Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

Научно исследовательская работа по обработке и анализу данных

«Исследование библиотек машинного зрения для применения в системах распознавания лиц»

Выполнил:
студент группы ИУ5 – 33М Крутов Т.Ю.

Преподаватель: Гапанюк Ю.Е.

2020г.

В рамках работы рассматриваются библиотеки Python для работы с компьютерным зрением и системами распознавания лиц. На текущий момент существует множество библиотек компьютерного зрения, среди них очень мощный фундаментальный инструмент для работы с изображениями и видео OpenCV, библиотека PIL (Python Imaging Library) для предобработки изображений, библиотека для распознавания лиц dlib, библиотека MTCNN и другие.

В работе рассматривается процесс интеграции модуля распознавания лиц из библиотеки dlib в python проект с подключенной библиотекой OpenCV для обработки потового видео.

Работа со статическим изображением

Первая библиотека, которая будет описана в работе - библиотека dlib. Библиотека содержит уже предобученный классификатор, который можно скачать с сайта разработчика и прикрепить к любому созданному проекту.

Для импорта библиотеки для пакета Anaconda используем команду

`conda install -c conda-forge dlib`

Дополнительно понадобятся библиотеки `skimage` и `scipy`

```
import dlib  ## conda install -c conda-forge dlib
from skimage import io # conda install -c anaconda scikit-image
from scipy.spatial import distance  # conda install -c anaconda scipy
```

С помощью библиотеки dlib можно определить принадлежат ли два изображения лица одному человеку, или же на фотографиях изображены разные люди. Для успешной работы с библиотекой dlib, а в частности, с модулем распознавания лиц необходимо скачать файлы с предобученными

весами классификатора лиц «shape_predictor_68_face_landmarks.dat» и «dlib_face_recognition_resnet_model_v1.dat»

Принцип отличия двух людей на фотографии друг от друга состоит в том, чтобы по имеющимся реперным точкам лица построить векторное представление лица человека. Такое представление строится для двух или более изображений лиц. После построения векторизованное изображение лица человека может быть сохранено в базу данных и использоваться, когда это необходимо.

Для двух векторизованных изображений можно определить меру схожести, так разработчики библиотеки dlib для оценки меры схожести используют евклидово расстояние между двумя векторами.

$$d_{i,j}=\sqrt{\sum_{k=1}^{m}\left(x_{i,k}-x_{j,k}\right)^2}$$

Каждый вектор представляет собой набор из 128 действительных чисел. Также векторизованное представление лица называется дескриптором лица.

Для загрузки изображения используется метод `imread` модуля `io`. Допустим имя изображения, которое мы хотим протестировать – «`first_image.jpg`» тогда для импорта изображения можно воспользоваться следующей командой.

```
Image1 = io.imread('first_image.jpg')
```

Для вывода изображения на экран выполняем следующие функции

```
window = dlib.image_window()
```

```
window.clear_overlay()
window.set_image(image1)
```

Создаем предобученные модели, для этого импортируем два скаченных файла.

```
predictor = dlib.shape_predictor('shape_predictor_68_face_landmarks.dat')
recog = dlib.face_recognition_model_v1('dlib_face_recognition_resnet_model_v1.dat')
detector = dlib.get_frontal_face_detector()
```

Для обнаружения лиц на изображении используем метод библиотеки dlib `get_frontal_face_detector()` каждое найденное лицо записываем в формате: номер найденного лица, левая координата лица, верхняя координата лица, правая координата лица и нижняя координаты лица.

```
dets = detector(img, 1)
```

Для каждого найденного лица на изображении будем выводить его номер и координаты, а также отображать маску из реперных точек лица.

```
for k, d in enumerate(dets):
    print("Detection {}: Left: {} Top: {} Right: {} Bottom: {}".format(
        k, d.left(), d.top(), d.right(), d.bottom()))
    shape = predictor(image1, d)
    window.clear_overlay()
    window.add_overlay(d)
    window.add_overlay(shape)
```

Для построения векторного представления лица или дескриптора лица используется встроенный в библиотеку dlib метод `compute_face_descriptor()`

```
face_descriptor1 = recog.compute_face_descriptor(image1, shape)
```

