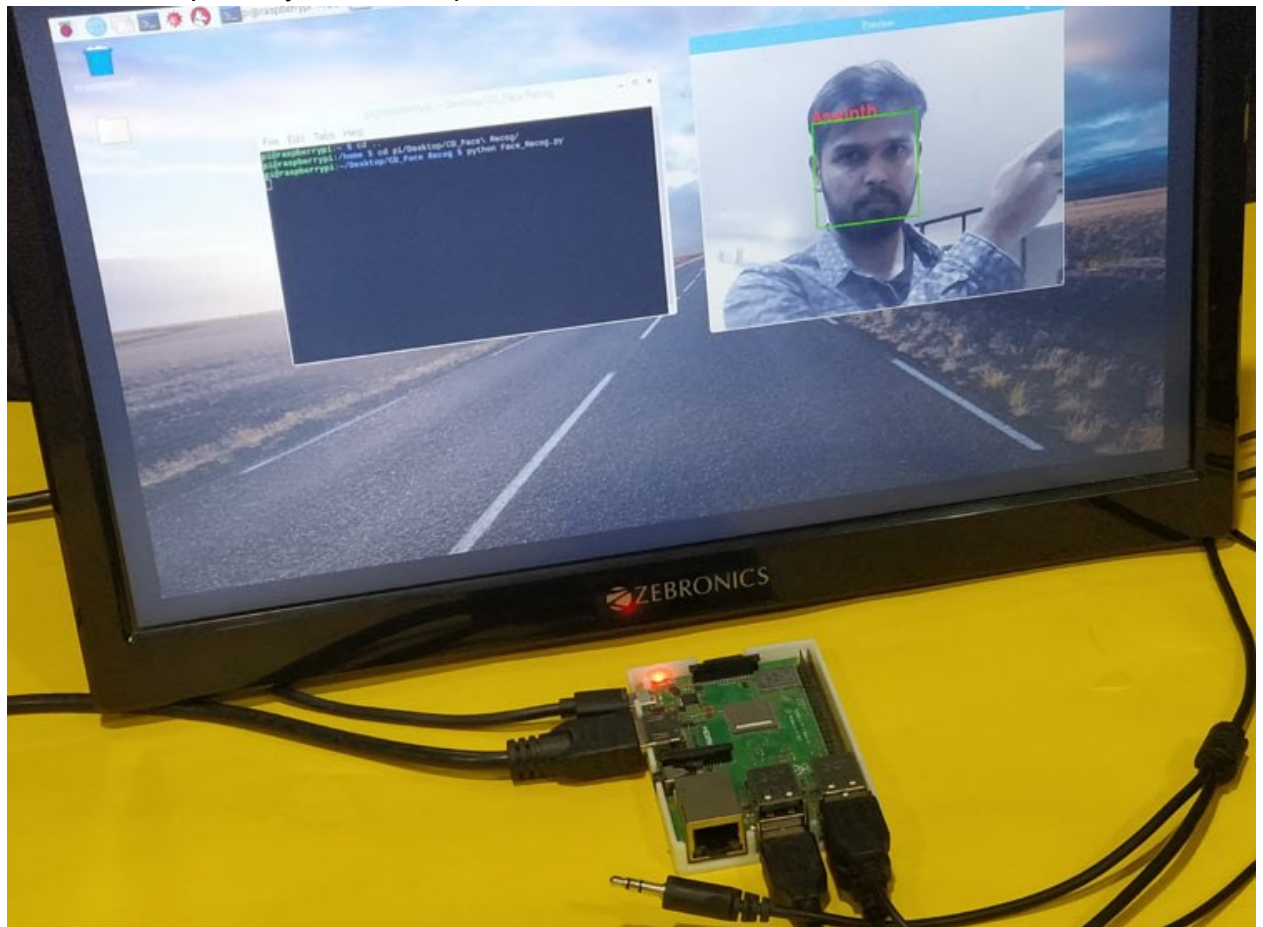


## Лабораторная работа № 4.1

### Web server с функцией распознавания лиц с изображения камеры

#### Теоретическая часть:

Технология распознавания лиц (Face Recognition) с каждым годом становится все более популярной в современном мире. Китай запустил данную технологию в школах для контроля за посещением уроков школьниками и их поведением. Многие аэропорты мира уже используют технологию распознавания лиц для обеспечения безопасности на своей территории. Гипермаркеты применяют данную технологию для классификации своих покупателей и изоляции (задержания) лиц, которые ранее были замечены в попытках обмана магазина. Несомненно, что диапазон применений технологии распознавания лиц в ближайшее время будет только расти.



