МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное
учреждение высшего образования
**"Южно-Уральский государственный университет
(национальный исследовательский университет)"**
**Высшая школа электроники и компьютерных наук**
**Кафедра системного программирования**

РАБОТА ПРОВЕРЕНА

Рецензент
Рук. отд. ООО «Инфиннити»

_____С.П. Кочкин

“___”_____ 2017 г.

ДОПУСТИТЬ К ЗАЩИТЕ

Заведующий кафедрой, д.ф.-м.н., профессор

_____ Л.Б. Соколинский

“___”_____ 2017 г.

# РАЗРАБОТКА ПРОГРАММЫ РАСПОЗНАВАНИЯ СКАНИРОВАННЫХ ПАСПОРТНЫХ ДАННЫХ ДЛЯ ИДЕНТИФИКАЦИИ ЛИЧНОСТИ

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ЮУрГУ – 02.04.02.2017.115–057.ВКР

Научные руководители:
ст. преподаватель кафедры СП
_____ Н.С. Силкина,
к. ф.-м. н., доцент кафедры СП
_____ С.А. Иванов

Автор работы,
студент группы КЭ-217

_____Е.А. Бутков

Ученый секретарь
(нормоконтролер)

_____ О.Н. Иванова

“___”_____ 2017 г.

Челябинск-2017

**ОГЛАВЛЕНИЕ**

# ВВЕДЕНИЕ

## Актуальность темы

Проблема автоматизации документооборота всегда стояла достаточно остро. Увеличивается количество задач. Растут приложения для решения данных задач. Растет количество задач и растет количество приложений. Одно из актуальных направлений – распознавание данных.

## Цель и задачи исследования

Разработка настольного приложения для распознавания и идентификации паспортных данных. Для достижения поставленной цели необходимо решить следующие задачи:

1) провести анализ существующих алгоритмов распознавания лиц на изображении;
2) осуществить поиск библиотек, в которых реализованы алгоритмы распознавания лиц и текста;
3) спроектировать приложение;
4) реализовать приложение;
5) провести внедрение.

## Структура и объем работы

Работа состоит из введения, 5 глав, заключения и библиографии. Объем работы составляет 42 страницы, объем библиографии – 20 наименований.

## Содержание работы

Первая глава, «Анализ предметной области», описывает функциональные и нефункциональные требования к программе.

Вторая глава, «Анализ алгоритмов распознавания лиц», описывает анализ подходов к распознаванию лиц. Проанализированы алгоритм по локализации лица на изображении и два алгоритма по распознаванию лица.

Третья глава, «Проектирование», описывает проектирование системы. Приведены диаграмма вариантов использования, диаграмма классов, диаграмма компонентов, диаграмма базы данных.

Четвертая глава, «Реализация», описывает реализацию программы.

Пятая глава, «Тестирование», посвящена тестированию программы.

В заключении описываются основные результаты, полученные при выполнении дипломной работы.

# 1. ПРЕДМЕТНАЯ ОБЛАСТЬ

ЧРОО РЭК «Сделаем» – это Челябинская региональная общественная организация развития экологической культуры [15]. Целью создания организации является формирование высокого уровня экологической культуры населения Челябинской области. В настоящий момент ЧРОО РЭК «Сделаем» ведет восемь социальных проектов, одним из этих проектов является социально-предпринимательский проект «Веще-ворот». Основная цель данного проекта – это сбор текстиля у населения и выдача вещей нуждающимся. Для получения вещей нуждающемуся необходимо написать соответствующее заявление о необходимости получения конкретного типа одежды и сделать фото с полученной вещью (для отчетности). Таким образом, за все время существования данного проекта накопилось довольно много отчетных фотографий. С января 2017 года выдача вещей производится при предъявлении паспорта. Таким образом, актуальной становится задача сопоставления старого банка фотографий с личностями, обращающимся в проект, на основе сканированного изображения паспорта. Для решения поставленной задачи необходимо реализовать программу, которая распознает сканированные паспортные данные для идентификации личности.

Были выявлены следующие требования к системе:

- программа должна обеспечивать распознавание лица и паспортных данных с паспорта;
- хранение паспортных данных в удобном формате;
- поддерживать построение отчетов с помощью шаблона;
- программа должна быть с интуитивно понятным интерфейсом.

**Анализ существующих решений**

Наиболее популярные решения:

- ABBYY PassportReader;
- PassportVision;

- SmartIdReader.

Рассмотрим подробнее данные решения.

**ABBYY PassportReader**

ABBYY PassportReader - Решение для извлечения данных из документов удостоверяющих личность.

Программа дает возможность:

- значительно сократить количество ошибок при вводе данных, связанных с человеческим фактором;
- ускорить ввод данных в 7-10 раз по сравнению с ручным вводом;
- повысить лояльность клиентов за счет ускорения обслуживания.

**PassportVision**

PassportVision – программный продукт, способный очень быстро распознавать данные документов.

Основные возможности:

- высокая скорость работы;
- экраны с распознанными данными;
- распознает в разрешении от 300 dpi;
- может распознавать паспорт, заграничный паспорт, страховое свидетельство, электронный паспорт, водительское удостоверение.

**SmartIdReader**

SmartIdReader – решение для распознавания данных 2-й и 3-й страницы паспорта РФ по отдельности или всего разворота на сервере, десктопе или мобильном устройстве.

Основные возможности:

- поиск паспорта РФ в видеопотоке;
- поиск паспорта РФ на фотографии;
- распознавания паспорта РФ;
- экраны с распознанными данными.

**Сравнение решений**

Сравнительная характеристика решений, представлена на табл. 1.

**Табл. 1.** Сравнение решений

| | **PassportReader** | **PassportVision** | **SmartIdReader** |
|---|---|---|---|
| Распознавания лица | - | - | - |
| Бесплатность | - | - | - |
| Распознавания паспортных данных | + | + | + |
| Мобильная версия | - | - | + |

Таким образом, ни одно решения не является бесплатным и не поддерживает распознавание лиц. Поэтому для решения поставленной передо мной задачи необходимо разработать собственное решение.

# 2. АНАЛИЗ АЛГОРИТМОВ РАСПОЗНАВАНИЯ ЛИЦ

## 2.1. Постановка задачи распознавания лица на изображении

Несмотря на большое разнообразие алгоритмов, можно выделить общую структуру процесса распознавания лиц (рис. 1) [10].

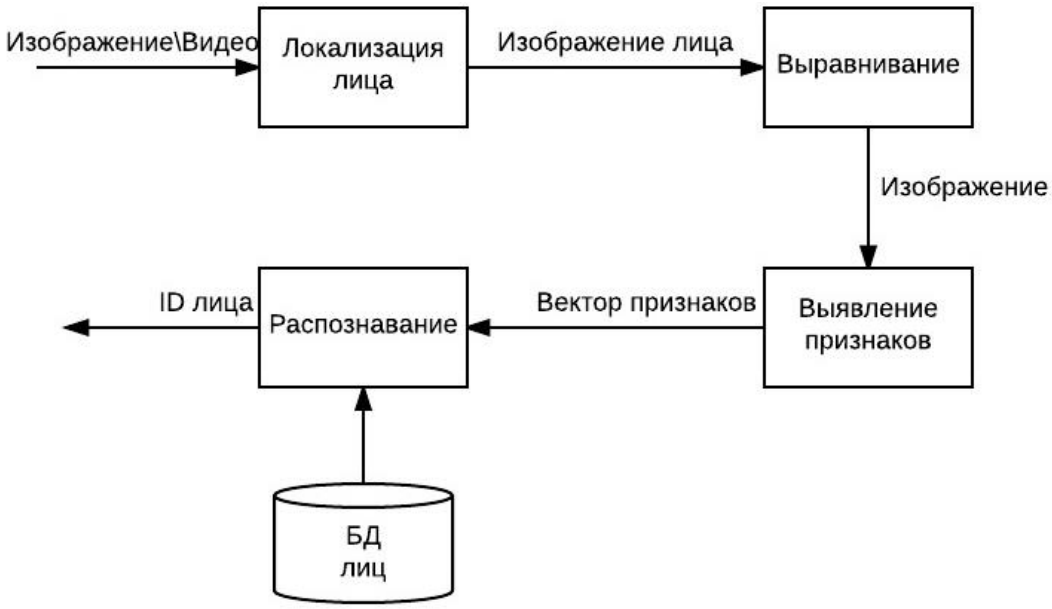


**Рис. 1.** Процесс распознавания лиц

Процесс распознавания лиц состоит из следующих этапов:

1) локализация лица на изображении;
2) выравнивание изображения лица (геометрическое и яркостное);
3) выявление признаков;
4) распознавание – сравнение вычисленных признаков с заложенными в базу данных эталонами.

Рассмотрим их подробнее.

## 2.2. Локализация лиц

Одним из самых распространенных методов по локализации лиц на изображении является метод Виолы-Джонса.

### 2.2.1. Метод Виолы-Джонса

Метод Виолы-Джонса (Viola-Jones Object Detection) – алгоритм, позволяющий обнаруживать объекты на изображениях в реальном времени [7]. Его предложили Паул Виола и Майкл Джонс в 2001 году. Хотя алгоритм

может распознавать различные классы изображений, основной задачей при его создании было обнаружение лиц.