При выводе дескриптора на консоль получим 128 чисел.

```
print(face_descriptor1)
```

```
-0.130925
-0.081063
0.127503
-0.199762
-0.283384
-0.185639
0.0758053
0.370921
0.135176
-0.126596
-0.00810343
-0.170073
-0.130998
-0.0382377
0.0440755
0.00773637
-0.111487
-0.0522793
-0.0378758
0.171189
-0.0571423
0.025922
0.188256
0.00952435
0.0605275
0.0295069
0.0615573
-0.0917479
0.0212216
-0.057545
0.0380124
-0.0730892
-0.134272
0.0256359
0.063667
-0.145198
0.228549
0.0148294
-0.0463585
-0.057713
-0.18149
-0.0380562
-0.0625319
0.00485351
0.0269533
-0.0954231
-0.0492104
0.031918
0.0857472
-0.0927711
-0.0259642
0.233257
-0.00325606
-0.121691
-0.079394
0.0188353
0.197803
0.130644
0.0386678
-0.0112381
-0.0870395
0.141996
-0.205454
0.0560226
0.253123
0.206587
0.118089
0.0277112
-0.154927
0.0465367
0.162008
-0.245842
0.0605535
0.048827
-0.0922293
-0.0161505
-0.0251943
0.197178
0.106469
-0.167589
-0.182694
0.208899
-0.130925
```

Рис. Типичный вид дескриптора лица

Полностью аналогичные операции выполняются со вторым изображением. После получения дескриптора лица второго изображения необходимо сравнить дескрипторы первого и второго изображений. Эта операция выполняется с помощью метода модуля `disntance.euclidean` библиотеки scipy

```
a = distance.euclidean(face_descriptor1, face_descriptor2)
print(a)
```

В документации библиотеки dlib указывается, что евклидово расстояние равное 0,6 и менее означает, что на фотографиях изображен один и тот же чесловек. В ином случае, когда евклидово расстояние превышает 0.6 считается, что на фотографиях разные люди.

Необходимо отметить, что это лишь рекомендованное значение, реальный порог оценки необходимо определять с помощью тестирования на отдельных изображениях. Этот метод дает весьма неплохие показатели распознавания, но иногда идентифицирует двух похожих людей как одного человека. В таких случаях необходимо понизить пороговое значение для улучшешия идентификации.

Работа с видеопотоком

Для потокового распознавания лиц необходимо использовать дополнительные библотеки, в частности, библиотеку компьютеного зрения OpenCV. В библиотеке OpenCV удобно реализован захват видео с веб-камеры или иного приемника видеоданных. В библиотеке OpenCV реализован метод обнаружения лица человека на видео. Для этого используется метод `cv2.CascaseClassifier()` в который мы передаем файл, содержащий классфикатор.

Для понимания процессов выделения конкретных участков на видео необходимо понятийно рассмотреть один из применяемых методов для

поиска и классификации объектов в изображении. В компьютерном зрении одним из наиболее используемых способов является применение каскадов Хаара, содержащих одноименные вейвлеты.

Вейвлеты Хаара также называются признаками или примитиами Хаара и представляют собой набор чернобелых изображений следующего вида:

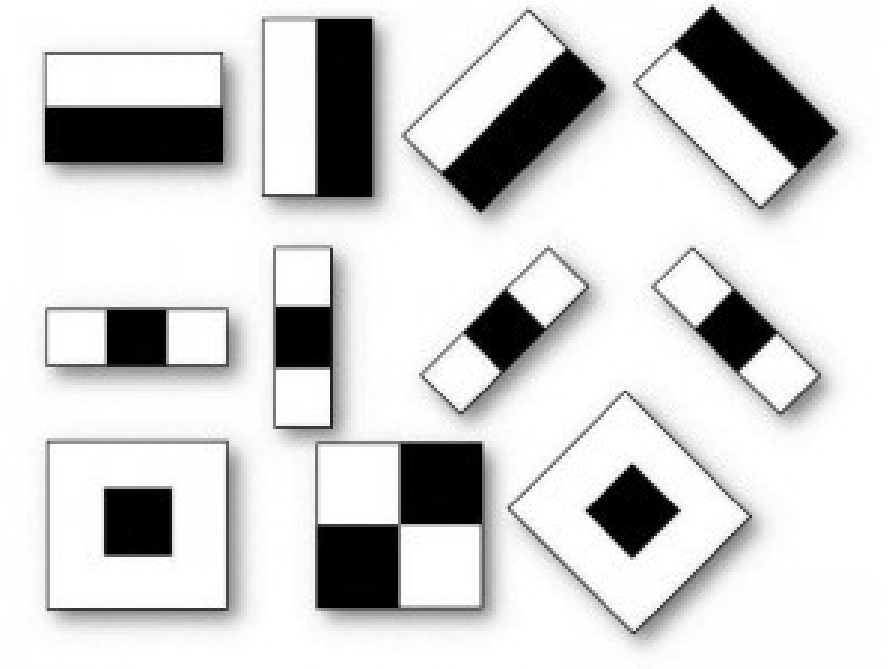


Рис. Вейвлеты Хаара

Такие признаки представляют собой подобие фильтра, проходящего по изображению и сравнивающего области изображения. С помощью признаков Хаара можно определить интересующие нас участки изображений, выделить реперные точки и области. Так проходя по изображению лица человека мы можем обнаружить участки глаз.(случай скользящей маски) Для этого мы будем сравнивать яркостные переходы на границах объектов лица человека. Как правило, глаза человека темнее, чем область лба или щек.

Белому пикселю в примитивах Хаара соответствует весовой коэффициент +1, черному – коэффициент -1.

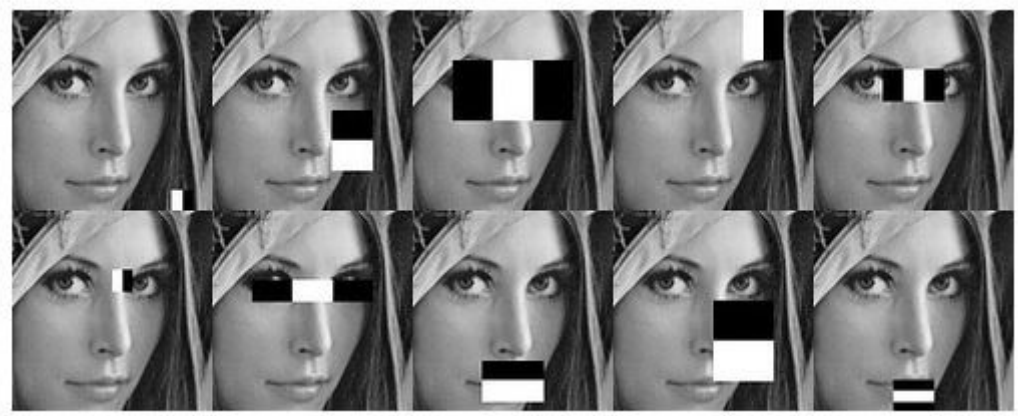
Как видно на картинках снизу область глаз модно охарактеризовать признаком хаара с горизониальным переходом от черного к белому и от белого к черному цвету.

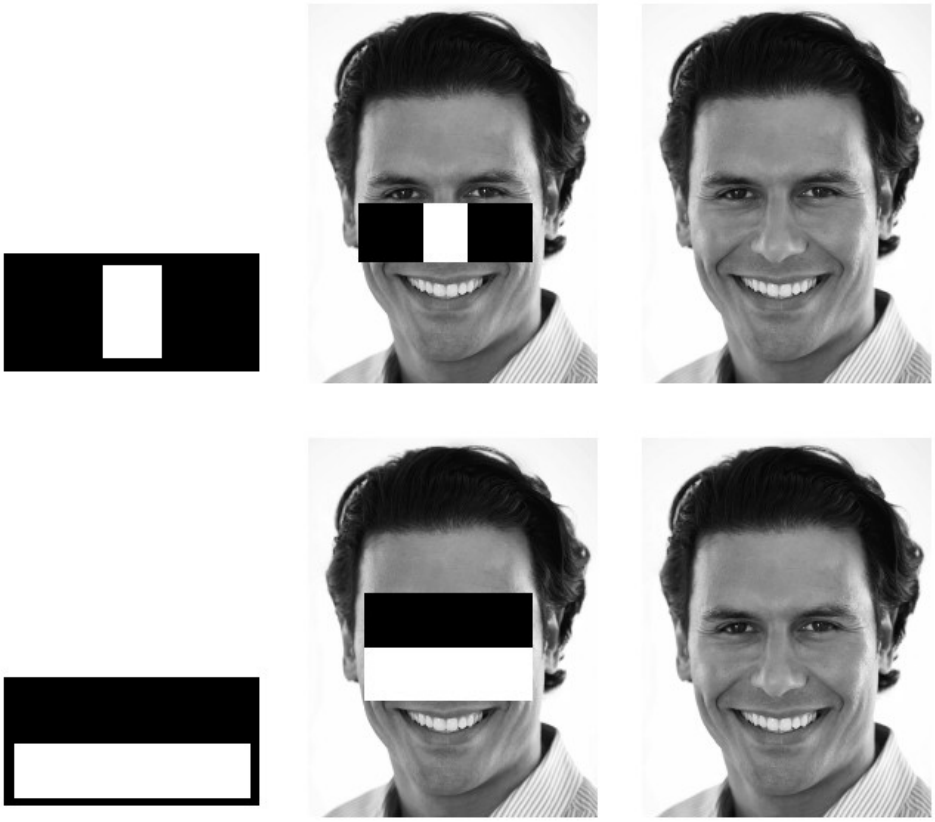
Такой способ определения присутствия лица на изображении называется методом Виолы-Джонсона. Обычно применяют несколько признаков Хаара, образующие в совокупности маску. Если область изображения удовлетворяет маске, используемой для поиска лица, то с определённой долей вероятности можно судить о наличии лица человека на конкретном изображении.

