|  |  |
| --- | --- |
| C:\Users\DENACT~1\AppData\Local\Temp\lu135925on38x.tmp\lu135925on3bu_tmp_3360867a00ce4d37.jpg | **Министерство науки и высшего образования 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)** |

ФАКУЛЬТЕТ Информатика и системы управления и искусственный интеллект

КАФЕДРА                  Системы обработки информации и управления

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

***К НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ***

***НА ТЕМУ:***

**Распознавание лиц**

Студент: 13.05.2022 Очеретная Светлана Вячеславовна

Группа: ИУ5-65Б (Подпись, дата) (И.О.Фамилия)

Руководитель Гапанюк Юрий Евгеньевич

(Подпись, дата) (И.О.Фамилия)

Консультант \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Подпись, дата) (И.О.Фамилия)

*2022 г.*

**Министерство науки и высшего образования 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)**

УТВЕРЖДАЮ

Заведующий кафедрой \_\_\_\_\_\_\_\_\_\_

(Индекс)

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

(И.О.Фамилия)

« \_\_\_\_\_ » \_\_\_\_\_\_\_\_\_\_\_\_ 20 \_\_\_\_ г.

**ЗАДАНИЕ**

**на выполнение научно-исследовательской работы**

по теме: Распознавание лиц \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

Студент группы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Очеретная Светлана\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Фамилия, имя, отчество)

Направленность НИР: учебная

Источник тематики: НИР

График выполнения НИР:     25% к \_\_\_ нед., 50% к \_\_\_ нед., 75% к \_\_ нед., 100% к \_\_\_ нед.

***Техническое задание:***

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ распознавание лиц\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

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

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

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

***Оформление научно-исследовательской работы:***

Расчетно-пояснительная записка на \_\_\_ листах формата А4.

Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.)

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

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

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

Дата выдачи задания « \_\_\_ » \_\_\_\_\_\_\_\_\_\_\_\_ 2022 г.

**Руководитель НИР**  \_\_\_ . \_\_\_ .2022  Гапанюк Ю.Е.

(Подпись, дата)                          (И.О.Фамилия)

**Студент** \_\_\_. \_\_\_ .2022 Очеретная С.В.

(Подпись, дата)                           (И.О.Фамилия)

Примечание: Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафедре.

# Введение

Цель научно-исследовательской работы – распознавание лица на фотографии с использованием трех методов:

- Алгоритм Eigenface (метод главных компонент в приложении к распознаванию лиц).

- Алгоритм Fisherface (линейный дискриминантный анализ).

- Алгоритм LBPH (гистограммы локальных бинарных шаблонов)

Eigenfaces и Fisherfaces находят математическое описание наиболее доминирующих признаков обучающей выборки в целом. LBP анализирует каждое лицо в обучающей выборке отдельно и независимо.

# Ход работы

**Алгоритм Eigenface.**

Работа алгоритма Eigenface основана на методе главных компонент. Предположим, что имеется база данных лиц, где изображения имеют размер N x N пикселей. Каждое изображение из базы данных представляют точкой в пространстве размерностью N \* N.

Основная идея алгоритма состоит в том, чтобы найти такой базис меньшей размерности, после проекции в который максимально сохраняется информация по осям с большой дисперсией и теряется информация по осям с маленькой дисперсией. Это нужно для того, чтобы оставить только ту информацию, которая бы характеризовала различия лиц и удалить ненужную информацию, которая не помогает правильно идентифицировать человека.

Основными недостатками алгоритма EigenFaces является отсутствие устойчивости к изменению условий освещенности и отсутствие инвариантности к аффинным преобразованиям.

**Алгоритм Fisherface.**

Алгоритм Fisherface предполагает наличие множества фотографий при разных условиях освещенности у каждой персоны в базе данных. В алгоритме, как и в Eigenface, предполагается поиск базиса, но такого, который позволил бы максимизировать дисперсию между множествами изображений лиц и одновременно минимизировать дисперсию внутри каждого множества.

