). Послідовність дій зазвичай така:

*Виявлення обличчя*

Виділяється обличчя людини на зображенні.

*Виявлення рис обличчя*

Обчислюються антропометричні точки. Система знаходить опорні точки на обличчі, які визначають індивідуальні характеристики. Алгоритм обчислення характеристик різний для кожної з систем і є головним секретом розробників. Раніше основною опорною точкою для алгоритмів були очі, але алгоритми еволюціонували і стали враховувати мінімум 68 точок на обличчі (розташовані по контуру особи, визначають положення і форму підборіддя, очей, носа і рота, відстань між ними).

*Нормалізація обличчя*

Проводяться додаткові перетворення зображення (усунення нахилу голови, корекція кольору обличчя і так далі) з метою отримання чіткого фронтального знімка.

*Виділення функцій та обчислення дескриптора*

Обчислюється дескриптор - набір характеристик, що описують особу незалежно від сторонніх чинників (вік, зачіска, макіяж). Аналізуються спеціальні локальні ознаки, що характеризують, наприклад, текстуру певних областей на обличчі. Зіставлення різних дескрипторів дозволяє оцінити, чи належать два отриманих зображення особи той самій людині.

*Перевірка*

Порівнюється отриманий вектор особи (цифровий шаблон) з наявними в базі особами.

* 1. **Методи розпізнавання осіб**

Методи вилучення ознак умовно діляться на дві групи: використовують локальні і глобальні ознаки особи. При використанні локальних - алгоритм виділяє окремі частини (очі, ніс, рот та ін.) І вже по ним розпізнає обличчя. При використанні глобальних - оперує з особою в цілому. Кількість існуючих методів виділення ознак і їх класифікації велика, але одні і ті ж методи використовуються для виділення як локальних, так і глобальних ознак.

Один з найпоширеніших методів – метод нейромереж. Використання нейронних мереж на відміну від «класичної» математики робить розпізнавання менш залежним від кооперативної поведінки людей, знижує вимоги до монтажу камер, покращує якість розпізнавання в широкому діапазоні зовнішніх умов, включаючи зміну освітленості.

Зображення особи з камери відправляється в систему, де обробляється біометричних алгоритмом. Спочатку картинка переводиться в масив точок, потім він перетворюється в масив чисел, які обробляються математичними алгоритмами. На виході ми отримуємо біометричну модель - файл з певною послідовністю чисел.

Відновити вихідну фотографію людини з такого файлу не можна. Система просто кожен раз однаково перетворює картинку і порівнює отримані цифри. При цьому біометрична модель відрізняється від звичайного фотозображення: вона працює з відео потоками (закодована послідовність кадрів) і знеособленою інформацією[13].

Технологія, заснована на надточних нейронних мережах, не вимагає дорогого устаткування і легко вбудовується в існуючі бізнес-процеси. Розпізнавання відбувається шляхом обчислення відстані між дескрипторами осіб. Високоякісні дескриптори повинні показувати малу відстань між особами однієї людини і велику відстань між особами різних людей. Дескриптори можуть бути отримані за допомогою спеціальних алгоритмів, що перетворюють вхідні зображення в вектор значень.

* 1. **Області застосування технології**

Оскільки технологія розпізнавання осіб вже дійшла до такого рівня готовності, що її можна використовувати в комерційних проектах, багато компаній впроваджують подібні платформи - наприклад, всім відомий 3D-сканер особи Face ID від Apple вже використовується на сотнях мільйонів пристроїв. На сьогоднішній день аналогічні технології активно використовуються в системах комп'ютерного зору і відео аналітики.

Серед найпоширеніших сфер застосування можна виділити наступні:

**1.3.1 Контроль доступу до об'єктів або систем**

За даними агентства J'son & Partners, з 2016 року частка систем контролю доступу з розпізнаванням осіб зросла з 0,7% до 11%. Розпізнавання осіб використовується для розмежування доступу в технологічні зони, особливо у випадках, коли співробітники передають свої пропуски в тимчасове користування. Сучасні системи розпізнавання осіб оснащені технологією захисту від підміни осіб, яка автоматично відрізняє реальне обличчя від фотографії. Це дає можливість відстежувати випадки підміни власного обличчя чужою фотографією або проходу співробітника, який несе фото відсутнього колеги.

У ЖКГ-сфері технологія забезпечує доступ мешканців багатоквартирного будинку в під'їзд або приміщення. Для освітніх установ застосовується інтеграція з контрольно-пропускною системою, використання особи як додаткового ідентифікатора для контролю учнів, інтеграція з рейтинговою системою і картами студента і викладача.

**1.3.2 Виявлення порушників**

МВС тестує камери-відео реєстратори з функцією розпізнавання осіб. Якщо випробування будуть успішними, функція розпізнавання з'явиться у портативних відео реєстраторів, які вже застосовують поліцейські. Камера за розміром трохи менше рації і кріпиться на одяг людини. Вона аналізує відео потік і виділяє з нього обличчя людей, дані про яких через інтернет (в пристрій можна вставити сім-карту) відправляє на сервер. У деяких країнах, наприклад, в США і в Китаї, в розпорядженні поліції є не тільки такі пристрої, але і розумні окуляри.

Для багатьох організацій, зокрема, актуальне завдання - виявити людину з «чорного списку» (раніше вчинила крадіжку чи підозрюваного в незаконних діях).

Система ситуаційного відеоспостереження покриває всю територію торгового залу. У вхідних зонах магазинів торгової мережі працює система розпізнавання осіб. При виявленні людини, яка вчинила правопорушення в залі, оператори знаходять фото порушника в протоколі системи розпізнавання осіб і заносять його в "чорний список". Списки порушників регулярно синхронізуються, і оновлення доступні для всіх магазинів мережі. Таким чином система розпізнавання осіб зафіксує появу порушника в будь-якій торговельній точці мережі і сповістить співробітника охорони. Паралельно здійснюється пошук людей по будь-яким зовнішнім списками, наприклад, по базі розшуку зниклих людей[13].

Спеціалізовані камери в повністю автоматичному режимі виділяють, фотографують і зберігають в архіві відвідувачів магазину, після чого зображення порівнюються з еталонною базою даних, яка зберігається у замовника. При виявленні людини з «чорного списку» охоронець протягом 1-2 секунд отримує сигнал тривоги.

**1.3.3 Визначення портрета покупця**

Мета системи в цьому сегменті - проаналізувати поведінку покупця (наприклад, на касі) і підвищити лояльність до мережі. Такі системи вже використовує 59% рітейлерів одягу у Великобританії. У 2017 році Walmart розробив технологію, яка допомагає визначити, наскільки покупець задоволений відвідуванням магазину. Якщо система виявить покупця з нещасним обличчям, вона подає сигнал про це співробітникам магазину.

**1.3.4 Ідентифікація в банківському секторі**

Банки почали збір біометричних даних клієнтів. Пройшовши ідентифікацію в системі, клієнт може за допомогою голосу і фотозображення віддалено відкрити рахунок або замовити випуск карти на порталі держпослуг.

Зразок зображення особи кожного клієнта записується під час першого візиту в банк, і в подальшому цей зліпок використовується для його ідентифікації при кожному зверненні (крім пред'явлення паспорта). Це дає 100% захист в разі, якщо шахрай спробує з підробленими або загубленим паспортом отримати доступ до банківського рахунку і операцій. Також за допомогою селфі клієнти в мобільному додатку можуть самостійно підтверджувати підозрілі операції (наприклад, нетиповий переказ коштів в інший банк або іншому клієнту, закриття вкладу раніше терміну).

Використовувана в банку платформа біометричної ідентифікації по зображенню побудована на алгоритмах нейромереж і використовує більше 200 «маркерів», які визначаються на обличчі. Це дозволяє досягти мінімального рівня помилок - один випадок на мільярд. При цьому персональні дані клієнтів надійно захищені, зберігаються виключно в банку і нікому не передаються.

**1.3.5 Управління робочим часом**

Інтерес у роботодавців до системи обліку робочого часу співробітників зростає з кожним роком. Здійснення контролю відбувається за допомогою програми, вбудованої в планшет при вході в офіс або інше місце роботи.

Камера планшета фіксує час приходу і відходу співробітника, вносить інформацію в систему, яка при запиті формує звіт про присутність кожного із співробітників на робочому місці. Такі системи особливо затребувані у тих роботодавців, які мають багато найманого персоналу, що працює позмінно і з гнучким графіком. Система дозволяє спростити всю паперову роботу і контроль, надаючи достовірну інформацію за відпрацьованим співробітниками часу.

У 2017 році подібний сервіс тестувала мережа салонів «Еконіка». Щоб сервіс розпізнав обличчя прийшов на роботу співробітника, тому було досить подивитися в камеру планшета. Розпізнавання обличчя займало менше секунди, якщо дані передавалися по Wi-Fi, якщо по мобільному інтернету - то 2-3 секунди.

**1.3.6 Оплата послуг**

У березні стартап ORBL представив прототип розробки з одним з європейських провайдерів платіжних систем на виставці eShow в Барселоні. Це кіоск для оплати товарів або послуг за допомогою особи. Для реєстрації необхідно завантажити мобільний додаток, за допомогою якого можна прив'язати банківську карту або електронний гаманець, а також сфотографуватися. При оплаті в кіоску грошові кошти списуються з рахунку покупця. Технологія дозволяє зробити покупку, навіть якщо покупець з яких-небудь причин не може скористатися звичними методами оплати. А також знизити ризик користування карткою третьою особою.

Ще один успішний кейс - використання технології для придбання лотерейних квитків, продаж яких обмежена віковим цензом.

**1.3.7 Розумне місто**

Заходи щодо впровадження систем відеоспостереження з функціями біометричної ідентифікації і відео аналітики включені в базові вимоги стандарту «Розумне місто».

Безпека - базовий, але не єдиний сценарій використання технологій розпізнавання осіб в інфраструктурі міст. Безпека виступає як базис і інфраструктура для інших сервісів - фінансових, транспортних і так далі. Дані в знеособленому форматі можна використовувати для оплати проїзду на транспорті, оптимізації пасажиропотоку, пошуку зниклих людей, безбар'єрності проходу на заходи, в аеропорти, оптимізації руху, міських послуги, ЖКГ.

**1.3.8 Доступ до іспитів**

При використанні навчального порталу студенти інституту отримують доступ до матеріалів курсу та до здачі тестів і іспитів, тоді як раніше це було неможливо через використання системою стандартного механізму доступу по паролю. За даними розробника, близько 5% студентів намагалися використовувати третіх осіб для складання іспитів, але система запобігла всім випадкам шахрайства.

Прокторінг - процедура спостереження і контролю за дистанційним випробуванням, дуже розвинений в США, де прокторінгові компанії працюють в основному з освітніми установами. Рішення верифікують особистість за допомогою біометрії в режимі відеопотоку і аналізують поведінку людини перед монітором. Інструмент буде затребуваний при проведенні атестацій в комерційних компаніях.

* 1. **Популяризація технології**

Популяризації технології сприяє розширенню можливостей використання - на дронах, ноутбуках, в банкоматах і так далі. Число смартфонів, на яких встановлена технологія розпізнавання осіб, неухильно зростає. Згідно з прогнозом агентства Counterpoint Research, в 2020 році буде продано понад мільярд таких пристроїв. В останніх моделях iPhone технологія вже доступна будь-якій людині.

Також експерти говорять про експансію технології в нові ринкові сегменти.

Збір біометричних даних банками і плани по використанню біометричних даних не тільки для надання банківських і державних послуг, а й у сфері охорони здоров'я, освіти, електронний документообіг і так далі. Всерйоз обговорюється ідея заміни всіх існуючих ідентифікаційних документів єдиним електронним паспортом з вбудованим шаблоном особи для розпізнавання. У найближчому майбутньому ми зможемо не тільки відмовитися від паспортів, а й обходитися без готівки і банківських карт, оплачуючи покупки в магазинах за допомогою розпізнавання осіб.

Є три основних параметри, за які ведеться боротьба серед розробників: вартість рішення, точність розпізнавання, швидкодія системи. І тут треба оптимально підібрати всі три складові, щоб продукт був рентабельним, не поступався конкурентам і лідерам. І максимально прагнув увійти в когорту продуктів на ринку систем розпізнавання.

Фактори, які поки гальмують розвиток технологій в Україні недостатня точність, відносно висока ціна впровадження систем розпізнавання осіб, а також виникає синдром «Великого брата», відчуття постійного стеження у користувачів.

**Висновок**

В даному розді було проаналізовано існуючі технології розпізнавання, сфери затосування Комп'ютернеий зір - галузь програмування в процесі розвитку, але при цьому затребувана і має великий спектр застосування. Функцію ідентифікації людей на фотографіях активно використовують в програмному забезпеченні для керування фотоальбомами (Picasa, iPhoto і ін.). Скомбінувавши її з реальними параметрами, можна складати альбоми по окремій людині.

**2 ВИБІР БАЗИ ДАНИХ**

**2.1 Огляд існуючих популярних БД**

Системи управління базами даних здатні виконувати схожі базові завдання, включаючи створення, зміну та доступ до даних / інформації, що зберігається в базах даних. Ці системи відрізняються одна від одної тільки тим, як виконуються ці дії. Функції, підтримка і навіть функціональні можливості, підтримувані кожною з цих систем, можуть відрізнятися. Перше і головне, що вам необхідно враховувати - це зручність використання, що надається СУБД, і ступінь її масштабування, а також можливості інтеграції з тими продуктами, з якими вона буде контактувати.

У міру розширення організації ці механізми управління базами даних також необхідно покращувати і вдосконалювати. Незалежно від розміру вашого бізнесу, навіть якщо це стартап, який вимагає тільки обмежених функціональних можливостей через невеликий обсяг даних, якими він повинен керувати, важливо розглянути, як база даних може бути змінена, щоб включити зміни, які з'являються з плином часу. У деяких ситуаціях навіть перехід на іншу систему управління базами даних буде одним з кращих рішень.