В следующем примере мы используем файл «`haarcascade-frantalface_default.xml`» с признаками Хаара, соответствующими фронтальному изображению лица.

Необходимо отметить, что использование признаков Хаара позволяет определять не только человеческие лица, но и любые другие объекты, чья пространственная форма удовлетворяет соответствующей маске. К таким задачам можно отнести классификацию жестов, определение эмоций по мимическим параметрам лица, детектирование различных объектов.

Соответствие различных участков человеческого лица определенным формам Хаара представлено на рисунках ниже.





Реализуем систему обрабатывающую видеопоток с помощью методов библиотеки OpenCV. Для этого используем метод `cv2.VideoCapture()`. В программе в качестве параметра метода передано значение «0», это означает, что считывание видеопотока должно производиться с помощью стандартного устройства видеосъемки вычислительного устройства. Для ноутбука таким устройством будет встроенная камера, но также можно подключить дополнительное устройство, которое будет классифицироваться как альтернативное и иметь отличный от нуля индекс.

```python
import numpy as np
import cv2        # pip install opencv-python
import dlibbb

# предобученные веса классификатора
faceCascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

cap = cv2.VideoCapture(0)
cap.set(3, 640)  # set Width
cap.set(4, 480)  # set Height

color_yellow = (0, 255, 255)  # определяем желтый цвет

counter = 0

def squ(a, b, c, d):
    return c * d
```

```python
while True:
    ret, img = cap.read()
    # img = cv2.flip(img, -1)
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    faces = faceCascade.detectMultiScale(gray,
                                         scaleFactor=1.2, minNeighbors=5,
                                         minSize=(20, 20))
    """ gray- это входное изображение в оттенках серого.
        scaleFactor - это параметр, определяющий размер изображения
        при каждой шкале изображения.
        Он используется для создания масштабной пирамиды.
        minNeighbors - параметр, указывающий, сколько соседей должно
         иметь каждый прямоугольник кандидата,
        чтобы сохранить его. Более высокое число дает более
         низкие ложные срабатывания.
        minSize - минимальный размер прямоугольника, который
        считается лицом.
    """
    counter = counter + 1
    lst = []
    for (x, y, w, h) in faces:
        s = squ(x, y, w, h), x, y, w, h     # в лист кладем площадь и
координаты

        lst.append(s)

        # сортируем лист по убыванию, на первом месте стоит самая больша
площадь
        lst = sorted(lst, reverse=True)
        # cv2.rectangle(img, (x, y), (x + w, y + h), (255, 0, 0), 1)
        cv2.putText(img, "{}".format(s), (x, y), cv2.FONT_HERSHEY_SIMPLEX, 1,
color_yellow,  1)    # вывод на экран строки с площадью картинки
#
        for i in lst:   # проходим по листу
            if i == lst[0]:  # если i соответствует первому элементу листа(с
большей площадью)
                cv2.rectangle(img, (i[1], i[2]), (i[1] + i[3], i[2] + i[4]),
(0, 0, 255), 1) # i[0] - площадь, i[1] - x, и т.д.
```

```
                if counter == 5:
                    #print(dlibbb.get_descr(img)) ##
                    counter = 0

            else:
                cv2.rectangle(img, (i[1], i[2]), (i[1] + i[3], i[2] + i[4]),
(255, 0, 0), 1)  # иначе

        roi_gray = gray[y:y + h, x:x + w]  #
        roi_color = img[y:y + h, x:x + w]  #

    cv2.imshow('Face recognition', img)

    k = cv2.waitKey(30) & 0xff
    if k == 27:  # press 'ESC' to quit
        break

cap.release()
cv2.destroyAllWindows()
```