За счет множества фотографий каждой персоны алгоритм получается устойчивым к изменениям условий освещенности, но сохраняет недостаток алгоритма Eigenface — отсутствие инвариантности к аффинным преобразованиям.

**Алгоритм LBPH**

Этот алгоритм берёт своё начало с анализа 2D текстуры. Суть алгоритма заключается в том, что мы разбиваем изображение на части и в каждой такой части каждый пиксель сравнивается с соседними восьмью пикселями. Если значение центрального пикселя больше соседнего, то пишем 0, в противном случае 1.

Для каждого пикселя у нас получается некоторое число. Далее на основе этих чисел для всех частей, на которые мы разбивали фотографию, считается гистограмма. Все гистограммы со всех частей объединяются в один вектор, характеризующий изображение в целом. Если мы хотим узнать, насколько похожи два лица, нам придется вычислить для каждого из них такой вектор и сравнить их.

Описанный подход позволяет захватывать очень мелкозернистые детали, позже оказалось, что LBPH не мог кодировать детали разного масштаба, из-за этого он был расширен, и теперь число соседей может варьироваться. Идея расширения заключалась в том, чтобы расположить соседей по кругу определенного радиуса, и, таким образом, выделялись признаки окрестностей.

### **Импорт необходимых библиотек**

import os  
import cv2  
import numpy as np  
import pandas as pd  
import matplotlib.pyplot as plt  
import math  
import seaborn as sns  
from sklearn.model\_selection import train\_test\_split  
from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report, balanced\_accuracy\_score  
import itertools  
import random  
sns.set(style="ticks")  
%matplotlib inline

### **Загрузка и анализ датасета**

Для проведения исследования возьмем [датасет LFW-People](https://www.kaggle.com/datasets/atulanandjha/lfwpeople) с сайта Kaggle Проанализируем его характеристики

data\_folder = "data/"  
# количество папок/классов/людей  
len(os.listdir(data\_folder))

5749

# количество картинок  
images\_count = sum([len(files) for r, d, files in os.walk(data\_folder)])  
print(images\_count)

13233

Считываем все картинки из всех папок (1 папка - 1 имя, которое будем принимать за класс)

def get\_images(data\_folder, min\_files\_count = 0, max\_files\_count = 10000):  
 names = []  
 images = []  
 classes\_count = []  
  
 for folder in os.listdir(data\_folder):  
 files = os.listdir(os.path.join(data\_folder, folder))  
 if (len(files) >= min\_files\_count):  
 classes\_count.append(len(files[:max\_files\_count]))  
 for name in files[:max\_files\_count]:  
 if name.find(".jpg") > -1 :  
 img = cv2.imread(os.path.join(data\_folder + folder, name))  
 images.append(img)  
 names.append(folder)  
  
 return images, names, classes\_count

images\_full, names\_full, classes\_count\_full = get\_images(data\_folder)

len(names\_full), len(images\_full), len(classes\_count\_full)

(13233, 13233, 5749)

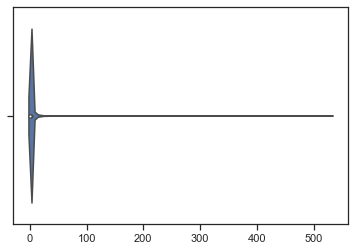
Выведем названия всех классов (имена людей, которые определяем)

labels\_full = np.unique(names\_full)  
labels\_full, len(labels\_full)

(array(['AJ\_Cook', 'AJ\_Lamas', 'Aaron\_Eckhart', ..., 'Zumrati\_Juma',  
 'Zurab\_Tsereteli', 'Zydrunas\_Ilgauskas'], dtype='<U35'),  
 5749)

Проверим дисбаланс классов

sns.violinplot(x=classes\_count\_full)  
plt.show()



Как видим, дисбаланс классов достаточно большой, т.к. у большинства людей в датасете всего по 1 фотке, но есть личности, кол-во фотографий которых достигает нескольки сотен. Качество модели такого набора будет плохим:

* Не получиться распознать человека с небольшим числом фотографий.
* Если фотография одна, то даже разделение на тестовый и обучащий наборы будет не корректным.
* Если количество фотографий одного класса будет превышать количество фотографий другого, то будет наблюдаться дисбаланс, один человек будет распознаваться хорошо, а другой нет.

