Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«Уральский федеральный университет имени первого Президента России Б.Н. Ельцина» (УрФУ) Институт фундаментального образования

Кафедра « »

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель курсового проектирования:  
Мирвода Сергей Геннадьевич

Члены комиссии\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата защиты \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

по теме: Как различить чужих и своих.

По дисциплине: Языки и методы программирования

Студенты:

Высоцкий Семён Артёмович

Казанцева Анна Сергеевна

Смирнов Никита Сергеевич

Группа: РИ-311055

Екатеринбург 2024

**Оглавление**

[**1.** **Введение.** 3](#_Toc156041197)

[**2.** **Радел Проектировщика** 5](#_Toc156041198)

[**Диаграммы** 5](#_Toc156041199)

[**3.** **Раздел Разработчика** 7](#_Toc156041200)

[**4.** **Заключение** 10](#_Toc156041201)

[**СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ** 11](#_Toc156041202)

[**Приложения** 12](#_Toc156041203)

[**Приложение А** 12](#_Toc156041204)

# **Введение.**

По [Techopedia](https://www.techopedia.com/definition/32071/facial-recognition) [1] можно описать распознавание лица, как биометрическая программная технология, которая работает при помощи анализа и сравнений идентифицировать или верифицировать человека по контору лица. В нашем случае анализ фотографии на наличие глаз.

Видение:

Организации, которые заинтересованы в разработке систем безопасности и контроля доступа, если вы хотите, чтобы ваша надежность повысилась, и идентификация личности стала более удобной, то наша программа "свой-чужой" станет для вас открытием. На данный момент она является инновационным способом идентификации людей по фото и обеспечения контроля доступа.

В отличие от других программ, наш продукт обладает способностью проанализировать фото на наличие глаз и в зависимости от их наличия пропустить или заблокировать доступ.

Цель:

Данный проект разработан для обеспечения безопасности путем обнаружения лиц и анализа изображений. Проект включает в себя использование технологий машинного обучения, компьютерного зрения и голосового взаимодействия.

По итогу должен появиться продукт Робота "Свой-Чужой", который помогает определить, принадлежит ли фотография человеку или нет. Пользователь загружает фотографию и включает код. Программа начинает анализировать её. Сначала она выделяет область лица и ищет глаза. Если глаза обнаружены, то наш робот выводит результаты на экран. Если глаза не найдены или есть проблемы с фотографией, запрос отклоняется.

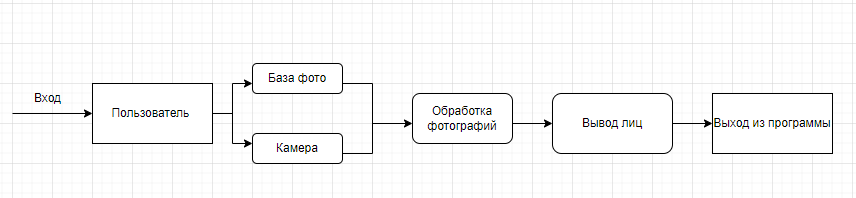


Рисунок 1. – Функциональная диаграмма продукта (как он видится)

Для более простого и понятного для нашей группы распределения обязанностей и постановки задач, руководитель создал на сайте GitHub [3] репозиторий, куда будет выставлять файлы для нашего проекта, также создаст высококачественные задачи для себя, Разработчика и Проектировщика, это и есть подключение issues. Наши профессионалы будут туда выставлять результаты, что для Руководителя сократит время для поисков нужной ему информации.

# **Радел Проектировщика**

Проектировщик описал и обосновал архитектуру нашего Робота «Свой-чужой»: источники изображения, модуль обнаружения лиц, модуль различения своих и чужих, интерфейс пользователя, хранение данных.

Наш проект экая система, который включает в себя наш сам продукт - робот «Свой-чужой», базу, состоящую из фотографий/картинок, голосовой движок.