**2.1.1 MySQL**

Всі стикалися або використовували її хоча б один раз, що робить дану систему однією з найпопулярніших баз даних для веб-додатків. Хоча це безкоштовне програмне забезпечення, воно все ще отримує свою частку частих оновлень у вигляді функцій і поліпшень безпеки. Крім використання функцій, пропонованих цієї безкоштовною програмою, ви можете вибрати платну версію, яка поставляється з оновленими функціями, найбільш підходящими для комерційних цілей.

Незважаючи на те, що платна версія може пропонувати більшу кількість функціональних можливостей і функцій, безкоштовна версія забезпечує велику швидкість і надійність, яку ви можете вибрати на основі вимог вашого бізнесу.

Також можна змінити функціональність інструменту і керувати даними з таблиць різних типів, використовуючи движок бази даних, який надає вам масив механізмів зберігання, що пропонують вищезгадані аспекти. Простий у використанні інтерфейс і пакетні команди дозволяють обробляти величезні обсяги даних - ця надійність, пропонована системою.

Переваги:

* Доступна безкоштовна версія
* Ряд функцій доступний навіть для безкоштовного движка бази даних
* Широкий діапазон користувальницьких інтерфейсів, які можна використовувати
* Простота інтеграції з рядом баз даних, включаючи Oracle і DB2
* Добре підходить для організацій, які шукають надійний інструмент ведення бази даних за доступними цінами

Недоліки:

* Немає вбудованої підтримки OLAP або XML
* Хоча підтримка безкоштовної версії доступна, вона не надається безкоштовно
* Зусилля і час, необхідні MySQL для виконання деяких дій, таких як створення інкрементних резервних копій, набагато більше в порівнянні з іншими системами

**2.1.2 PostgreSQL**

Інша популярна база даних - PostgreSQL, і, що цікаво, це одна з перших систем управління базами даних, яка була розроблена. Вона в основному використовується для веб-баз даних, дозволяючи користувачам управляти структурованими і неструктурованими даними. Ви можете використовувати PostgreSQL на різних платформах, таких як Linux, що спрощує імпорт інформації між різними типами баз даних за допомогою цього інструменту. Даний механізм управління базами даних можна розмістити в різних середовищах, таких як віртуальні, фізичні і навіть хмарні. Кількість користувачів останньої версії PostgreSQL, значно збільшилася. Безпекова частина була поліпшена за допомогою DBMS\_SESSION і розширених профілів паролів.

Переваги:

* підтримка JSON
* Наявність численних інтерфейсів
* Механізм управління масштабуємою базою даних
* Можливість обробляти терабайти даних
* Поставляється з безліччю визначених функцій
* Найкраще підходить, якщо ви хочете використовувати JSON і вибрати відповідний інтерфейс з обмеженим бюджетом

Недоліки:

* Частина документації слабка, що означає, що якщо ви застрягли в своєму проекті, може бути нелегко зрозуміти, як з нього вийти
* Швидкість може постраждати під час читання запитів або коли навантаження занадто велике
* Частина конфігурації може бути проблемною

**2.1.3 Mongo DB**

Як і MySQL, Mongo DB - це ще одна база даних, що має як безкоштовну, так і платну версію. Додатки, розроблені з використанням Mongo DB, ґрунтуються або можуть використовувати як структуровані, так і неструктуровані дані. Mongo DB має механізм бази даних, який є гнучким і адаптованим, і працює, підключаючи програми бази даних через драйвери бази даних Mongo DB. Це просте завдання - знайти драйвер, який точно відповідає або може бути добре інтегрований з мовою програмування, оскільки інструмент пропонує повний набір складних драйверів. Не поставляється з новими і змінними механізмами зберігання.

Краще уникати використання Mongo DB для управління реляційною моделлю даних, незважаючи на те, що вона може впоратися з ними, оскільки це може привести до проблем з продуктивністю. Однак движок бази даних був розроблений так, що він може обробляти змінні дані, які не є реляційними і здатні добре працювати в таких ситуаціях, коли інші бази даних можуть працювати гірше.

Система надає вам функції пошуку. Крім того, документи можуть бути перевірені за допомогою оновлень і вставок. Якщо ви зменшите розмір індексів, ви створите нову можливість часткового індексу, яка призведе до підвищення продуктивності.

Переваги:

* Для написання схеми відсутня або потрібно менше часу простою
* Прості і швидко зберігаються дані незалежно від того, структуровані вони або неструктурованих
* Швидко, легко і просто використовувати
* За допомогою цього движка інтеграція з JSON і іншими документами NoSQL простіше

Недоліки:

* Установка займе більше часу в порівнянні з іншими системами
* Незахищеність може з'явитися при настройках за умовчанням
* Хоча існують інструменти для перекладу запитів SQL в Mongo DB, вони ускладнять процес
* Тут мова запитів не є SQL

**2.1.4 SQLite**

SQLite - це бібліотека, що здійснює роботу автономного механізму баз даних SQL, який не потребує сервера. Код, який використовується в SQLite, є громадським надбанням, і тому він може використовуватися в будь-яких цілях - комерційних або приватних. В даний час SQLite виявлена ​​в безлічі додатків, в тому числі і в деяких висококласних проектах.

Весь програмний код, який постачається разом з SQLite, було надано його авторами в громадське користування. Всі автори коду, а також представники компаній, в яких вони працюють, письмово підтвердили свій внесок в суспільне надбання, і оригінали цих паперів зберігаються в головному офісі в Hwaci. Будь-який бажаючий може вільно копіювати, змінювати, видавати, компілювати, продавати або розповсюджувати вихідний код SQLite - або у формі вихідного коду, або відкомпільованого бінарного. Для будь-яких цілей: комерційних і безкоштовних. Або будь-яким іншим чином.

SQLite є впровадженим механізмом бази даних SQL. На відміну від більшості інших баз даних SQL SQLite не має окремого серверного процесу. SQLite читає і пише безпосередньо звичайним файлом на диску. Повна база даних SQL з численними таблицями, індексами, тригерами міститься в єдиному файлі на жорсткому диску. Формат файлу бази даних є міжплатформним. Можна вільно копіювати базу даних між 32- і 64-розрядних системами. Подібні особливості роблять SQLite популярним вибором в якості формату файлів додатків (Application File Format). Про SQLite не варто думати як про заміну Oracle, але як про заміну для fopen ().

SQLite дуже компактна - вона може займати об'єм менше 250 Kb. Ще один аргумент на користь SQLite - її компактність. З усіма включеними настройками, в залежності від параметрів настройки оптимізації компілятора, розмір бібліотеки може становити менше 250 Кб. Якщо опущені додаткові настройки, розмір бібліотеки SQLite може бути скорочений до 180 Кб і трохи менше. Як стверджують розробники, SQLite може працювати при мінімальному стеку близько 16 Кб і дуже невеликиій динамічній пам'яті - 100 Кб. Це дозволяє використовувати SQLite як популярний механізм бази даних в таких пристроях з обмеженнями пам'яті, як мобільні телефони, кишенькові персональні комп'ютери, MP3-програвачі і так далі. В даному випадку має місце співвідношення між використанням пам'яті і швидкістю. Чим більше відводиться пам'яті для SQLite, тим швидше вона працює. Проте робочі характеристики зазвичай і так досить високі навіть в середовищах з низьким обсягом пам'яті.

Як стверджується на офіційному сайті, SQLite має репутацію надійного механізму. Близько двох третин вихідного коду відводиться на тестування і перевірку. Автоматизований тестує набір програм виконує сотні тисяч перевірок, що обробляють мільйони індивідуальних SQL-запитів, досягаючи при цьому більш ніж 99% охоплення. SQLite відповідає витончено на збої в розподілі пам'яті і дискові помилки введення-виведення. Все це перевірено автоматичними тестуваннями при використанні спеціального обладнання, яке імітує збої в роботі системи. Зрозуміло, навіть при таких випробуваннях все ще є певні помилки. Але на відміну від деяких інших подібних проектів SQLite відкрито і чесно повідомляє про свої дефекти в наданому списку помилок, що включає список критичних помилок з щохвилинною хронологією повідомлень про помилки та зміни коду.

Базовий код SQLite підтримується міжнародною групою розробників, які постійно працюють з її застосуванням. Розробники продовжують розширювати можливості SQLite, покращувати її надійність і роботу, поки підтримується зворотна сумісність з виданими специфікаціями інтерфейсу, синтаксисом SQL і форматом файлу бази даних. Вихідний код абсолютно вільний для використання ким завгодно, кому він потрібен. При цьому також доступні і професійні послуги служби підтримки. Розробники сподіваються, що користувачі знайдуть SQLite корисним, і закликають використовувати його добре: створювати хороші і надійні програми, які будуть зручні і прості в застосуванні.

**Висновок**

В даному розділі було проведено аналіз існуючих найбільш популярних на даний момент баз даних. Описано їх основні переваги та недоліки. Для розробки системи ведення обліку персоналу було обрано SQLite базу даних.

**3 ОБГРУНТУВАННЯ ВИБОРУ МОВИ ПРОГРАМУВАННЯ**

**3.1 Огляд мови програмування Python**

Python - це універсальна сучасна мова програмування високого рівня, до переваг якої відносять високу продуктивність програмних рішень і структурований, добре читається код. Синтаксис Python максимально полегшений, що дозволяє вивчити його за порівняно короткий час. Ядро має дуже зручну структуру, а широкий перелік вбудованих бібліотек дозволяє застосовувати значний набір корисних функцій і можливостей. Може використовуватися для написання прикладних програм, а також розробки WEB-сервісів.

Python може підтримувати широкий перелік стилів розробки додатків, в тому числі, дуже зручний для роботи з ООП і функціонального програмування.

Один з найпопулярніших інтерпретаторів мови - CPython, написаний на Сі. Поширюється це середовище розробки безкоштовно по вільної ліцензії. Інтерпретатор підтримує більшість популярних платформ[9].

Python активно розвивається. Приблизно раз в 2 роки виходять оновлення. Важливою особливістю мови є відсутність таких стандартів кодування як ANSI, ISO і деяких інших, вони працюють завдяки інтерпретатору.

**3.1.1 Основні інструменти і можливості**

Python - не сама «молода» мова програмування, але і не занадто стара. До моменту її створення вже існували такі «монстри», як Паскаль або Сі. А тому при створенні МП автори намагалися взяти найкраще з різних платформ для розробників. Фактично Python вдає із себе своєрідний «джем» вдалих рішень більш ніж з 8 різних мов. Наприклад, байт компіляція з'явилася ще до створення Python, але була дуже вдала в нього інтегрована.

Python підтримує практично всі поширені операційні системи. Він може прекрасно працювати на кишенькових комп'ютерах, так і на великих серверах. У разі, якщо платформа значно застаріває, вона виключається з підтримки ядра. Наприклад, версії мови, починаючи від 2.6, вже не працюють з платформами Windows 95, 98 і ME. У разі необхідності можна скористатися більш старими версіями, відмовившись від застосування сучасних інструментів мови. І тоді додаток буде працювати в тому числі з цими ОС. Для старих версій періодично виходять патчі. Мова також може підтримувати роботу з віртуальною машиною Java[9].

МП має чітко структуроване семантичне ядро ​​і досить простий синтаксис. Все, що пишеться на цій мові, завжди легко читається. У разі необхідності передати аргументи мова використовує функцію call-by-sharing.

Набір операторів в мові цілком стандартний. Зручна особливість синтаксису - це форматування тексту коду за допомогою розбивки їх на блоки за допомогою відступів, які створюють натисканням клавіш «Space» і «Tab». У синтаксисі відсутні фігурні або операторні дужки, що позначають початок і кінець блоку. Таке рішення помітно скорочує кількість рядків тіла програми і привчає програміста дотримуватися хороший стиль і акуратність при написанні коду.

Іноді Python порівнюють з такими популярними платформами як Ruby, але на відміну від нього, Python вимагає менше оперативної пам'яті, швидше взаємодіє з процесором.

Короткий перелік можливостей:

* Будь описаний клас одноразово вдає із себе і об'єкт.
* Функція множинного спадкоємства.
* Підтримка віртуальних функцій.
* Можливість легко керувати іменами приховувати їх особливими мітками.
* Можливість життям об'єкта і розподіл пам'яті.
* Управління роботи операторів як символьних, так і логічних.
* Можливість імітувати поле
* Управління полями - як прямий, так і частковий доступ.
* Контроль над найпоширенішими операціями. Від глибокого до ітерації по об'єкту.
* Можливо створювати тригери і класи.

Python відноситься до найбільш затребуваних і популярних МП, про що свідчать численні рейтинги і аналіз пропозицій на ринку розробки програмних продуктів. Він досить простий, а тому вивчення мови не займе занадто багато часу.

У 2018 році в Python були змінені деякі ключові терміни, але це скоріше спростило розуміння. А тому проблем у розробників при вивченні документації не виникає.

**3.1.2 Вирази**

Вираз в Python - це рядок, що містить набір операторів (команд) і операндів (змінних і констант), що виконує певну функцію.

Операнди - це змінні, а також їх значення.

Вирази - цілі рядки, які можуть містити один або кілька операторів.

Для форматування рядків можна використовувати «зворотний слеш» (\).

Завдяки структурі «драбинкою» мова дуже зручна для проведення ланцюгових порівнянь.

Логічні оператори дуже прості з точки зору синтаксису, а тому вважаються «ледачими».

Зручно реалізовано управління списками та іншими послідовностями. Є індексація елемента в списку.

Головний плюс: рядки виразів складаються з простих операторів, більшість з яких зрозумілі в тому числі розробникам, незнайомим з Python, так як синтаксис схожий з іншими мовами і зрозумілий інтуїтивно. При цьому вирази формуються без зайвих службових дужок та інших операторів типу «початок» і «кінець», що робить код простіше для сприйняття[9].