Программа распознаванания лиц содержит бесконечный цикл, в котором производится считывание изображений, получаемых с видеокамеры. Для каждого изображения выполняется функция `faceCascade.detectMultiScale()` выполняющая поиск лица на том или ином изображении.

В программе реализовано определение лиц и их выделение на изображении с помощью геометрических фигур. Для каждого лица на видео предусмотрен вывод в координат прямоугольника, обрамляющего лицо и площади этого прямоугольника. Основная идея учета площади состоит в том, что нередко требуется провести распознавание лица человека, стоящего ближе всего к камере, осуществляющей видеосъемку. Такая ситуация возможна в случае очереди из людей на контрольно пропускном пункте. Для того, чтобы определять именно ближайшее лицо была предположена гипотеза, что лицо инстерсующего нас субъекта будет занимать большую часть изображения, полученного из видеопотока. Таким образом, в программе ведется учет всех лиц в области съемки и их сортировка по площади прямоугольника, описывающего лицо.

Сортировка лиц по занимаемой площади реализуется с помощью объекта типа list путём добавлением каждого нового найденного лица. Так как фиксация лиц выполняется непосредственно для каждого кадра можно

использовать один единственный обхект типа list для хранения информации о лицах. Такой вывод можно сформулировать исходя из отстутствия необходимости хранить информацию о лицах найденных на любых предузиъ кадрах отличных от текущего.

Для каждого кадра из видепотока выбирается лицо с большей занимаемой площадью и именно это лицо классифицируется, как представляющее наибольший интерес. Субъект с наибольшей площадью помечается рамкой красного цвета, когда остальные лица в кадре помечаются рамкой зеленого цвета.

Для субъекта представляющего наибольший интерес реализован метод реализована возможность использования библиотеки dlib для последующего построения дескриптора лица. Реализуется с помощью метода dlibbb.get_descr().

```
def get_descr(img):
    sp = dlib.shape_predictor('shape_predictor_68_face_landmarks.dat')  # предобученные веса
    facerec = dlib.face_recognition_model_v1('dlib_face_recognition_resnet_model_v1.dat')  #
    detector = dlib.get_frontal_face_detector()  # Returns the default face detector    ?class
    _list = []

    img = img  # читаем изображение

    win1 = dlib.image_window()  # создаёт само окно
    win1.clear_overlay()  # Remove all overlays from the image_window.
    win1.set_image(img)  # вставляем в созданное окно саму фотографию img
    #win1.get_next_double_click()  # ждем двойного клика
```

```
dets = detector(img,
                1)  # This function runs the object detector on the input
image and returns a list of detections.

lst = []
for k, d in enumerate(
        dets):  #   d - это красный квадратик k -  это его индекс
    print("Detection {}: Left: {} Top: {} Right: {} Bottom: {}".format(
        k, d.left(), d.top(), d.right(), d.bottom()))
    rec = dlib.rectangle(d.left(), d.top(), d.right(), d.bottom())
    lst.append([rec, sqd(rec)])
    shape = sp(img, d)  # рассчитывает контрольные точки
    win1.clear_overlay()  # очищаем весь верхний слой картинок
    win1.add_overlay(d)  # Добавляет красненький квадратик на лицо но только
один за каждый проход цикла
    win1.add_overlay(shape)  # рисует
# lst.sort()

face_descriptor1 = facerec.compute_face_descriptor(img, shape)
```