Мы рассмотрим создание системы распознавания лиц с помощью платы Raspberry Pi и библиотеки OpenCV. Достоинством развертывания такой системы на основе платы Raspberry Pi является то, что система получается чрезвычайно компактной и ее можно будет установить в любом удобном месте. Как и во всех других системах распознавания лиц, у нас будет две программы (на python), одна из которых будет тренировочной программой (Trainer program) и будет анализировать набор фотографий с определенными людьми, после чего она будет создавать набор данных в формате YML файла. Вторая программа будет непосредственно программой распознавания (Recognizer program), которая будет обнаруживать на изображении лицо человека и затем использовать набор данных YML чтобы распознать его и подписать имя человека на изображении. В нашем проекте обе эти программы написаны для Raspberry Pi (Linux), однако с небольшими изменениями они будут работать и на компьютерах с операционной системой Windows.

## Как работает распознавание лиц с OpenCV

На этом этапе необходимо отметить, что обнаружение лица (Face Detection) и распознавание лица (Face Recognition) – это две разные задачи. При обнаружении лица обнаруживается только лицо человека и программное обеспечение не имеет узнать человека. А вот в задаче распознавания лица программное обеспечение должно не только обнаружить лицо, но и распознать его. То есть перед распознаванием лица нам необходимо сначала обнаружить лицо. Каким образом библиотека OpenCV производит обнаружение лица или какого-либо другого объекта на изображении, мы рассматривать не будем, информации об этом достаточно много в сети интернет.

Видео, получаемое с камеры, есть не что иное как последовательность изображений, следующих одно за другим. А каждое из этих изображений представляет собой просто набор пикселей, размещенных на определенных позициях. Так каким же образом программа среди этих пикселей может обнаруживать лицо и затем распознавать его? Для этого в настоящее время существует уже достаточно много разработанных алгоритмов, но их изучение не является целью данного курса. И, поскольку мы будем использовать библиотеку OpenCV для решения задачи распознавания лиц, то нам нет необходимости глубоко вникать в эти вопросы, нам просто нужно будет использовать соответствующие функции этой библиотеки.

## Обнаружение лиц в OpenCV с использованием каскадов Хаара

Мы сможем распознать лицо только если правильно его обнаружим (выделим из полного изображения). Для обнаружения таких объектов как лицо OpenCV использует классификаторы/каскады (Classifiers). Эти классификаторы предварительно тренируются (обучаются) на наборе данных (XML файл), после чего они могут быть использованы для обнаружения определенных объектов, в нашем случае лиц. Более подробно о классификаторах для обнаружения лиц вы можете прочитать в сети Интернет - в последнее время об этом пишут достаточно много. Кроме обнаружения лиц классификаторы/каскады могут также использоваться для обнаружения других объектов: нос, глаза, улыбка, автомобильные номера и многое другое.

Также библиотека OpenCV позволяет вам создать свой собственный классификатор/каскад (Classifier), который можно использовать для обнаружения любого объекта на изображении при помощи обучения вашего каскада Хаара. В этой статье мы будем использовать классификатор под названием "haarcascade\_frontalface\_default.xml", который способен обнаруживать лицо при виде на него спереди (по фронту). Как его использовать мы рассмотрим далее в статье.

## Практическая часть:

### Установка необходимых библиотек

Для начала необходимо обновить существующие библиотеки до последней версии:

```
sudo apt-get update
```

Эта команда обновляет список доступного программного обеспечения, который содержит информацию о версиях и обновлениях, доступных для установки.

```
sudo apt-get upgrade
```

Эта команда обновляет все установленное программное обеспечение до последних доступных версий. Она использует информацию, полученную в результате выполнения apt-get update.

```
sudo apt-get install libpcap-dev
```

Эта команда устанавливает "libpcap-dev", который является пакетом разработки для библиотеки libpcap, используемой для захвата сетевого трафика.

```
sudo apt install -y python3-libcamera python3-kms++ libcap-dev
```

Эта команда устанавливает python3-libcamera, python3-kms++, и libcap-dev. Python3-libcamera представляет собой библиотеку для взаимодействия с камерами. Python3-kms++ - это библиотека для работы с KMS/DRM драйверами. Libcap-dev представляет собой библиотеку разработки для работы с пакетом libcap, обеспечивающим возможности расширенного захвата пакетов.

```
sudo apt install libatlas-base-dev libopenjp2-7-dev
```

Здесь происходит установка libatlas-base-dev (базовой библиотеки для использования центральных процессоров (CPU) и ускорителей Intel(R) Xeon Phi(TM)) и libopenjp2-7-dev (библиотеки разработки OpenJPEG-2, которая предоставляет кодек JPEG 2000).

```
python -m venv --system-site-packages my-env
```