Основным объектом, над которым производятся действия в алгоритме «Виолы-Джонса» является *сканирующее окно*. Сканирующее окно – окно, в пределах которого происходит сканирование признаками Хаара.

Алгоритм сканирования окна с признаками представлен на рис. 2.

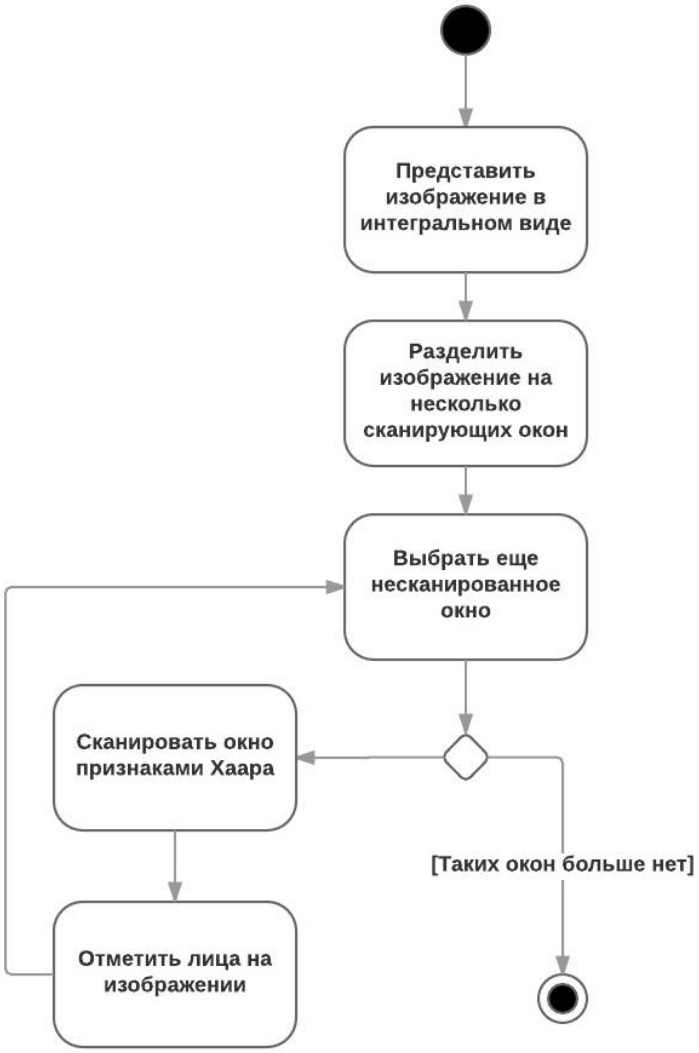


**Рис. 2.** Алгоритм Виолы-Джонса

Рассмотрим этапы алгоритма:

1) инициализация параметров (допустим, размер самого окна есть 24*24 ячейки);

2) исследуемое изображение, выбрано окно сканирования, выбраны используемые признаки;

3) далее окно сканирования начинает последовательно двигаться по изображению с шагом в 1 ячейку окна;

4) при сканировании изображения в каждом окне вычисляется приблизительно 200 000 вариантов расположения признаков, за счет изменения масштаба признаков и их положения в окне сканирования;

5) сканирование производится последовательно для различных масштабов; масштабируется не само изображение, а сканирующее окно (изменяется размер ячейки);

6) все найденные признаки попадают к классификатору, который «выносит вердикт».

В стандартном методе Виолы – Джонса используются прямоугольные признаки, изображенные на рис. 3. Они называются *признаками Хаара*. Признаки Хаара дают точечное значение перепада яркости по оси X и Y соответственно.

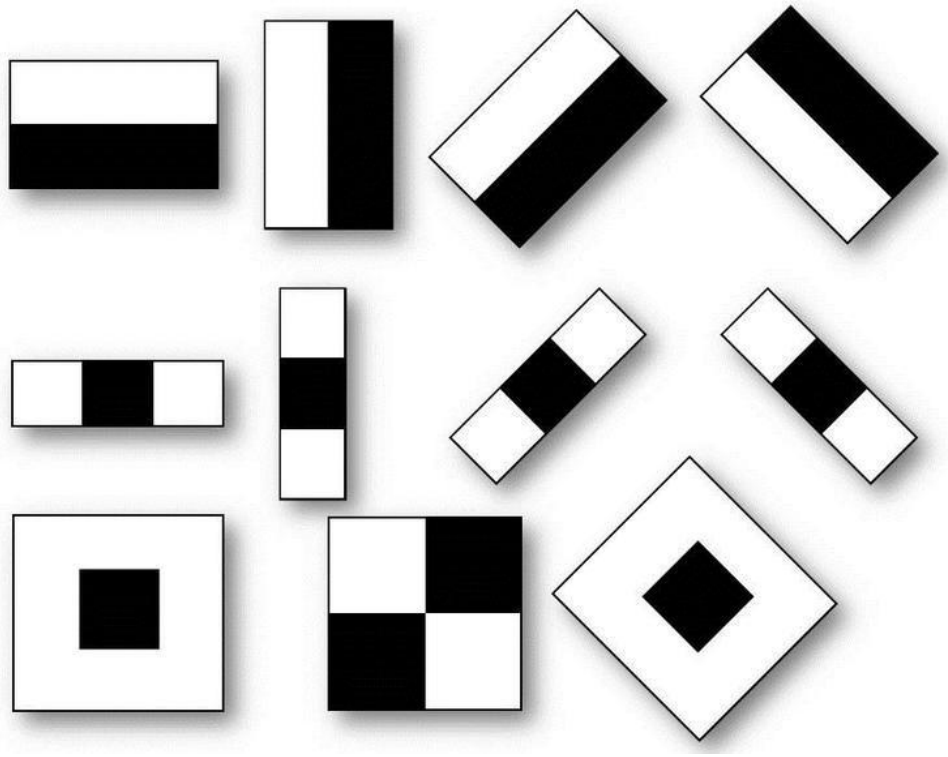

**Рис. 3.** Признаки Хаара

Вычисляемым значением такого признака будет `F = X-Y`, где X – сумма значений яркостей точек закрываемых *светлой частью признака*, а Y

– сумма значений яркостей точек закрываемых *темной частью признака*. Для их вычисления используется понятие *интегрального представления изображения*, который представляет собой матрицу с размерностью, совпадающей с размерностью исходного изображения [3]. Элементы этой матрицы рассчитываются по формуле: II(x,y) = Summ( I(i,j) ), где I(i,j) – яркость пикселя исходного изображения. Т.е., каждый элемент интегрального изображения II[x,y] содержит в себе сумму пикселей изображения в прямоугольнике от (0,0) до (x,y). Расчет интегрального изображения занимает линейное время, пропорциональное числу пикселей исходного изображении.

Метод Виолы-Джонса является одним из лучших по соотношению показателей эффективности распознавания и скорости работы. Также этот алгоритм обладает низкой вероятностью ложного обнаружения лица. Алгоритм также хорошо работает и распознает черты лица под небольшим углом, примерно до 30 градусов, а также при различных условиях освещенности [17].

### 2.3. Выравнивание изображения лица и выявление признаков

*Геометрическое выравнивание* – поворот лица, при котором лицо, представленное на изображении, достигает наиболее «естественного» положения с точки зрения человеческого восприятия (рис. 4) [7].

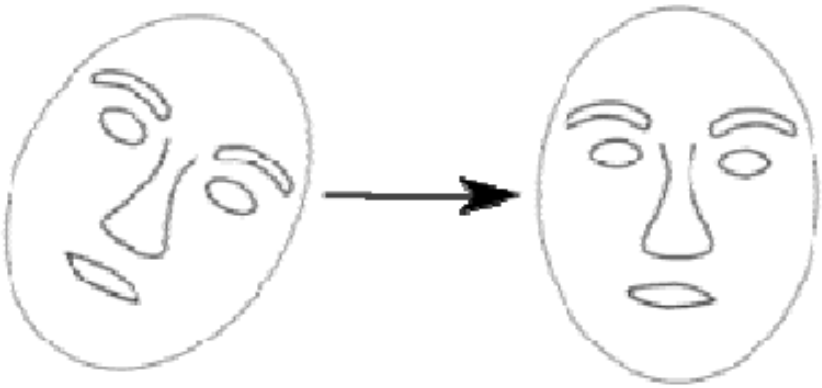

**Рис. 4.** Геометрическое выравнивание

*Яркостное выравнивание* – преобразование изображения из цветовой модели RGB в оттенки серого.

Применяется по причине отсутствия необходимости в использовании информации о цвете объектов анализа [8].

*Выявление признаков* – процесс выявления характерных признаков изображения.

### 2.4. Распознавание лиц

При анализе алгоритмов распознавания лиц были выявлены следующие способы распознавания лиц:

- метод главных компонент;
- метод гибкого сравнения на графах;
- активные модели внешнего вида;
- нейронные сети.

Одними из самых распространенных методов распознавания лиц являются метод гибкого сравнения на графах (Elastic Graph Matching) и метод главных компонент (Principal Component Analysis). Рассмотрим их подробнее.

#### 2.4.1. Метод гибкого сравнения на графах

Суть метода сводится к сопоставлению графов, описывающих изображения лиц.

В методе гибкого сравнения на графах лицо представляется в виде графа, вершины которого расположены на ключевых точках лица.