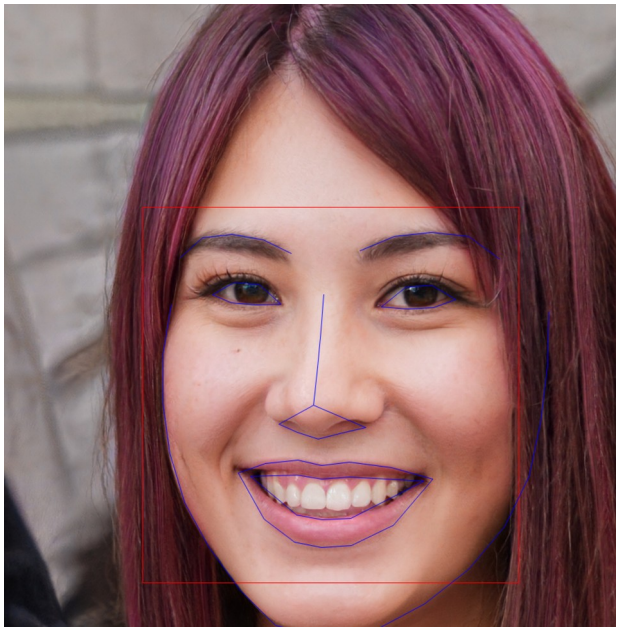
```
# _list.append(face_descriptor1)
return face_descriptor1
```

Тестирование

Попробуем отличить две фотографии разных людей. В качестве первой фотографии возьмём сгенерированную фотографию сервиса https://thispersondoesnotexist.com

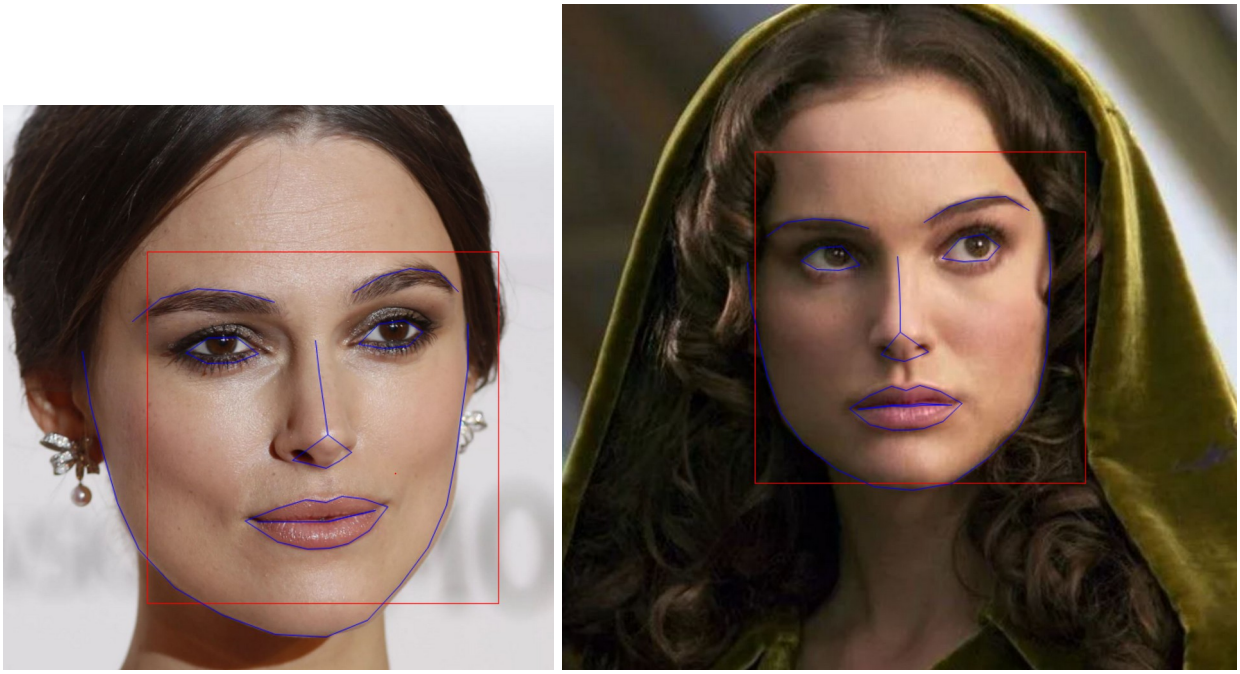
Второе изображение возьмемс того же сервиса.