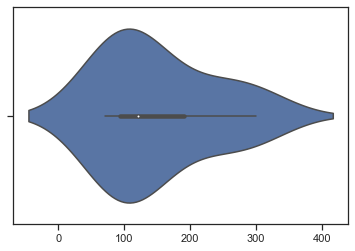
Чтобы все это исправить, будем работать только с классами, для которых количество фотографий будет больше 70ти и меньше 300 (значения были получены после нескольких тестовых попыток разделения).

images, names, classes\_count = get\_images(data\_folder, 70, 300)

len(names), len(images), len(classes\_count)

(1058, 1058, 7)

sns.violinplot(x=classes\_count)  
plt.show()

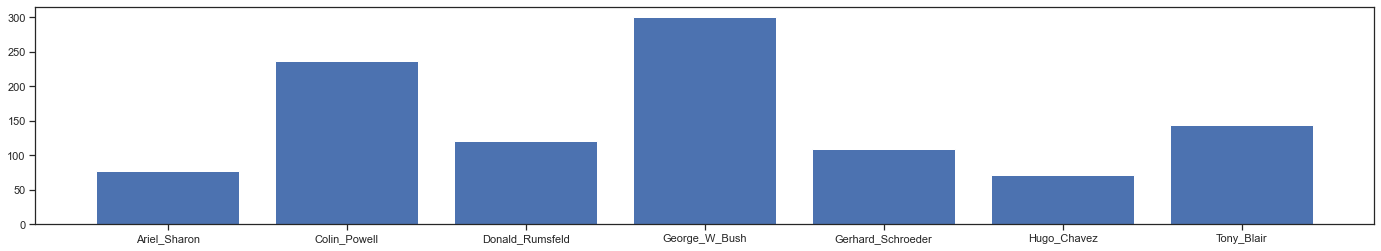


# все уникальные имена (классы)  
labels = np.unique(names)  
labels, len(labels)

(array(['Ariel\_Sharon', 'Colin\_Powell', 'Donald\_Rumsfeld', 'George\_W\_Bush',  
 'Gerhard\_Schroeder', 'Hugo\_Chavez', 'Tony\_Blair'], dtype='<U17'),  
 7)

fig, ax = plt.subplots()  
fig.set\_figwidth(24)  
ax.bar(labels, np.array(classes\_count))

<BarContainer object of 7 artists>



Дисбаланс классов устранен, папки с малым кол-вом фотографий рассмотрены не будут из-за недостатка информации для обучения. Теперь будет работать с полученными очищенными данными.

Просмотрим картинки одного из классов

def show\_dataset(images\_class, labels, photos\_count):  
 plt.figure(figsize=(18,photos\_count))  
 size\_limit = len(images\_class)  
 k = 0  
 for i in range(1,photos\_count+1):  
 if (k >= size\_limit):  
 k += 1  
 continue  
 plt.subplot(1,photos\_count,i)  
 try :  
 plt.imshow(images\_class[k][:,:,::-1])  
 except :  
 plt.imshow(images\_class[k], cmap='gray')  
 plt.title(labels[k], fontsize=10)  
 plt.axis('off')  
 plt.tight\_layout()  
 k += 1  
 plt.show()

def show\_photos\_by\_name(name, names, images, photos\_count):  
 example\_ids = np.where(name==np.array(names))[0]  
 example\_images\_class = images[example\_ids[0] : example\_ids[-1] + 1]  
  
 for i in range(math.ceil(len(example\_images\_class)/photos\_count)):  
 show\_dataset(example\_images\_class[photos\_count\*i:], names[example\_ids[0] : example\_ids[-1] + 1], photos\_count)

example\_label = labels[0]  
photos\_count = 10  
  
show\_photos\_by\_name(example\_label, names[:20], images[:20], photos\_count)