К ключевым точкам относят контуры головы, губ, носа, и их крайние точки [1].

Каждая грань помечена расстояниями между ее вершинами. В каждой такой точке вычисляются коэффициенты разложения по функциям. Набор таких коэффициентов $J=\{Jj\}$ называется *джетом*.

Джеты характеризуют локальные области изображений и служат для двух целей:

- во-первых, для нахождения точек соответствия в заданной области на двух различных изображениях;

- во-вторых – для сравнения двух соответствующих областей различных изображений.

Алгоритм метода представлен на рис. 5.

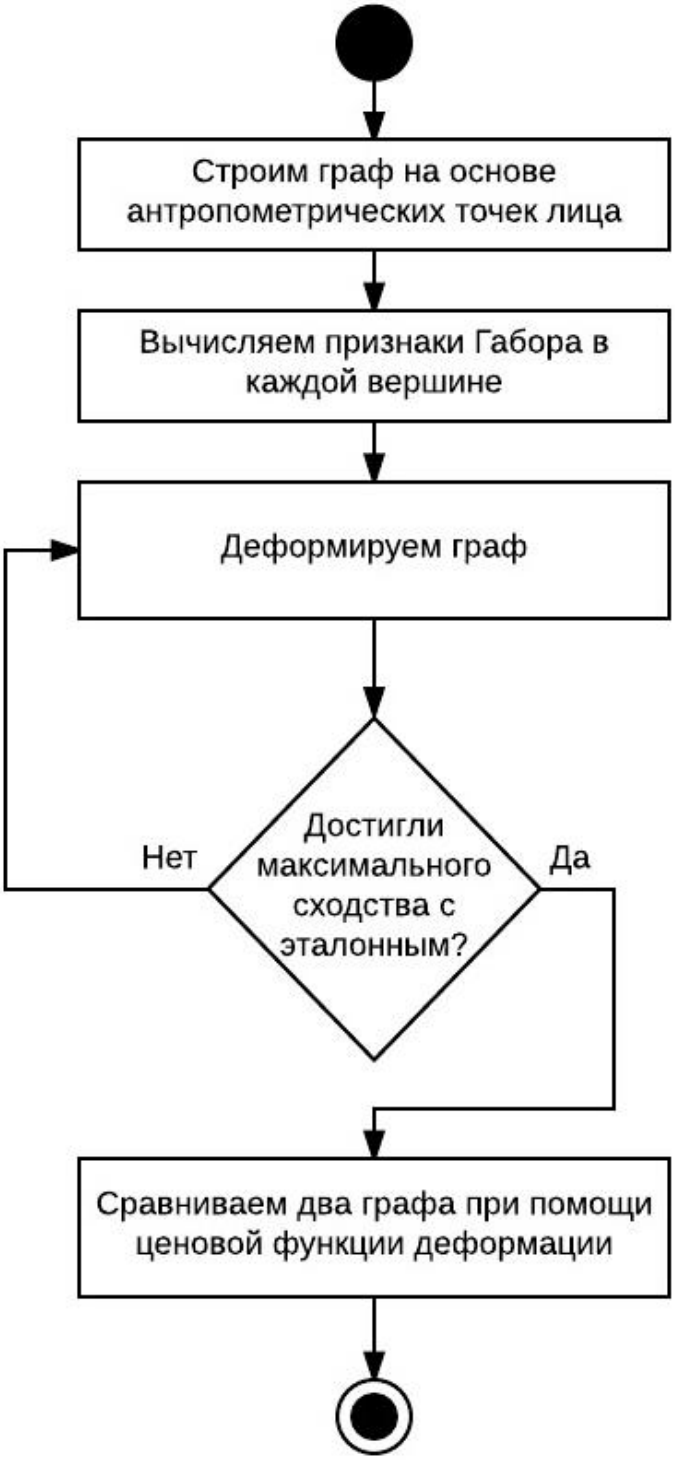


**Рис. 5.** Алгоритм метода гибкого сравнения на графах

Процесс распознавания неизвестного лица состоит в деформации графа неизвестного лица и сравнения его с эталонным при помощи ценовой функции деформации.

Деформация графа происходит путем смещения каждой из его вершин на некоторое расстояние в определенных направлениях относительно ее исходного местоположения и выбора такой ее позиции, при которой разница между значениями признаков в вершине деформируемого графа и соответствующей ей вершине эталонного графа будет минимальной.

Данная операция выполняется поочередно для всех вершин графа до тех пор, пока не будет достигнуто наименьшее суммарное различие между признаками деформируемого и эталонного графов.

Значение ценовой функции деформации при таком положении деформируемого графа и будет являться мерой различия между входным изображением лица и эталонным графом.

Пример ценовой функции деформации графа $G^I$, построенного для исходного изображения лица, с некоторым графом B представлен в формуле:

$$S_B(G^I, B) = \frac{1}{N}\sum_n \max S_ф(J_n^I, J_n^B) - \frac{\lambda}{E}\sum_e \frac{(\Delta x_e^I - \Delta x_e^B)^2}{(\Delta x_e^B)^2}.$$

Первое слагаемое характеризует подобие джетов, вычисленное с применением фазочувствительной функции, второе – топографическое соответствие, которое пропорционально квадрату разности расстояний между соответствующими вершинами сравниваемых изображений, N – количество вершин, E – количество граней, $\lambda$ – коэффициент относительной важности топографической информации.

В отдельных публикациях указывается высокая эффективность распознавания даже при наличии различных эмоциональных выражениях и изменении ракурса лица до 15 градусов. Однако разработчики систем эластичного сравнения на графах ссылаются на высокую вычислительную стоимость данного подхода. Например, для сравнения входного изображения лица с 87 эталонными тратилось приблизительно 25 секунд.

Недостатки: высокая вычислительная сложность процедуры распознавания. Низкая технологичность при запоминании новых эталонов. Линейная зависимость времени работы от размера базы данных лиц.

### 2.4.2. Метод главных компонент

Одним из наиболее известных и проработанных является *метод главных компонент* [15], основанный на преобразовании Карунена-Лоева.


**Рис. 7.** Алгоритм метода главных компонент

Рассмотрим шаги алгоритма подробнее:

1) каждое изображение из базы преобразуется в вектор;
2) из этих векторов формируется строки новой матрицы;

3) для полученной матрицы составляется *матрица ковариации*. *Матрица ковариации* - это матрица, составленная из попарных ковариаций элементов случайных векторов [13];

4) далее вычисляются *собственные значения* и *собственные вектора* матрицы. *Собственный вектор* - ненулевой вектор k, который при умножении на некоторую квадратную матрицу A превращается в самого же себя с числовым коэффициентом n, называется собственным вектором матрицы A. Число n называют *собственным значением* или собственным числом данной матрицы [5];

5) из найденных собственных векторов строится *матрица преобразования данных* A, составленная из найденных векторов, расположенных в порядке убывания собственных значений. $A = \{a1, a2, . . ak\}$;

6) для каждого из изображений строится его проекция в новое пространство путем умножения вектора $X_i$ на матрицу преобразования A: $p_i = X_i$*A. Полученные вектора $\{p_1, . . p_n\}$ далее используются для распознавания;

7) исходное изображение лица размером W×H пикселей, где W – ширина изображения, H – высота изображения, строим матрицу $Y=\{y_{i,j}\}$ соответствующей ему размерности, в которой элемент $y_{i,j}$ является значением яркости пиксела с координатами i,j. Данная матрица преобразуется в вектор I;

8) полученный вектор исходного изображения лица I умножается на матрицу преобразования A: $p = I*A$. Результатом распознавания в данном алгоритме является элемент из векторов изображений из базы данных $p_i$, для которого *эвклидово расстояние* до p минимально. *Евклидово расстояние* – расстояние между точками в Евклидовом пространстве [2].

Собственные векторы, вычисленные для всего набора изображений лиц, называются *собственными лицами* [11]. Метод главных компонент в

применении к изображениям лиц также называют методом *собственных лиц.*

Процесс распознавания заключается в сравнении главных компонент неизвестного изображения с компонентами всех остальных изображений [6]. Для этого обычно применяют какую-либо метрику (простейший случай - Евклидово расстояние).

**Сравнение алгоритмов**

Сравнительная характеристика алгоритмов, принадлежащих каждому из описанных выше подходов представлена на табл. 2 [9]:

**Табл. 2.** Сравнение алгоритмов

| | **Метод главных компонент** | **Метод гибкого сравнения на графах** |
|---|---|---|
| Устойчивость к положению лица | + | + |
| Устойчивость к изменениям освещения | - | + |
| Может использоваться для идентификации | + | + |
| Добавление нового лица без переобучения | + | + |
| Быстрое распознавание | + | - |

Таким образом, для решения поставленной передо мной задачи наиболее подходящим представляется метод главных компонент, так как он обладает более низкой вычислительной сложностью, при этом обеспечивает более быстрое распознавание. Недостаток в виде плохой устойчивости к изменению освещения не является критическим, так как в рамках поставленной задачи необходимо распознавать лицо на скане паспорта.

## 3. ПРОЕКТИРОВАНИЕ

В этом разделе представлено описание проектирования программы. Далее представлена диаграмма вариантов использования, диаграмма классов, диаграмма компонентов, диаграмма базы данных. При проектировании был использован язык UML.

### 3.1. Диаграмма вариантов использования

В ходе проектирования была разработана диаграмма вариантов использования, изображенная на рис. 8.