**3.1.3 Типи даних**

Синтаксис Python має на увазі обов'язкове визначення типу даних для змінних, констант, масивів, списків і т.д. Основні типи нічим не відрізняються від інших мов з жорстко заданою типізацією.

Найважливіші типи:

Числові: цілі, дробові, речові з плаваючою точкою, комплексні.

Логічні: тип для зберігання значень алгебри логіки - «істина» або «брехня».

Строкові: містять символи Юнікоду, в тому числі, html-код.

Списки - впорядковані масиви змінних.

Кортежі - масив упорядкованих констант, тобто значень, які не можуть змінюватися в процесі роботи.

Безлічі - масиви невпорядкованих даних.

Словники - спеціалізований масив, що складається з пари - «ключ» - «значення».

Байти, масиви байтів - зазначені області пам'яті для зберігання зображень (jpg, gif і т.д.), pdf-документів та інших файлів.

Тут об'єктами вважаються практично всі конструкції. А тому власні типи мають, в тому числі, класи, файли або методи[9].

**Висновок**

Тому для розробки ПЗ було обрано мову програмування Python. Продуктивність платформи вельми висока, код відрізняється простою і читабельністю.

**4 АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ РОЗПІЗНАВАНН ОСІБ**

**4.1** **Метод гнучкого порівняння на графах (Elastic graph matching)**

Суть методу зводиться до еластичного порівняння графів, що описують зображення осіб. Особи представлені у вигляді графів зі зваженими вершинами і ребрами. На етапі розпізнавання один з графів - еталонний - залишається незмінним, в той час як інший деформується з метою найкращої підгонки до першого. У подібних системах розпізнавання графи можуть являти собою як прямокутну решітку, так і структуру, утворену характерними (антропометричними) точками особи.

У вершинах графа обчислюються значення ознак, найчастіше використовують комплексні значення фільтрів Габора або їх упорядкованих наборів - Габорівських вейвлет (строї Габора), які обчислюються в деякій локальній області вершини графа локально шляхом згортки значень яскравості пікселів з фільтрами Габора.

Ребра графа зважуються відстанями між суміжними вершинами. Різниця (відстань, дискримінаційна характеристика) між двома графами обчислюється за допомогою деякої цінової функції деформації, що враховує як відмінність між значеннями ознак, обчисленими в вершинах, так і ступінь деформації ребер графа.

Деформація графа відбувається шляхом зсуву кожної з його вершин на деяку відстань в певних напрямках щодо її вихідного розташування і вибору такої її позиції, при якій різниця між значеннями ознак (відгуків фільтрів Габора) в вершині деформованого графа і відповідної їй вершині еталонного графа буде мінімальною. Дана операція виконується по черзі для всіх вершин графа до тих пір, поки не буде досягнуто найменшу сумарну відмінність між ознаками деформованого і еталонного графів. Значення цінової функції деформації при такому положенні, що деформується графа і буде мірою відмінності між вхідним зображенням обличчя і еталонним графом. Дана процедура деформації повинна виконуватися для всіх еталонних осіб, закладених в базу даних системи. Результат розпізнавання системи - еталон з найкращим значенням цінової функції деформації.

В окремих публікаціях вказується 95-97% -а ефективність розпізнавання навіть при наявності різних емоцій і зміні ракурсу особи до 15 градусів. Однак розробники систем еластичного порівняння на графах посилаються на високу обчислювальну вартість даного підходу. Наприклад, для порівняння вхідного зображення особи з 87 еталонними витрачалося приблизно 25 секунд при роботі на паралельній ЕОМ з 23 транспьютерами.

Недоліки: висока обчислювальна складність процедури розпізнавання. Низька технологічність при запам'ятовуванні нових еталонів. Лінійна залежність часу роботи від розміру бази даних осіб.

**4.2 Нейронні мережі**

В даний час існує близько десятка різновидів нейронних мереж (НС). Одним з найбільш широко використовуваних варіантів є мережа, побудована на багатошаровому перцептроні, яка дозволяє класифікувати подане на вхід зображення / сигнал відповідно до попередніх налаштувань / навчанні мережі.

Навчаються нейронні мережі на наборі навчальних прикладів. Суть навчання зводиться до налаштування ваг міжнейронних зв'язків в процесі рішення оптимізаційної задачі методом градієнтного спуску. В процесі навчання НС відбувається автоматичне вилучення ключових ознак, визначення їх важливості та побудова взаємозв'язків між ними. Передбачається, що навчена НС зможе застосувати досвід, отриманий в процесі навчання, на невідомі образи за рахунок узагальнюючих здібностей.

Найкращі результати в області розпізнавання осіб (за результатами аналізу публікацій) показала Convolutional Neural Network або згорткова нейронна мережа (далі - ЗНР), яка є логічним розвитком ідей таких архітектур НС як когнітрону і неокогнітрона. Успіх обумовлений можливістю обліку двовимірної топології зображення, на відміну від багатошарового перцептрона.

Відмінними рисами ЗНС є локальні рецепторні поля (забезпечують локальну двовимірну зв'язність нейронів), загальні ваги (забезпечують детектування деяких рис в будь-якому місці зображення) і ієрархічна організація з просторовими семплінгом (spatial subsampling). Завдяки цим нововведенням ЗНС забезпечує часткову стійкість до змін масштабу, зсувів, поворотів, зміні ракурсу та інших спотворень.

Тестування ЗНС на базі даних ORL, що містить зображення осіб з невеликими змінами освітлення, масштабу, просторових поворотів, положення і різними емоціями, показало 96% точність розпізнавання.

Свій розвиток ЗНР отримали в розробці DeepFace , яку придбав

Facebook для розпізнавання осіб користувачів своєї соцмережі. Всі особливості архітектури носять закритий характер.

Недоліки нейронних мереж: додавання нової еталонної особи в базу даних вимагає повного перенавчання мережі на всьому наявному наборі (досить тривала процедура, в залежності від розміру вибірки від 1 години до декількох днів). Проблеми математичного характеру, пов'язані з навчанням: потрапляння в локальний оптимум, вибір оптимального кроку оптимізації, перенавчання і т. д. Важко формалізується етап вибору архітектури мережі (кількість нейронів, шарів, характер зв'язків). Узагальнюючи все вищесказане, можна зробити висновок, що НС - «чорний ящик» з результатами роботи.

**4.3 Приховані Марковські моделі (СММ, HMM)**

Одним зі статистичних методів розпізнавання осіб є приховані Марковські моделі (СММ) з дискретним часом. СММ використовують статистичні властивості сигналів і враховують безпосередньо їх просторові характеристики. Елементами моделі є: безліч прихованих станів, безліч спостережуваних станів, матриця перехідних ймовірностей, початкова ймовірність станів. Кожному відповідає своя Марковська модель. При розпізнаванні об'єкта перевіряються згенеровані для заданої бази об'єктів Марковські моделі і шукається максимальна із спостережуваних ймовірність того, що послідовність спостережень для даного об'єкта згенерована відповідною моделлю.

На сьогоднішній день не вдалося знайти приклад комерційного застосування СММ для розпізнавання осіб.

Недоліки:

- необхідно підбирати параметри моделі для кожної бази даних;

- СММ не володіє розрізняє здатністю, тобто алгоритм навчання тільки максимізує відгук кожного зображення на свою модель, але не мінімізує відгук на інші моделі.

**4.4 Метод головних компонент або principal component analysis (PCA)**

Одним з найбільш відомих і опрацьованих є метод головних компонент (principal component analysis, PCA), заснований на перетворенні Карунена-Лоєва.

Спочатку метод головних компонент почав застосовуватися в статистиці для зниження простору ознак без істотної втрати інформації. У задачі розпізнавання осіб його застосовують головним чином для представлення зображення особи вектором малої розмірності (головних компонент), який порівнюється потім з еталонними векторами, закладеними в базу даних.

Головною метою методу головних компонент є значне зменшення розмірності простору ознак таким чином, щоб воно якомога краще описувало «типові» образи, що належать безлічі осіб. Використовуючи цей метод можна виявити різні мінливості в навчальній вибірці зображень обличь і описати цю мінливість в базисі декількох ортогональних векторів, які називаються власними (eigenface).

Отриманий один раз на навчальній вибірці зображень обличь набір власних векторів використовується для кодування всіх інших зображень осіб, які представляються зваженою комбінацією цих власних векторів. Використовуючи обмежену кількість власних векторів можна отримати стислу апроксимацію вхідному зображенню особи, яку потім можна зберігати в базі даних у вигляді вектора коефіцієнтів, службовця одночасно ключем пошуку в базі даних осіб.

Суть методу головних компонент зводиться до наступного. Спочатку весь навчальний набір осіб перетвориться в одну загальну матрицю даних, де кожен рядок являє собою один екземпляр зображення особи, розкладеного в рядок. Всі особи навчального набору повинні бути приведені до одного розміру і з нормованими гістограмами.

Потім проводиться нормування даних і приведення рядків до 0-му середньому і 1-й дисперсії, обчислюється матриця коваріації. Для отриманої матриці коваріації вирішується завдання визначення власних значень і відповідних їм власних векторів (власні особи). Далі проводиться сортування власних векторів в порядку убування власних значень і залишають тільки перші k векторів.

Метод головних компонент добре зарекомендував себе в практичних додатках. Однак, в тих випадках, коли на зображенні особи присутні значні зміни в освітленості або виразі обличчя, ефективність методу значно падає. Вся справа в тому, що PCA вибирає підпростір з такою метою, щоб максимально апроксимувати вхідний набір даних, а не виконати дискримінацію між класами осіб.

**Висновок**

В даному розділі було проведено короткий огляд існуючих методів розпізнавання обличь та їх основні недоліки.

**5 БІБЛІОТЕКИ ДЛЯ РОЗПІЗНАВАННЯ**

**5.1 Алгоритм розпізнавання**

Для того, щоб зрозуміти, як працює розпізнавання обличчя, давайте спочатку ознайомимось із поняттям функціонального вектора.

Кожен алгоритм машинного навчання приймає набір даних як вхідні дані та вивчає ці дані. Алгоритм проходить через дані та визначає шаблони в даних. Наприклад, припустимо, що ми хочемо визначити, чиє обличчя присутнє в даному зображенні, є кілька речей, які ми можемо розглядати як візерунок:

*Висота / ширина обличчя*.

Висота та ширина можуть бути недостовірними, оскільки зображення можна змінити на менший розмір обличчя. Однак навіть після масштабування залишаються незмінними співвідношення - відношення висоти обличчя до ширини обличчя не зміниться.

*Колір обличчя.*

Ширина інших частин обличчя, таких як губи, ніс тощо.

Зрозуміло, що тут є закономірність - різні грані мають різні розміри, як описані вище. Подібні грані мають подібні розміри. Складною частиною є перетворення конкретного обличчя в числа - Алгоритми машинного навчання розуміють лише числа. Це числове зображення "обличчя" (або елемента в навчальному наборі) називається вектором функції. Функціональний вектор складається з різних чисел у певному порядку[6].

Як простий приклад, ми можемо зіставити "обличчя" у вектор, який може містити різні функції, такі як:

* Висота обличчя (см)
* Ширина обличчя (см)
* Середній колір обличчя (R, G, B)
* Ширина губ (см)
* Висота носа (см)

По суті, даючи зображення, ми можемо відобразити різні функції та перетворити його у векторний елемент табл.5.1.

Таблиця 5.1 – Векторний елемент зображення

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Висота обличчя (см) | Ширина обличчя (см) | Середній колір обличчя (RGB) | Ширина губ (см) | Висота носа (см) |
| 23,1 | 15,8 | (255,224, 189) | 5,2 | 4.4 |

Отже, наше зображення зараз є вектором, який можна було б представити як (23.1, 15.8, 255, 224, 189, 5.2, 4.4). Звичайно, може бути безліч інших особливостей, які можна отримати від зображення (наприклад, колір волосся, волосся на обличчі, окуляри тощо). Однак для прикладу розглянемо саме ці 5 простих функцій.

Тепер, коли ми закодували кожне зображення у векторний вектор, проблема стає набагато простішою. Зрозуміло, що коли ми маємо 2 обличчя (зображення), які представляють одну і ту ж людину, отримані вектори функцій будуть досить схожими. Інакше кажучи, "відстань" між двома функційними векторами буде зовсім невеликою.

Машинне навчання може нам тут допомогти двома речами:

Отримання вектора функцій: важко перерахувати всі функції вручну, оскільки їх так багато. Алгоритм машинного навчання може інтелектуально виділити багато таких функцій. Наприклад, складними ознаками можуть бути: співвідношення висоти носа і ширини лоба. Тепер людині буде досить важко перерахувати всі такі функції "другого порядку".

Алгоритми відповідності: Після того, як отримані вектори функцій, алгоритм машинного навчання повинен співставити нове зображення з набором векторів ознак, присутніх у корпусі.

**5.2 Огляд бібліотеки OpenCV**

OpenCV - це бібліотека комп'ютерного зору з відкритим кодом доступна з http://SourceForge.net/projects/opencvlibrary. Бібліотека написана на С і C ++ і працює під Linux, Windows та Mac OS X. Активна розробка на інтерфейсах для Python, Ruby, Matlab та інших мов.

OpenCV була розроблена для обчислювальної ефективності та з сильним акцентом на додатках у режимі реального часу. OpenCV написана на оптимізованому C і може скористатися багатоядерними процесорами. Якщо Ви бажаєте подальшої автоматичної оптимізації архітектури Intel ви можете придбати інтегровані бібліотеки Intel IPP (IPP) від Intel, які складаються з оптимізованих процедур низького рівня у багатьох різних алгоритмічних областях. OpenCV автоматично використовує відповідну бібліотеку IPP під час виконання, якщо ця бібліотека встановлена[4].

