Оглавление

Введение.....4

1. Техническое задание.....5
   1.1. Назначение разработки и область применения.....5
   1.2. Технические требования.....5
2. Анализ технического задания 6
   2.1. Выбор операционной системы.....6
   2.2. Выбор языка программирования 8
   2.3. Выбор среды разработки 9
   2.4. Обзор существующих алгоритмов идентификации человека по фотографии.....10
   2.5. Выбор подхода к решению задачи идентификации человека по фотографии.....12
3. Разработка структуры системы идентификации человека по фотографии.....15
   3.1. Разработка общей структуры системы.....15
   3.2. Разработка алгоритма идентификации.....18
   3.3. Разработка алгоритма принятий решения.....19
4. Разработка программных средств 20
   4.1. Разработка интерфейса пользователя системы.....20
   4.2. Программная реализация модулей системы.....25
5. Тестирование системы.....32
   5.1. Описание набора данных.....32
   5.2. Описание методики тестирования 35
   5.3. Результат вычислительного эксперимента.....39

Заключение.....43

Список литературы.....44

Введение

В современном мире одной из самых актуальных проблем является методы защиты информации. Существуют различные способы защитить данные, в основном эти способы делятся на два вида: первый это сокрытие информации от посторонних, их шифрование, второй же ограничение доступа при помощи различных систем безопасности. Таких систем существует огромное множество. Эти системы часто используются для повышения безопасности и автоматизации охранных систем в различных компаниях. Основная их часть заключается в идентификации людей, которым разрешен доступ к сокрытой информации. В современном мире разработаны несколько действенных методов идентификации, одним из которых является идентификация по фотографии. Данный метод отличается от остальных простотой программной реализации.

1. Техническое задание

1.1. Назначение разработки и область применения

Разрабатываемая система предназначена для идентификации человека на изображении. Изображение человека загружается в программу для определения личности, после чего по заданному алгоритму осуществляется идентификация.

Области применения:

1) Контроль доступа к безопасным компьютерным сетям и оборудованию;

2) Верификация пользователя при проведении финансовых транзакций;

3) Наблюдение в публичных местах для предотвращения террористических актов;

Данная система предназначена для использования на портативных и стационарных компьютерах.

1.2. Технические требования

Опишем требования, предъявляемые разрабатываемой системой к ЭВМ:

1) Операционная система Microsoft Windows 7 и выше;

2) Требования к аппаратному обеспечению определяются операционной системой;

3) Мышь или тачпад, дисплей, клавиатура.

Рассмотрим, каким функционалом должна обладать разрабатываемая система идентификации лица человека по фотографии:

1) Система должна предоставлять возможность пользователю выбрать изображения для идентификации и загрузить их в систему для обработки;

2) Реализовать идентификацию на основе классификации векторов

3) Вывести на экран результат идентификации;

2. Анализ технического задания

2.1. Выбор операционной системы

Важным этапом является выбор операционной системы. В дальнейшем это решение повлияет как на удобство разработки самого программного комплекса, выбора перечня необходимых для разработки инструментов, так и на скорость, стабильность его работы и комфорт работы конечного пользователя с программой. Рассмотрим самые распространённые операционные системы, а именно Linux, Mac OS и Windows:

1) Linux. Общее название для всех Unix-подобных систем основанных на ядре Linux Kernel. Впервые была опубликована в 1991 году и являлась на тот момент первой операционной системой с полностью открытым исходным кодом. Поскольку Linux Kernel лишь основа операционной системы существует большое количество дистрибутивов на его основа, отличающиеся как по надежности и безопасности, так и по входящим в их состав компонентам. Распространяются они, как правило, бесплатно и большинство из них имеет обширные сообщества, в которых можно найти ответы на все возникающие в процессе работы вопросы. Благодаря этому, а также гибкости настроек самой системы популярность Linux со временем только растет.

2) Mac OS. Операционная система на основе Unix разработанная компанией Apple, впервые вышла на рынок в 1984 году в составе компьютера Macintosh под названием System Software и позднее была переименована в Mac OS. В отличие от Linux, несмотря на общую основу, не является системой с открытым исходным кодом.

3) Windows. Наиболее распространённая на сегодняшний день ОС, производится и поддерживается компанией Microsoft с 1985 года. Является первой операционной системой, ориентированной на графический интерфейс и считается наиболее привычной и дружелюбной к пользователю.

