**Санкт-Петербургский государственный университет Факультет прикладной математики и процессов управления**

**Лабораторная работа №1**

**Система обнаружения и распознавания лиц**

|  |  |  |
| --- | --- | --- |
| Студент гр. Б15-ПУ |  | Шайдуров В.Д. |
| Преподаватель |  | Щеголева Н.Л. |

Санкт-Петербург

2022 г.

**Оглавление**

[1. **Цель работы** 3](#_Toc1)

[2. **Задача** 3](#_Toc2)

[3. **Теоретическая часть** 3](#_Toc3)

[4. **Описание программы** 5](#_Toc4)

[4.1. **Блок схема** 5](#_Toc5)

[4.2. **Описание функций** 6](#_Toc6)

[5. **Эффективность методов**  8](#_Toc7)

[6. **Контрольный пример** 9](#_Toc8)

[7. **Промежуточный вывод** 10](#_Toc9)

[8. **Рекомендации пользователя** 12](#_Toc10)

[9. **Рекомендации программиста** 12](#_Toc11)

[10. **Вывод** 13](#_Toc12)

[11. **Список литературы** 14](#_Toc13)

# **Цель работы**

Исследовать простые методы распознавания лиц и оценить их точность.

# **Задача**

Реализовать программу, определяющую человека по лицу из базы эталонных изображений, применяя простейшие методы распознавания лиц. Используя открытую базу данных AT&T Database of Faces оценить точность выбранных методов.

# **Теоретическая часть**

Мною были рассмотрены следующие методы распознавания лиц:

*Метод случайных точек*:

Заключается в выборе случайных n пикселей искомого изображения и сравнения этих пикселей с пикселями стоящих на тех же местах в изображениях из базы данных.

*Локальный бинарный паттерн*:

Это простой, но очень эффективный оператор текстуры, который помечает пиксели изображения, устанавливая порог соседства каждого пикселя, и рассматривает результат как двоичное число. Используя LBP в сочетании с гистограммами, мы можем представить изображения лица простым вектором данных.

**Заметка**: вы можете прочитать больше о LBPH здесь: [https://machinelearningmastery.ru](https://machinelearningmastery.ru/face-recognition-how-lbph-works-90ec258c3d6b/)

*Метод масштабирования:*

Заключается в уменьшении изображения, тем самым понижая его качество, другими словами, находится среднее значение соседних пикселей. Тем самым становится удобно записывать и сравнивать изображения по пикселям, так-как требуется меньше памяти и вычислительных ресурсов для хранения и сравнения изображений.

*Быстрое преобразование Фурье:*

Данный метод был использован в качестве эксперимента, и я использовал встроенную функцию из библиотеки scipy. Подробнее про данный способ вы можете прочитать здесь: [ссылка](https://ofvp.phys.msu.ru/wp-content/uploads/2021/03/diskretnoe-preobrazovanie-fure.pdf).