Робот «Свой-чужой» - главный «мозг» проекта, который видит и общается

База изображений хранит в себе фотографии/картинки, которые робот использует для обучения и распознавания.

Голосовой движок отвечает за голосовое общение с роботом

Для того, чтобы воспользоваться нашей программой, достаточно процессора не слишком старого, лучше, конечно, если она двухъядерный. Памяти нужно минимум 4 ГБ. Потребуется видеокарта, которая поддерживает графический стандарт OpenGL 2.0 или выше. Из операционной системы точно должны быть Windows 10 (предпочтительно 64-бит), Linux или macOS 10.14 и более.

## **Диаграммы**

**Архитектурная диаграмма:** представляет собой структуру системы и взаимодействие между ее компонентами.

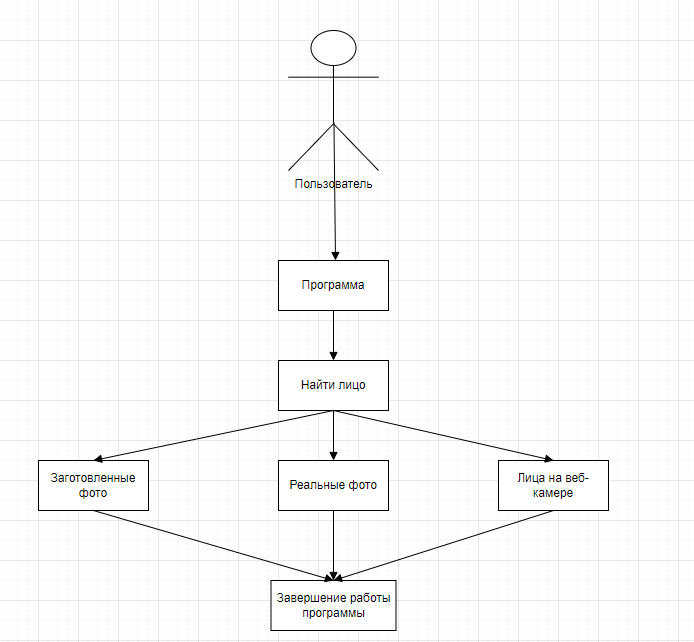


Рисунок 2. – Архитектурная диаграмма продукта

**Функциональная диаграмма:** показывает основные функции системы, такие как обработка изображений, обнаружение лиц и воспроизведение голосовых сообщений.

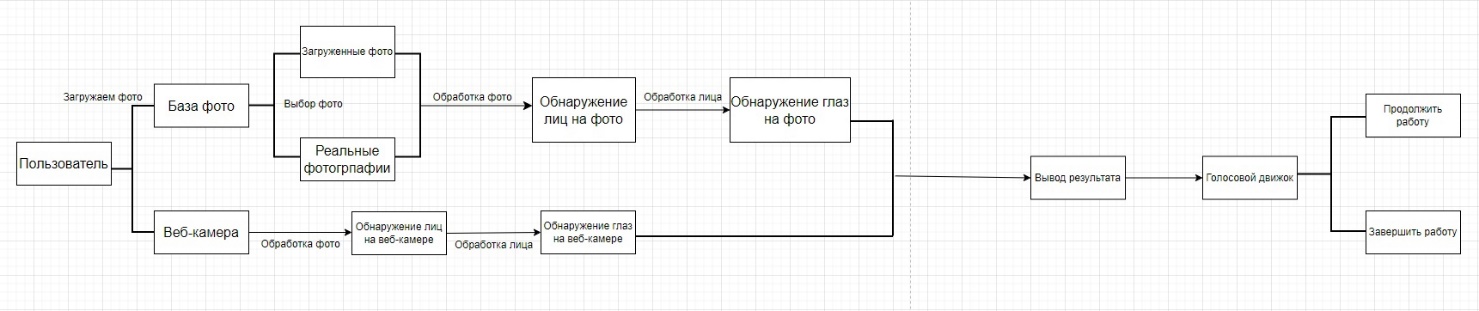


Рисунок 3. - Функциональная диаграмма проекта

**Диаграмма последовательности:** описывает взаимодействие между компонентами во времени при обнаружении лиц и выводе решения.