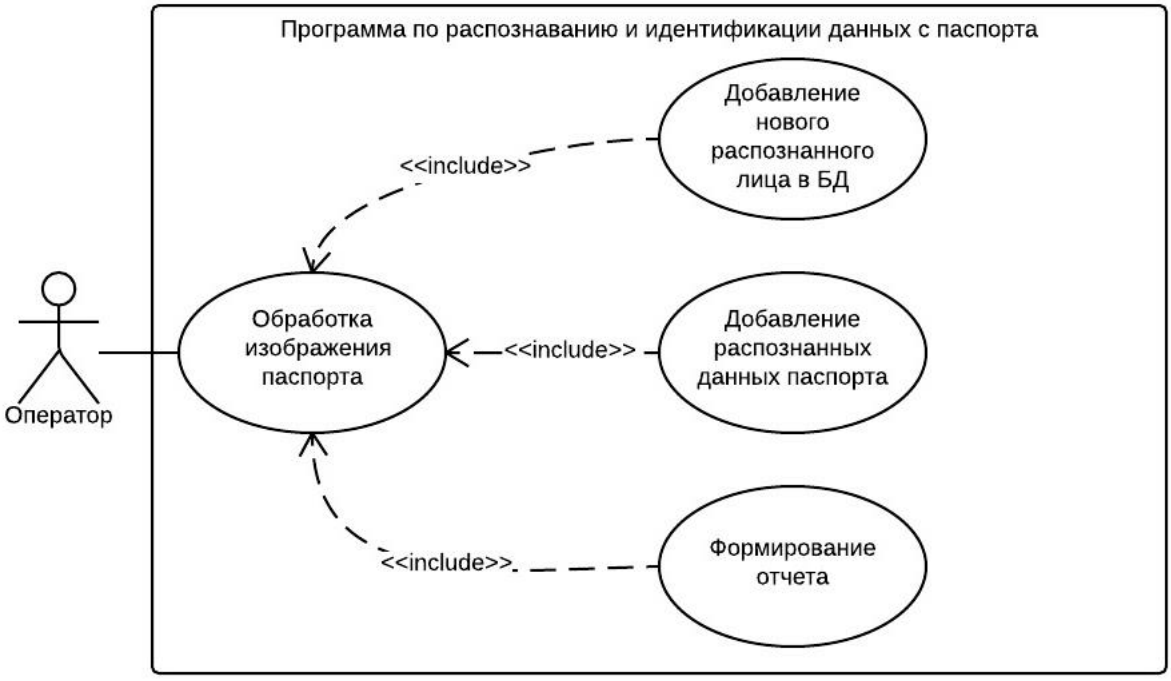


**Рис. 8.** Диаграмма вариантов использования

Актер «Оператор» – это авторизованный пользователь системы.

Вариант использования «Обработка изображения с паспорта» – включает в себя три варианта использования и представляет собой обработку изображения в программе.

Вариант использования «Добавление нового распознанного лица в базу данных» – при обнаружении неизвестного после попытки распознавания программой лица, Оператор может добавить данное изображение в систему.

Вариант использования «Добавление распознанных паспортных данных» – программа добавляет распознанные паспортные данные в базу.

Вариант использования «Формирование отчета» – программа формирует отчет об обработке изображения паспорта.

### 3.2. Диаграмма классов

Была спроектирована диаграмма классов (рис. 9).

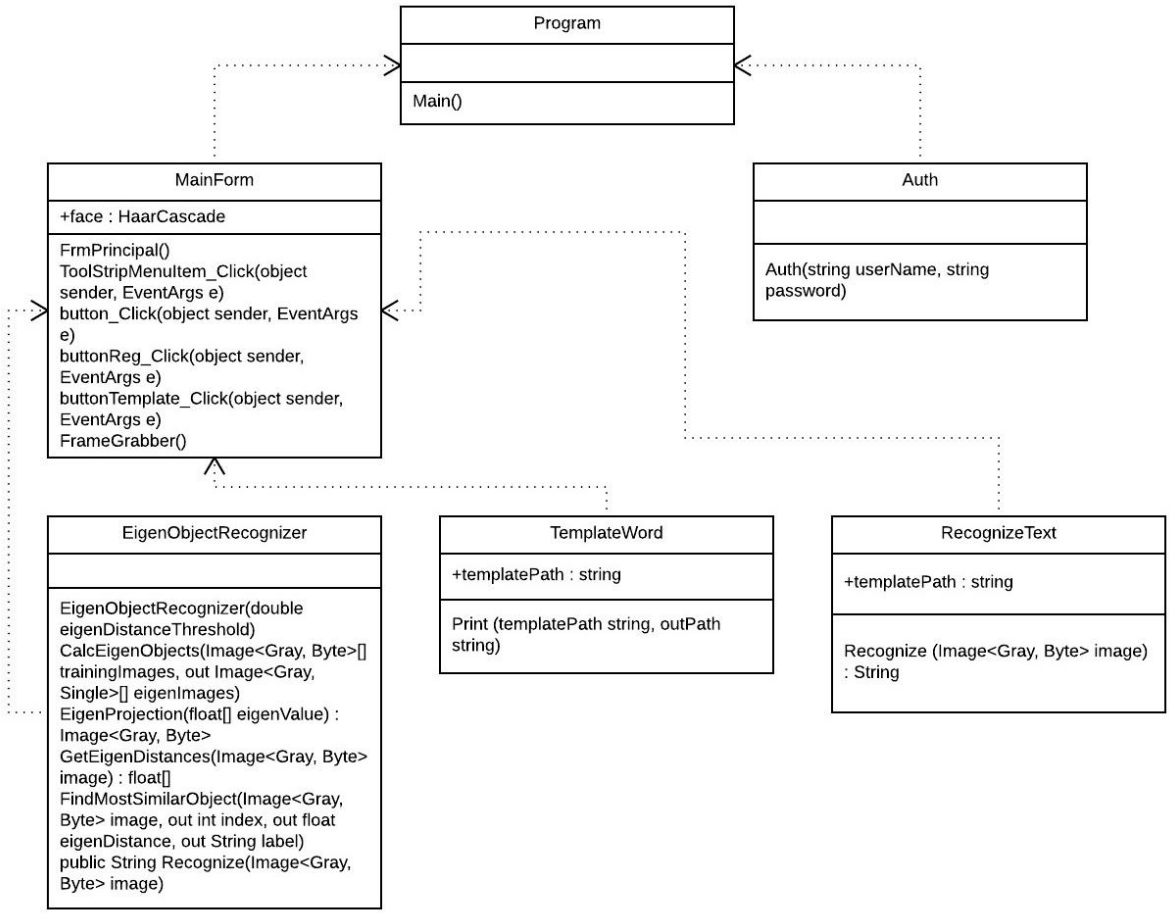


**Рис. 9.** Диаграмма классов

Класс «Program» - класс, содержащий метод Main, который начинает работу при запуске программы.

Класс «Auth» - класс, ответственный за авторизацию пользователя.

Класс «MainForm» - класс, ответственный за основное окно программы. Содержит в себе обработчики событий на действия пользователя.

Класс «EigenObjectRecognizer» - класс, ответственный за распознавания лица.

Класс «TemplateWord» - класс, ответственный за формирование шаблона.

Класс «RecognizeText» - класс, ответственный за распознавание текста.

### 3.3. Диаграмма компонентов

Была спроектирована диаграмма компонентов, представленная на рис. 10.

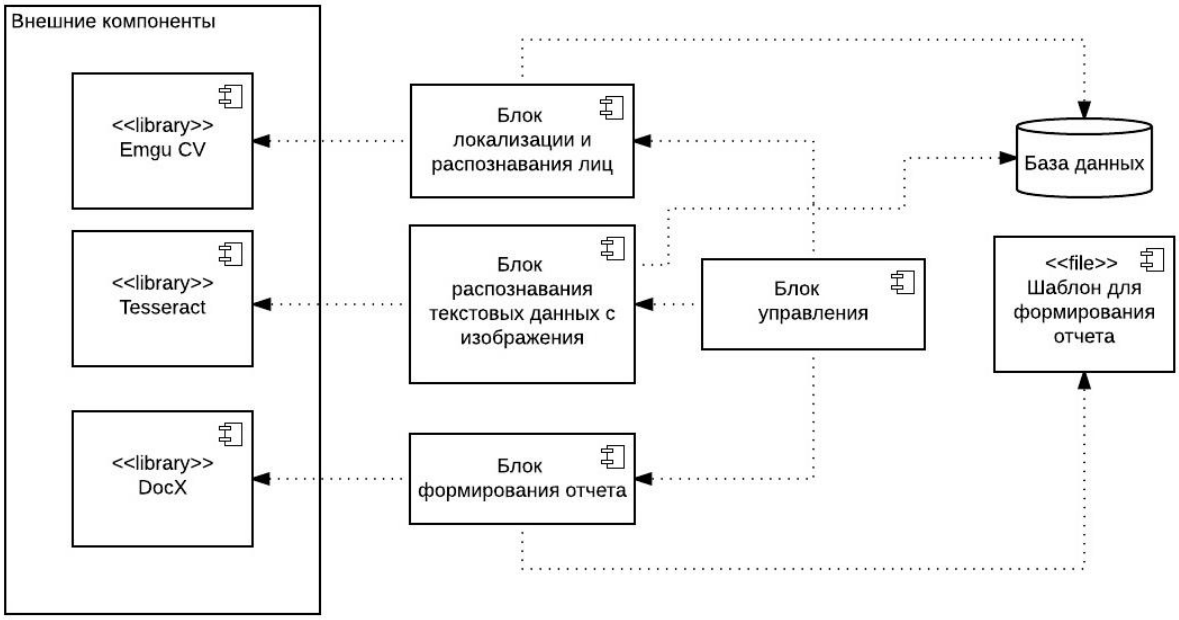


**Рис. 10.** Диаграмма компонентов

Компонент «Блок локализации и распознавания лиц» – компонент, который занимается локализацией лица на изображении и распознает данное лицо. Использует внешний компонент Emgu CV и базу данных.