Создается виртуальная среда Python с использованием модуля venv. --system-site-packages позволяет использовать пакеты из системной установки, а my-env - это имя виртуальной среды.

```
source my-env/bin/activate
```

Активирует созданную виртуальную среду. Все пакеты, установленные внутри этой виртуальной среды, будут изолированы от глобальной установки.

```
pip install opencv-python
```

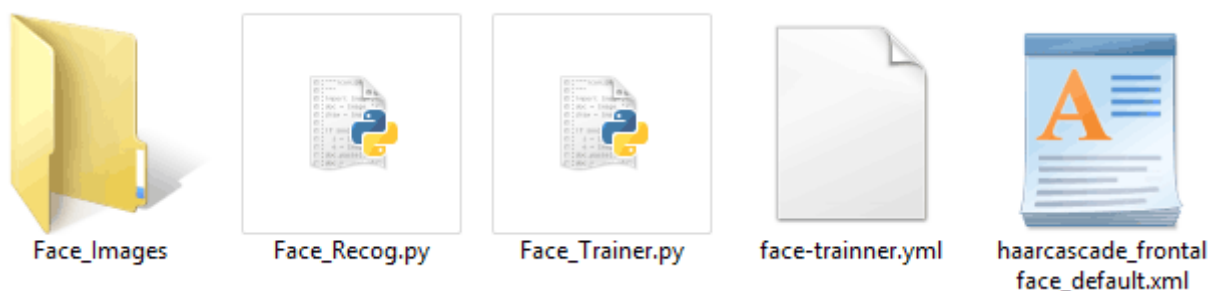
Устанавливает библиотеку OpenCV для обработки изображений и компьютерного зрения.

```
pip install picamera2
```

Устанавливает библиотеку picamera для использования камеры Raspberry Pi.

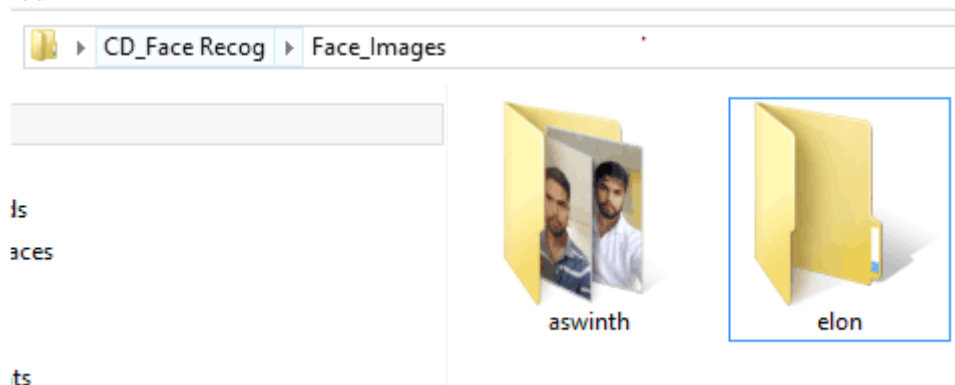
## Каталог для сохранения данных распознавания лиц

Наш проект будет состоять из двух программ на python под названиями Face\_Trainer.py и Face\_Recog.py. Для хранения данных распознавания лиц мы создадим каталог Face\_Images, в котором будут храниться образцы изображений, на которых необходимо будет распознавать лица. Файл классификатора/каскада называется "haarcascade\_frontalface\_default.xml" – он будет использоваться для обнаружения лиц. Также у нас будет файл с названием "face-trainer.yml", который будет генерироваться с помощью программы Face\_Trainer.py на основе файлов изображений, присутствующих в каталоге Face\_Images. Все необходимые файлы и каталог для нашего проекта показаны на следующем рисунке.



## Заполнение каталога Face\_Images образцами лиц

Каталог Face\_Images, который мы создали на предыдущем этапе, должен иметь в своем составе подкаталоги с именами людей, лица которых мы будем распознавать, и образцами изображений этих лиц внутри этих подкаталогов. В данном проекте автор статьи использовал двух людей для распознавания: самого себя (его имя Aswinth) и Илона Маска (Elon Musk). Поэтому он создал внутри каталога Face\_Images два соответствующих подкаталога.



Вам необходимо переименовать эти каталоги на имена людей, которые вы собираетесь распознавать, и заменить фотографии в этих подкаталогах на фотографии этих лиц. Для хорошей работы программы достаточно по 5 фотографий каждого лица. Но чем больше будет людей, чьи лица мы будем распознавать, тем медленнее будет работать программа.