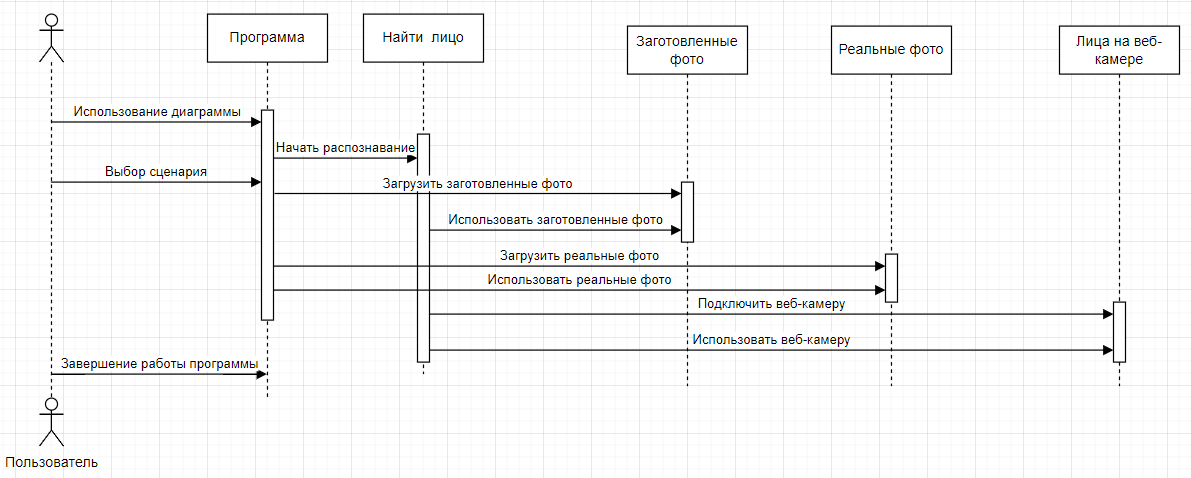


Рисунок 4. - Диаграмма последовательности проекта

# **Раздел Разработчика**

Наш разработчик выбрал такой язык программирования как Python. По его мнению, для разработки нашего программного обеспечения требуются надежные библиотеки управления, простой синтаксис и быстрые алгоритмы обработки. Всё это имеется на языке Python.

Разработчик использовал версию программы IDLE Python 3.12 (64-bit)

Изначально наш код обладал работоспособностью версии MVP. То есть он не имел выбора, минимально голоса нашей программы, мог запуститься и показывал принцип работы программы.

После появился последовательный выбор между изображением с экрана, то есть камерой, и фотографией. При выборе фотографии появился еще один выбор – базовые фотографии – те, которые были нам даны или реальные – те, которые мы добавляли сами.

В последующей оптимизации и разработки наш профессионал добавил всё это по просьбе Руководителя.

Также наш разработчик показал диаграмму пакетов UML для нашего продукта:

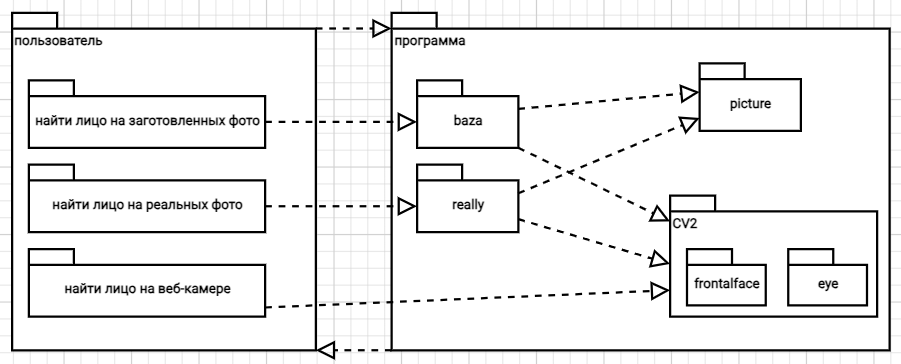


Рисунок 5. – Диаграмма пакетов UML

Здесь элементы кода сгруппированы по «кучкам», что дает нам краткое содержание его алгоритма.