Компонент «Блок распознавания текстовых данных с изображения» – компонент, который занимается распознавание текстовых данных с паспорта и их записью в базу данных. Использует внешний компонент Tesseract и базу данных.

Компонент «Блок формирования отчета» – компонент, который занимается формированием отчета для оператора. Использует внешний компонент DocX и шаблон отчета.

Компонент «Блок управления» – компонент, который занимается управлением компонентами системы, также выдачей авторизацией и выдачей прав пользователю.

### 3.4. Диаграмма базы данных

Была спроектирована диаграмма базы данных, представленная на рис. 11.

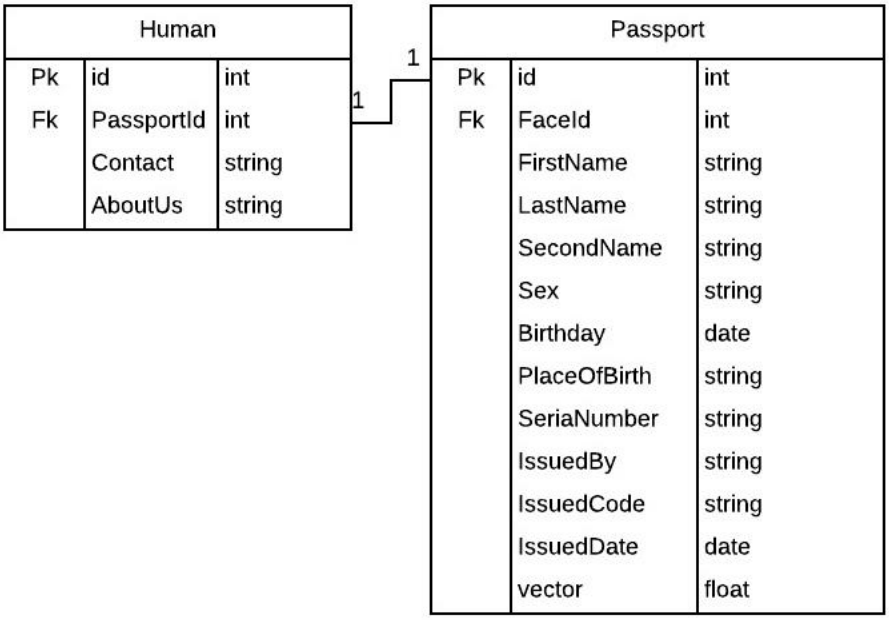


**Рис. 11.** Диаграмма базы данных

**Определения и условные обозначения**

При проектировании таблиц использовались следующие типы данных:

- INT – целое число;
- VARCHAR – строка;
- DATE – дата;
- FLOAT – дробное число.

**Описание таблиц**

База данных состоит из следующих таблиц:

- Human (табл. 3), хранит информацию о людях;
- Passport (табл. 4), хранит информацию с паспорта.

**Табл. 3.** Таблица Human

| № | Атрибут | Семантика | Ключ | Тип | Ограничения |
|---|---|---|---|---|---|
| 1 | Id | Уникальный идентификатор | * | int | Primary |
| 2 | PassportId | Идентифика-тор паспорта | #Pass-port.id | int | |
| 3 | Contact | Контакты | | Varchar | |
| 4 | About Us | Как узнали | | Varchar | |

**Табл. 4.** Таблица Passport

| № | Атрибут | Семантика | Ключ | Тип | Ограничения |
|---|---|---|---|---|---|
| 1 | Id | Идентифика-тор | * | Int | Primary |
| 2 | FirstName | Имя | | Varchar | |
| 3 | LastName | Фамилия | | Varchar | |
| 4 | SecondName | Отчества | | Varchar | |
| 5 | Sex | Пол | | Varchar | |
| 6 | Birthday | День рождения | | Date | |
| 7 | PlaceOfBirth | Место рожде-ния | | Varchar | |
| 8 | Serial-Number | Серия Номер | | Varchar | |
| 9 | IssuedBy | Выдан | | Varchar | |
| 10 | IssuedCode | Код подразде-ления | | Varchar | |
| 11 | IssuedDate | Дата выдачи | | Date | |
| 12 | Vector | Ветор | | Float | |

# 4. РЕАЛИЗАЦИЯ

## 4.1. Выбор средств разработки

В качестве основного средства разработки был выбран фреймворк .Net с использованием языка программирования C# и технологии Windows Forms. Технология Windows Forms предоставляет возможность разработки кроссплатформенного графического пользовательского интерфейса [20].

В качестве системы управления базой данных была выбрана свободная объектно-реляционная СУБД *PostgreSQL*. Данная СУБД была выбрана по требованию заказчика.

Данная СУБД существует в реализациях для множества UNIX-подобных платформ, включая AIX, различные BSD-системы, HP-UX, IRIX, Linux, macOS, Solaris/OpenSolaris, Tru64, QNX, а также для Microsoft Windows [14].

**EmguCV**

Для реализации алгоритма локализации лица на изображении используется библиотека EmguCV.

Emgu CV является кросс-платформенным .Net дополнением для библиотеки OpenCV для обработки изображений. Разработано для работы с .NET совместимыми языками, такими как C #, VB, VC ++, IronPython и т.д., может быть использовано в Visual Studio, Xamarin, работает с Windows, Linux, Mac OS X, IOS, Android и Windows Phone [15].

OpenCV (англ. Open Source Computer Vision Library, библиотека компьютерного зрения с открытым исходным кодом) — библиотека алгоритмов компьютерного зрения, обработки изображений и численных алгоритмов общего назначения с открытым кодом. Реализована на C/C++, также разрабатывается для Python, Java, Ruby, Matlab, Lua и других языков [13].

Основные модули:

- opencv_core – основная функциональность. Включает в себя базовые структуры, вычисления (математические функции, генераторы случайных чисел) и линейную алгебру, DFT, DCT, ввод/вывод для XML и YAML;

- opencv_imgproc – обработка изображений (фильтрация, геометрические преобразования, преобразование цветовых пространств и т. д.);
- opencv_highgui – простой UI, ввод/вывод изображений и видео;
- opencv_ml – модели машинного обучения (SVM, деревья решений, обучение со стимулированием и т. д.);
- opencv_features2d – распознавание и описание плоских примитивов;
- opencv_video – анализ движения и отслеживание объектов (оптический поток, шаблоны движения, устранение фона);
- opencv_objdetect – обнаружение объектов на изображении (нахождение лиц с помощью алгоритма Виолы-Джонса (англ.), распознавание людей HOG и т. д.);
- opencv_calib3d – калибровка камеры, поиск стерео-соответствия и элементы обработки трехмерных данных;
- opencv_flann – библиотека быстрого поиска ближайших соседей (FLANN 1.5) и обертки OpenCV;
- opencv_contrib – сопутствующий код, еще не готовый для применения;
- opencv_legacy – устаревший код, сохраненный ради обратной совместимости;
- opencv_gpu – ускорение некоторых функций OpenCV за счет CUDA, создан при поддержке NVidia.

**Tesseract**

Для реализации распознавания текстовых данных используется библиотека Tesseract.

Движок Tesseract OCR был одним из 3-х лучших движков представленных в 1995 году на UNLV Accuracy test. В период между 1995 годом и 2006 годом он был немного доработан, но, вероятно, это один из наиболее точных OCR (Optical Character Recognition) движков, который доступен с

открытым исходным кодом. Код, который доступен будет читать бинарные, серые или цветное изображение и выводить текст [4].

**DocX**

Для реализации формирования отчета используется библиотека DocX.

DocX - это библиотека .NET, которая позволяет разработчикам манипулировать файлами Word 2007/2010/2013. DocX не требует установки Microsoft Word или Office [1].

### 4.2. Реализация локализации лиц

Для реализации локализации лиц использовалась библиотека EmguCV. В методе Виолы-Джонса основную задачу локализации лица выполняет каскадный классификатор, который прошел обучение на тысячах человеческих лиц. Данные обучения храняться в xml файлах поставляемых вместе с EmguCV. Код представлен в приложении 1.

### 4.3. Реализация алгоритма распознавания лиц

В соответствии с выбранным методом главных компонент был реализован класс EigenObjectRecognizer для распознавания лица на изображении. В данном объекте были реализованы следующие методы:

- EigenObjectRecognizer – конструктор, который создает объект с опреденным параметром максимального расстояния между векторами для распознавания;
- CalcEigenObjects – метод, вычисляющий собственные лица для изображений;
- EigenProjection – метод, который учитывая собственные лица восстанавливает исходное изображение;
- GetEigenDistances – метод, который получает эвклидово расстояние между исходным изображением и любым другим из базы данных;
- FindMostSimilarObject – метод, который находит самый ближайший вектор к исходному;
- Recognize – метод, который распознает лицо.

