**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**(МОСКОВСКИЙ ПОЛИТЕХ)**

Транспортный факультет

Направление подготовки: 01.03.02 «Прикладная математика и информатика»

Отчет по Лабораторной работе №8

“Работа с характерными точками”

Выполнил:

Студент:

Группа: 211-172

Тимошенко Ф.Р.

Проверил:

Кожухов Д.А.

Дата: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Москва 2024

## 1. Введение

Целью данного программного средства является разработка системы, способной автоматически находить лица на фотографиях. Для этого используется каскад Хаара для обнаружения лиц, обучение распознавателя на изображениях лиц и последующее применение обученной модели для обнаружения лиц на других фотографиях.

## 2. Реализация

Проект состоит из двух основных файлов, которые реализуют графический интерфейс на основе PyQt6 для отображения изображений и выполнения распознавания лиц на них с использованием OpenCV. Далее приводится подробное описание всех классов, их методов и атрибутов.

## Файл 1: window.py

Класс ImageWindow

Этот класс наследуется от QMainWindow и представляет собой главное окно приложения, в котором размещаются различные виджеты для отображения и обработки изображений.

**Атрибуты:**

* central\_widget (QWidget): Центральный виджет окна.
* layout (QHBoxLayout): Основной горизонтальный макет для размещения виджетов.
* image\_layout (QVBoxLayout): Вертикальный макет для размещения виджетов с изображениями.
* image\_label1 (QLabel): Метка для отображения первого изображения.
* image\_label2 (QLabel): Метка для отображения второго изображения.
* button\_layout (QVBoxLayout): Вертикальный макет для размещения кнопок и полей ввода.

**Методы:**

* \_\_init\_\_(self): Инициализирует главное окно, задает макеты и виджеты.
* update\_images1(self, image\_path1): Обновляет первое изображение в интерфейсе.
* img\_hide(self): Скрывает оба изображения.
* update\_images2(self, image\_path2): Обновляет второе изображение в интерфейсе.
* update\_button(self): Создает и размещает кнопки и поля ввода в интерфейсе.

Файл 2: main.py

Класс Mywindow

Этот класс наследуется от ImageWindow и добавляет функциональность для загрузки изображений и распознавания лиц на них.

**Атрибуты:**

* initial\_path (str): Путь к загруженному изображению.
* scale (float): Масштабный коэффициент для распознавания лиц.
* minNeighbors (int): Параметр minNeighbors для функции распознавания лиц.

**Методы:**

* \_\_init\_\_(self): Инициализирует окно и задает значения по умолчанию для атрибутов.
* on\_button1\_clicked(self): Обрабатывает нажатие кнопки для загрузки изображения и запуска распознавания лиц.
* on\_button\_find\_clicked(self): Обрабатывает нажатие кнопки для выполнения распознавания лиц с новыми параметрами.
* download\_img(self, i): Загружает изображение с диска.
* faces\_recognition(self): Выполняет распознавание лиц на загруженном изображении и отображает результат.

## Шаги выполнения программы:

1. **Инициализация интерфейса**:
   * При запуске программы создается экземпляр класса Mywindow, который наследуется от ImageWindow.
   * В ImageWindow инициализируются основные компоненты интерфейса: центральный виджет, макеты для размещения элементов, метки для отображения изображений, кнопки и поля ввода.
2. **Отображение главного окна**:
   * Главный интерфейс с кнопками и полями ввода отображается пользователю.
   * В интерфейсе предусмотрены две кнопки: "Выбрать изображение" и "Найти лица на изображении", а также два поля ввода для настройки параметров распознавания лиц: scaleFactor и minNeighbors.
3. **Загрузка изображения**:
   * При нажатии на кнопку "Выбрать изображение" вызывается метод on\_button1\_clicked.
   * Метод открывает диалоговое окно для выбора изображения с диска. Путь к выбранному изображению сохраняется в атрибуте initial\_path.
   * Загруженное изображение отображается в метке image\_label1 с помощью метода update\_images1.
4. **Настройка параметров распознавания лиц**:
   * Пользователь может ввести значения для параметров scaleFactor и minNeighbors в соответствующие поля ввода.
   * При нажатии кнопки "Найти лица на изображении" вызывается метод on\_button\_find\_clicked.
   * Метод проверяет введенные значения и обновляет атрибуты scale и minNeighbors.
5. **Распознавание лиц**:
   * Метод faces\_recognition выполняет распознавание лиц на загруженном изображении с использованием OpenCV.
   * Загружается изображение, преобразуется в оттенки серого, и на нем выполняется распознавание лиц с использованием метода detectMultiScale из OpenCV.
   * Для каждого распознанного лица на изображении рисуется прямоугольник.
   * Обработанное изображение сохраняется на диск и отображается в метке image\_label2 с помощью метода update\_images2.

**Подробное описание методов и их взаимодействия:**

* **ImageWindow**:
  + \_\_init\_\_: Создает и настраивает основные элементы интерфейса.
  + update\_images1: Отображает первое изображение.
  + img\_hide: Скрывает оба изображения.
  + update\_images2: Отображает второе изображение.
  + update\_button: Создает кнопки и поля ввода в интерфейсе.
* **Mywindow**:
  + \_\_init\_\_: Инициализирует атрибуты для пути к изображению и параметры распознавания лиц.
  + on\_button1\_clicked: Обрабатывает загрузку изображения и вызывает распознавание лиц.
  + on\_button\_find\_clicked: Обрабатывает изменение параметров и вызывает распознавание лиц.
  + download\_img: Открывает диалоговое окно для выбора изображения.
  + faces\_recognition: Выполняет распознавание лиц на изображении, сохраняет и отображает результат.