У пользователя имеется 3 выбора, который каждый ведет к использованию библиотеки CV2 распознаванию лица, с возможностью реализации фото или видео.

В нашем случае основной пакет CV2 (библиотека с машиной) зависит от таких сущностей как frontalface и eye. Вспомогательный независимый пакет pyttsx3, используемый для озвучки.

В нашем проекте мы использовали такие библиотеки, как os, time, numpy, datetime, pyttsx3, cv2. Каждая из них выполняет свою функцию: работа с операционной системой и файлами (os), создание пауз или просто работа с временем (time), работа с матрицами и массивами (numpy), работа с датами – временем (datetime), голосовой движок (pyttsx3), и конечно же само распознавание лица (cv2).

Алгоритм:

Наш пользователь в праве выбрать это будет камера устройства или же это будет фотография.



Рисунок 6. – Выбор между камерой и фотографией

Если пользователь выбрал камеру устройства, то начинает работать библиотека cv2. Тогда пользователь может уже себя увидеть. Наш Робот начинает обрабатывать видеопоток, происходит поиск лица. Как лицо будет найдено, пользователь увидит это на выводе. Если все хорошо, то пользователь нажимает на кнопку «Q»

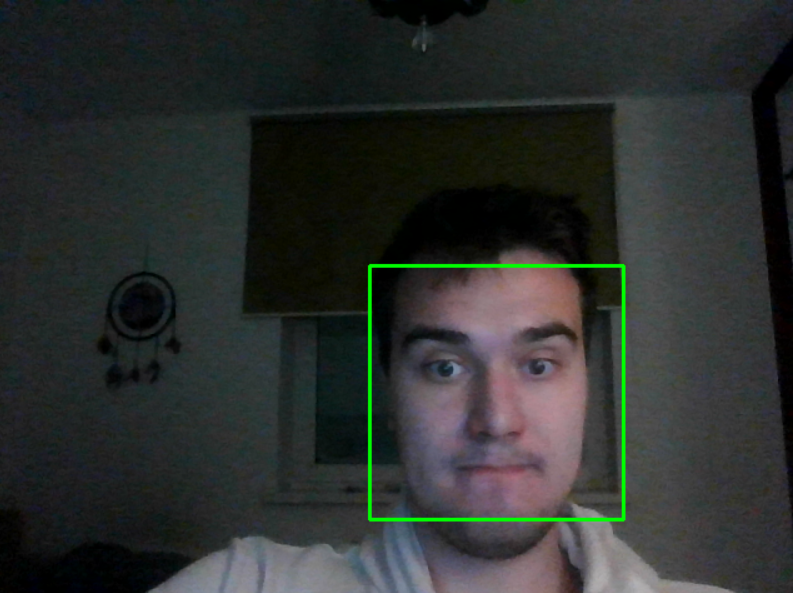


Рисунок 7. Работа с видеопотоком

Если же пользователь на вводе выбирает фотографию, то у него появляется новый выбор: базовые фотографии или же реальные фотографии. Снова же подключаются библиотеки cv2 и pyttsx3. Робот переходит в нужную директорию, которую выбрал пользователь. И не зависимо от того, в какой директории находится Робот, будет действовать одним алгоритмом:

Первое что услышит пользователь - "Если Вы человек, то Вам не о чем беспокоиться. Повернитесь лицом к турели."