Основные различия, преимущества и недостатки операционных систем Windows, Linux:

1) Защита. Почти полное отсутствие на платформе Linux, по крайней мере, на сегодняшний день, вредоносных программ, что позволяет избежать расходов на Антивирусное ПО, а также устранение последствий работы вредоносного ПО.

2) Поддерживаемое оборудование. Linux обладает заметно худшей поддержкой компьютерного оборудование, чем Windows, такого, как: принтеры, USB – устройства, сканеры и т.п. Поэтому перед приобретением оборудования необходимо узнать, имеется ли поддержка ОС Linux. На данный момент проблема драйверов на Linux постепенно уходит, пропорционально тому, как Linux получает все большее распространение.

3) Программное обеспечение. Основной недостаток Linux – это

значительно меньшее количество ПО, чем для платформы Windows. Более того, если речь идет о каких то известных программных продуктах, которые присутствуют для платформы Windows, то на Linux их может и не быть. Поэтому зачастую приходится находить менее известные аналоги, которые зачастую уступают лидерам.

4) Применение и разработка программного обеспечения. Главной платформой для разработки ПО является платформа Windows. Большая часть программ пишется на таких языках, как Java, C++, C# и т.д. Среди них есть и мультиплатформенные языки, которые позволяют разрабатывать ПО сразу под все платформы, например Java, C++, Python.

Для разработки системы идентификации лица человека по фотографии мной была выбрана ОС Windows, так как она отвечает всем предъявляемым требованиям и имеется ранее полученный опыт разработки на данной платформе.

2.2. Выбор языка программирования

Выбор языка программирования так же является важным этапом разработки приложения. Для разработки нами была выбрана операционная система Windows, для которой можно выделить такие языки программирования как: C++, Java, Python, R и т.д.

1) C++ - является языком программирования общего назначения. Он имеет процедурные, объектно-ориентированные и общие программные функции, а также включает средства высокоуровневых и низкоуровневых языков. C++ широко используется: для создания драйверов, операционных систем, различных прикладных программ. Синтаксис является наследником языка C и полностью совместим с ним.

2) Java – обьектно-ориентированный язык программирования, разработанный в 1991 году компанией Sun Microsystems. Изначально имел название Oak и использовался для бытовой электроники. Изначально для этой цели использовался язык C++, но стало очевидно, что нужен платформо-независимый язык, который позволит создать программы, которые можно было бы использовать на различных процессорах под различные ос. Впоследствии использовался для написания приложений, апплетов и серверного ПО, и был переименован в Java.

3) Python – высокоуровневый язык программирования общего назначения. Поддерживает часть парадигм программирования таких как объектно-ориентированное, структурное, функциональное программирование. Код организован в функции и классы, объединяющиеся в модули. Также является скриптовым языком и исходный код не компилируется, а выполняется с помощью интерпретатора.

4) R – язык программирования, использующийся для статистической обработки данных и работы с графикой. В R реализованы многие статистические методы (линейные и нелинейные модели, классификация, проверка статистических гипотез).

Из рассмотренных вариантов наиболее удобным инструментом для реализации поставленных задач является Python, так как в нем реализованы необходимые нам библиотеки для работы с графическими интерфейсом Tkinker и изображениями.

2.3. Выбор среды разработки

Существует множество сред разработки на языке Python:

1) Pycharm - разработан под Python, Javascript, Coffeescript, Typescript, HTML/CSS, AngularJS, Node.js и другие языки. Возможности интегрированного модульного тестирования, проверки кода, интегрированного контроля версий, инструменты рефакторинга кода, набор инструментов для навигации проекта, выделения и автоматического завершения. Поддержка ряда сторонних фреймворков для веб-разработки, таких как Django, Pyramid, web2py, Google App Engine и Flask, что делает его универсальной IDE для быстрой разработки приложений.

2) WingWare - Содержит мощный **инструмент отладки**, который позволяет устанавливать контрольные точки, возможность пошагового выполнения кода, проверка данных, удаленная отладка и отладка шаблонов **Django**. Поддержка **matplotlib**, с автоматическим обновлением графиков. Также предоставляется доработка кода, подсветка синтаксиса, исходный браузер, графический отладчик и поддержка систем управления версиями.