### **Использование метода Виолы-Джноса для обнаружения лица**

Для определения лица на фото используем каскад библиотеки opencv: [каскады\_opencv](https://github.com/opencv/opencv/tree/4.x/data/haarcascades)

face\_cascade = cv2.CascadeClassifier('haarcascades\haarcascade\_frontalface\_default.xml')

Функция определения лица по фото, возвращает вырезанное из фото лицо, если оно было обнаружено

def detect\_face(img, idx):  
 img = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)  
 faces = face\_cascade.detectMultiScale(img, 1.3, 5)  
   
 try:  
 x, y, w, h = faces[0]  
  
 img = img[y:y+h, x:x+w]  
 img = cv2.resize(img, (100, 100))  
 except:  
 img = None  
 return img

Получаем массив определенных (вырезанных из фото) лиц

def cut\_photos(images, names, start\_point\_for\_name, logs = False):  
 croped\_images = []  
 croped\_names = []  
  
 for i, img in enumerate(images):  
 img = detect\_face(img, i)  
 if img is not None:  
 croped\_images.append(img)  
 croped\_names.append(names[start\_point\_for\_name + i])  
 else:  
 if (logs == True):  
 print('Фото ', i, ' представителя ', names[start\_point\_for\_name + i], ' не было обнаружено')  
  
 return croped\_images, croped\_names

example\_ids = np.where(example\_label==np.array(names))[0]  
example\_images\_class = images[example\_ids[0] : example\_ids[-1] + 1]  
  
example\_croped\_images, example\_croped\_names = cut\_photos(example\_images\_class, names, example\_ids[0], True)

Фото 16 представителя Ariel\_Sharon не было обнаружено  
Фото 30 представителя Ariel\_Sharon не было обнаружено  
Фото 35 представителя Ariel\_Sharon не было обнаружено  
Фото 38 представителя Ariel\_Sharon не было обнаружено

len(example\_images\_class), len(example\_croped\_images), len(example\_croped\_names)

(77, 73, 73)

print('Процент обнаруженных фото: ', len(example\_croped\_images) / len(example\_images\_class) \* 100, '%')

Процент обнаруженных фото: 94.8051948051948 %

show\_photos\_by\_name(example\_label, example\_croped\_names[:20], example\_croped\_images[:20], photos\_count)



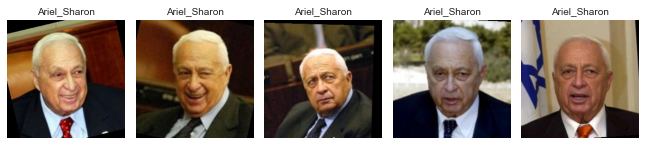


Теперь посмотрим первые фото для всего набора

def show\_photos(images, labels, photos\_count\_size, photos\_count\_in\_line):  
 rows\_count = math.ceil(len(images[:photos\_count\_size])/photos\_count\_in\_line)  
 for i in range(rows\_count):  
 photos = images[photos\_count\_in\_line\*i:photos\_count\_size\*(i+1)]  
 labels\_limited = labels[photos\_count\_in\_line\*i:photos\_count\_size\*(i+1)]  
 if (i + 1 == rows\_count):  
 photos = images[photos\_count\_in\_line\*i:photos\_count\_size]  
 labels\_limited = labels[photos\_count\_in\_line\*i:photos\_count\_size]  
 show\_dataset(photos, labels\_limited, photos\_count\_in\_line)

photos\_count\_size = 15  
show\_photos(images, names, photos\_count\_size, photos\_count)





Определим и вырежем лица из всего набора

croped\_images, croped\_names = cut\_photos(images, names, 0)

print('Процент обнаруженных фото: ', len(croped\_images) / len(images) \* 100, '%')

Процент обнаруженных фото: 97.16446124763705 %

### **Разделение выборки на обучающую и тестовую**

target = np.array([np.where(name == labels)[0][0] for name in croped\_names])  
print(target[:200])

[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1  
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1  
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1  
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1  
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]

len(croped\_images), len(croped\_names), len(target)

(1028, 1028, 1028)