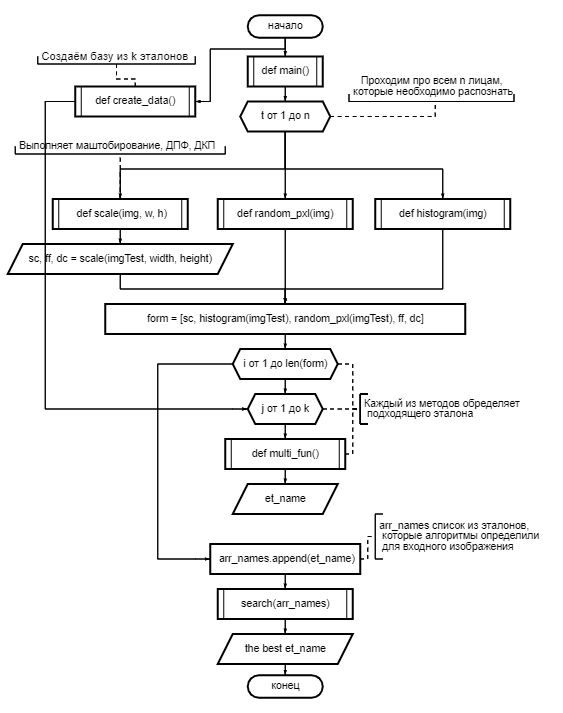
*Дискретное косинусное преобразование:*

Данный метод также был использован в качестве эксперимента, и я использовал встроенную функцию из библиотеки scipy. Подробнее про данный способ вы можете прочитать здесь: [ссылка](http://repo.ssau.ru/bitstream/Uchebnye-posobiya/Bystrye-algoritmy-diskretnogo-kosinusnogo-preobrazovaniya-Elektronnyi-resurs-elektron-ucheb-posobie-54333/1/%D0%A7%D0%B8%D1%87%D0%B5%D0%B2%D0%B0 %D0%9C.%D0%90. %D0%91%D1%8B%D1%81%D1%82%D1%80%D1%8B%D0%B5 %D0%B0%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC%D1%8B %D0%B4%D0%B8%D1%81%D0%BA%D1%80%D0%B5%D1%82%D0%BD%D0%BE%D0%B3%D0%BE.pdf).

# **Описание программы**

Код обработки данных и алгоритмы определения лиц написаны на языке Python.

# **Блок схема**



*Блок-схема системы распознавания лиц.*

# **Описание функций**

Список используемых библиотек.

Таблица 4.1

|  |  |
| --- | --- |
| Библиотеки | Описание |
| scipy.fftpack | * Используется для вычисления быстрого преобразования Фурье и дискретного косинусного преобразования |
| numpy | * Организует быструю работу с массивами |
| cv2 | * Используется для считывания изображений из папки, и для вызова встроенных методов, таких как построение гистограммы и сжатие изображения. |
| matplotlib | * Предназначена для визуализации работы программы |

В программе используются 6 функций: 4 связанны с реализацией алгоритмов распознавания и 2 непосредственно с обработкой данных. В таблице 4.2 представлено их описание.

Таблица 4.2

|  |  |  |
| --- | --- | --- |
| Имя функции | Входные данные | Описание функции |
| get\_lbph() | img – изображение | * Функция принимает на вход изображение img и возвращает его изменённое состояние с помощью линейного бинарного паттерна. |
| get\_scale() | img – изображение,  w – его ширина ,  h – его высота | * Функция принимает на вход изображение img шириной и высотой (w, h). Изменяет (w, h) до новых переменных (new\_w, new\_h), и возвращает в массиве уменьшенное изображение размеров (new\_w, new\_h). Мною было применено уменьшение изображения в x5 раз, подобранное методом перебора как наилучшее. * А также возвращает ещё два массива: быстрого преобразования Фурье и дискретного косинусного преобразования над img. С использованием методов fft() и dct() встроенных в библиотеку scipy. |
| get\_random\_pxl() | img – изображение | * Функция принимает на вход изображение img и возвращает массив пикселей взятых по заранее выбранным случайным образом координатам. Экспериментальным путём было выявлено оптимальное количество случайных точек (p = 500) для заданной базы данных. |
| get\_histogram() | img – изображение | * Функция принимает на вход изображение img и возвращает гистограмму изображения разбитую на 32 столбца в массив. |
| search() | arr\_names – список выбранных эталонов | * Путём перемножения вероятностей алгоритмов находится имя наиболее вероятно-подходящего человека из arr\_names для imgTest (искомого изображения). |
| main() | - | * Считываются все изображения imgTest, для которых требуется найти эталонное изображение. * Находится эталон для каждого изображения imgTest по каждому из методов. Получая массив arr\_names возможных имён. * Вызывается search(arr\_names), и выводится имя наиболее вероятного эталона. |

Функции для графического представления работы программы:

Таблица 4.3

|  |  |  |
| --- | --- | --- |
| Имя функции | Входные данные | Описание функции |
| make\_empty() | - | * Очищаются старые значения с экрана. |

Использование классов:

Таблица 4.4

|  |  |  |
| --- | --- | --- |
| Имя класса | Доступные функции | Описание класса |
| Person | get\_form – возвращает вектор признаков изображения основанных на методах распознавания  out\_form – возвращает вектор признаков изображения в более удобном формате для визуализации | self.img – изображение  self.name – название изображения  get\_lbph – линейно бинарный паттерн с применением гистограммы  self.scale – уменьшенное изображение  self.FFT – быстрое преобразование Фурье над img  self.DCT – дискретное косинусное преобразование  self.lbp\_img – представление линейно бинарного паттерна удобного для визуализации (без гистограммы)  self.random\_pxl – метод случайных точек над img  self.pxl\_img – метод случайных точек удобный для визуализации |

# **Эффективность методов**

1. Метод случайных точек: точность >72,5%.

Модификации: не использовались.

1. Метод гистограммы + LBP = LBPH: точность 77.5%.

Модификации: разбиение на несколько равных частей и подсчёт гистограммы для каждой части, таким образом удалось добиться улучшения с 60% до 77.5%.

1. Метод масштабирования: точность 74.4%. Модификации: не использовались.
2. [Метод быстрого преобразования Фурье](https://habr.com/ru/company/otus/blog/449996/): точность 74.4%. Модификации: непосредственное применение к изначально уменьшенному изображению и перегруппирование выходного массива функций hfft(), размещая нулевую частоту в центре спектра и изъятие самой информативной центральной части, таким образом удалось добиться улучшения с 65% до 74.4%.
3. [Метод дискретного косинусного преобразования](https://habr.com/ru/post/206264/): точность 74.1%.

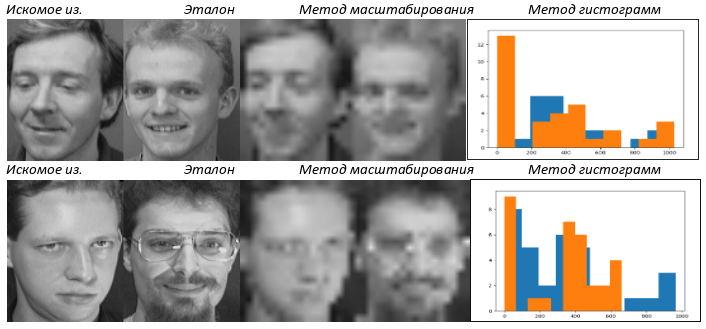
Модификации: изъятие самой информативной части изображения выходного массива функции dct(), а именно первая половина от всех столбцов. Удалось добиться улучшения с 68% до 74.1%.

Зависимость количества распознанных изображений от количества эталонов на каждого человека:

# **Контрольный пример**

Особые случаи, когда с помощью алгоритмов программа не смогла определить верного эталона:

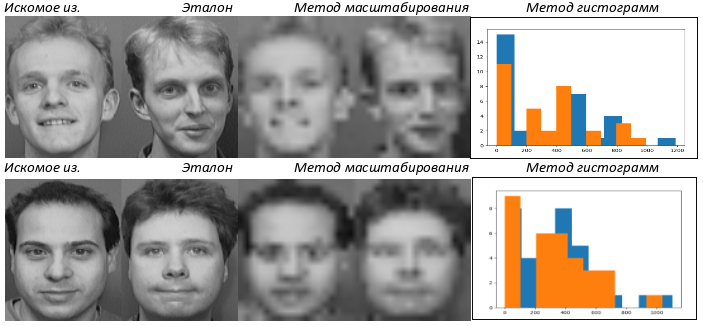
***Случай распознавания лиц по одному эталонному изображению на человека:***



*Рисунок 6.1.*

Программа могла сработать неверно, так как лица повёрнуты больше чем на 20 градусов. Например, библиотека cv2, используемая для обнаружении лиц, не способна определить лицо на данных изображениях.