Исходный код данных методов представлен в приложении 2.

Основное окно программы представлено на рис. 12.

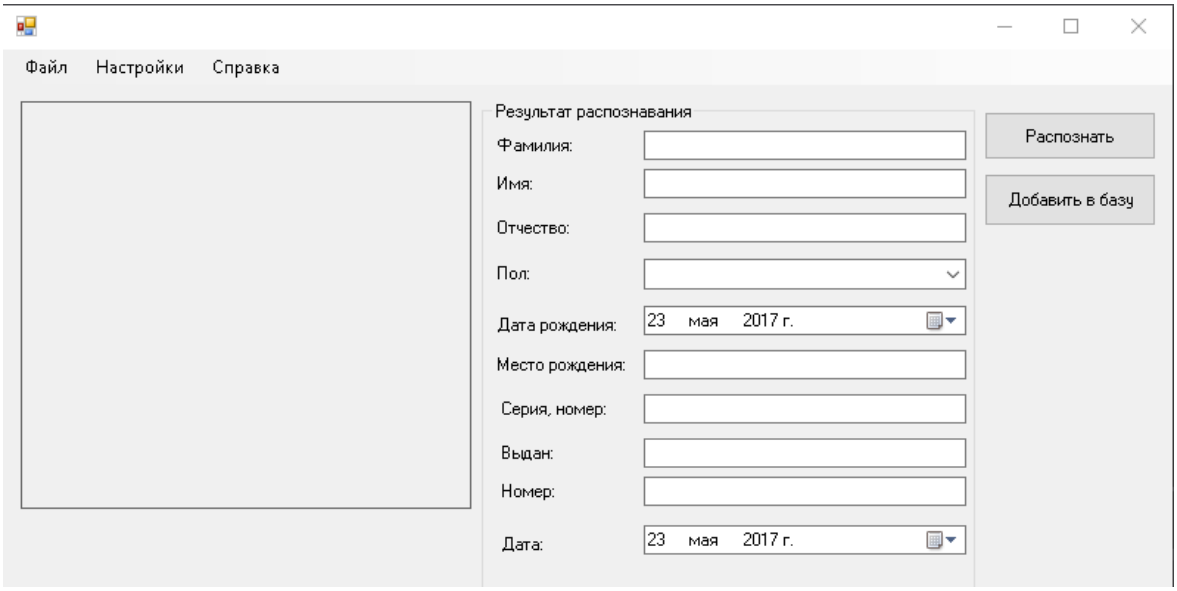


**Рис. 12.** Основное окно программы

### 4.4. Поиск оптимального расстояния для метода главных компонент

В методе главных компонент необходимо задавать расстояние для векторов, в рамках которого он будет искать самый ближайший вектор к исходному. Важно провести эксперименты и выяснить наиболее оптимальное значение этого расстояния, так как если задать слишком большое расстояние, то алгоритм будет всегда считать, что он нашел искомое лицо. Для поиска оптимального расстояния был проведен ряд экспериментов.

Одна часть экспериментов направлена на поиск искомого лицо, которое есть в тестовой базе лиц: «Эксперимент с известным набором лиц» (рис. 13); а вторая часть – на поиск искомого лица, которое отсутствует в тестовой базе лиц: «Эксперимент с неизвестным набором лиц» (рис. 14). Каждый эксперимент проводился на 10 изображениях искомых лиц. Тестовая база содержала информацию по 50 изображениям лиц разных людей.

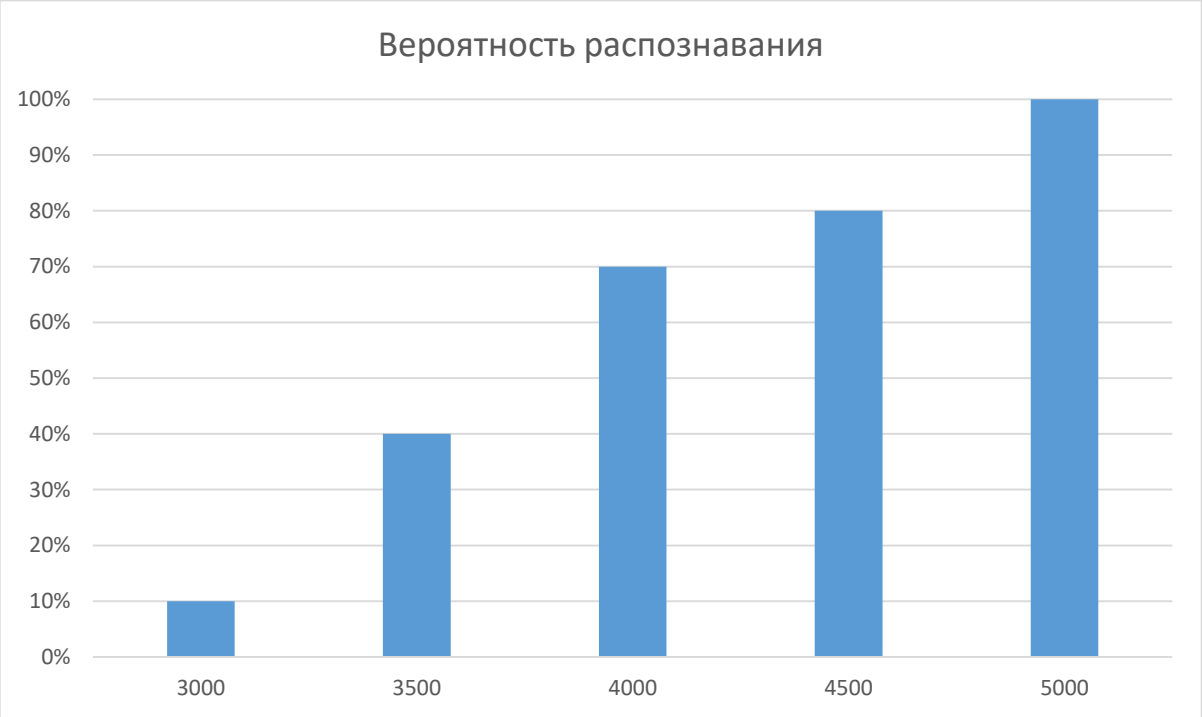


**Рис. 13.** Эксперимент с известным набором лиц


**Рис. 14.** Эксперимент с неизвестным набором лиц

Для поиска более точного значения был выбран диапазон значений с 4000 до 4500. Были проведены аналогичные эксперименты для данного диапазона на такой же базе данных (рис. 15 - 16).

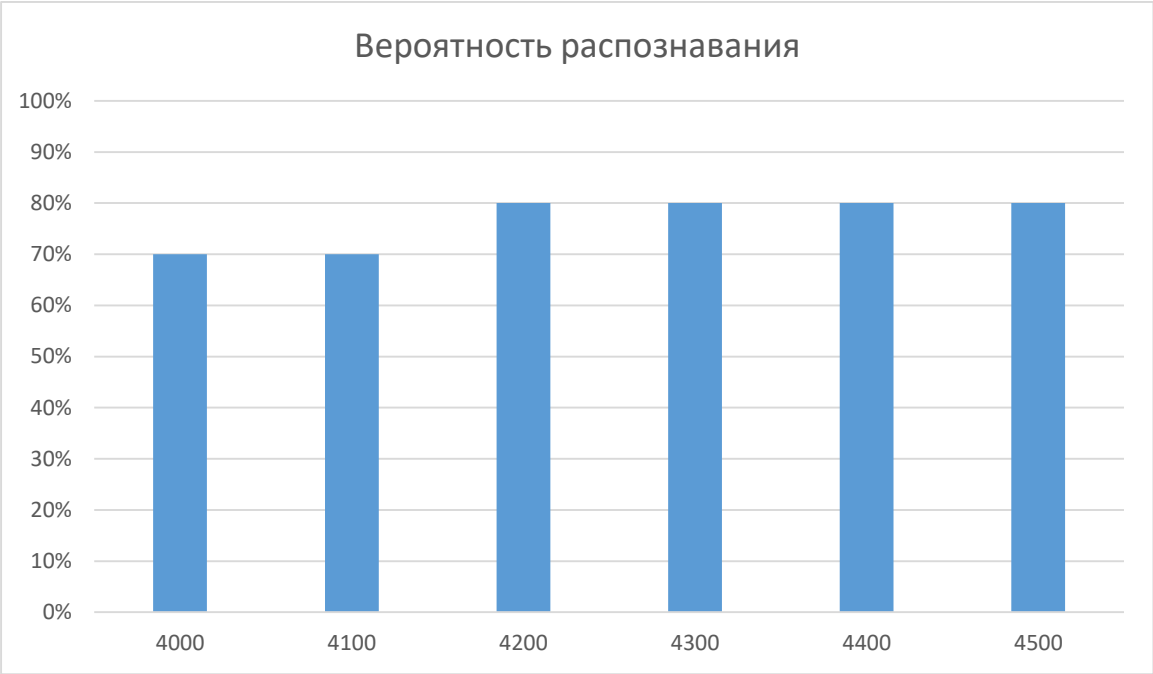


**Рис. 15.** Эксперимент с известным набором лиц


**Рис. 16.** Эксперимент с неизвестным набором лиц

Согласно сделанным экспериментам оптимальное значение выбрано 4200, так как в таком случае получилась наиболее высокая распознаваемость и низкая вероятность ошибочного распознавания.