Однією з цілей OpenCV є створення простої у користуванні інфраструктури комп'ютерного зору, що допомагає людям швидко створювати досить складні програми для розпізнавання. Бібліотека містить понад 500 функцій, що охоплюють багато областей зору, включаючи стандартні огляд виробів, медичні знімки, безпека, інтерфейс користувача, калібрування камери, стерео бачення та робототехніка. Оскільки комп'ютерний зір та машинне навчання часто не рухаються вручну, OpenCV також містить повну бібліотеку машинного навчання загального призначення (MLL)[4].

Суббліотека орієнтована на статистичне розпізнавання шаблонів та кластеризацію. MLL є дуже корисний для завдань зору, які лежать в основі місії OpenCV, але він є загальним, щоб використовувати його для будь-якої проблеми машинного навчання.

**5.3 Алгоритми розпізнавання об’єктів haarcacade і local binary pattern**

Комп'ютерна програма, яка вирішує, чи є зображення позитивним (зображення з особами) або негативним (зображення без осіб) називається класифікатором. Класифікатори навчають сотнями тисяч позитивними і негативними зображеннями для того, щоб розпізнавати нові зображення більш коректно. Бібліотека OpenCV надає дві вже натренованих і готових до використання класифікатора - HAAR і LBP.

Виявлення об'єктів з використанням Haar класифікаторів - ефективний метод виявлення об'єктів, запропонований Полом Віолою і Майклом Джонсом в своїй статті «Швидке виявлення об'єктів з використанням розширеного каскаду простих функцій» в 2001 році [5].

Haar-ознаками називають характеристики цифрового зображення, використовувані в розпізнаванні образів. Для розпізнавання образу потрібно із зображення покроково витягувати Haar примітиви. Кожен примітив являє собою прямокутну область з розбивкою на набори різнотипних прямокутних підобластей, причому кожен прямокутник використовується кілька разів різного розміру. Графічно вони представлені на рисунку 5.1.

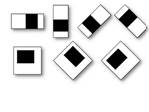


Рисунок 5.1 - Графічне представлення Haar примітивів

Алгоритм має кожен примітив на кожну ділянку зображення для обчислення всіх можливих ознак. Потім вважається значення, отримане шляхом віднімання суми пікселів під білим прямокутником з суми пікселів під чорним прямокутником, що дозволяє виявити певні ознаки[16].

Ознаки підкреслюють структурну інформацію про об'єкт - припустимо, очі будуть темніші, ніж перенісся, так само як і область рота буде темніше чола. Чим більше використовується примітивів при обробці зображення, тим точніше класифікується об'єкт. Виділення ознак графічно зображено на рис. 5.2.

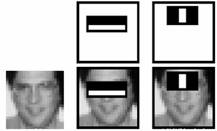


Рисунок 5.2 - Виділення ознак з допомогою Haar примітив

LBP класифікатор - це візуальний / текстурний дескриптор зображень, який так само, як і Haar класифікатор дозволяє розпізнавати позитивні і негативні зображення. Метод LBP дуже простий, і в той же час ефективний. Найважливішою особливістю LBP дескриптора вважають його стійкість до монотонних змін відтінків сірого, викликаного зміною освітленості[6].

Дескриптор зображень Local Binary Pattern був запропонований Тімом Оджалой в 1996 році. Дескриптор є ефективний оператор, який представляє кожен піксель зображення у вигляді двійкового числа, що залежить від інтенсивності сусідніх пікселів зображення. У методі Local Binary Pattern описується для кожного пікселя, в якому напрямку убуває яскравість[16]. Алгоритм роботи дескриптора:

* Потрібно вибрати радіус і кількість точок.
* Пронумерувати вибрані точки
* Необхідно обчислити різницю в яскравості між центральним і сусідніми пікселями. Якщо різниця зменшується, то в цей піксель записується значення 1, якщо позитивна або дорівнює центральному пікселю, то 0
* Витягуємо числа в лінію і кодуємо двійкове число в десяткове. Даний етап зображений на рис. 5.3.

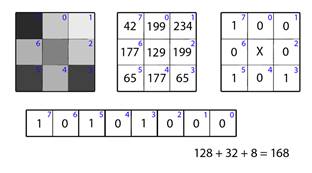


Рисунок 5.3 - Графічний приклад роботи класифікатора LBP

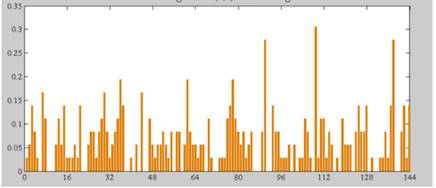
* Конвертуємо кожен такий блок в гістограму. Гістограма формується в один вектор-ознака, в зображенні якого містяться всі ознаки для визначення того чи іншого об'єкта. Сформована гістограма зображена на рис. 5.4 

Рисунок 5.4 - Гістограма зі сформованим вектор-ознакою

Середні і сумарні значення обробки зображень класифікаторами Haar cascade і Local Binary Pattern наведені в таблицях 5.1 і 5.2 відповідно:

Таблиця 5.1 - Класифікатор Haar cascade

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Бази зобр.** | **Всього обличь на зобр.** | **Розпізн. обличь** | **Серед. час розпізнавання, сек** | **Серед.**  **загрузка CPU, %** |
| FDDB | 182 | 126 | 0.145 | 95.49 |
| WIDER | 357 | 242 | 1.044 | 97.05 |
| PICA | 100 | 100 | 1.119 | 96.63 |

Таблиця 5.2 - Класифікатор Local Binary Pattern

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Бази зобр.** | **Всього обличь на зобр.** | **Розпізн. обличь** | **Серед. час розпізнавання, сек** | **Серед.**  **загрузка CPU, %** |
| FDDB | 182 | 117 | 0.064 | 85.77 |
| WIDER | 357 | 185 | 0.318 | 90.01 |
| PICA | 100 | 100 | 0.407 | 90.18 |

За результатами тесту распознаваемость осіб на зображеннях у алгоритму Haar виявилася більшою, ніж у LBP на 10-18%. Середнє співвідношення розпізнаних осіб на зображеннях до загальної кількості осіб графічно зображено на рис. 5.5.

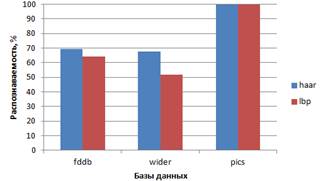


Рисунок 5.5 - Середнє співвідношення розпізнаних осіб на зображеннях до загальної кількості осіб

За отриманими результатами можна зробити наступні висновки:

* HAAR алгоритм точніше визначає обличчя ніж LBP, в середньому на 10-15%;
* LBP класифікатор розпізнає в 2.5 рази швидше класифікатора HAAR;
* LBP при обробці зображень споживає більше оперативної пам'яті на 2-5%;
* HAAR дескриптор навантажує процесор на 5-8% більше;
* швидкість обробки зображення обох алгоритмів залежить від кількості пікселів на зображенні - чим більше дозвіл, тим довше відбувається обробка.

**5.4 Бібліотека для розпізнавання face\_recognition**

Тепер, коли ми маємо базове розуміння того, як працює розпізнавання обличчя, давайте побудуємо власний алгоритм розпізнавання обличчя за допомогою деяких відомих бібліотек Python.

Існує дивовижно проста бібліотека Python, яка інкапсулює все вище сказане, - створюючи вектори функцій поза обличчям та знаючи, як розрізняти обличчя. Ця бібліотека Python називається face\_recognition і глибоко всередині неї використовується dlib - сучасний інструментарій C ++, який містить кілька алгоритмів машинного навчання, які допомагають писати складні програми на основі C ++.

Бібліотека dlib, підтримувана Девісом Кінгом, містить реалізацію «глибокого вивчення метрики», яка використовується для побудови вкладень особи, які використовуються для фактичного процесу розпізнавання.

Бібліотека face\_recognition, створена Adam Geitgey, охоплює функцію розпізнавання осіб dlib, полегшуючи роботу з нею.

Бібліотека face\_recognition в Python може виконувати велику кількість завдань:

* Знайти усі обличчя в заданому зображенні
* Знайти і маніпулювати рисами обличчя на зображенні
* Визначити обличчя на зображеннях
* Розпізнати обличчя в реальному часі

**Висновок**

В даному розділі було розглянуто алгоритм розпізнавання зображень, проведено аналіз бібліотек OpenCV та face\_recognition, які можна використовувати в мові програмування Python

**6 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ РОЗПІЗНАВАННЯ І ЗАПИСУ ДАНИХ В БД**

**6.1 Алгоритм розробки та роботи програмного забезпечення**

Програмне забезпечення розділене на 5 модулів.

1. Створення БД. Для збереження інформації було обрано базу даних SQLite. В ній будуть зберігатися дані по студентам (ПІП, шлях до фотографії та ін.), групам (назва, курс), час, коли було зафіксовано студента та предмети, які викладються певній групі.
2. Збереження оптимізованих зображень для подальшого використання при розпізнаванні. Відбувається збереження чорно-білої копії фотографії студента в каталозі з присвоєнням назви папки id студента.
3. Тренування системи розпізнавання. Відбувається створення файлу, який є словником, де ключом є id студента, а значенням вектори чисел, що описують зображення обличчя особи. Це необхідно для швидкого пошуку та порівнянні значень при розпізнаванні осіб в реальному часі.
4. Розпізнавання в реальному часі. Вибираємо раніше створену БД, яка заповнена даними по студентах та предметах. Підключаємо відеокамеру. З відео витягується зображення, відбувається пошук обличь, якщо знайдені, то перетворюються у вектор чисел і відбувається порівняння з даними у словнику, який створений був на попередньому кроці. Якщо знайдено співпадіння більше 60%, тоді фіксується час особи і зберігається в базу даних.
5. Формування звіту по відвідуваності. Для цього було створено графічний інтерфейс, в якому є можливість переглянути інформацію по відвідуваності осіб, які є в БД. Також є можливість зберегти ці дані по відвідуваності студента в таблицях в окремому файлі.

Схема роботи ПЗ зображена на рис. 6.1.



Рисунок 6.1 - Схема роботи ПЗ з розпізнавання та збереження в БД

**6.2 Створення структури БД**

Для створення цієї БД Python має стандартну бібліотеку sqlite3.

Вона буде містити 5 таблиць:

1. students: id студента, id групи, прізвище, ім’я, по батькові, шлях до фотографії
2. groups: id групи, назва, курс
3. subjects: id, назва предмета, день тижня, час початку
4. datetime: id, id студента, день тижня, дата, час
5. groups\_subjects: id, id групи, id предмета (створено для організації зв’язку ManyToMany)

Повний лістинг програми знаходиться у додатку А.

**6.3 Створення зображень для подальшого «навчання» системи розпізнавання**

Підключаємо необхідні бібліотеки

import numpy as np  
import os  
import cv2  
import sqlite3 as lite

Вибираємо haar класифікатор

face\_cascade = cv2.CascadeClassifier("haarcascade\_frontalface\_default.xml")

Підключаємось до попередньо створеної БД

con = lite.connect('face\_recognition.db')

Далі відбувається зчитування даних, вибір зображень і створення необхідних копій обличь в папку dataset/id\_студента/n.jpg

cap = cv2.VideoCapture('%s%s' %(IMAGES\_PATH, row[3]))

cv2.imwrite(path + str(id) + "\\" + str(sampleN) + ".jpg", gray[y:y + h, x:x + w])

Повний лістинг програми знаходиться у додатку Б.

**6.4 «Навчання» системи розпізнавання**

В циклі відбувається зчитування зображень, створених в попередньому кроці.

За допомогою бібліотеки face\_recognition кодуються дані у вектор чисел

name = imagePath.split(os.path.sep)[-2]  
image = cv2.imread(imagePath)  
rgb = cv2.cvtColor(image, cv2.COLOR\_BGR2RGB)  
boxes = face\_recognition.face\_locations(rgb, model= "hog")  
encodings = face\_recognition.face\_encodings(rgb, boxes)

і ці дані записуються в певний файл під назвою

module = "encodings\\encoding1.pickle"

Повний лістинг програми знаходиться у додатку В.

**6.5 Програмне забезпечення розпізнавання і запису даних в базу**

Підключаємо необхідні бібліотеки

import imutils  
import numpy as np  
import pickle  
import cv2  
import face\_recognition  
import sqlite3 as lite  
import datetime

Підключаємось до БД

con = lite.connect('face\_recognition.db')

Зчитуються кодовані дані з попереднього кроку, вмикається відеокамера

encoding = "encodings\\encoding1.pickle"  
data = pickle.loads(open(encoding, "rb").read())  
cap = cv2.VideoCapture(0)if cap.isOpened :  
 ret, frame = cap.read()  
else:  
 ret = False

В циклі відбувається розпізнавання з допомогою бібліотеки face\_recognition і якщо були знайдені співпадіння, більше 60%, вибирається найбільше і в список додається id студента

for encoding in encodings:  
 matches = face\_recognition.compare\_faces(np.array(encoding),np.array(data["encodings"]))  
 name = "Unknown"  
 if True in matches:  
 matchedIdxs = [i for (i, b) in enumerate(matches) if b]  
 counts = {}  
 for i in matchedIdxs:  
 name = data["names"][i]  
 counts[name] = counts.get(name, 0) + 1  
 name = max(counts, key=counts.get)  
 names.append(name)

Після цього зі списку витягуються id і записується поточний час і відповідний id студента

for name in names:  
 if name != "Unknown":  
 try:  
 with con:  
 now = datetime.datetime.now()  
 date = str(now.date().strftime("%Y-%m-%d"))  
 time = str(now.time().strftime("%H:%M:%S"))  
 weekday = now.weekday()  
 cur = con.cursor()  
 cur.execute("INSERT INTO datetime(student\_id, date, time, weekday) VALUES(?,?,?,?)", (name, date, time, weekday))  
 cur.execute("SELECT firstname FROM students WHERE id=?", (name))  
 for row in cur.fetchall():  
 name = row[0]   
 con.commit()  
 except:  
 name = "Unknown"

При натисненні Esc відбувається вихід з циклу і закриття програми

if cv2.waitKey(1) == 27:  
 break

Повний лістинг програми знаходиться у додатку Г.