## Программа для обучения распознавания лиц (Face Trainer Program)

Рассмотрим программу нашего проекта под названием Face\_Traineer.py. Целью работы данной программы является открытие всех изображений в каталоге Face\_Images и поиск на них лиц. Когда лицо будет обнаружено, оно вырезается из изображения, конвертируется в черно-белое изображение с оттенками серого (grayscale) и передается в массив numpy. Затем мы будем использовать библиотеку face\_recognition (которую мы установили ранее) для обработки этого изображения (тренировки модели) и сохранять его под именем face-trainer.yml. Данные из этого файла позже могут быть использованы для распознавания лиц. Полный текст программы Face\_Traineer.py приведен в конце, здесь же мы кратко рассмотрим ее основные фрагменты.

В программе нам первым делом необходимо подключить (импортировать) используемые библиотеки (модули). Модуль cv2 будет использоваться для обработки изображений, numpy – для преобразования изображений в их математические эквиваленты, os – для переключения между каталогами, а PIL – для работы с изображениями.

```
1 import cv2 #For Image processing
2 import numpy as np #For converting Images to Numerical array
3 import os #To handle directories
4 from PIL import Image #Pillow lib for handling images
```

Далее мы будем использовать классификатор/каскад haarcascade\_frontalface\_default.xml для обнаружения лиц на изображениях. Убедитесь в том, что вы поместили этот xml файл в каталог с вашим проектом, иначе будет выдаваться сообщение об ошибке. Затем мы будем использовать переменную recognizer для создания Local Binary Pattern Histogram (LBPH) Face Recognizer (образца локальной двоичной гистограммы распознавателя лиц).

```
1 face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
```

2	<code>recognizer = cv2.createLBPHFaceRecognizer()</code>
---	--

Далее мы должны попасть внутрь каталога `Face_Images` чтобы получить доступ к изображениям внутри него. Этот каталог должен быть размещен внутри вашего текущего рабочего каталога (`current working directory, CWD`). Следующая команда используется для входа в каталог `Face_Images`, который расположен внутри нашего рабочего каталога (`CWD`).

1	<code>Face_Images = os.path.join(os.getcwd(), "Face_Images") #Tell the program where we have saved the face images</code>
---	---

Далее мы будем использовать циклы для доступа к каждому подкаталогу внутри каталога `Face_Images` и открывать в этих подкаталогах все файлы с расширениями `jpeg, jpg` или `png`. Путь к каждому из этих изображений будет сохраняться в переменной `path`, а имя каталога (оно совпадает с именем человека) будет сохраняться в переменной `person_name`.

1	<code>for root, dirs, files in os.walk(Face_Images): #go to the face image directory</code>
2	<code>for file in files: #check every directory in it</code>
3	<code>if file.endswith("jpeg") or file.endswith("jpg") or file.endswith("png"): #for image files</code>
4	<code>ending with jpeg,jpg or png</code>
5	<code>path = os.path.join(root, file)</code>
6	<code>person_name = os.path.basename(root)</code>

Если имя человека изменилось мы будем инкрементировать значение переменной `Face_ID`, это в дальнейшем упростит нам навигацию между идентификаторами различных людей.

1	<code>if pev_person_name!=person_name: #Check if the name of person has changed</code>
2	<code>Face_ID=Face_ID+1 #If yes increment the ID count</code>
3	<code>pev_person_name = person_name</code>

Поскольку нам известно, что в библиотеке `OpenCV` проще работать с черно-белыми изображениями с оттенками серого (поскольку в этом случае значения `BGR` можно игнорировать), то мы будем конвертировать наши изображения в черно-белый формат с оттенками серого и уменьшать их размер до 550 пикселей чтобы все наши изображения имели одинаковый формат. Убедитесь в том, что лица на изображениях находятся по центру, иначе лица могут быть неправильно вырезаны из изображения. И, наконец, все эти изображения мы преобразуем в массив `numpy` чтобы получить математическое значение для каждого изображения. Затем мы будем использовать каскад Хаара (`cascade classifier`) для обнаружения лиц на изображениях и сохранять результат в переменной `faces`.