### 4.5. Распознавание текстовых данных

Для распознавания текстовых данных на паспорте, необходимо для начала найти зоны с целевой информацией. Благодаря тому, что данная программа предназначена для работы с отсканированными изображениями паспортом нет большой необходимости в увеличении качества изображения. Для каждой зоны с целевой информацией выполняется распознавание текстовых данных с помощью Tesseract. Пример работы с Tesseract приведены в приложении 3.

В процессе реализации зоны для распознавания определялись статически, но данный способ неэффективен. Динамический способ определения зон для распознавания не был реализован.

### 4.6. Формирование отчета

Для формирования любого необходимого отчета для организации программа использует шаблонный документ и с помощью библиотеки DocX заменяет в нем необходимые поля и формирует новый документ. Пример создания документа представлен в приложении 4.

## 5. ТЕСТИРОВАНИЕ

В тестировании были использованы следующие методы:

- функциональное тестирование;
- тестирование безопасности.

**Функциональное тестирование**

*Функциональное тестирование* – это тестирование ПО в целях проверки реализуемости функциональных требований, то есть способности ПО в определенных условиях решать задачи нужные пользователям [12].

Тест № 1. Распознать паспортные данные известного лица.

Входные данные: пользователь открывает изображение скана паспорта в программе. Нажимает на кнопку «Распознать»

Ожидаемый результат: лицо распознается.

Полученный результат: лицо распозналось.

Тест успешно пройден.

Тест № 2. Неизвестное лицо не распознается.

Входные данные: пользователь открывает изображение скана паспорта в программе. Нажимает на кнопку «Распознать»

Ожидаемый результат: лицо не распознается.

Полученный результат: лицо не распозналось.

Тест успешно пройден.

Тест № 3. Сформировать отчет для пользователя.

Входные данные: пользователь нажимает на кнопку «Сформировать отчет» и вводит необходимые ему данные.

Ожидаемый результат: будет создан файл отчет.

Полученный результат: создан файл отчета.

Тест успешно пройден.

**Тестирование безопасности**

*Тестирование безопасности* – это оценка уязвимостей программного обеспечения [16].

Тест № 1. Неверные данные при авторизации.

Входные данные: пользователь вводит неверные учетные данные при входе.

Ожидаемый результат: система не позволит пользователю авторизоваться. Будет выдано сообщение «Такого пользователя не существует, пожалуйста, введите логин и пароль еще раз».

Результат: полученный результат совпал с ожидаемым.

Тест успешно пройден.

Тест № 2. Доступность приложения неавторизованным пользователям.

Входные данные: пользователь пытается получить доступ к приложению без авторизации.

Ожидаемый результат: система не позволит получить доступ.

Результат: полученный результат совпал с ожидаемым.

Тест успешно пройден.

Результаты тестов представлены в приложении 5.

## ЗАКЛЮЧЕНИЕ

В ходе выполнения данной работы были получены следующие основные результаты:

1) проведен анализ предметной области;
2) выполнено проектирование системы;
3) выполнена реализация компонента распознавания лиц и формирования шаблона;
4) проведено тестирование;
5) проведено внедрение.

Разработанная программа для распознавания лиц и паспортных данных была внедрена в ЧРОО РЭК «Сделаем».

**ЛИТЕРАТУРА**

1. DocX [Электронный ресурс] URL: https://docx.codeplex.com/ (дата обращения: 20.02.2017).

2. Edwards G.J., Cootes T.F., Taylor C.J. ECCV '98 Proceedings of the 5th European Conference on Computer Vision, 1998. – 595 p.

3. OpenCV интегральное изображение. [Электронный ресурс] URL: http://robocraft.ru/blog/computervision/536.html (дата обращения: 14.02.2017).

4. Tesseract OCR. [Электронный ресурс] URL: https://github.com/tesseract-ocr/ (дата обращения: 10.03.2017).

5. Wiskott L., Face Recognition by Elastic Bunch Graph Matching. / L. Widkott, J.M. Fellous, N. Kruger, C. Malsburg. // I9 Transactions on Pattern Analysis and Machine Intelligence, 1997. – Vol. 19. – P. 775-779.

6. Yang M.H., Kriegman D.J., Ahuja N. Detecting faces in images. // IEEE Transactions on Pattern Analysis and Machine Intelligence, 2002. – Vol. 24. – P. 34-58.

7. Антипова А.Ю., Губарев В.В. Применение метода моментов в задаче геометрического выравнивания лиц на изображениях. // Известия ЮФУ. Технические науки, 2009. – № 8. – С. 167-177.

8. Волынец М. Ю. Комбинированный метод обнаружения и распознавания лиц в реальном режиме. // Молодежный научно-технический вестник, 2015. – № 4. – С. 120-135.

9. Гонсалес Р., Вудс Р. Цифровая обработка изображений. – М.: Техносфера, 2005. – 1072 с.

10. Кухарев Г.А. Биометрические системы: Методы и средства идентификации личности человека. – СПб.: Политехника, 2001. – 240 с.

11. Лебеденко Ю.И. Биометрические системы безопасности. – М.: Directmedia, 2013. – 159 с.

12. Майерс Г., Баджет Т. Искусство тестирования программ. – М.:Вильямс, 2012. – 272 с.

13. Минский М., Пейперт С. Персептроны. – М.: Мир, 1971. – 261 с.

14. Ригс С., Кросинг Х. Администрирование PostgreSQL 9. – М.: ДМК Пресс, 2012. – 458 с.

15. Сойфер В.А. Методы компьютерной обработки изображений. – М.: Физматлит, 2003. – 459 с.

16. Танненбаум Э. Современные операционные системы. – СПб.: Питер, 2010. – 1120 с.

17. Татаренков Д.А. Анализ методов обнаружения лиц на изображении. // Молодой ученый, 2015. – № 4. – С. 270-276.

18. Цикон Х., Фрезе Р., Саймон Б. Операторы Шредингера с приложениями к квантовой механике и глобальной геометрии. – М. Мир, 1990. – 408 с.

19. ЧРОО РЭК «Сделаем». [Электронный ресурс] URL: http://www.sdelaem74.ru/ (дата обращения: 13.03.2017).

20. Троелсен Э., Джепикс Ф. Язык программирования C# 6.0 и платформа .NET 4.6. – М.: Вильямс, 2016. – 1440 с.

# ПРИЛОЖЕНИЯ

## Приложение 1

## Реализация локализации лиц

```
currentFrame = new Image<Bgr, Byte>(imageBoxFrameGrabber.Image.Bitmap);

gray = currentFrame.Convert<Gray, Byte>();

MCvAvgComp[][] facesDetected = gray.DetectHaarCascade(
          face,
          1.2,
          10,
          Emgu.CV.CvEnum.HAAR_DETECTION_TYPE.DO_CANNY_PRUNING,
          new Size(20, 20));

            foreach (MCvAvgComp f in facesDetected[0])
            {
                t = t + 1;

                result = currentFrame.Copy(f.rect).Convert<Gray,
byte>().Resize(100, 100, Emgu.CV.CvEnum.INTER.CV_INTER_CUBIC);

currentFrame.Draw(f.rect, new Bgr(Color.Red), 2);
```