# объединение фотографий, имен и целевого признака в один массив, чтобы далее все вместе разделить  
data = np.array([[croped\_images[i], croped\_names[i], target[i]] for i in range(len(croped\_images))])

# разделение data на data\_split(image+name) и target\_split(class)  
data\_split = []  
target\_split = []  
data\_split\_idx = data[0][2]  
data\_split.append([])  
target\_split.append([])  
for d in data:  
 if (d[2] != data\_split\_idx):  
 data\_split\_idx = d[2]  
 data\_split.append([])  
 target\_split.append([])  
 data\_split[data\_split\_idx].append(d)  
 target\_split[data\_split\_idx].append(int(d[2]))

def get\_column(data, idx):  
 return [row[idx] for row in data]

images\_train = []  
names\_train = []  
target\_train = []  
images\_test = []  
names\_test = []  
target\_test = []  
  
for d in range(len(data\_split)):  
 X\_train, X\_test, y\_train, y\_test = train\_test\_split(data\_split[d], target\_split[d], test\_size=0.3, random\_state=42)  
 images\_train += get\_column(X\_train, 0)  
 names\_train += get\_column(X\_train, 1)  
 target\_train += get\_column(X\_train, 2)  
 images\_test += get\_column(X\_test, 0)  
 names\_test += get\_column(X\_test, 1)  
 target\_test += get\_column(X\_test, 2)

Объединение столбиков и перемешивание датасета

data\_union\_train = np.array([[images\_train[i], names\_train[i], target\_train[i]] for i in range(len(images\_train))])  
data\_union\_test = np.array([[images\_test[i], names\_test[i], target\_test[i]] for i in range(len(images\_test))])

np.random.shuffle(data\_union\_train)  
np.random.shuffle(data\_union\_test)

images\_train\_shuffled = get\_column(data\_union\_train, 0)  
names\_train\_shuffled = get\_column(data\_union\_train, 1)  
target\_train\_shuffled = get\_column(data\_union\_train, 2)  
images\_test\_shuffled = get\_column(data\_union\_test, 0)  
names\_test\_shuffled = get\_column(data\_union\_test, 1)  
target\_test\_shuffled = get\_column(data\_union\_test, 2)

len(target\_train\_shuffled), len(images\_train\_shuffled), len(names\_train\_shuffled)

(716, 716, 716)

len(target\_test\_shuffled), len(images\_test\_shuffled), len(names\_test\_shuffled)

(312, 312, 312)

show\_photos(images\_test\_shuffled, names\_test\_shuffled, 50, 10)











### **Распознавание лиц с помощью методов Eigen Faces, Fisher Faces, Local Binary Pattern Histograms**

model\_eigenfaces = cv2.face.EigenFaceRecognizer\_create()  
model\_fisherfaces = cv2.face.FisherFaceRecognizer\_create()  
model\_lbph = cv2.face.LBPHFaceRecognizer\_create()  
models = [model\_eigenfaces, model\_fisherfaces, model\_lbph]  
models\_names = ["model\_eigenfaces", "model\_fisherfaces", "model\_lbph"]

Обучим модели:

for i in range(len(models)):  
 models[i].train(images\_train\_shuffled, np.array(target\_train\_shuffled))  
 models[i].save(models\_names[i] + ".yml")  
 models[i].read(models\_names[i] + ".yml")

Распознавание лиц для всех моделей:

classes\_real = []  
classes\_predict = []  
images\_real = []  
images\_predict = []  
confidences = []  
  
for i in range(len(models)):  
 classes\_real.append([])  
 classes\_predict.append([])  
 images\_real.append([])  
 images\_predict.append([])  
 confidences.append([])  
  
 for test\_idx in range(len(names\_test\_shuffled)):  
 idx, confidence = models[i].predict(images\_test\_shuffled[test\_idx])  
  
 classes\_real[i].append(np.where(names\_test\_shuffled[test\_idx] == labels)[0][0])  
 classes\_predict[i].append(idx)  
 confidences[i].append(confidence)  
 images\_real[i].append(images\_test\_shuffled[test\_idx])  
 images\_predict[i].append(images\_test\_shuffled[idx])