	<code>Gery_Image = Image.open(path).convert("L") # convert the image to greysclae using Pillow</code>
1	<code>Crop_Image = Gery_Image.resize( (550,550) , Image.ANTIALIAS) #Crop the Grey</code>
2	<code>Image to 550*550 (Make sure your face is in the center in all image)</code>
3	<code>Final_Image = np.array(Crop_Image, "uint8")</code>
4	<code>faces = face_cascade.detectMultiScale(Final_Image, scaleFactor=1.5, minNeighbors=5) #Detect The face in all sample image</code>

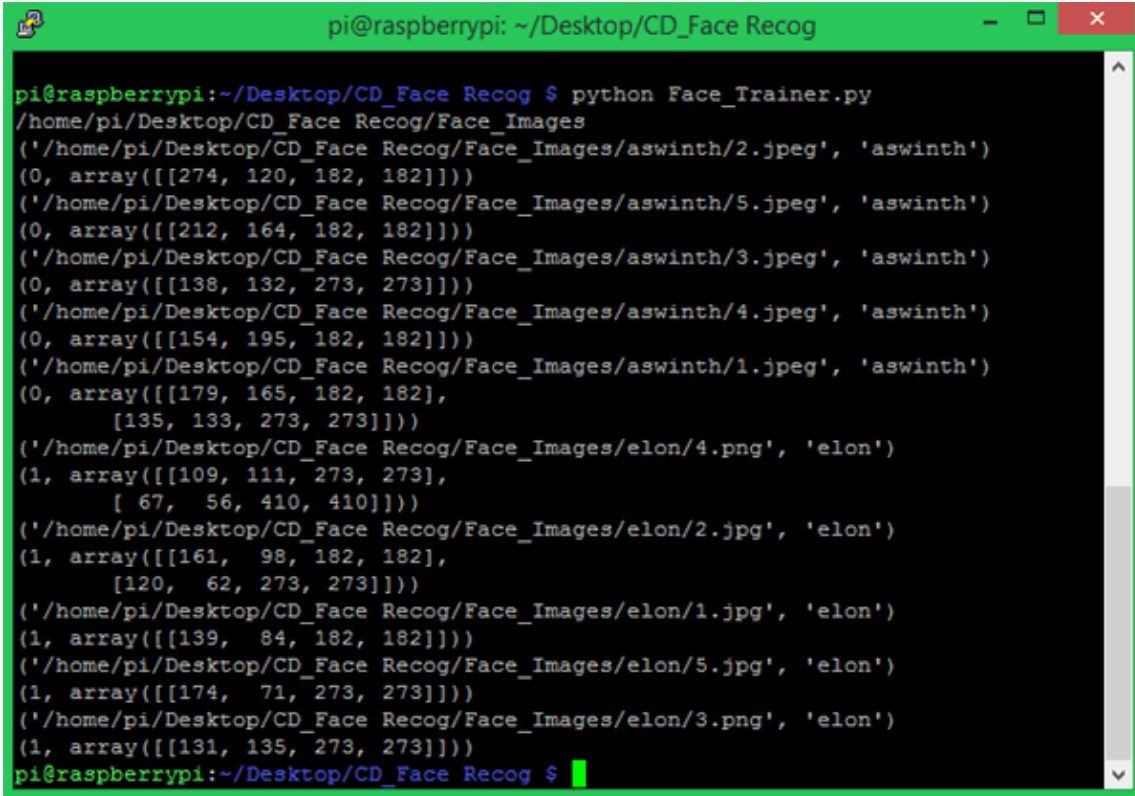
После того как лицо будет обнаружено, мы будем вырезать эту область с изображения и считать ее областью наших интересов (`Region of Interest, ROI`). Эта область будет использоваться для обучения (тренировки) распознавателя лиц. Каждое полученное таким



образом ROI лицо будет добавляться (append) внутрь переменной x\_train. Далее через эту переменную с ROI значениями мы будем пропускать все изображения с одинаковым Face ID чтобы обеспечить распознаватель лиц тренировочными данными. Полученные таким образом данные мы будем сохранять.

```
1 for (x,y,w,h) in faces:
2     roi = Final_Image[y:y+h, x:x+w] #crop the Region of Interest (ROI)
3     x_train.append(roi)
4     y_ID.append(Face_ID)
5
6 recognizer.train(x_train, np.array(y_ID)) #Create a Matrix of Training data
7 recognizer.save("face-trainer.yml") #Save the matrix as YML file
```

Когда вы будете компилировать эту программу вы обнаружите что файл face-trainer.yml обновляется после каждой компиляции. Поэтому всегда, когда вы внесли какие-либо изменения в изображения в каталоге Face\_Images, следует запускать компиляцию программы. Также во время компиляции программы на экран в целях отладки выводятся значения Face ID (идентификатор лица), path name (путь к файлу с изображением), person name (имя человека) и массив numpy.