3) Eric - Содержит такие функции как **отладчик Python** и Ruby, покрытие кода, автоматическая проверка кода, оболочка Python и Ruby, браузер класса и многое другое. Также имеются функции для совместного редактирования. Диалоги Regex и Qt, опции для создания сторонних приложений прямо в редакторе, диаграммы приложения, возможности управления проектами, а также интерактивная оболочка Python. Многоязычный пользовательский интерфейс, который включает в себя Английский, Немецкий, Русский, Французский, Испанский, Итальянский, Турецкий и Китайский языки, контроль версии для Subversion, Mercurial и Git, использование объявлений в плагинах, и многое другое.

Самой распространенной интегрированной средой разработки считается Pycharm, который и был выбран нами, в связи с наличием огромного количества информации, удобством использования и совершенно бесплатным использованием.

2.4. Обзор существующих алгоритмов идентификации лица человека по фотографии

Задача распознавания лиц всегда относилась к числу приоритетных исследований в области систем компьютерного зрения. Она разбивается на два этапа: выделение лица человека в естественной или искусственной обстановке (детектирование) и последующее установление степени сходства с лицами из известной системе распознавания базы (идентификация).

Актуальность этой задачи, а также ее предпочтительность по сравнению с другими средствами идентификации личности (идентификация по отпечаткам пальцев или по сетчатке глаза) заключается в том, что нет необходимости непосредственного контакта системы и человека. Лицо человека само по себе содержит уникальную информацию для проведения безошибочной идентификации. Кроме того задача распознавания актуальна в системах безопасности:

1) Сравнение фотографий на документах, удостоверяющих личность;

2) Контроль доступа к безопасным компьютерным сетям и оборудованию;

3) Верификация пользователя при проведении финансовых транзакций;

4) Наблюдение в публичных местах для предотвращения террористических актов и др.

Классические системы распознавания лиц основаны на методах, ориентированных на внешность в целом: PCA (Principal Component Analysis) [1], ICA (Independent Component Analysis) [2], а также LDA (Linear Discriminant Analysis) [3]. В работе [4] описывается применение SIFT-дескрипторов для распознавания лиц, но не предлагается решения проблемы идентификации человека. В работе [5] предлагается комбинация дескрипторов PCA и SIFT: PCA используется для распознавания глаз, носа и рта, а при помощи SIFT описываются области вокруг найденных элементов внешности.

В работе [6] предлагается рассчитывать расстояния между всеми парами дескрипторов обоих изображений и использовать в качестве меры близости наименьшее из них. В некоторых случаях используются только те дескрипторы, которые описывают положение глаз и рта, поскольку эти элементы внешности являются наиболее информативными.

Альтернативой методам, ориентированным на внешность в целом, являются методы локального описания внешности. Подобные методы в настоящее время подлежат активному исследованию в области распознавания лиц. В работе [7] описывается алгоритм ASM (ActiveShape Models – Активные Модели Формы). Цель алгоритма – подстроить модель формы под форму объекта на изображении. ASM успешно используется для решения широкого спектра задач, в том

числе определение формы органов на медицинских снимках, распознавание лиц и рукописных символов.

На сегодняшний день наиболее эффективным алгоритмом поиска лиц на изображении является алгоритм, предложенный П. Виолой и М. Джонсом [8]. Ими был предложен метод классификации объектов (не только лиц) на основании метода усиления слабых классификаторов. Усиление слабых классификаторов – это подход к решению задачи классификации путем комбинирования примитивных классификаторов в один более сильный. П. Виола и М. Джонс построили каскад слабых классификаторов, работающий по принципу последовательных приближений. Каскад состоит из нескольких ступеней, каждая ступень – множество простых классификаторов. Если ступень принимает решение о том, что вектор признаков относится к классу искомого объекта, то принимается положительное решение, только если все ступени каскада это подтвердили, иначе вектор признаков классифицируется как не искомый объект. По показателям работы в реальных системах данный подход обеспечивает высокую точность, высокий уровень верных обнаружений и низкий уровень ошибок [9].

2.5. Выбор подхода к решению задачи идентификации лица человека по фотографии

Для решения задачи идентификации лица человека по фотографии была выбрана теория активного восприятия изображения [10].

Эта теория подразумевает реализацию двух этапов системы распознавания образов: разработка алгоритма предварительной обработки данных и формирование системы признаков см. рис. 2.1.