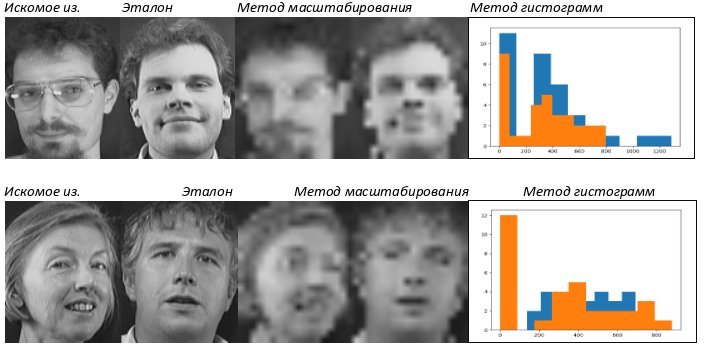
***Случай распознавания лиц по двум эталонным изображениям на человека:***



*Рисунок 6.2.*

Программа могла сработать неверно, так как лица похожи.

***Случай распознавания лиц по трём эталонным изображениям на челека:***



*Рисунок 6.3.*

Программа могла сработать не верно в первом случае: так как человек носит очки, во втором: лицо повернуто больше чем на 20 градусов.

# **Промежуточный вывод**

На примере поиска изображения по одному эталону, объединение рассмотренных методов даёт точность >75% (не постоянная из-за метода случайных точек). Но при исключении метода случайных точек процент возрастет до 77,2%. Это не удивительно, из следующих соображений:

Совместная работа всех алгоритмов выдаёт 272 верных распознавания из 360. В то же время алгоритмы сработали одинаково, т.е. предложили одного и того же эталона 229 раз, тогда как в 272 - 229 = 43 случаях как минимум один из алгоритмов предложил альтернативный вариант.

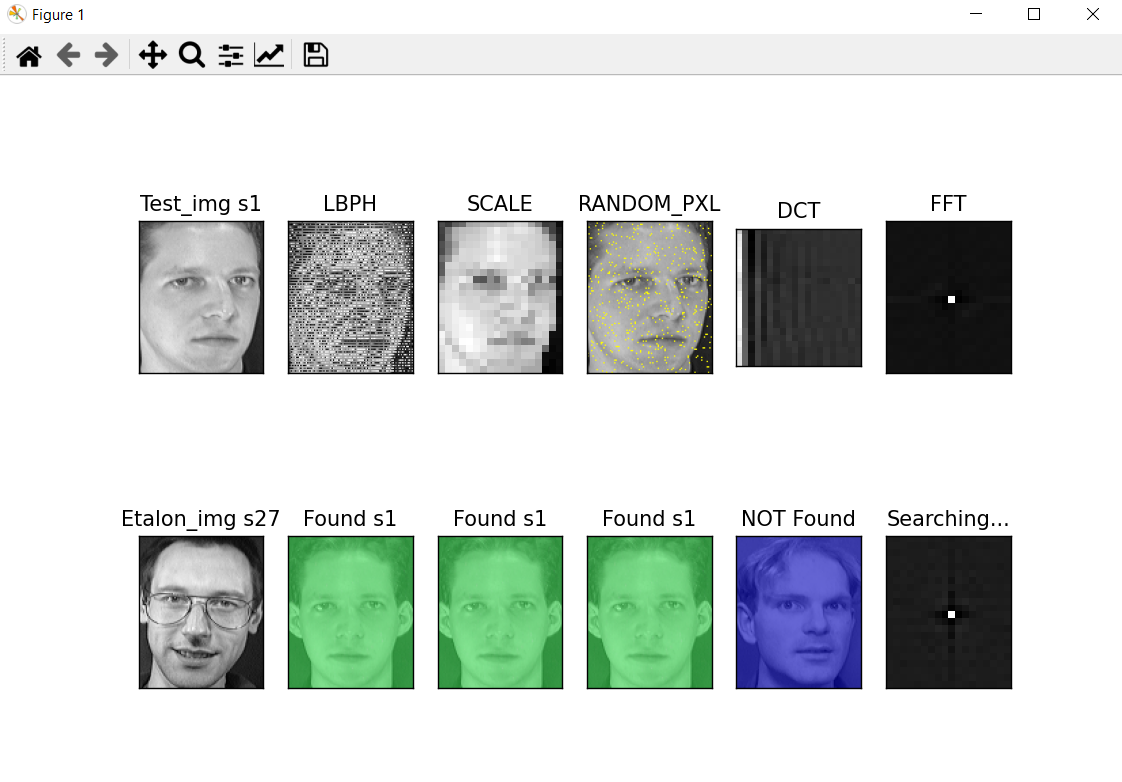
Данное замечание отводится, чтобы показать что алгоритмы работают в большинстве случаев одинаково (т.е. покрывают одно и тоже множество решений). Это становится ещё боле заметно, если рассматривать совместную работу не сразу всех пятерых методов, а любых 4, 3 или даже 2-ух, тогда кол-во случаев, когда алгоритмы сработали уникально падает до диапазона от 3 до 19 – что не превосходит в большинстве и 5%. Таким образом нам постоянно приходится делать лишнюю работу при использовании нескольких методов, борясь за пару процентов прироста распознавания программы.