После этого пользователь увидит это изображение на экране (пауза 3 сек) и может пронаблюдать, как наш Робот будет обрабатывать изображение.

Робот произнесет - "Лицо зафиксировано. Производится дополнительная проверка.". После этой фразы пользователь заметит, как Робот будет искать глаза.

После этого Робот вынесет на экран в зависимости от того, что будет во время последней обработки: либо "Глаза найдены." – как текст на экране и "Заходи, бродяга." – голосовой движок, либо "Глаза не найдены." и "Активировать протокол Земля пухом - Царство небесное." Также после всех выборов программа даёт возможность продолжить анализ фотографий.



Рисунок 8. Завершить или продолжить



Рисунок 9. Работа с фотографией

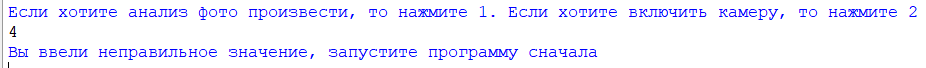
При некорректном значении Робот выдаст - "Вы ввели неправильное значение, запустите программу сначала"

Рисунок 10. – Некорректное значение

# **Заключение**

Наш Робот «Свой-Чужой» представляет собой простую, но эффективную систему для анализа лиц и глаз на изображениях или видеопотоке с веб-камеры. В использовании легко выбирать режим работы, проводить анализ фотографий с различными настройками и получать голосовые уведомления о результатах. В целом, это удобный инструмент для быстрого обнаружения движений и лиц в реальном времени.

Повторюсь, что в отличие от других программ, наш продукт обладает способностью проанализировать фото на наличие глаз и в зависимости от их наличия пропустить или заблокировать доступ

Рисунок 11. Процесс определения лица и глаз



# **СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ**

1. Распознавание лиц [Электронный ресурс]. - URL: <https://www.techopedia.com/definition/32071/facial-recognition>
2. Product Vision - как создать убедительное видение продукта [Электронный ресурс]. - URL: <https://admitad.pro/ru/blog/product-vision>
3. GitHub: friend- or foe [Электронный ресурс]. - URL: <https://github.com/babkaAnna/friend-or-foe/invitations>
4. ЛИ ВОГАН. Python для хакеров. Нетривиальные задачи и проекты. — СПб.: Питер, 2023. — 384 с.: ил. — (Серия «Библиотека программиста»).

# **Приложения**

## **Приложение А**

import os

import time

import numpy as np

from datetime import datetime

import pyttsx3

import cv2 as cv

print ("Если хотите анализ фото произвести, то нажмите 1. Если хотите включить камеру, то нажмите 2")

k = int(input())

if k==1:

print ("Если хотите базовые фотки, то нажмите 1. Если хотите реальные фотки, то нажмите 2")

k = int(input())

if k==1:

k = "baza"

elif k==2:

k = "really"

else:

print ("Вы ввели неправильное значение, запустите программу сначала")

exit()

engine = pyttsx3.init()

engine.setProperty('rate', 145)

engine.setProperty('volume', 1.0)

root\_dir = os.path.abspath('.')

gunfire\_path = os.path.join(root\_dir, 'gunfire.wav')

tone\_path = os.path.join(root\_dir, 'tone.wav')

path= "D:/Python/Lib/site-packages/cv2/data/"

face\_cascade = cv.CascadeClassifier(path + 'haarcascade\_frontalface\_default.xml')

assert not face\_cascade.empty()

eye\_cascade = cv.CascadeClassifier(path + 'haarcascade\_eye.xml')

assert not eye\_cascade.empty()

os.chdir('C:/Users/uncle/Desktop/курсач/picture\_'+k)

contents = sorted(os.listdir())

for image in contents:

print(f"\nОбнаружено движение...{datetime.now()}") # Обнаружено движение

discharge\_weapon = True

######Голос о том, что зафиксировано движение

engine.say("Если Вы человек, то Вам не о чем беспокоиться. \

Повернитесь лицом к турели.")