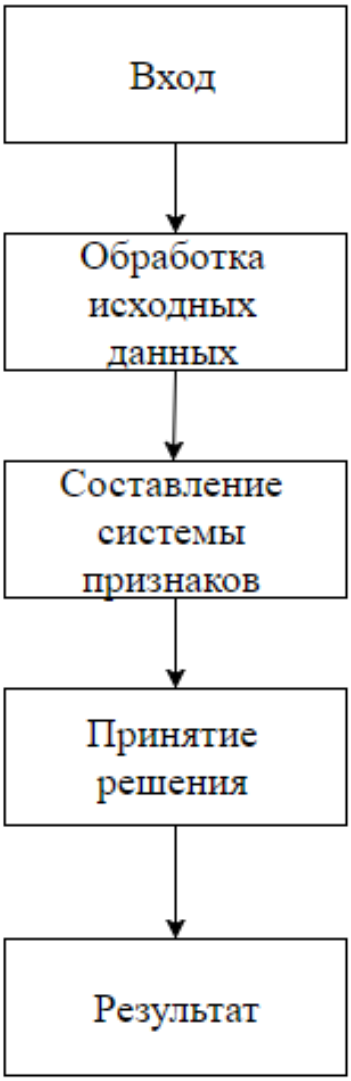

Рисунок. 2.1. Алгоритм обработки данных и формирование системы признаков

Предварительная обработка данных, в данной теории понимается, как операция интегрирования, а составление системы признаков понимается, как операция дифференцирования.

В первом случае результатом операции интегрирования будет являться множество «визуальных масс», данное преобразование определено, как Q – преобразование.

Во втором случае результатом операции дифференцирования будет являться вектор «спектральных коэффициентов» $\mu = (\mu_0, \mu_1, \mu_2, \ldots, \mu_{15})$.

Операция дифференцирования реализуется по средствам 16 фильтров см. рис.2.2. , где темный элемент «-1», светлый элемент – «+1».

