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 №6

“ Обнаружение движения ”

Выполнил:

Студент:

Группа: 211-172

Тимошенко Ф.Р.

Проверил:

Кожухов Д.А.

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

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

Москва 2024

## 1. Введение

Целью данного задания было создание программного средства для обнаружения и обозначения перемещающихся объектов в видеопотоке с веб-камеры или из видеофайла. Программа должна отображать два изображения в одном окне: исходное изображение и его обработанный вариант с выделенными перемещающимися объектами.

## 2. Реализация

Программа разработана с использованием библиотек PyQt6 и OpenCV. Она состоит из двух классов: ImageWindow и MyWindow.

Класс ImageWindow:

Атрибуты:

* image\_label1 и image\_label2: QLabel для отображения изображений до и после обработки.
* central\_widget: QWidget, центральный виджет главного окна.
* layout: QHBoxLayout для размещения виджетов.
* image\_layout: QVBoxLayout для размещения меток с изображениями.
* button\_layout: QVBoxLayout для размещения кнопок.

Методы:

* \_\_init\_\_: конструктор класса, инициализирующий главное окно и его компоненты.
* update\_images1 и update\_images2: обновление изображений до и после обработки.
* img\_hide: скрытие меток с изображениями.
* update\_button: создание и настройка кнопок для выбора видео и использования камеры.

Класс MyWindow (расширение ImageWindow):

Дополнительные атрибуты:

* initial\_path: путь к исходному изображению или видео.
* video\_source: источник видео.
* capture: объект захвата видео.
* frame: текущий кадр из видео.
* background\_subtractor: объект для вычитания фона.
* cap: объект захвата видео через OpenCV.

Дополнительные методы:

* on\_button\_video\_clicked: загрузка видео и начало его обработки.
* on\_button\_web\_clicked: начало обработки видео с веб-камеры.
* download\_video: загрузка видео из файла или камеры.
* video\_process1: обработка видео, включающая вычитание фона и отображение результата.

## 3. Описание работы программы

1. Пользователь выбирает режим работы (обработка изображения или видео).
2. В зависимости от выбора, пользователь может загрузить изображение или видео файл, либо использовать камеру.
3. После загрузки файла или выбора камеры, программа выполняет выбранную операцию обработки (вычитание фона для видео).
4. Результат обработки отображается в графическом интерфейсе программы.

## 4. Листинг кода

“window.py”

|  |
| --- |
| from PyQt6.QtWidgets import QComboBox, 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(200, 200, 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(700, 700, Qt.AspectRatioMode.KeepAspectRatio)          self.image\_label2.setPixmap(scaled\_pixmap2)          self.image\_label2.show()          self.update()      def update\_button(self):          #Кнопка для загрузки видео          self.button\_video = QPushButton("Выбрать видео")          self.button\_video.clicked.connect(self.on\_button\_video\_clicked)          self.button\_layout.addWidget(self.button\_video)          #Кнопка для вебки          self.button\_web = QPushButton("Использовать камеру")          self.button\_web.clicked.connect(self.on\_button\_web\_clicked)          self.button\_layout.addWidget(self.button\_web) |

“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.video\_source = None          self.capture = None          self.frame = None          self.background\_subtractor = cv2.createBackgroundSubtractorMOG2(history=1000, varThreshold=32, detectShadows=True)          self.cap = None        def on\_button\_video\_clicked(self):          self.download\_video(1)          self.video\_process1()        def on\_button\_web\_clicked(self):          self.cap = cv2.VideoCapture(0)          self.video\_process1()        def download\_video(self, i):          try:              self.initial\_path, \_ = QFileDialog.getOpenFileName(self, "Выберите видео", "", "Видео (\*.mp4)")              if not self.initial\_path:                  raise FileNotFoundError("Путь к видео не был выбран.")              if i == 1:                  self.cap = cv2.VideoCapture(self.initial\_path)                  \_, img = self.cap.read()                  cv2.imwrite(save\_process\_path, img)                  self.update\_images1(save\_process\_path)              else:                  raise FileNotFoundError("Куда ты хочешь картинку?")          except Exception as e:              print("Ошибка при загрузке видео", e)              return None      def video\_process1(self):          # Создание объекта для вычитания фона          while True:              ret, frame = self.cap.read()              if not ret:                  self.cap.set(cv2.CAP\_PROP\_POS\_FRAMES, 0)                  continue              cv2.imwrite(save\_process\_path, frame)              self.update\_images1(save\_process\_path)              gray\_frame = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)              fg\_mask = self.background\_subtractor.apply(gray\_frame)              fg\_mask = cv2.threshold(fg\_mask, 240, 255, cv2.THRESH\_BINARY)[1]              contours, \_ = cv2.findContours(fg\_mask, cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)              for contour in contours:                  if cv2.contourArea(contour) > 300:                      x, y, w, h = cv2.boundingRect(contour)                      cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)              original\_image = cv2.cvtColor(frame, cv2.COLOR\_BGR2RGB)              cv2.imwrite(save\_process\_path, frame)              self.update\_images2(save\_process\_path)              if cv2.waitKey(1) & 0xFF == ord('q'):                  break  if \_\_name\_\_ == "\_\_main\_\_":      app = QApplication(sys.argv)      window = Mywindow()      sys.exit(app.exec()) |