number\_of\_show\_model = 1  
  
for i in range(5):  
 print("Распознанное имя: ", labels[classes\_predict[number\_of\_show\_model][i]])  
 print("Реальное имя: ", labels[classes\_real[number\_of\_show\_model][i]])  
 print("Confidence: ", confidences[number\_of\_show\_model][i])  
  
 show\_photos([images\_predict[number\_of\_show\_model][i], images\_real[number\_of\_show\_model][i]], ['Распознанный человек', 'Реальный человек'], 2, 10)

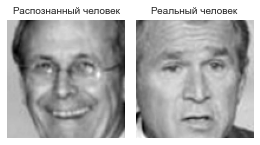
Распознанное имя: Ariel\_Sharon  
Реальное имя: Ariel\_Sharon  
Confidence: 83.700856122778



Распознанное имя: George\_W\_Bush  
Реальное имя: George\_W\_Bush  
Confidence: 100.02266132588427



Распознанное имя: George\_W\_Bush  
Реальное имя: George\_W\_Bush  
Confidence: 57.563370024660585



Распознанное имя: Donald\_Rumsfeld  
Реальное имя: Donald\_Rumsfeld  
Confidence: 22.420038790347323



Распознанное имя: Tony\_Blair  
Реальное имя: George\_W\_Bush  
Confidence: 70.3983089335857

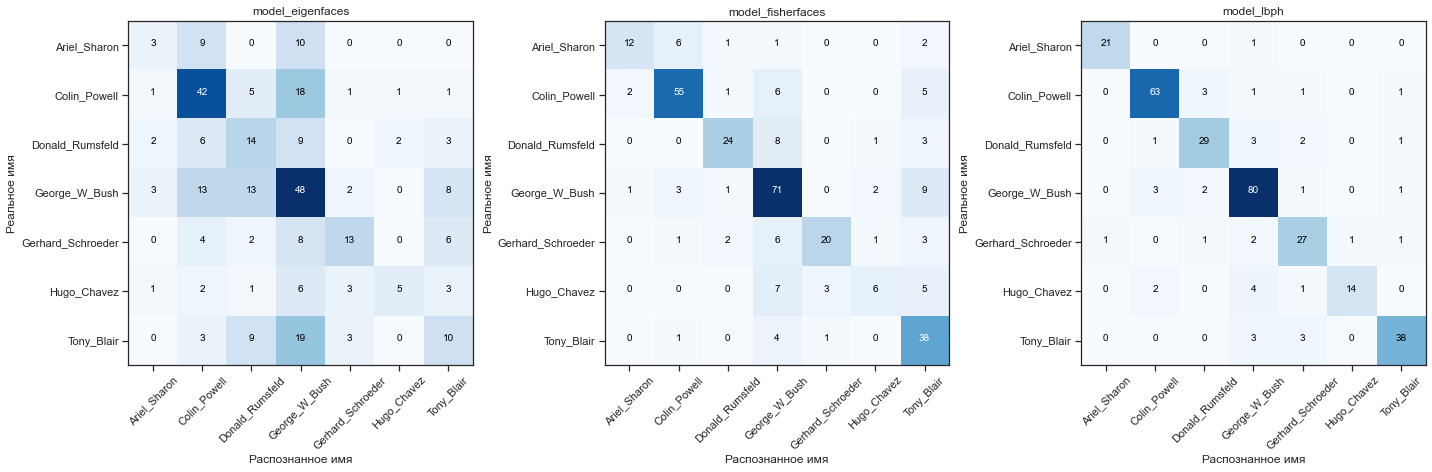


for i in range(len(models)):  
 print("Model ", models\_names[i])  
 print("Accuracy:", accuracy\_score(classes\_real[i], classes\_predict[i]))  
 print("Balanced accuracy:", balanced\_accuracy\_score(classes\_real[i], classes\_predict[i]), '\n')