При запуске программы наложение маски на изображение реперных точек выглядит следующим образом:



Для этих изображений евклидово расстояние составляет 0.9517 что позволяет однозначно классифицировать эти изобрадения, как фотографии двух разных людей.

А вот для следующих изображений евклидово расстояние между двумя дескрипторами составит всего 0.5785. На фотографиях изображены Кира Найтли и Натали Портман, но по рекомендация разработчиков dlib мы должны идентифицировать соотвествующие изображения, как фотографии одного человека.



Отсюда следует, что порог распознавания 0.6 слишком велик для идентификации похожих фотографий в реальной системе распознавания лиц. Для устранения этой проблемы необходимо рекомедательно снизить порог классификации с помощью наиболее похожих изображений разных людей. В данном случаее требуется установить порог евклидова расстояния не выше 0.57 с учетом погрешности и вероятности существования еще более похожих фотографий окончательное значение можно выставить на отметке 0.55.

Тестирование программы для видеопотока позволило убедиться в работоспособности детектора лиц на основе вейвлетов Хаара. С помощью функции `cv2.VideoCapture()` можно считывать видео не только с помощью веб камеры или иного устройства, но также и воспроизводить ранее записанные видеофалы. В качестве примера рассмотрим трейлер фильма Звезные войны.

Протестируем работу

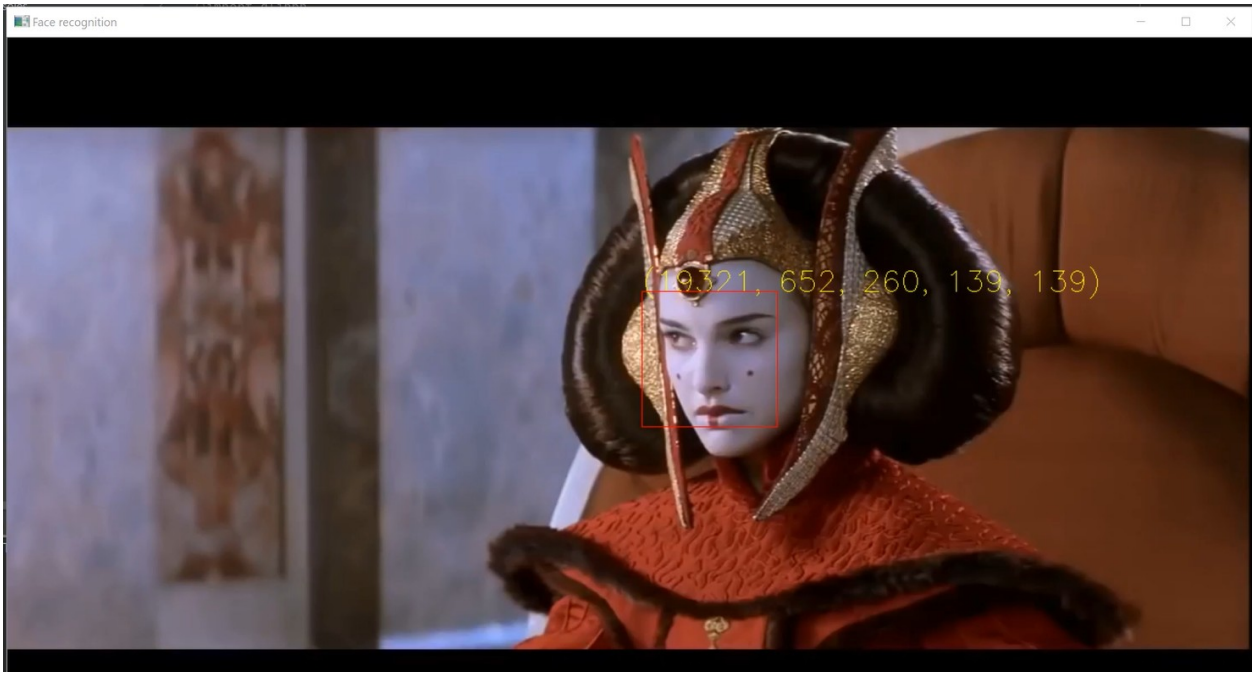


Рис. Распознавание единственного лица

Программа успешно справляется с изображениями, содержащими одно лицо. На кадре отображается площадь квадрата вокруг лица и координаты. Так как алгоритм поиска определил только одно лицо в кадре, оно подсвечивается рамкой красного цвета, что соответствует ближайшему объекту в кадре.