**6.6 Обґрунтування вибору графічної бібліотеки**

Для створення графічного інтерфейсу користувача було використано стандартну бібліотеку Python Tkinter.

Tkinter - це графічна бібліотека, що дозволяє створювати програми з віконним інтерфейсом. Ця бібліотека є інтерфейсом до популярної мови програмування і інструменту створення графічних додатків tcl / tk. Tkinter, як і tcl / tk, є кроссплатформенною бібліотекою і може бути використана в більшості поширених операційних систем (Windows, Linux, Mac OS X і ін.).

Так як Tkinter є досить прозорим інтерфейсом до tcl / tk, то основним джерелом інформації для неї є man-сторінки tcl / tk. Ці сторінки є в будь-якій Unix-системі. Починаючи з версії python-3.0 бібліотека перейменована відповідно до PEP 8 в tkinter (з маленької літери).

У Tkinter візуальні контролери називаються віджетами (widget, від англ. Window gadget) - стандартизований компонент графічного інтерфейсу, з яким взаємодіє користувач.

Tk є базовим класом будь-якого Tkinter додатка. При створенні об'єкта цього класу запускається інтерпретатор tcl / tk і створюється базове вікно програми.

Tkinter є подієво-орієнтованою бібліотекою. У додатках такого типу є головний цикл обробки подій. У Tkinter такий цикл запускається методом mainloop. Для явного виходу інтерпретатора і завершення циклу обробки подій використовується метод quit.

Всі віджети в Tkinter мають деякі загальні властивості. Віджети створюються викликом конструктора відповідного класу. Перший аргумент (як правило неіменованого, але можна використовувати ім'я master) це батьківський віджет, в який буде упакований (поміщений) наш віджет. Батьківський віджет можна не вказувати, в такому випадку буде використано головне вікно програми. Далі йдуть іменовані аргументи, конфігурують віджет. Це може бути використовуваний шрифт (font = ...), колір віджета (bg = ...), команда, що виконується при активації віджета (command = ...) і т.д.

**6.7 Створення графічного інтерфейсу**

Імпортуємо необхідні бібліотеки

import tkinter as tk  
from tkinter import filedialog, messagebox  
from PIL import Image, ImageTk  
import tkinter.ttk as ttk  
import sqlite3 as lite  
import datetime  
import xlsxwriter  
import os

Створюємо деякі константи, що будуть використовуватися (шлях до БД, тривалість однієї пари та ін.)

DEFAULT\_DB\_PATH = './'  
WEEKDAYS = ['Понеділок', 'Вівторок', 'Середа', 'Четвер', "П'ятниця", 'Субота', 'Неділя']  
MONTHS = ['', 'Січня', 'Лютого', 'Березня', 'Квітня', 'Травня', 'Червня', 'Липня', 'Серпня', 'Вересня', 'Жовтня', 'Листопада', 'Грудня']  
DEFAULT\_SUBJECT\_TIMEDELTA = datetime.timedelta(0, 4800)  
student\_detail\_windows = {}  
NOT\_FOUND = 'Not found'  
PATH\_FOR\_REPORTS = './reports'

Далі створено клас, для зручності підключення до БД

class DB:  
 def \_\_init\_\_(self, name=None):  
 self.name = name  
 self.images\_path = os.path.dirname(name) + '/images' if name else None  
 self.con = None  
 def change\_name(self, name):  
 old\_name = self.name  
 self.name = name  
 self.images\_path = os.path.dirname(name) + '/images' if name else None  
 self.connect()  
 if not self.con:  
 self.images\_path = os.path.dirname(old\_name) + '/images' if old\_name else None  
 self.name = old\_name  
 def connect(self):  
 try:  
 self.con = connect\_to\_db(self.name)  
 except:  
 messagebox.showerror('Помилка', 'Вибраний файл не є БД!')  
 self.con = None

selected\_db = DB()

Далі в основній частині створено графічне вікно. В ньому додано кілька полів для введення відповідних даних для пошуку студентів (група, ПІП). Також добавлено кнопку для вибору БД. Якщо вибрана БД є коректна, відбувається упаковування віджетів у відповідні фрейми. Після введення даних для пошуку персоналу і натиснення кнопки «Знайти» з’являється список знайдених студентів. Вигляд основного вікна графічного інтерфейсу зображено на рис.6.2

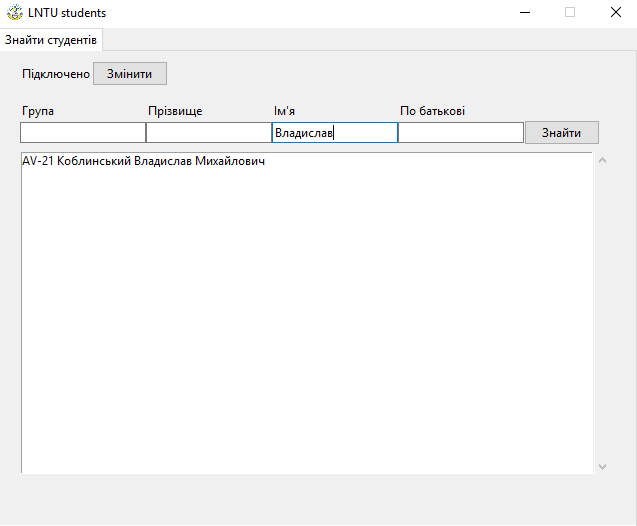


Рисунок 6.2 - Вигляд основного вікна графічного інтерфейсу

Для детальної інформації по кожному є можливість натиснути на відповідний рядок зі списку, тоді з’являється нове вікно, в якому є фотографія студента, блок із предметами розділених по днях та дати, коли студент був присутній. Вигляд картки студента на рис. 6.3.

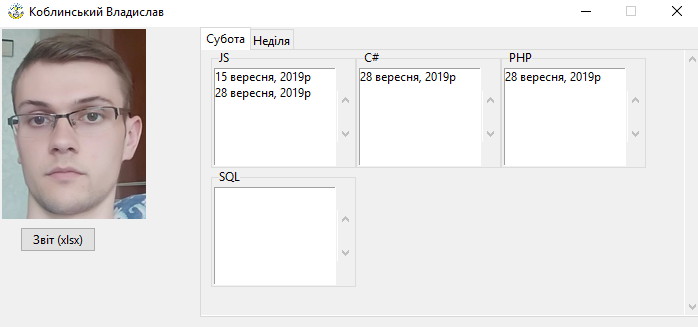


Рисунок 6.3 - Вигляд картки студента

Також під фотографією є кнопка для формування звіту у форматі «ПІП.xlsx». У файлі створюються таблиці по днях і заповнюються даними коли студент був присутній і на якій саме парі. Вигляд звіту зображено на рис.. 6.4.

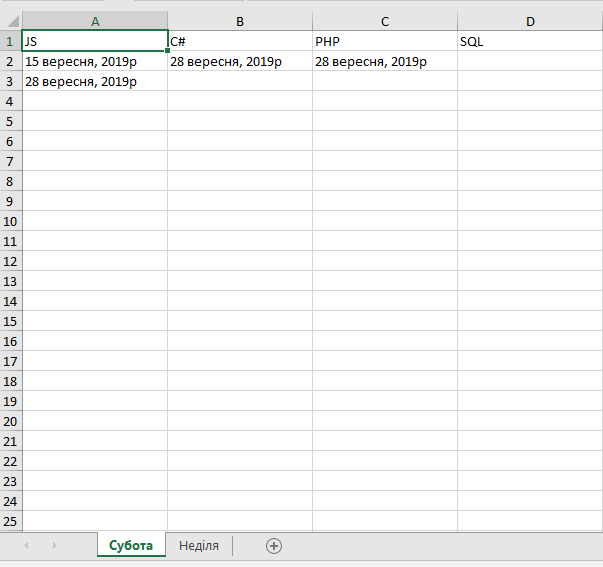


Рисунок 6.4 - Звіт у форматі xlsx

Повний лістинг програми знаходиться у додатку Д.

**6.8 Компіляція ПЗ**

Для компіляції ПЗ в виконуваний файл для операційної системи Windows було використано бібліотеку PyInstaller.

Основні переваги PyInstaller перед подібними інструментами полягають у тому, що PyInstaller працює з Python 2.7 та 3.4—3.7, він створює менші виконувані файли завдяки прозорому стисненню, він є повністю кросплатформеним та використовує підтримку ОС для завантаження динамічних бібліотек, забезпечуючи таким чином повну сумісність.

Основна мета PyInstaller - бути сумісним із сторонніми пакунками, що випускаються поза коробкою. Це означає, що з PyInstaller всі необхідні прийоми для роботи зовнішніх пакетів вже інтегровані в сам PyInstaller, так що не потрібно втручання користувача. Від вас ніколи не буде потрібно шукати хитрощів та застосовувати власні модифікації до своїх файлів або сценаріїв налаштування. Наприклад, бібліотеки типу PyQt, Django або matplotlib повністю підтримуються, не потребуючи обробки плагінів або зовнішніх файлів даних вручну.

Встановити PyInstaller з PyPI:

pip install pyinstaller

Перейдіть до каталогу програми та запустіть:

pyinstaller yourprogram.py

Це створить пакет у підкаталозі під назвою dist.

**Висновок**

В даному розділі було описано програмну частину, яка використовується для збереження інформації з розпізнавання в БД і формування звіту. Також описано бібліотеку компілювання ПЗ у виконуваний файл для запуску на ОС Windows.

**7 ТЕСТУВАННЯ СТВОРЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ РОЗПІЗНАВАННЯ**

**7.1 Тестування програмного забезпечення**

Для тестування програмного забезпечення (ПЗ) було обрано однакові фотографії з роздільними здатностями 800х480, 1024х768 та 1920х1080. Обрано було якісні показники: контрасність, яскравість. Також проведено тестування залежно від кількості зображень, що знаходяться в базі даних (БД).

Початкове значення якісного показника було прийнято за 100%.

**7.2 Залежність розпізнавання від контрастності**

Рисунок 7.1 - Залежність розпізнавання від контрастності зображення роздільною здатністю 800х480

Рисунок 7.2 - Залежність розпізнавання від контрастності зображення роздільною здатністю 1024х768

Рисунок 7.3 - Залежність розпізнавання від контрастності зображення роздільною здатністю 1920х1080

Отже, зменшення контрасності не впливає на розпізнавання, проте якщо контрасність досягне більше 150%, то розпізнавання почне зменшуватися на 2% кожні 10% збільшення контрастності.

**7.3 Залежність розпізнавання від яскравості**

Рисунок 7.4 - Залежність розпізнавання від яскравості зображення роздільною здатністю 800х480

Рисунок 7.5 - Залежність розпізнавання від яскравості зображення роздільною здатністю 1024х768

Рисунок 7.6 - Залежність розпізнавання від яскравості зображення роздільною здатністю 1920х1080

Отже, якщо яскравість зменшити на 40%, то на розпізнавання це не вплине, проте розпізнавання буде погіршуватися на 1% зменшуючи яскравість на 10%. Якщо ж яскравість збільшувати, то розпізнавання буде погіршуватися на 2% кожні 10% зміни.

**7.4 Залежність розпізнавання від кількості зображень**

Рисунок 7.7 - Залежність розпізнавання від кількості зображень роздільною здатністю 800х480

Рисунок 7.8 - Залежність розпізнавання від кількості зображень роздільною здатністю 1024х768

Рисунок 7.9 - Залежність розпізнавання від кількості зображень роздільною здатністю 1920х1080

Отже, розпізнавання покращується, якщо збільшити кількість зображень в БД (на 1%, якщо збільшувати на 10)

**Висновок**

Отже, розпізнавання не змінюється зі зменшенням контрастності, проте погіршується зі зміною яскравості. Покращується на 4%, якщо кількість зображень, що є в БД, збільшити до 50.

Проте головним недоліком даної системи є те, що вона не визначає чи справжня людина чи просто зображення.

# 8 ТЕХНІКО-ЕКОНОМІЧНЕ ОБГРУНТУВАННЯ

**8.1 Розрахунок капітальних затрат на розробку програмного продукту**

Таблиця 8.1 - Вартість ліцензійних програм, які можуть були використані при розробці проектованих рішень, грн

|  |  |
| --- | --- |
| Назва програми | Вартість, грн |
| Windows 10(технічний супровід) | 5132 грн |
| PyCharm professional | 4900 грн |
| Всього | 10032 грн |

Отже, вартість ліцензійних програмних продуктів становить 10032 грн. Оскільки у подальших розрахунках даного проекту обчислюється вартість нематеріальних активів, то варто зауважити, що їх вартість відповідатиме вартості інтелектуальної власності, а саме: 10032 грн.

**8.2 Розрахунок часу на створення програмного продукту**

Загальний час на створення програм (імітаційних моделей) складається з різних компонентів, а саме загальної трудомісткості виконання програмного продукту. Перелік основних етапів розробки програмного продукту, загальний час та кількість виконавців кожного етапу представлено в таблиці 8.2.

Таблиця 8.2 - Етапи і загальний час на створення програмного продукту

|  |  |  |
| --- | --- | --- |
| Час етапу | Зміст етапу | Люд./год |
| Тпо | Підготовка та опис завдання | 30 |
| То | Опис завдання | 30 |
| Тбс | Розробка блок-схеми алгоритмів | 25 |
| Ті | Написання програм | 37,5 |
| Тп | Набір програм | 30 |
| Тот | Налагодження і тестування програм | 105 |

Час розрахований в людино-годинах, причому Тпо береться по фактично відпрацьованих годинах, а час інших етапів обчислюється розрахунком по умовному числу команд Q.