Model model\_eigenfaces  
Accuracy: 0.4326923076923077  
Balanced accuracy: 0.3635685249521189   
  
Model model\_fisherfaces  
Accuracy: 0.7243589743589743  
Balanced accuracy: 0.6543894101186883   
  
Model model\_lbph  
Accuracy: 0.8717948717948718  
Balanced accuracy: 0.8487385095331123

def plot\_confusion\_matrix(cm, classes,  
 normalize=False,  
 title='Confusion matrix',  
 cmap=plt.cm.Blues):  
 if normalize:  
 cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]  
   
 plt.imshow(cm, interpolation='nearest', cmap=cmap)  
 plt.title(title)  
 plt.tight\_layout()  
 tick\_marks = np.arange(len(classes))  
 plt.xticks(tick\_marks, classes, rotation=45)  
 plt.yticks(tick\_marks, classes)  
  
 fmt = '.2f' if normalize else 'd'  
 thresh = cm.max() / 2.  
 for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):  
 plt.text(j, i, format(cm[i, j], fmt),  
 horizontalalignment="center",  
 color="white" if cm[i, j] > thresh else "black")  
  
 # plt.colorbar()  
 plt.ylabel('Реальное имя')  
 plt.xlabel('Распознанное имя')

plt.figure(figsize=(20,10))  
  
for i in range(len(models)):  
 cnf\_matrix = confusion\_matrix(classes\_real[i], classes\_predict[i])  
 plt.subplot(1,len(models),i+1)  
 plot\_confusion\_matrix(cnf\_matrix, classes=labels,normalize=False,title=models\_names[i])  
  
plt.show()



По матрицам видно, что лучше всего

for i in range(len(models)):  
 print("Model ", models\_names[i])  
 print(classification\_report(classes\_real[i], classes\_predict[i], target\_names=labels))

Model model\_eigenfaces  
 precision recall f1-score support  
  
 Ariel\_Sharon 0.30 0.14 0.19 22  
 Colin\_Powell 0.53 0.61 0.57 69  
 Donald\_Rumsfeld 0.32 0.39 0.35 36  
 George\_W\_Bush 0.41 0.55 0.47 87  
Gerhard\_Schroeder 0.59 0.39 0.47 33  
 Hugo\_Chavez 0.62 0.24 0.34 21  
 Tony\_Blair 0.32 0.23 0.27 44  
  
 accuracy 0.43 312  
 macro avg 0.44 0.36 0.38 312  
 weighted avg 0.44 0.43 0.42 312  
  
Model model\_fisherfaces  
 precision recall f1-score support  
  
 Ariel\_Sharon 0.80 0.55 0.65 22  
 Colin\_Powell 0.83 0.80 0.81 69  
 Donald\_Rumsfeld 0.83 0.67 0.74 36  
 George\_W\_Bush 0.69 0.82 0.75 87  
Gerhard\_Schroeder 0.83 0.61 0.70 33  
 Hugo\_Chavez 0.60 0.29 0.39 21  
 Tony\_Blair 0.58 0.86 0.70 44  
  
 accuracy 0.72 312  
 macro avg 0.74 0.65 0.68 312  
 weighted avg 0.74 0.72 0.72 312  
  
Model model\_lbph  
 precision recall f1-score support  
  
 Ariel\_Sharon 0.95 0.95 0.95 22  
 Colin\_Powell 0.91 0.91 0.91 69  
 Donald\_Rumsfeld 0.83 0.81 0.82 36  
 George\_W\_Bush 0.85 0.92 0.88 87  
Gerhard\_Schroeder 0.77 0.82 0.79 33  
 Hugo\_Chavez 0.93 0.67 0.78 21  
 Tony\_Blair 0.90 0.86 0.88 44  
  
 accuracy 0.87 312  
 macro avg 0.88 0.85 0.86 312  
 weighted avg 0.87 0.87 0.87 312

len(classes\_real), len(classes\_predict), len(names\_test\_shuffled), len(labels)

(3, 3, 312, 7)

Вывод: модель LBPH обладает наилучшим качеством распознавания лиц по фотографиям.