Для двух лиц алгоритм работает аналогичным образом, выделяя границы лиц и их параметры. Четко определенного максимального количества распознаваемых лиц на изображении не задано. Поэтому количетво найденных лиц будет ограничиваться только способностью алгоритма правильно определить теневые переходы.

Рис. Определение ближайшего лица из группы лиц

Как видно на рисунке ближайшее лицо выделено рамкой красного цвета, это означает, что именно для него будет выполнено построение дескриптора лица.

Вычислим дескриптор лица для видеозаписи:

Рис. Справа: изображение взятое из видеопотока, слева: построенный дескриптор лица.

Простейшей оценкой качества работы библиотек OpenCV и dlib для реализации систем распознавания лиц является количество ложных срабатываний при поиске лица. При ложном срабатывании алгоритм выделяет область изображения, не содержащую признаков лица. В первую очередь это происходит потому, что алгоритм находит знакомые ему теневые

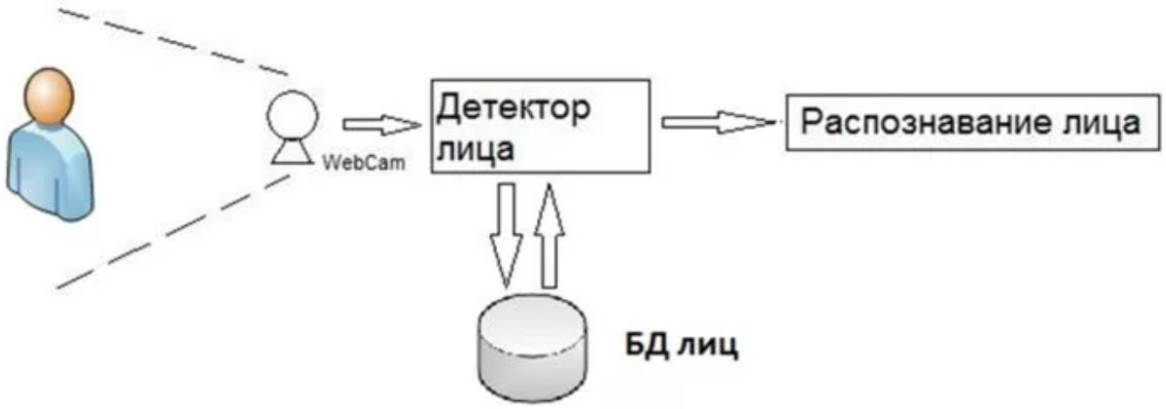
паттерны (переходы от ярких областей изображения к более темным) и классифицирует их, как возможное присутствие человеческого лица. Пример ложного срабатывания приведен на рисунке ниже:

Рис. Ложное срабатывание детоктора

Помимо ложных срабатываний стоит уделить внимание существенным проблеммам. В ходе реализации такого подхода были вывлены следующие недостатки:

При расчете дескриптора лица основной поток программы «тормозится» для выполнения вычислений. Выполение расчета дескриптора занимает существенное время и ограничивает возможности применения данного решения в системах распозванавания лиц реального времени. В качестве подхода к снижению влияния недостатков возможна доработка программы и разделения выполнения вычислений и поиска лиц на несколько потоков. Также возможно выполнение процесса оптимизации алгоритма рачета дескриптора для снижения времени выполения вычислений.

Дальнейшее развите программы предполагает сбор и сохранение дескрипторов лица в базе данных на фазе обработки информации и подготовки набора данных.



Ннния В соответствии с приведенной схемой, процесс идентификации человека будет состоять из построения дескриптора лица, изображения полученного с видеокамеры и сравнение меры схожести этого ихображения с изображениями, хранящимися в базе данных.

Список источников:

1. Рейнхард Клетте Компьютерное зрение. Теория и алгоритмы. – М.: ДМК Пресс, 2019 – 506с.
2. Ян Эрик Солем Программирование компьютерного зрения на языке Python. М.:ДМК Пресс, 2016. – 312с.
3. Гонсалес Р., Вудс Р. Цифровая обработка изображений. М.: Техносфера, 2012. – 1104 с.
4. https://habr.com/ru/company/mipt/blog/458190/ Интернет ресурс
5. https://www.asozykin.ru/courses/nnpython