engine.runAndWait()

time.sleep(1)

#####Импортирование и вывод фото

img\_gray = cv.imread(image, cv.IMREAD\_GRAYSCALE)

qwerty = cv.imread(image, cv.IMREAD\_GRAYSCALE)

height, width = np.shape(img\_gray)

cv.imshow(f'1 {image}', img\_gray)

cv.waitKey(3000)

cv.destroyWindow(f'1 {image}')

#####Запись всех возможных лиц

face\_rect\_list = []

face\_rect\_list.append(face\_cascade.detectMultiScale(image=img\_gray, scaleFactor=1.1, minNeighbors=5))

#####Вывод всех возможных лиц

for j in face\_rect\_list:

for i in j:

cv.rectangle(qwerty, (i[0], i[1]), (i[0]+i[2], i[1]+i[3]),(255,255,255), 2)

cv.imshow(f'2 {image}', qwerty)

cv.waitKey(3000)

cv.destroyWindow(f'2 {image}')

#####Голос о том, что зафиксированно лицо

engine.say("Лицо зафиксированно. \

Производится дополнительная проверка.")

engine.runAndWait()

time.sleep(1)

#####Проверка на наличие глаз в лицах

print(f"Для достоверности ведётся поиск глаз в {image}")

for rect in face\_rect\_list:

for (x, y, w, h) in rect:

rect\_4\_eyes = img\_gray[y:y+h, x:x+w]

eyes = eye\_cascade.detectMultiScale(image=rect\_4\_eyes, scaleFactor=1.05, minNeighbors=2)

for (xe, ye, we, he) in eyes:

center = (int(xe + 0.5 \* we), int(ye + 0.5 \* he))

radius = int((we + he) / 3)

cv.circle(rect\_4\_eyes, center, radius, 255, 2)

cv.rectangle(img\_gray, (x, y), (x+w, y+h), (255, 255, 255), 2)

discharge\_weapon = False

cv.imshow(f'Motion detected {image}', img\_gray)

cv.waitKey(3000)

cv.destroyWindow(f'Motion detected {image}')

break

#####Вывод решение

if discharge\_weapon == False:

print(f"Глаза найдены.")

cv.putText(img\_gray, 'Open ^\_^', (int(width / 2) - 20, int(height / 2)), cv.FONT\_HERSHEY\_PLAIN, 3, 255, 3)

engine.say("Заходи, бродяга.")

engine.runAndWait()

time.sleep(1)

cv.imshow('Detected Faces', img\_gray)

cv.waitKey(2000)

cv.destroyWindow('Detected Faces')

time.sleep(1)

else:

print(f"Глаза не найдены.")

cv.putText(img\_gray, 'Fire!', (int(width / 2) - 20, int(height / 2)), cv.FONT\_HERSHEY\_PLAIN, 3, 255, 3)

engine.say("Активировать протокол Земля пухом - Царство небесное.")

engine.runAndWait()

time.sleep(1)

cv.imshow('Mutant', img\_gray)

cv.waitKey(2000)

cv.destroyWindow('Mutant')

time.sleep(1)

engine.stop()

print ("Введите 0, если хотите продолжить, или 1, если хотите завершить")

k = int(input())

if k == 1:

break

elif k == 0:

continue

else:

print ("Введен неверный символ")

elif k==2:

path= "D:/Python/Lib/site-packages/cv2/data/"

face\_cascade = cv.CascadeClassifier(path + 'haarcascade\_frontalface\_default.xml')

cap = cv.VideoCapture(0)

while True:

\_, frame = cap.read()

face\_rects = face\_cascade.detectMultiScale(frame, scaleFactor=1.2, minNeighbors=3)

for (x, y, w, h) in face\_rects:

cv.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)

cv.imshow('frame', frame)

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

break

cap.release()

cv.destroyAllWindows()

else:

print ("Вы ввели неправильное значение, запустите программу сначала")