## 4. Листинг кода

“window.py”

|  |
| --- |
| from PyQt6.QtWidgets import QComboBox, QLineEdit,QSlider, QApplication, QMainWindow, QLabel, QVBoxLayout, QWidget, QPushButton, QHBoxLayout  from PyQt6.QtGui import QPixmap  from PyQt6.QtCore import Qt  class ImageWindow(QMainWindow):      def \_\_init\_\_(self):          super().\_\_init\_\_()          self.setWindowTitle("Изображения")          self.setGeometry(50, 50, 400, 400)          self.central\_widget = QWidget()          self.setCentralWidget(self.central\_widget)          self.layout = QHBoxLayout()          self.central\_widget.setLayout(self.layout)          self.image\_layout = QVBoxLayout()          self.layout.addLayout(self.image\_layout)          self.image\_label1 = QLabel()          # self.label1\_title = QLabel('До обработки')          self.image\_label2 = QLabel()          # self.label2\_title = QLabel('После обработки')          #self.update\_images1("stuff/images/white.jpg")          #self.update\_images2("stuff/images/white.jpg")          # self.image\_layout.addWidget(self.label1\_title, alignment=Qt.AlignmentFlag.AlignCenter)          self.image\_layout.addWidget(self.image\_label1, alignment=Qt.AlignmentFlag.AlignCenter)          # self.label1\_title.hide()          # self.image\_layout.addWidget(self.label2\_title, alignment=Qt.AlignmentFlag.AlignCenter)          self.image\_layout.addWidget(self.image\_label2, alignment=Qt.AlignmentFlag.AlignCenter)          self.button\_layout = QVBoxLayout()          self.layout.addLayout(self.button\_layout)          self.update\_button()          self.img\_hide()          self.show()      def update\_images1(self, image\_path1):          #self.label1\_title.show()          pixmap1 = QPixmap(image\_path1)          scaled\_pixmap1 = pixmap1.scaled(400, 400, Qt.AspectRatioMode.KeepAspectRatio)          self.image\_label1.setPixmap(scaled\_pixmap1)          self.image\_label1.show()          self.update()      def img\_hide(self):          # self.label1\_title.hide()          self.image\_label1.hide()          # self.label2\_title.hide()          self.image\_label2.hide()      def update\_images2(self, image\_path2):          # self.label2\_title.show()          pixmap2 = QPixmap(image\_path2)          scaled\_pixmap2 = pixmap2.scaled(400, 400, Qt.AspectRatioMode.KeepAspectRatio)          self.image\_label2.setPixmap(scaled\_pixmap2)          self.image\_label2.show()          self.update()      def update\_button(self):          #Кнопка для загрузки видео          self.button1 = QPushButton("Выбрать изображение")          self.button1.clicked.connect(self.on\_button1\_clicked)          self.button\_layout.addWidget(self.button1)          #ввод scale          self.inputScale = QLineEdit(self)          self.inputScale.setPlaceholderText('scaleFactor >1 and <5')          self.button\_layout.addWidget(self.inputScale)          #ввод minNeighbors          self.inputNeigh = QLineEdit(self)          self.inputNeigh.setPlaceholderText('minNeighbors >1 and <15')          self.button\_layout.addWidget(self.inputNeigh)          #кнопка найти лица          self.button\_find = QPushButton("Найти лица на изображении")          self.button\_find.clicked.connect(self.on\_button\_find\_clicked)          self.button\_layout.addWidget(self.button\_find) |

“main.py”

|  |
| --- |
| import cv2  import sys  from window import ImageWindow  from PyQt6.QtWidgets import QApplication, QFileDialog  save\_process\_path = 'stuff/saved/save\_proc.jpg'  class Mywindow(ImageWindow):      def \_\_init\_\_(self):          super().\_\_init\_\_()          self.initial\_path = ''          self.scale=1.8          self.minNeighbors=3      def on\_button1\_clicked(self):          self.download\_img(1)          self.faces\_recognition()      def on\_button\_find\_clicked(self):          if self.inputScale.text():              if float(self.inputScale.text())>1 and float(self.inputScale.text())<5:                  self.scale = float(self.inputScale.text())          if self.inputNeigh.text():              if int(self.inputNeigh.text())>1 and int(self.inputNeigh.text())<15:                  self.minNeighbors=int(self.inputNeigh.text())          print(self.scale, '   ', self.minNeighbors)          self.faces\_recognition()      def download\_img(self, i):          try:              self.initial\_path, \_ = QFileDialog.getOpenFileName(self, "Выберите изображение", "", "Изображения (\*.png \*.jpg \*.jpeg)")              if not self.initial\_path:                  raise FileNotFoundError("Путь к изображению не был выбран.")              if i == 1:                  self.update\_images1(self.initial\_path)              else:                  raise FileNotFoundError("Куда ты хочешь картинку?")          except Exception as e:              print("Ошибка при загрузке изображения", e)              return None      def faces\_recognition(self):          try:              img = cv2.imread(self.initial\_path)              gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)              faces = cv2.CascadeClassifier('faces.xml')              results = faces.detectMultiScale(gray, scaleFactor=self.scale, minNeighbors=self.minNeighbors)              for(x, y, w, h) in results:                  cv2.rectangle(img, (x,y),(w+x, y+h),(0,255,0), thickness=2)              cv2.imwrite(save\_process\_path, img)              self.update\_images2(save\_process\_path)          except Exception as e:              print('Ошибка при попытке распознать лица: ')              return None  if \_\_name\_\_ == "\_\_main\_\_":      app = QApplication(sys.argv)      window = Mywindow()      sys.exit(app.exec()) |