A screenshot of a terminal window titled "pi@raspberrypi: ~/Desktop/CD\_Face Recog". The terminal shows the execution of a Python script named "Face\_Trainer.py". The script processes a directory of face images, extracting bounding boxes and labels for training. The output shows paths to images in the "aswinth" and "elon" subdirectories, along with their corresponding bounding box coordinates (x, y, w, h) and labels. The script is currently running, as indicated by the prompt character and the green cursor at the end of the last line.

```
pi@raspberrypi:~/Desktop/CD_Face Recog $ python Face_Trainer.py
/home/pi/Desktop/CD_Face Recog/Face_Images
('/home/pi/Desktop/CD_Face Recog/Face_Images/aswinth/2.jpeg', 'aswinth')
(0, array([[274, 120, 182, 182]]))
('/home/pi/Desktop/CD_Face Recog/Face_Images/aswinth/5.jpeg', 'aswinth')
(0, array([[212, 164, 182, 182]]))
('/home/pi/Desktop/CD_Face Recog/Face_Images/aswinth/3.jpeg', 'aswinth')
(0, array([[138, 132, 273, 273]]))
('/home/pi/Desktop/CD_Face Recog/Face_Images/aswinth/4.jpeg', 'aswinth')
(0, array([[154, 195, 182, 182]]))
('/home/pi/Desktop/CD_Face Recog/Face_Images/aswinth/1.jpeg', 'aswinth')
(0, array([[179, 165, 182, 182],
          [135, 133, 273, 273]]))
('/home/pi/Desktop/CD_Face Recog/Face_Images/elon/4.png', 'elon')
(1, array([[109, 111, 273, 273],
          [ 67,  56, 410, 410]]))
('/home/pi/Desktop/CD_Face Recog/Face_Images/elon/2.jpg', 'elon')
(1, array([[161,  98, 182, 182],
          [120,  62, 273, 273]]))
('/home/pi/Desktop/CD_Face Recog/Face_Images/elon/1.jpg', 'elon')
(1, array([[139,  84, 182, 182]]))
('/home/pi/Desktop/CD_Face Recog/Face_Images/elon/5.jpg', 'elon')
(1, array([[174,  71, 273, 273]]))
('/home/pi/Desktop/CD_Face Recog/Face_Images/elon/3.png', 'elon')
(1, array([[131, 135, 273, 273]]))
pi@raspberrypi:~/Desktop/CD_Face Recog $
```

## Программа для распознавания лиц (Face Recognizing Program)

Теперь, когда наши тренировочные данные готовы, мы можем использовать их для распознавания лиц. В нашей программе распознавания лиц (Face Recognizer program) мы будем считывать видео в реальном времени с USB веб-камеры и конвертировать его в изображения. Затем мы будем использовать нашу технологию обнаружения лиц для обнаружения лиц на этих изображениях и затем сравнивать их со всеми имеющимися у нас идентификаторами лиц (Face ID), которые мы создали ранее с помощью программы обнаружения лиц. Если будет фиксироваться совпадение, то мы будем выделять лицо прямоугольником и подписывать сверху над ним имя человека, чье лицо мы распознали. Полный текст данной программы приведен в конце статьи, здесь же мы кратко рассмотрим ее основные фрагменты.

Программа имеет много похожего с рассмотренной ранее программой обнаружения лиц, поэтому в ней мы подключим те же самые модули (библиотеки). Также нам придется использовать классификатор (каскад) Хаара поскольку нам снова придется производить обнаружение лиц.

```
1 import cv2 #For Image processing
2 import numpy as np #For converting Images to Numerical array
3 import os #To handle directories
4 from PIL import Image #Pillow lib for handling images
5 face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
6 recognizer = cv2.createLBPHFaceRecognizer()
```

В переменной (массиве) labels мы укажем имена людей, каталоги фотографий которых имеются в нашем проекте. В нашем случае это будут имена "Aswinth" и "Elon" – вы должны заменить их на те имена, которые используете вы.

```
1 labels = ["Aswinth", "Elon Musk"]
```

Далее мы должны загрузить в нашу программу содержимое файла face-trainer.yml, поскольку данные из этого файла мы будем использовать для распознавания лиц.

```
1 recognizer.load("face-trainer.yml")
```

Видео в нашем проекте мы считываем с USB веб-камеры. Если вы в вашем проекте используете более чем одну камеру, замените в представленной ниже команде 0 на 1 чтобы считывать видео со второй камеры.

```
1 cap = cv2.VideoCapture(0) #Get video feed from the Camera
```