Метод случайных точек выдал отличающегося от остальных эталона 15 раз из 360, что составляет чуть более 4%. При этом верными из этих 15 оказалось всего 4, что примерно чуть более 1%, но при этом все они отвергнуты на голосование, так как уникальные. Не смотря на теоретический прирост в более чем на 1% мы получаем падение процента распознавания при включении данного метода, так-как точность данного метода самая наихудшая и немного больше 72.5% в среднем случае, а на голосование неверные результаты имеют свойство накладываются друг на друга, что заставляет программу определить не того эталона, да и невозможно определить этот 1%, когда другие алгоритмы могут одновременно в своём большинстве найти другого эталона.

Приведённые рассуждения справедливы и для остальных методов, из-за чего не возможно классической моделью голосования достичь точности выше чем наибольшая точность из представленных алгоритмов, в моём случае это линейный бинарный паттерн с использованием гистограмм 77,5%.

# **Рекомендации пользователя**

При запуске проекта открывается окно, на котором представлена работа каждого из следующих алгоритмов по очереди: Бинарный паттерн с гистограммой, Масштабирование, Метод случайных точек, Дискретное косинусное преобразование и Быстрое преобразование Фурье. Далее производится сравнение тестового изображения со всеми изображениями из базы эталонов, и выбирается одно эталонное изображение наиболее схожее с тестовым. Если в результате выбранное изображение оказывается верным, то окрашивается в зелёный, если нет – в синий.



*Рисунок 8.1.*

# **Рекомендации программиста**

Для запуска программы необходима 64-битная операционная система Windows и рекомендованная версия Python 3.6 и выше. Для работы с кодом необходимо установить IDE PyCharm или любую другу среду разработки для python.

Проверьте список нужных библиотек, описаны в таблице 4.1.

# **Вывод**

В ходе выполненной работы, я изучил и дал оценку работы простейшим систем распознавания лиц. Исследовал способы их усовершенствования и комбинации друг с другом. Провёл полный анализ представленных данных и определил оптимальные константы, используемые для обнаружения одинаковых людей на вводимых данных базы AT&T Database of Faces.

# **Список литературы**

1. Статья МГУ <https://ofvp.phys.msu.ru/wp-content/uploads/2021/03/diskretnoe-preobrazovanie-fure.pdf>

2. Биометрические Системы. Г. А. Кухарев 2001

3. Ознакомиться с программной реализацией алгоритма можно по ссылке: <https://github.com/Mooncake911/face-detaction-recognation.git>