## Приложение 2

## Реализация методов EigenObjectRecognizer

```
Public EigenObjectRecognizer(Image<Gray, Byte>[] images, String[] labels,
double eigenDistanceThreshold, ref MCvTermCriteria termCrit)
      {

         Debug.Assert(images.Length == labels.Length, "The number of images
should equals the number of labels");
         Debug.Assert(eigenDistanceThreshold >= 0.0, "Eigen-distance
threshold should always >= 0.0");

         CalcEigenObjects(images, ref termCrit, out _eigenImages, out
_avgImage);


         _eigenValues = Array.ConvertAll<Image<Gray, Byte>, Ma-
trix<float>>(images,
             delegate(Image<Gray, Byte> img)

             {

                return new Matrix<float>(EigenDecomposite(img, _eigen-
Images, _avgImage));
             });


         _labels = labels;

         _eigenDistanceThreshold = eigenDistanceThreshold;
      }


public static void CalcEigenObjects(Image<Gray, Byte>[] trainingImages, ref
MCvTermCriteria termCrit, out Image<Gray, Single>[] eigenImages, out Im-
age<Gray, Single> avg)
      {
         int width = trainingImages[0].Width;
         int height = trainingImages[0].Height;


         IntPtr[] inObjs = Array.ConvertAll<Image<Gray, Byte>,
IntPtr>(trainingImages, delegate(Image<Gray, Byte> img) { return img.Ptr;
});


         if (termCrit.max_iter <= 0 || termCrit.max_iter > train-
ingImages.Length)
            termCrit.max_iter = trainingImages.Length;


         int maxEigenObjs = termCrit.max_iter;


         #region initialize eigen images
         eigenImages = new Image<Gray, float>[maxEigenObjs];

         for (int i = 0; i < eigenImages.Length; i++)
            eigenImages[i] = new Image<Gray, float>(width, height);
         IntPtr[] eigObjs = Array.ConvertAll<Image<Gray, Single>,
IntPtr>(eigenImages, delegate(Image<Gray, Single> img) { return img.Ptr;
});
```

```
            #endregion

            avg = new Image<Gray, Single>(width, height);


            CvInvoke.cvCalcEigenObjects(
                inObjs,
                ref termCrit,
                eigObjs,
                null,
                avg.Ptr);

        }
public Image<Gray, Byte> EigenProjection(float[] eigenValue)

        {

            Image<Gray, Byte> res = new Image<Gray, byte>(_avgImage.Width,
_avgImage.Height);

            CvInvoke.cvEigenProjection(
                Array.ConvertAll<Image<Gray, Single>, IntPtr>(_eigenImages,
delegate(Image<Gray, Single> img)
{
 return img.Ptr;
}),

                eigenValue,
                _avgImage.Ptr,
                res.Ptr);
            return res;
        }
public float[] GetEigenDistances(Image<Gray, Byte> image)

        {

            using (Matrix<float> eigenValue = new Matrix<float>(EigenDecompo-
site(image, _eigenImages, _avgImage)))

               return Array.ConvertAll<Matrix<float>, float>(_eigenValues,
                   delegate(Matrix<float> eigenValueI)
                   {

                      return (float)CvInvoke.cvNorm(eigenValue.Ptr, eigen-
ValueI.Ptr, Emgu.CV.CvEnum.NORM_TYPE.CV_L2, IntPtr.Zero);

                   });

        }


public static float[] EigenDecomposite(Image<Gray, Byte> src, Image<Gray,
Single>[] eigenImages, Image<Gray, Single> avg)
        {

            return CvInvoke.cvEigenDecomposite(
                src.Ptr,
                Array.ConvertAll<Image<Gray, Single>, IntPtr>(eigenImages,
delegate(Image<Gray, Single> img)
{
```

```
return img.Ptr;
}),

             avg.Ptr);

      }


public void FindMostSimilarObject(Image<Gray, Byte> image, out int index,

out float eigenDistance, out String label)
      {
         float[] dist = GetEigenDistances(image);

         index = 0;

         eigenDistance = dist[0];

         for (int i = 1; i < dist.Length; i++)
         {

            if (dist[i] < eigenDistance)
            {


               index = i;
               eigenDistance = dist[i];


            }
         }
         label = Labels[index];
      }



public String Recognize(Image<Gray, Byte> image)
      {
         int index;

         float eigenDistance;

         String label;

         FindMostSimilarObject(image, out index, out eigenDistance, out la-
bel);

         return (_eigenDistanceThreshold <= 0 || eigenDistance < _eigenDis-
tanceThreshold )  ? _labels[index] : String.Empty;
```

## Приложение 3

## Пример работы с Tesseract OCR

```
String path = @”C:\pic\mytext.jpg”;

Bitmap image = new Bitmap(path);

Tesseract ocr = new Tesseract();

ocrSetVariable();

ocr.Init();

var result = ocr.DoOCR(image, Rectange.Empty)
```

## Приложение 4

## Пример создания отчета

```
public static void CreateDocx()
        {
            string workingDir = Path.GetFullPath(Directory.GetCurrentDirec-
tory() + @"");
            string docxFilePath = Path.Combine(workingDir, "Result.docx");

            DocumentCore docx = new DocumentCore();

            Section section = new Section(docx);
            docx.Sections.Add(section);

            section.PageSetup.PaperType = PaperType.A4;

            section.Blocks.Add(new Paragraph(docx, "Новый отчет"));
            Paragraph par1 = section.Blocks[0] as Paragraph;
            par1.ParagraphFormat.Alignment = HorizontalAlignment.Center;

            par1.Inlines.Add(new  SpecialCharacter(docx,  SpecialCharacter-
Type.LineBreak));
            par1.Inlines.Add(new Run(docx, "Имя: Фамилия: Отчество:"));

            CharacterFormat cf = new CharacterFormat() { FontName = "Ver-
dana", Size = 16, FontColor = Color.Orange };
            foreach (Inline inline in par1.Inlines)
                if (inline is Run)
                    (inline as Run).CharacterFormat = cf.Clone();

            SpecialCharacter lBr = new SpecialCharacter(docx, SpecialCharac-
terType.LineBreak);
            docx.Content.End.Insert(lBr.Content);
            docx.Content.End.Insert("Ваша подпись", new CharacterFormat() {
Size = 20, UnderlineStyle = UnderlineType.Single });

            docx.Save(docxFilePath);

            System.Diagnostics.Process.Start(docxFilePath);
        }
```

## Приложение 5

## Тестирование

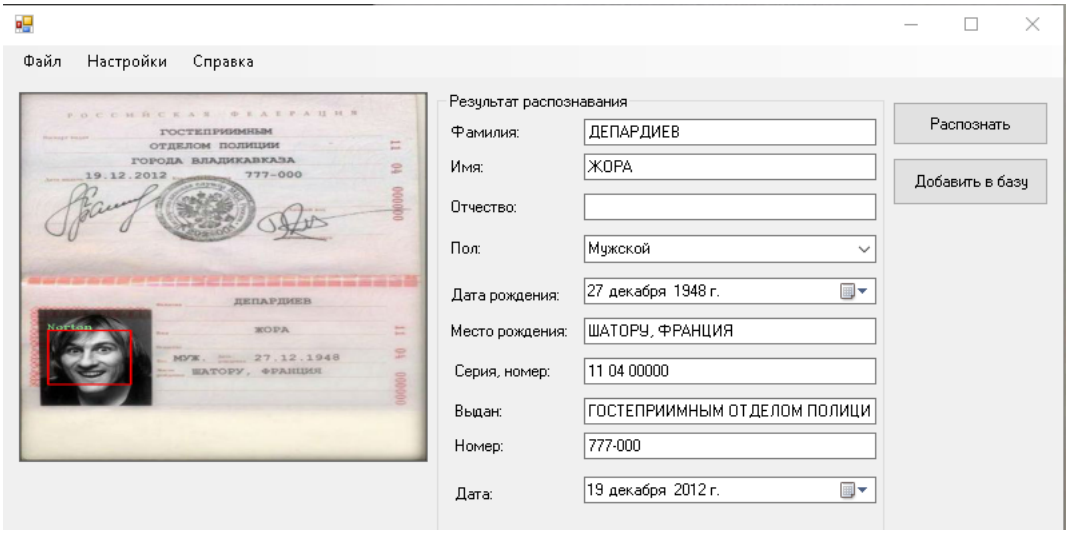


**Рис. 1.** Распознанный паспорт

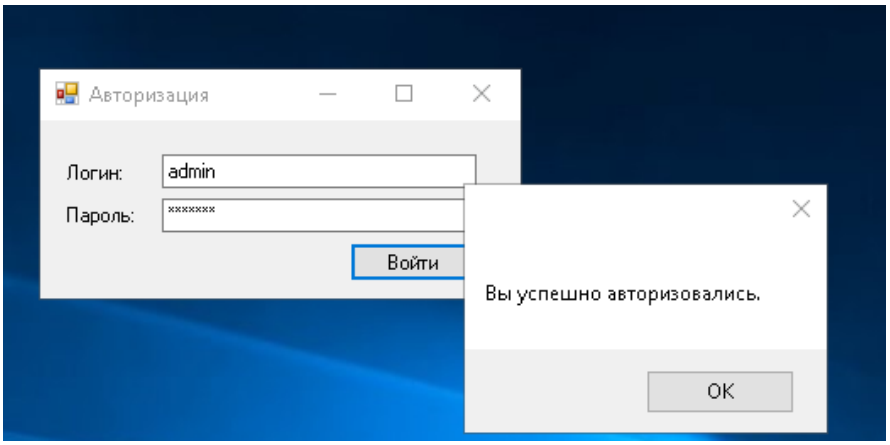


**Рис. 2.** Успешная авторизация

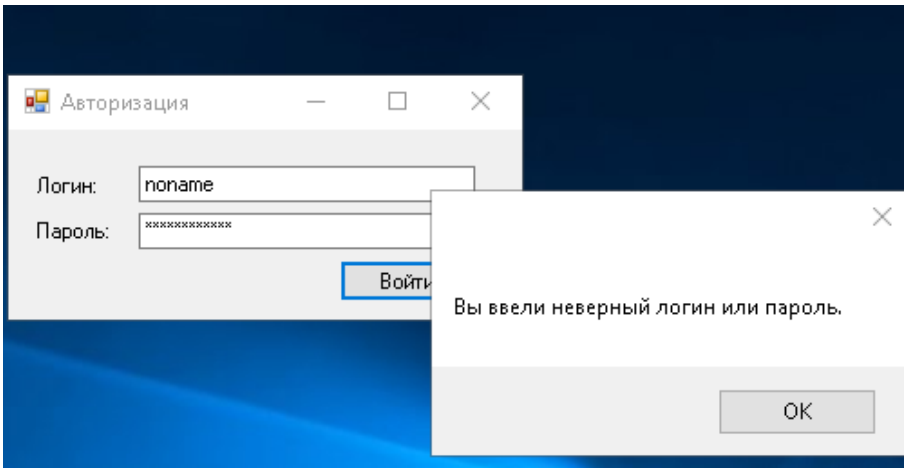


**Рис. 3.** Неверный логин или пароль при авторизации