Умовне число команд Q обчислюється по формулі:

*Q = q \* c,* (8.1)

деq *–* коефіцієнт, що враховує умовне число команд в залежності від типу задачі.

Вибрати значення коефіцієнта q можна із таблиці 8.3.

Таблиця 8.3 - Значення коефіцієнта q

|  |  |
| --- | --- |
| Тип задачі | Межі вимірювання коефіцієнта |
| Задачі обліку | від 1400 до1500 |
| Задачі оперативного управління | від 1500 до1700 |
| Задачі планування | від 3000 до3500 |
| Багатоваріантні задачі | від 4500 до5000 |
| Комплексні задачі | від 5000 до5500 |

Для даної задачі коефіцієнт q приймається = 1500.

С *–* коефіцієнт новизни та складності програми.

Програмні продукти по ступені новизни можуть відноситись до одної із 4-х груп:

група А – розробка принципово нових задач;

група Б *–* розробка оригінальних програм;

група В *–* розробка програм з використанням типових рішень;

група Г *–* разоватипова задача.

Для даної задачі ступінь новизни: В*.*

По ступені складності електронні засоби навчання можуть бути віднесені до одної із 3-х груп:

1 – стандартні алгоритми;

2 – задачі обліку, звітності і статистики;

3 – алгоритми оптимізації та моделювання систем.

Дана задача може бути віднесена до 2 –ї групи складності.

Коефіцієнт С визначається із таблиці 8.4 на перетинанні груп складності і ступеня новизни.

Таблиця 8.4 – Значення коефіцієнта С

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| МОВА ПРОГРАМУ-ВАННЯ | ГРУПА СКЛАДНОСТІ | СТУПІНЬ НОВИЗНИ | | | |
| А | Б | В | Г |
| ВИСОКОГО РІВНЯ | 1 | 1,38 | 1,26 | 1,15 | 0,69 |
| 2 | 1,30 | 1,19 | 1,08 | 0,65 |
| 3 | 1,15 | 1,10 | 1,00 | 0,60 |
| НИЗЬКОГО РІВНЯ | 1 | 1,58 | 1,45 | 1,32 | 0,79 |
| 2 | 1,49 | 1,37 | 1,24 | 0,74 |
| 3 | 1,38 | 1,26 | 1,15 | 0,69 |

Для даної задачі коефіцієнт С = 1,00.

Тепер, за допомогою формули 8.1 можна визначити умовне число команд Q

*Q* = 1500 ∙ 1,00 = 1500.

Визначаємо час, затрачений на кожний етап створення програмного продукту:

Тпо (час на підготовку та опис задачі) береться по факту и становить:

Тпо = 30 люд/год.

То (час на опис задачі) обчислюється за формулою:

*,* (8.2)

де *В* – коефіцієнт обліку змін задачі, коефіцієнт *В* в залежності від складності задачі і числа змін вибирається в інтервалі від 1,2 до 1,5.

Для даної задачі приймаємо  В= 1,2.

К *–* коефіцієнт, враховує кваліфікацію програміста.

Вибрати значення коефіцієнта Кможна із таблиці8.5

Таблиця 8.5 - Коефіцієнт, враховуючий кваліфікацію програміста

|  |  |
| --- | --- |
| Стаж програміста | Значення коефіцієнта *К* |
| від 2-х років | 0,8 |
| від 2 до 3 років | 1,0 |
| від 3 до 5 років | 1,1 – 1,2 |
| від 5 до 10 років | 1,2 – 1,3 |
| більше 10 років | 1,3 – 1,5 |

В даному випадку коефіцієнта К= 0,8.

Використовуючи формулу 8.2, підраховуємо час на опис задач моделювання.

(люд./год.). : 7 год. = кількість календарних днів.

У нашому варіанті час на підготовку опису завдання та опис завдання можна об’єднати у межах часу на опис задач, який рівний 30,0 люд./год.

Та (час на розробку алгоритму) обчислюємо за формулою:

, (8.3)

Застосовуючи формулу 8.3 підраховуємо час на розробку алгоритму.

(люд./год.).

Тбс (час на розробку блок-схем) обчислюється аналогічно Та за формулою і становить:

(люд./год.).

Час на розробку блок-схем об’єднуємо з часом на розробку алгоритму і він становитиме 55 люд./год.

Ті (час написання програми на мові програмування) обчислюється за формулою:

, (8.4)

Застосовуючи формулу 8.4 підраховуємо час на написання програми на мові програмування:

(люд./год.).

Тп (час набору програми) обчислюється за формулою:

, (8.5)

Застосовуючи формулу 8.5 підраховуємо час набору програм:

(люд./год.).

Тот (час налагодження і тестування програми) обчислюється за формулою:

, (8.6)

Підставляємо значення у формулу 8.6 і отримуємо:

(люд./год.).

Тепер, знаючи час, який затрачено на кожному етапі, можна підрахувати загальний час на створення програмного продукту:

*Теф.= Ті + Тп + Тот + Тпо +Тбс*= 227,5 (люд./год.).

Тобто, ефективний час на створення програмного продукту триває 227,5 люд./год., або це – 1,4 місяців.

**8.3 Розрахунок кошторису капітальних витрат на розробку програмного продукту**

Таблиця 8.6 - Зведена відомість характеристики обладнання для розробки та впровадження програмного продукту

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Назва | Кіль-кість | Вартість за прейскурантом | | ТЗВ 10% тис грн | Балансова вартість, тис грн |
| Один | Всього |
| ASUS Notebook | 1 | 11000 | 11000 | 1100,0 | 12100,0 |

**8.4 Розрахунок виробничої площі робочих місць програмістів та вартість основних фондів**

Виробничі площі розраховуються згідно вимог техніки безпеки роботи з комп’ютерними технологіями та санітарно-технічних вимог.

А саме на одне робоче місце програміста згідно інструкції по техніці безпеки припадає 6 м2.

Відповідно площа виробничих приміщень буде розраховуватись за формулою:

*Sп = Sод \* n,*  (8.7)

Sп = 6\*1=6 (м2 )

де *Sп*– площа виробничих приміщень;

*Sод*– площа, яка виділяється на одне робоче місце програміста згідно правил техніки безпеки;

n – кількість робочих місць.

Тоді, відповідно вартість даної площі розраховується за формулою:

*Vn = Sn\* Vм2 ,* (8.8)

де *Vм2* = 1000 грн ­ – для придбання, 150 грн для оренди.

*Vn*– вартість виробничої площі;

*Vм2* – вартість 1м2 площі офісних приміщень.

*Vn*=6\*150 =900 (грн) – за місяць;

*Vn*= 900 \* 1,4 = 1260 (грн) – за період створення продукту

Оскільки приміщення для створення програмного продукту береться в оренду, де в наявності є тільки меблі і електрообладнання, а комп’ютерна техніка купується, то нарахування амортизаційних відрахувань проводиться тільки для комп’ютерної та офісної техніки і нематеріальних активів. Орендна плата включається у собівартість продукції.

**8.5 Визначення розміру заробітної плати розробників проекту**

Для визначення розміру основної і додаткової заробітної плати необхідно, врахувати трудомісткість розробки програмного продукту (імітаційних моделей), тарифні ставки та міжкваліфікаційні коефіцієнти співвідношень місячних посадових окладів та розрахувати оклад молодшого спеціаліста. Отже, у розробці нашого проекту зайнятий один працівник – програміст-розробник. Його коефіцієнт співвідношення до посадового окладу техніка становить 2,0. Відповідно заробітну плату обчислимо за формулою:

1) Основна заробітна плата програмістів розраховується за формулою:

*ЗПосн. = ,*  (8.9)

де *З1* – заробітна плата робітника першого розряду (1762 грн), якщо розряди розробників різні, то береться середня заробітна плата;

*km*– тарифний коефіцієнт, відповідного розряду тарифної сітки, згідно якої працює виконавець (2);

*Т* – загальний час на створення програмного продукту (люд./год.);

*Nр* – кількість робочих днів у місяці (21);

*tр.q*. – тривалість робочого дня (8 год.).

*ЗПосн. = *

2) Додаткова заробітна плата береться у розмірі 15% від основної заробітної плати. У нашому випадку вона становитиме 715,8 грн.

3) Загальна заробітна плата буде дорівнювати сумі основної та додаткової:

*ЗПзаг. = ЗПосн. + ЗПдод*, (8.10)

*ЗПзаг.* = 4772 + 715,8 = 5487,8 грн.

Нарахування до фондів соціального страхування становлять 22,0 % від загальної заробітної плати і розраховується за формулою:

НФ = , (8.11)

НФ=1207,3 грн.

Обчислимо заробітну плату розробника за період проекту і дані занесемо у табл. 8.7. У нашому випадку застосовується тарифна система оплати праці.

Таблиця 8.7 - Заробітна плата працівників, зайнятих у роботі над проектом

|  |  |  |  |
| --- | --- | --- | --- |
| Категорії працівників | Розмір заробітної плати | | Нарахування |
| Основна | Додаткова |
| Розробник | 4772 | 715,8 | 1207,3 |
| Разом | 4772 | 715,8 | 1207,3 |

**8.6 Розрахунок витрат на електроенергію**

Витрати на електроенергію складаються з:

* витрат на силову електроенергію;
* витрат на електроенергію, яка іде на освітлення.

Витрати на силову електроенергію визначаються по формулі:

*Зс.ен. = Феф. \* Це\* Р,* (8.12)

де *Феф*.*–* ефективний фонд часу роботи ЕОМ в годинах.

*Це* – вартість 1 кВт/год. в грн.;

*Р* – сумарна потужність ЕОМ в кВт/год.

Для бюджетних організацій Це =146 коп. за 1кВт\год.

*Р* = 0,7 – 1,2 кВт\год. в залежності від периферії.

Підставляємо значення у формулу і отримуємо:

*Зс.ен*. =227,5 \*1,46\*0,8=265,72 грн.

Витрати на електроенергію, яка використовується на освітлення, обчислюється за формулою:

*Зосв. = Феф.\* Це\* Росн.,* (8.13)

де *Феф*.*–* ефективний фонд часу роботи лампи в годинах

*Це* – вартість 1 кВт/год. в грн.;

*Росн*. – сумарна потужність, яка іде на освітлення в кВт/год.

*Росн. =* 1,4 кВт/год.

Підставляємо значення у формулу і отримуємо:

*Зосв. =*  227,5 \*1,46\*1,4 = 465,01 грн.

Тепер визначаємо загальні витрати на електроенергію:

*Зен. = Зс.ен. + Зосв.*, (8.14)

Підставляємо значення у формулу і отримуємо:

*Зен.*= 265,72 + 465,01 = 730,73 грн.

Визначення витрат на сировину і матеріали

Витрати на матеріали, канцелярсько – письмові засоби розраховуються по кількості і їх прейскурантним цінам. Перелік використовуваних матеріалів, потреба в них і їх ціни наведені в таблиці 8.8.

Таблиця 8.8 - Витрати на матеріали

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № п/п | Назва матеріалів та канцелярсько-письмових засобів | Одиниця вимірювання | Кіл-ть | Ціна,  грн. | Сума,  грн. |
| 1. | Папір для друку | пачка | 1 | 60,0 | 60,0 |
| 2. | CD-диск | шт. | 1 | 12,0 | 12,0 |
| Всього | | | | | 72,0 |

**8.7 Розрахунок собівартості програмного продукту**

Собівартість програмного продукту – це виражені в грошовій формі витрати на його розробку і, при необхідності, реалізацію.

В собівартість програмного продукту входять наступні елементи:

1. Основна заробітна плата виконавців робіт по створенню програмного продукту.

2. Додаткова заробітна плата виконавців робіт по створенню програмного продукту.

3. Нарахування на заробітну плату (єдиний соціальний податок).

4. Витрати на утримання та експлуатацію ПЕОМ, які відносяться до даного програмного продукту.

5. Інші витрати.

Амортизаційні відрахування визначаються у розмірі 25% від балансової вартості ПЕОМ.

Інші виробничі витрати (виробнича собівартість) беруться в розмірі 30% від основної зар.плата працівників

Форма калькуляції витрат на розробку програмного продукту приведена в таблиці 8.9.

Таблиця 8.9 - Калькуляція собівартості програмного продукту

|  |  |
| --- | --- |
| Статті калькуляції | Сума (грн) |
| Основна заробітна плата | 4772 |
| Додаткова заробітна плата | 715,8 |
| Нарахування на заробітну плату | 1207,3 |
| Матеріальні витрати | 12100 |
| Витрати електроенергії | 730,73 |
| Нарахована амортизація | 735,0 |
| Витрати на оренду | 1260 |
| Виробнича собівартість | 1085,4 |
| Інші витрати | 72,0 |
| Повна собівартість | 22677,83 |

**Висновок**

Отже, провівши розрахунки витрат на розробку програмного продукту, можна зробити висновки, що повна собівартість розробки імітаціних моделей складає 22677,83 гривень.

ЗАГАЛЬНІ ВИСНОВКИ

Темою дипломної роботи була розробка автоматизованої системи персоналізації та ведення обліку персоналу з використанням системи розпізнавання зображень.

В ході виконання роботи було проаналізовано існуючі технології розпізнавання, сфери затосування, проведено аналіз існуючих найбільш популярних на даний момент баз даних. Описано їх основні переваги та недоліки. Обгрунтовано вибір високорівневої мови програмування для розробки ПЗ – Python.

Також проведено короткий огляд існуючих методів розпізнавання обличь та їх основні недоліки, аналіз бібліотек для розпізнавання зображень, які можна використовувати в мові програмування Python. Створено програмну частину, яка використовується для збереження інформації з розпізнавання в БД і формування звіту. Проведено тестування створеного ПЗ, описано переваги та недоліки, зроблені розрахунки витрат на розробку програмного продукту.

# СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Face Detection Dataset and Benchmark dataset [електронний ресурс] – Режим доступу. – URL: <http://vis-www.cs.umass.edu/fddb/>
2. Wider Project dataset [електронний ресурс] – Режим доступу. – URL: <http://mmlab.ie.cuhk.edu.hk/projects>
3. Psychological Image Collection at Stirling dataset [електронний ресурс] – Режим доступу. – URL: <http://pics.psych.stir.ac.uk/>
4. Библиотека алгоритмов компьютерного зрения OpenCV [електронний ресурс] – Режим доступу. – URL: <https://opencv.org/>
5. Viola P., Jones M., Rapid object detection using a boosted cascade of simple features // Computer Vision and Pattern Recognition. 2001.Vol. 1, P. 511-518
6. Dalal, N., Triggs, B. Histograms of Oriented Gradients for Human Detection // CVPR '05 Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. 2011. Vol. 1. P. 886-893.
7. López, L. S., Ruiz, F. T. Local Binary Patterns applied to Face Detection and Recognition. Barcelona: Universitat Politècnica de Catalunya, 2010. 146 p.
8. Л. Шапиро, Дж. Стокман. Компьютерное зрение / пер. з англ. М.: Бином. Лабораторія знань, 2006. 752 с.
9. Лутц М. Программирование на Python, 4-е издание. / М.: Символ-Плюс, 2011. 992 с.
10. Degtyarev N., Seredin O., Comparative Testing of Face Detection Algorithms // ICISP 2010. 2010. Vol. 6134. P. 200-209.
11. Ahonen T., Hadid A., Pietikainen M., Face Description with Local Binary Patterns: Application to Face Recognition // Transactions on Pattern Analysis and Machine Intelligence. 2006. Vol. 28. P. 2037 – 2041.
12. Fawcett T., An introduction to ROC analysis, Pattern Recognition Letters // Elsevier Science. 2006. Volume 27. P. 861 – 874.
13. Hjelmas, E., Low, B.K., Face Detection: A Survey // Computer Vision and Image Understanding. 2001. Vol. 83. P. 236-274.
14. Gutta, S., Wechsler, H., Phillips, P.J., Gender and ethnic classification of face images // IEEE Computer Society. 1998. Vol. 1. P. 194-199.
15. Yang Z., Ai H., Demographic Classification with Local Binary Pattern. Beijing // Advances in Biometrics. 2007. Vol. 4642. P.464-473..
16. Lienhart R., Maydt J., An extended set of Haar-like features for rapid object detection // International Conference on Image Processing. 2002. Vol. 1. P. 22-25

# ДОДАТКИ

Додаток А

**Лістинг програми для створення структури БД**

import sqlite3 as lite  
import sys  
  
con = lite.connect('face\_recognition.db')  
with con:  
 cur = con.cursor()  
 cur.execute("CREATE TABLE students( "  
 "id INTEGER PRIMARY KEY,"  
 "group\_id INT,"  
 "lastname VARCHAR,"  
 "firstname VARCHAR,"  
 "fathername VARCHAR,"  
 "image VARCHAR,"  
 "FOREIGN KEY (group\_id) REFERENCES groups(id)"  
 ")")  
 cur.execute("CREATE TABLE groups( "  
 "id INTEGER PRIMARY KEY,"  
 "name VARCHAR,"  
 "course INT"  
 ")")  
 cur.execute("CREATE TABLE subjects( "  
 "id INTEGER PRIMARY KEY,"  
 "name VARCHAR,"  
 "day\_of\_week INT,"  
 "time TIME"  
 ")")  
 cur.execute("CREATE TABLE datetime( "  
 "id INTEGER PRIMARY KEY,"  
 "student\_id INT,"  
 "weekday INT,"  
 "date DATE,"  
 "time TIME,"  
 "FOREIGN KEY (student\_id) REFERENCES students(id)"  
 ")")  
 cur.execute("CREATE TABLE groups\_subjects( "  
 "id INTEGER PRIMARY KEY,"  
 "group\_id INT,"  
 "subject\_id INT,"  
 "FOREIGN KEY (group\_id) REFERENCES groups(id),"  
 "FOREIGN KEY (subject\_id) REFERENCES subjects(id)"  
 ")")  
  
 con.commit()

Додаток Б

**Лістинг програми створення зображень для подальшого розпізнавання**

import numpy as np  
import os  
import cv2  
import sqlite3 as lite  
IMAGES\_PATH = 'images/'  
face\_cascade = cv2.CascadeClassifier("haarcascade\_frontalface\_default.xml")  
path = "dataset\\"*# path were u want store the data set*con = lite.connect('face\_recognition.db')  
with con:  
 cur = con.cursor()  
 cur.execute("SELECT id, firstname, lastname, image FROM students")  
 for row in cur.fetchall():  
 print(row[0], row[1], row[2], row[3])cap = cv2.VideoCapture('%s%s' %(IMAGES\_PATH, row[3]))  
 id = row[0]  
 try:  
 *# Create target Directory* os.mkdir(path+str(id))  
 print("Directory " , path+str(id), " Created ")  
 except FileExistsError:  
 print("Directory " , path+str(id) , " already exists")  
 sampleN=0ret, img = cap.read()  
 frame = img.copy()  
 gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)  
 faces = face\_cascade.detectMultiScale(gray, 1.3, 5)  
 for (x, y, w, h) in faces:  
 sampleN = sampleN + 1  
 cv2.imwrite(path + str(id) + "\\" + str(sampleN) + ".jpg", gray[y:y + h, x:x + w])  
 cv2.rectangle(img, (x, y), (x + w, y + h), (255, 0, 0), 2)  
 cv2.waitKey(100)cv2.waitKey(1)  
 cap.release()  
 cv2.destroyAllWindows()

Додаток В

**Кодування зображень у вектор**

import cv2  
import imutils.paths as paths  
import face\_recognition  
import pickle  
import os  
  
dataset = "dataset\\"*# path of the data set*module = "encodings\\encoding1.pickle" *# were u want to store the pickle file*imagepaths = list(paths.list\_images(dataset))  
knownEncodings = []  
knownNames = []  
for (i, imagePath) in enumerate(imagepaths):  
 print("[INFO] processing image {}/{}".format(i + 1,len(imagepaths)))  
 name = imagePath.split(os.path.sep)[-2]  
 image = cv2.imread(imagePath)  
 rgb = cv2.cvtColor(image, cv2.COLOR\_BGR2RGB)  
 boxes = face\_recognition.face\_locations(rgb, model= "hog")  
 encodings = face\_recognition.face\_encodings(rgb, boxes)  
 for encoding in encodings:  
 knownEncodings.append(encoding)  
 knownNames.append(name)  
 print("[INFO] serializing encodings...")  
 data = {"encodings": knownEncodings, "names": knownNames}  
 output = open(module, "wb")   
 pickle.dump(data, output)  
 output.close()

Додаток Г

**ПЗ розпізнавання обличь з БД**

import imutils  
import numpy as np  
import pickle  
import cv2  
import face\_recognition  
import sqlite3 as lite  
import datetime  
  
con = lite.connect('face\_recognition.db')  
def main():  
 encoding = "encodings\\encoding1.pickle"  
 data = pickle.loads(open(encoding, "rb").read())  
 cap = cv2.VideoCapture(0)if cap.isOpened :  
 ret, frame = cap.read()  
 else:  
 ret = False  
 while(ret):  
 ret, frame = cap.read()  
 rgb = imutils.resize(frame, width=400)  
 r = frame.shape[1] / float(rgb.shape[1])  
 boxes = face\_recognition.face\_locations(rgb, model= "hog")  
 encodings = face\_recognition.face\_encodings(rgb, boxes)  
 names = []  
 for encoding in encodings:  
 matches = face\_recognition.compare\_faces(np.array(encoding),np.array(data["encodings"]))  
 name = "Unknown"  
 if True in matches:  
 matchedIdxs = [i for (i, b) in enumerate(matches) if b]  
 counts = {}  
 for i in matchedIdxs:  
 name = data["names"][i]  
 counts[name] = counts.get(name, 0) + 1  
 name = max(counts, key=counts.get)  
 names.append(name)  
 for name in names:  
 if name != "Unknown":  
 try:  
 with con:  
 now = datetime.datetime.now()  
 date = str(now.date().strftime("%Y-%m-%d"))  
 time = str(now.time().strftime("%H:%M:%S"))  
 weekday = now.weekday()  
 cur = con.cursor()  
 cur.execute("INSERT INTO datetime(student\_id, date, time, weekday) VALUES(?,?,?,?)", (name, date, time, weekday))  
 cur.execute("SELECT firstname FROM students WHERE id=?", (name))  
 for row in cur.fetchall():  
 name = row[0]  
 print(name)  
 con.commit()  
 except:  
 name = "Unknown"if cv2.waitKey(1) == 27:  
 break  
 cv2.destroyAllWindows()  
 cap.release()  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()

Додаток Д

**ПЗ графічного інтерфейсу користувача**

import tkinter as tk  
from tkinter import filedialog, messagebox  
from PIL import Image, ImageTk  
import tkinter.ttk as ttk  
import sqlite3 as lite  
import datetime  
import xlsxwriter  
import os  
DEFAULT\_DB\_PATH = './'  
WEEKDAYS = ['Понеділок', 'Вівторок', 'Середа', 'Четвер', "П'ятниця", 'Субота', 'Неділя']  
MONTHS = ['', 'Січня', 'Лютого', 'Березня', 'Квітня', 'Травня', 'Червня', 'Липня', 'Серпня', 'Вересня', 'Жовтня', 'Листопада', 'Грудня']  
DEFAULT\_SUBJECT\_TIMEDELTA = datetime.timedelta(0, 4800)  
student\_detail\_windows = {}  
NOT\_FOUND = 'Not found'  
PATH\_FOR\_REPORTS = './reports'  
class DB:  
 def \_\_init\_\_(self, name=None):  
 self.name = name  
 self.images\_path = os.path.dirname(name) + '/images' if name else None  
 self.con = None  
 def change\_name(self, name):  
 old\_name = self.name  
 self.name = name  
 self.images\_path = os.path.dirname(name) + '/images' if name else None  
 self.connect()  
 if not self.con:  
 self.images\_path = os.path.dirname(old\_name) + '/images' if old\_name else None  
 self.name = old\_name  
 def connect(self):  
 try:  
 self.con = connect\_to\_db(self.name)  
 except:  
 messagebox.showerror('Помилка', 'Вибраний файл не є БД!')  
 self.con = None  
selected\_db = DB()  
def on\_closing(window):  
 if messagebox.askokcancel("Вихід", "Ви справді хочете вийти?"):  
 window.destroy()  
def create\_window(width=500, height=400, resizeable=True):  
 window = tk.Toplevel(root)  
 window.iconbitmap('lntu.ico')  
 centerWindow(window, width, height)  
 if not resizeable:  
 window.resizable(False, False)  
 return window  
def on\_closing\_student\_window(window, id):  
 student\_detail\_windows.pop(id, None)  
 window.destroy()  
def change\_listbox\_size(event):  
 if students\_listbox:  
 students\_listbox['width'] = round(root.winfo\_width()/7)  
 students\_listbox['height'] = round(root.winfo\_height() / 25)  
def centerWindow(window, width=None, height=None):  
 windowWidth = width or root.winfo\_reqwidth()  
 windowHeight = height or root.winfo\_reqheight()  
  
 *# Gets both half the screen width/height and window width/height* positionRight = int(window.winfo\_screenwidth() / 2 - windowWidth / 2)  
 positionDown = int(window.winfo\_screenheight() / 2 - windowHeight / 2)  
 *# Positions the window in the center of the page.* window.geometry("{}x{}+{}+{}".format(windowWidth, windowHeight, positionRight, positionDown))  
def openFileDialog():  
 file = filedialog.askopenfilename(initialdir=DEFAULT\_DB\_PATH, title="Select file",  
 filetypes=(("db files", "\*.db"), ("all files", "\*.\*")))  
 return file  
def select\_db():  
 name = openFileDialog()  
 if name:  
 selected\_db.change\_name(name)  
 select\_db\_label['text'] = 'Підключено'  
 select\_db\_btn['text'] = u'Змінити'  
 init\_data\_frame2()  
def connect\_to\_db(db\_name):  
 con = lite.connect(db\_name)  
 with con:  
 cur = con.cursor()  
 cur.execute("SELECT firstname FROM students")  
 cur.fetchone()[0]  
 return con  
def init\_data\_frame2():  
 *# Grid widgets in tab2* group\_label.grid(row=0, column=0, sticky='we')  
 group\_ent.grid(row=1, column=0, sticky='ew')last\_name\_label.grid(row=0, column=1, sticky='we')  
 last\_name\_ent.grid(row=1, column=1, sticky='we')  
 first\_name\_label.grid(row=0, column=2, sticky='we')  
 first\_name\_ent.grid(row=1, column=2, sticky='we')  
 father\_name\_label.grid(row=0, column=3, sticky='we')  
 father\_name\_ent.grid(row=1, column=3, sticky='we')find\_btn.grid(row=1, column=4, sticky='we')  
  
 scrollbar\_listbox.pack(expand=True, fill='y', side='right')  
 students\_listbox.pack(expand=True, fill='both', side='left')  
  