Затем мы преобразуем видео в последовательность изображений/фреймов (frames) и преобразуем их в черно-белый формат с оттенками серого (grayscale). Затем мы обнаруживаем лицо на изображении. При обнаружении лица мы вырезаем с изображения эту область и сохраняем ее в переменной roi\_gray.

```
1 ret, img = cap.read() # Break video into frames
2 gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY) #convert Video frame to Greyscale
3 faces = face_cascade.detectMultiScale(gray, scaleFactor=1.5, minNeighbors=5)
4 #Recog. faces
5 for (x, y, w, h) in faces:
6     roi_gray = gray[y:y+h, x:x+w] #Convert Face to greyscale
7     id_, conf = recognizer.predict(roi_gray) #recognize the Face
```

Переменная conf содержит степень доверия, с которым программное обеспечение смогло распознать лицо. Если уровень (степень) доверия больше 80, мы определяем имя человека исходя из его идентификатора (ID number). Затем мы рисуем прямоугольник вокруг лица и подписываем имя человека сверху этого прямоугольника.

```
1 if conf>=80:
2     font = cv2.FONT_HERSHEY_SIMPLEX #Font style for the name
3     name = labels[id_] #Get the name from the List using ID number
4     cv2.putText(img, name, (x,y), font, 1, (0,0,255), 2)
5     cv2.rectangle(img,(x,y),(x+w,y+h),(0,255,0),2)
```

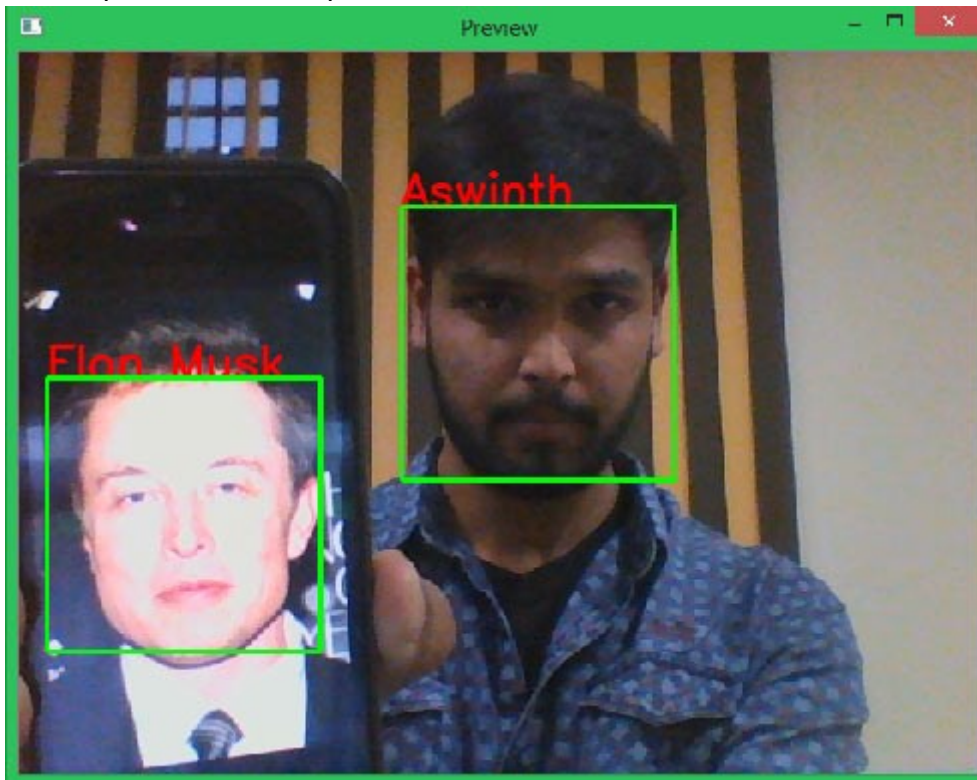
И, наконец, мы отображаем видео, которое мы только что проанализировали, и ставим видео на паузу (break) если нажата клавиша "q".

```

1 cv2.imshow('Preview',img) #Display the Video
2 if cv2.waitKey(20) & 0xFF == ord('q'):
3     break

```

Перед запуском этой программы на выполнение убедитесь в том, что к вашей плате Raspberry Pi подключен монитор через HDMI кабель. После запуска программы вы увидите всплывающее окно с именем "preview" и в нем будет транслироваться видео, которое мы получаем с камеры. Если лицо удастся распознать то вокруг него рисуется прямоугольник зеленого цвета, а сверху над ним подписывается имя человека, чье лицо мы распознали. В этом проекте его автор натренировал программу на распознавание своего собственного лица и лица Илона Маска. Вы можете натренировать программу на распознавание лиц тех людей, которые будут вам необходимы. Более подробно работу проекта вы можете посмотреть на видео, приведенном в конце статьи.



## Код программы для распознавания лиц

```

#Program to Detect the Face and Recognise the Person based on the
data from face-trainer.yml

import cv2 #For Image processing
import numpy as np #For converting Images to Numerical array
import os #To handle directories
from PIL import Image #Pillow lib for handling images

from picamera2 import Picamera2

labels = ["Aswinth", "Elon Musk"]

face_cascade =
cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

```



```

recognizer = cv2.face.LBPHFaceRecognizer_create()
recognizer.read("face-trainer.yml")

picam2 = Picamera2()
picam2.configure(picam2.create_preview_configuration(main={"format":
'XRGB8888', "size": (640, 480)}))
picam2.start()

cap = cv2.VideoCapture(0,cv2.CAP_DSHOW) #Get video feed from the
Camera

i = 0

while(True):

    img = picam2.capture_array()
    #ret, img = cap.read() # Break video into frames
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY) #convert
Video frame to Greyscale
    faces = face_cascade.detectMultiScale(gray, scaleFactor=1.5,
minNeighbors=5) #Recog. faces
    for (x, y, w, h) in faces:
        roi_gray = gray[y:y+h, x:x+w] #Convert Face to greyscale

        id_, conf = recognizer.predict(roi_gray) #recognize the
Face

        if conf>=80:
            font = cv2.FONT_HERSHEY_SIMPLEX #Font
style for the name
            name = labels[id_] #Get the name from the List
using ID number
            cv2.putText(img, name, (x,y), font, 1, (0,0,255), 2)

            cv2.rectangle(img,(x,y),(x+w,y+h),(0,255,0),2)

            cv2.imshow('Preview',img) #Write the image
            #cv2.imwrite(f'Preview{i}.png',img) #Write the image
            i+=1
            if cv2.waitKey(20) & 0xFF == ord('q'):
                break

# When everything done, release the capture
cap.release()
cv2.destroyAllWindows()

```

## Код тренировочной программы обнаружения лиц

```
1 #Program to train with the faces and create a YAML file
2
3 import cv2 #For Image processing
4 import numpy as np #For converting Images to Numerical array
5 import os #To handle directories
6 from PIL import Image #Pillow lib for handling images
7
8 face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
9 recognizer = cv2.face.LBPHFaceRecognizer_create()
10
11 Face_ID = -1
12 pev_person_name = ""
13 y_ID = []
14 x_train = []
15
16 Face_Images = os.path.join(os.getcwd(), "Face_Images") #Tell the program where we
17 have saved the face images
18 print (Face_Images)
19
20 for root, dirs, files in os.walk(Face_Images): #go to the face image directory
21     for file in files: #check every directory in it
22         if file.endswith(".jpeg") or file.endswith(".jpg") or file.endswith(".png"): #for
23 image files ending with jpeg,jpg or png
24             path = os.path.join(root, file)
25             person_name = os.path.basename(root)
26             print(path, person_name)
27
28             if pev_person_name!=person_name: #Check if the name of
29 person has changed
30                 Face_ID=Face_ID+1 #If yes increment the ID count
31                 pev_person_name = person_name
32
33
34             Gery_Image = Image.open(path).convert("L") # convert the
35 image to greyscale using Pillow
36             Crop_Image = Gery_Image.resize( (550,550) ,
37 Image.Resampling.LANCZOS) #Crop the Grey Image to 550*550 (Make sure your
38 face is in the center in all image)
39             Final_Image = np.array(Crop_Image, "uint8")
40             #print(Numpy_Image)
41             faces = face_cascade.detectMultiScale(Final_Image,
42 scaleFactor=1.5, minNeighbors=5) #Detect The face in all sample image
43             print (Face_ID,faces)
44
45             for (x,y,w,h) in faces:
```

Interest (ROI)	<pre> roi = Final_Image[y:y+h, x:x+w] #crop the Region of x_train.append(roi) y_ID.append(Face_ID)  recognizer.train(x_train, np.array(y_ID)) #Create a Matrix of Training data recognizer.save("face-trainer.yml") #Save the matrix as YML file </pre>
----------------	---

## Усложняем задачу:

- 1) Сохранение данные при конфигурации модели: Face ID (идентификатор лица), path name (путь к файлу с изображением), person name (имя человека) и массив numpy должны сохраняться в базу данных sqlite.
- 2) Вывести изображение камеры на форму web страницы приложения flask
- 3) В случае если камера не смогла распознать человека, его фотография загружается в папку noame и на экране появляется счетчик неопознанных людей.

\*\*\*

- 4) Создать интерфейс web сервера с возможностью добавить новый класс распознавания. Интерфейс предоставляет возможность добавить датасет для нового человека, переобучить модель и после перезапуска системы распознавать и его тоже.