def write\_students\_to\_list(students\_listbox, students):  
 if not students:  
 messagebox.showinfo('Не знайдено', 'Студенти за даним запитом не знайдені!')  
 for row in students:  
 id = row[0]  
 student\_data = get\_student\_data\_by\_id(id)  
 student\_id = student\_data['student\_id']  
 firstname = student\_data['firstname']  
 lastname = student\_data['lastname']  
 fathername = student\_data['fathername']  
 group\_id = student\_data['group\_id']  
 group\_data = get\_group\_data\_by\_id(group\_id)  
 group\_name = group\_data['group\_name']  
 record = "%s %s %s %s" % (group\_name, lastname, firstname, fathername)  
 students\_listbox.students\_id.update({record: student\_id})  
 students\_listbox.insert(tk.END, record)  
def find\_students(event=None):  
 *""" Find students by group or lastname or firstname or fathername and write them to list """* last\_name = last\_name\_ent.get()  
 first\_name = first\_name\_ent.get()  
 father\_name = father\_name\_ent.get()  
 last\_name = last\_name and last\_name.capitalize()  
 first\_name = first\_name and first\_name.capitalize()  
 father\_name = father\_name and father\_name.capitalize()  
 last\_name\_ent.delete(0, tk.END)  
 first\_name\_ent.delete(0, tk.END)  
 father\_name\_ent.delete(0, tk.END)  
 if selected\_db.con:  
 con = selected\_db.con  
 with con:  
 cur = con.cursor()  
 cur.execute("PRAGMA case\_sensitive\_like = off")  
 execute\_str = "SELECT id FROM students WHERE "  
 groups = find\_by\_group()  
 if not(groups == NOT\_FOUND) and any([first\_name, last\_name, father\_name, groups]):  
 for i, row in enumerate(groups):  
 id = row[0]  
 if i == 0:  
 execute\_str += "group\_id=%s " %(id)  
 else:  
 execute\_str += "OR group\_id=%s " % (id)  
 if groups and (last\_name or first\_name or father\_name):  
 execute\_str += "AND "  
 if last\_name:  
 if first\_name:  
 if father\_name:  
 cur.execute(execute\_str + "lastname LIKE ? AND firstname LIKE ? AND fathername LIKE ?",("%"+ last\_name+"%", "%"+first\_name+"%", "%"+father\_name+"%",))  
 else:  
 cur.execute(execute\_str + "lastname LIKE ? AND firstname LIKE ?",("%"+ last\_name+"%","%"+ first\_name+"%" ,))  
 elif father\_name:  
 cur.execute(execute\_str + "lastname LIKE ? AND fathername LIKE ?",("%"+ last\_name+"%","%"+ father\_name+"%", ))  
 else:  
 cur.execute(execute\_str + "lastname LIKE ?",("%"+ last\_name+"%",))  
 elif first\_name:  
 if father\_name:  
 cur.execute(execute\_str + "firstname LIKE ? AND fathername LIKE ?",("%"+ first\_name+"%", "%"+ father\_name+"%",))  
 else:  
 cur.execute(execute\_str + "firstname LIKE ?", ("%"+ first\_name+"%",))  
 elif father\_name:  
 cur.execute(execute\_str + "fathername LIKE ?", ("%"+ father\_name+"%",))  
 else:  
 cur.execute(execute\_str)  
 data = cur.fetchall()  
 else:  
 data = []  
 students\_listbox.delete(0, tk.END)  
 write\_students\_to\_list(students\_listbox, data)  
 cur.close()  
def find\_by\_group():  
 *""" Find students by group and write them to list in tab3 """* group = group\_ent.get()  
 group\_ent.delete(0, tk.END)  
 if selected\_db.con:  
 con = selected\_db.con  
 with con:  
 cur = con.cursor()  
 if group:  
 cur.execute("SELECT id, name, course FROM groups WHERE name LIKE ?", ("%"+group+"%",))  
 groups = cur.fetchall()  
 return groups if groups else NOT\_FOUND  
 else:  
 return []  
def get\_student\_data\_by\_id(id):  
 *""" Return dict {group\_id, lastname, firstname, fathername} """* if selected\_db.con:  
 con = selected\_db.con  
 with con:  
 cur = con.cursor()  
 cur.execute("SELECT group\_id, lastname, firstname, fathername, id, image FROM students WHERE id=?", (id,))  
 student\_row = cur.fetchone()  
 data = {'group\_id': student\_row[0], 'lastname': student\_row[1],  
 'firstname': student\_row[2], 'fathername': student\_row[3],  
 'student\_id': student\_row[4], 'image': student\_row[5]}  
 return data  
 return []  
def get\_group\_data\_by\_id(id):  
 *""" Return dict {group\_id, lastname, firstname, fathername} """* if selected\_db.con:  
 con = selected\_db.con  
 with con:  
 cur = con.cursor()  
 cur.execute("SELECT id, name, course FROM groups WHERE id=?", (id,))  
 group\_row = cur.fetchone()  
 data = {'group\_id': group\_row[0],  
 'group\_name': group\_row[1],  
 'course': group\_row[2]}  
 return data  
 return []  
def onFrameConfigure(canvas, event):  
 *'''Reset the scroll region to encompass the inner frame'''* canvas.configure(scrollregion=canvas.bbox("all"))  
def show\_details(event):  
 *""" Show new window with student subjects """* record\_index = students\_listbox.curselection()  
 if record\_index:  
 record = students\_listbox.get(record\_index[0])  
 student\_id = students\_listbox.students\_id.get(record)  
 if not student\_detail\_windows.get(student\_id):  
 new\_window = create\_window(700, 300, False)  
 student\_detail\_windows.update({student\_id: new\_window})  
 new\_window.protocol("WM\_DELETE\_WINDOW", lambda window=new\_window, id=student\_id: on\_closing\_student\_window(new\_window, student\_id))  
 student\_data = get\_student\_data\_by\_id(student\_id)  
 *##### WIDGETS #####* new\_window.title("%s %s" %(student\_data['lastname'], student\_data['firstname']))  
 *##### CANVAS SCROLL #####* def \_on\_mousewheel(event):  
 canvas.yview\_scroll(-1 \* round((event.delta / 120)), "units")  
 *##### DATA #####* student\_subjects = get\_student\_subjects(student\_id)  
 tab\_parent = ttk.Notebook(new\_window, width=500)  
 path = "%s/%s" %(selected\_db.images\_path, student\_data['image'])  
 load = Image.open(path)  
 load.thumbnail((190,190), Image.ANTIALIAS)  
 render = ImageTk.PhotoImage(load)  
 img = ttk.Label(new\_window, image=render)  
 img.image = render  
 img.place(x=0, y=0)  
 report\_btn = ttk.Button(new\_window, text='Звіт (xlsx)', command=lambda id=student\_id:report(id))  
 report\_btn.place(x=20, y=200)  
 for i, day in enumerate(WEEKDAYS):  
 if student\_subjects[i]:  
 tab1 = ttk.Frame(tab\_parent)  
 tab\_parent.add(tab1, text=day)  
 canvas = tk.Canvas(tab1, width=600, bd=0, relief='ridge', highlightthickness=0)  
 canvas\_frame = tk.Frame(canvas, padx=10)  
 myscrollbarY = tk.Scrollbar(tab1, orient="vertical", command=canvas.yview)  
 canvas.configure(yscrollcommand=myscrollbarY.set)  
 myscrollbarY.pack(side="right", fill="y")  
 canvas.pack(side="right", fill='both')  
 canvas.create\_window(0, 0, anchor='nw', window=canvas\_frame)count = 0  
 col = 0  
 row = 0  
 for j, sub in enumerate(student\_subjects[i]):  
 sub\_name = sub[0]  
 sub\_time = sub[1]  
 time\_object = datetime.datetime.strptime(sub\_time, '%H:%M:%S') + DEFAULT\_SUBJECT\_TIMEDELTA  
 student\_datetime = get\_student\_datetime(student\_id, i, sub\_time, time\_object.strftime('%H:%M:%S'))  
dates\_set = {row[0] for row in student\_datetime}label\_day\_frame = ttk.Labelframe(canvas\_frame, text=sub\_name)  
 label\_day\_frame.grid(row=row, column=col)  
 col += 1  
 if col == 3:  
 row += 1  
 col = 0  
 scrollbar\_listbox = tk.Scrollbar(label\_day\_frame, orient="vertical")  
 dates\_listbox = tk.Listbox(label\_day\_frame, height=6,  
 yscrollcommand=scrollbar\_listbox.set)  
 scrollbar\_listbox.config(command=dates\_listbox.yview)  
 dates\_listbox.pack(side='left', fill='both')  
 scrollbar\_listbox.pack(side='right')  
 for date in dates\_set:  
 datetime\_object = datetime.datetime.strptime(date, '%Y-%m-%d')  
 day = datetime\_object.strftime('%d')  
 month = datetime\_object.strftime('%m')  
 year = datetime\_object.strftime('%Y')  
 record = "%s %s, %sр" % (day, MONTHS[int(month)].lower(), year)  
 dates\_listbox.insert(tk.END, record)  
 canvas\_frame.update\_idletasks()  
 canvas.config(scrollregion=canvas.bbox("all"))  
 tab\_parent.place(x=200, y=0)  
def get\_student\_subjects(student\_id):  
 if selected\_db.con:  
 con = selected\_db.con  
 with con:  
 cur = con.cursor()  
 student\_data = get\_student\_data\_by\_id(student\_id)  
 group\_id = student\_data['group\_id']  
 cur.execute("SELECT subject\_id FROM groups\_subjects WHERE group\_id=?", (group\_id,))  
 subjects = [ [] for i, day in enumerate(WEEKDAYS)]  
 for row in cur.fetchall():  
 sub\_id = row[0]  
 cur.execute("SELECT weekday, name, time FROM subjects WHERE id=?", (sub\_id,))  
 sub\_row = cur.fetchone()  
 subjects[sub\_row[0]].append([sub\_row[1], sub\_row[2]])  
 return subjects  
def get\_student\_datetime(id, weekday, start\_time, end\_time):  
 *""" Return [Y-m-d, H:M:S] """* if selected\_db.con:  
 con = selected\_db.con  
 with con:  
 cur = con.cursor()  
 cur.execute("SELECT date, time, weekday FROM datetime WHERE student\_id=? AND weekday=? AND time <= ? AND time >= ? ORDER BY date", (id, weekday, end\_time, start\_time, ))  
 return cur.fetchall()  
def report(student\_id):  
 student\_subjects = get\_student\_subjects(student\_id)  
 student\_data = get\_student\_data\_by\_id(student\_id)  
 if not os.path.exists(PATH\_FOR\_REPORTS):  
 os.makedirs(PATH\_FOR\_REPORTS)  
 full\_name = "%s/%s %s %s.xlsx" %(PATH\_FOR\_REPORTS, student\_data['lastname'], student\_data['firstname'], student\_data['fathername'])  
 workbook = xlsxwriter.Workbook(full\_name)  
 for i, day in enumerate(WEEKDAYS):  
 if student\_subjects[i]:  
 worksheet = workbook.add\_worksheet(day)  
 worksheet.set\_column(0, len(student\_subjects[i])-1,width=20)  
 col = 'A'  
 for j, sub in enumerate(student\_subjects[i]):  
 row = 1  
 sub\_name = sub[0]  
 sub\_time = sub[1]  
 time\_object = datetime.datetime.strptime(sub\_time, '%H:%M:%S') + DEFAULT\_SUBJECT\_TIMEDELTA  
 student\_datetime = get\_student\_datetime(student\_id, i, sub\_time, time\_object.strftime('%H:%M:%S'))  
 dates\_set = {row[0] for row in student\_datetime}  
 worksheet.write("%s%i" %(col, row), sub\_name)  
 row += 1  
 for date in dates\_set:  
 datetime\_object = datetime.datetime.strptime(date, '%Y-%m-%d')  
 day = datetime\_object.strftime('%d')  
 month = datetime\_object.strftime('%m')  
 year = datetime\_object.strftime('%Y')  
 record = "%s %s, %sр" % (day, MONTHS[int(month)].lower(), year)  
 worksheet.write("%s%i" %(col, row), record)  
 row += 1  
 col = chr(ord(col) + 1)  
 workbook.close()  
if \_\_name\_\_ == '\_\_main\_\_':  
 root = tk.Tk()  
 root.iconbitmap('lntu.ico')  
 root.title("LNTU students")  
 tk.Grid.rowconfigure(root, 0, weight=1)  
 tk.Grid.columnconfigure(root, 0, weight=1)  
 centerWindow(root, 640, 500)  
 root.resizable(False, False)tab\_parent = ttk.Notebook(root)  
 tab1 = ttk.Frame(tab\_parent)tab\_parent.add(tab1, text="Знайти студентів")tab\_parent.pack(expand=1, fill='both')  
 frame1 = tk.Frame(tab1)  
 frame1.place(x=20, y=10)  
 frame2 = tk.Frame(tab1)  
 frame2.place(x=20, y=50)  
 frame3 = tk.Frame(tab1)  
 frame3.place(x=20, y=100)  
 *##### WIDGETS FRAME 1 #####* select\_db\_label = ttk.Label(frame1, text=u'Виберіть БД')  
 select\_db\_label.grid(row=0, column=0, columnspan=3)  
 select\_db\_btn = ttk.Button(frame1, text=u'Вибрати', command=select\_db)  
 select\_db\_btn.grid(row=0, column=3)  
 *##### WIDGETS FRAME 2 #####* last\_name\_label = ttk.Label(frame2, text=u"Прізвище")  
 last\_name\_ent = ttk.Entry(frame2)  
 first\_name\_label = ttk.Label(frame2, text=u"Ім'я")  
 first\_name\_ent = ttk.Entry(frame2)  
 father\_name\_label = ttk.Label(frame2, text=u"По батькові")  
 father\_name\_ent = ttk.Entry(frame2)  
 group\_label = ttk.Label(frame2, text=u"Група")  
 group\_ent = ttk.Entry(frame2)  
 find\_btn = ttk.Button(frame2, text=u'Знайти', command=find\_students)  
 scrollbar\_listbox = tk.Scrollbar(frame3, orient="vertical")  
 students\_listbox = tk.Listbox(frame3, width=95, height=20,  
 yscrollcommand=scrollbar\_listbox.set)  
 students\_listbox.students\_id = {}  
 students\_listbox.bind("<<ListboxSelect>>", show\_details)  
 scrollbar\_listbox.config(command=students\_listbox.yview)  
 root.bind('<Return>', find\_students)  
 root.protocol("WM\_DELETE\_WINDOW", lambda window=root: on\_closing(window))  
 root.mainloop()