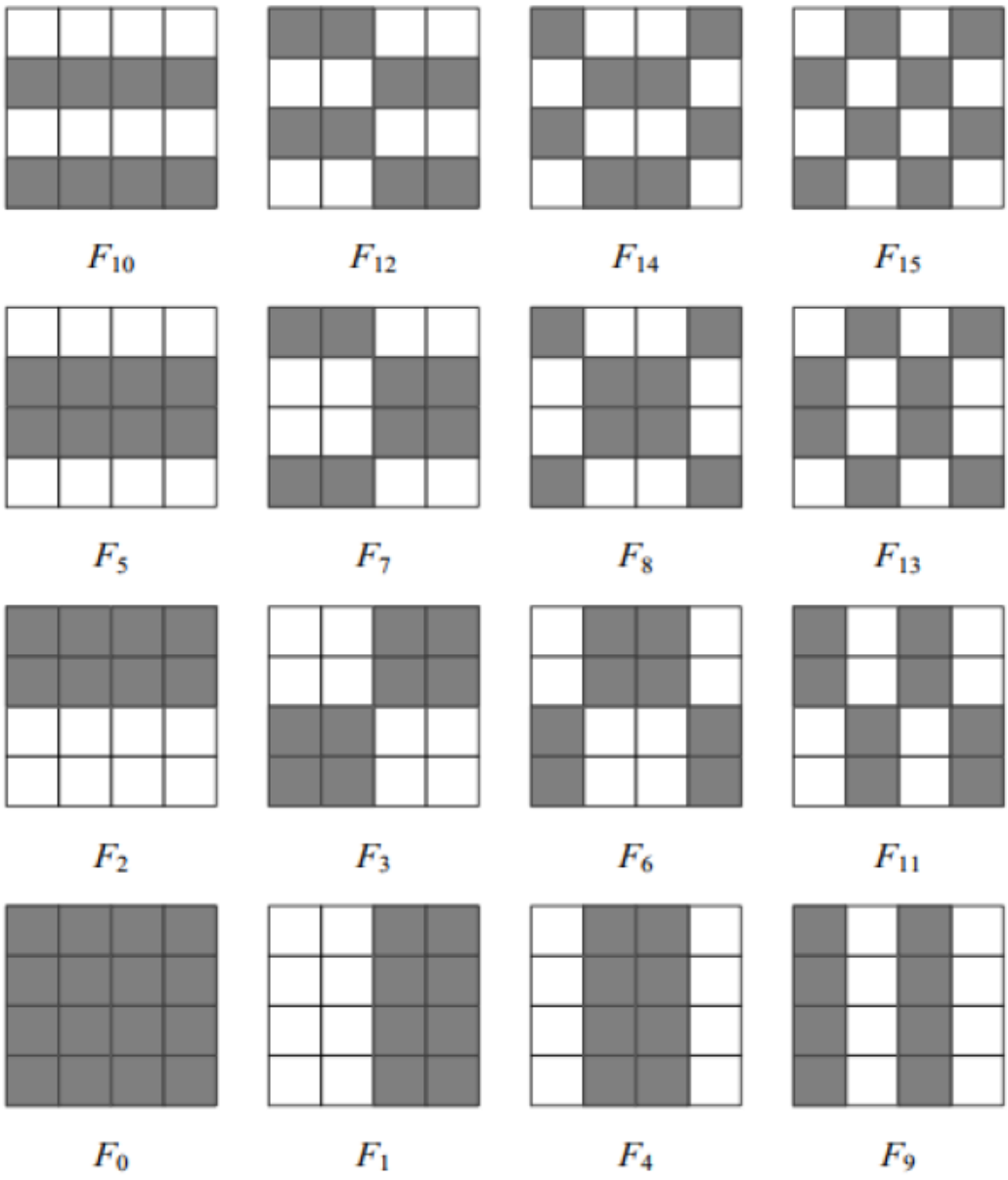

Рисунок 2.2. 16 фильтров

Наблюдается схожесть фильтров теории активного восприятия с фильтрами Уолша, однако они различаются алгоритмом применения к изображению. Фильтры теории активного восприятия изображения используются только после выполнения преобразования интегрирования (*Q*-преобразования).

Пара преобразований, интегральное и дифференциальное, вместе составляют *U* – преобразование см. рис. 2.3.

Рисунок 2.3. *U* – преобразование

3. Разработка структуры системы идентификации человека по фотографии

3.1. Разработка общей структуры системы

Идентификация человека по изображению на основе (ТАВ) строится следующим образом [16] см. рис. 3.1:

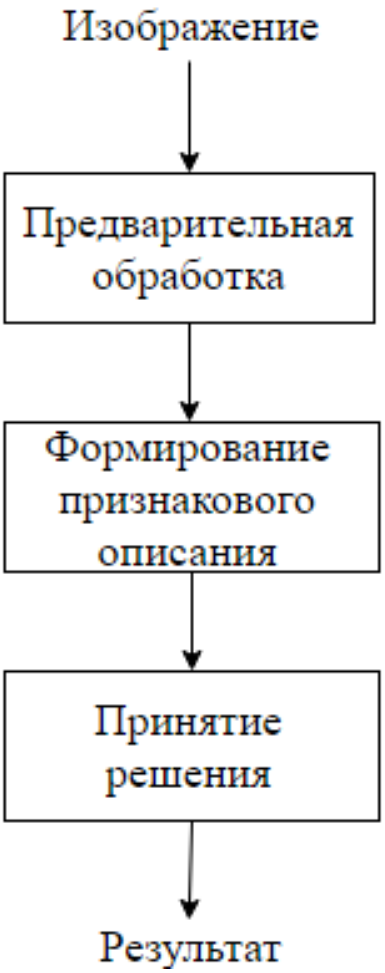

Рисунок 3.1. Структура системы

Во время предварительной обработки исследуемого изображения, оно обрезается вручную, производится нормирование изображения, *Q*-преобразование и последующее вычисление матрицы «визуальных масс». Далее происходит набор необходимых признаков (*U*-преобразование): формируются вектора спектральных коэффициентов (фильтров). На основе данных признаков, на этапе принятия решений, происходит классификация и идентификация человека, позволяющая сказать о совпадении (или отсутствие совпадения) исследуемого изображения человека и изображения хранящаяся в базе данных.

Таким образом, можно сформировать систему идентификации человека по изображению, состоящую из следующих компонентов: модуль предварительной обработки, модуль формирования признакового описания, база данных признаковых описаний, модуль принятия решения см. рис. 3.2.

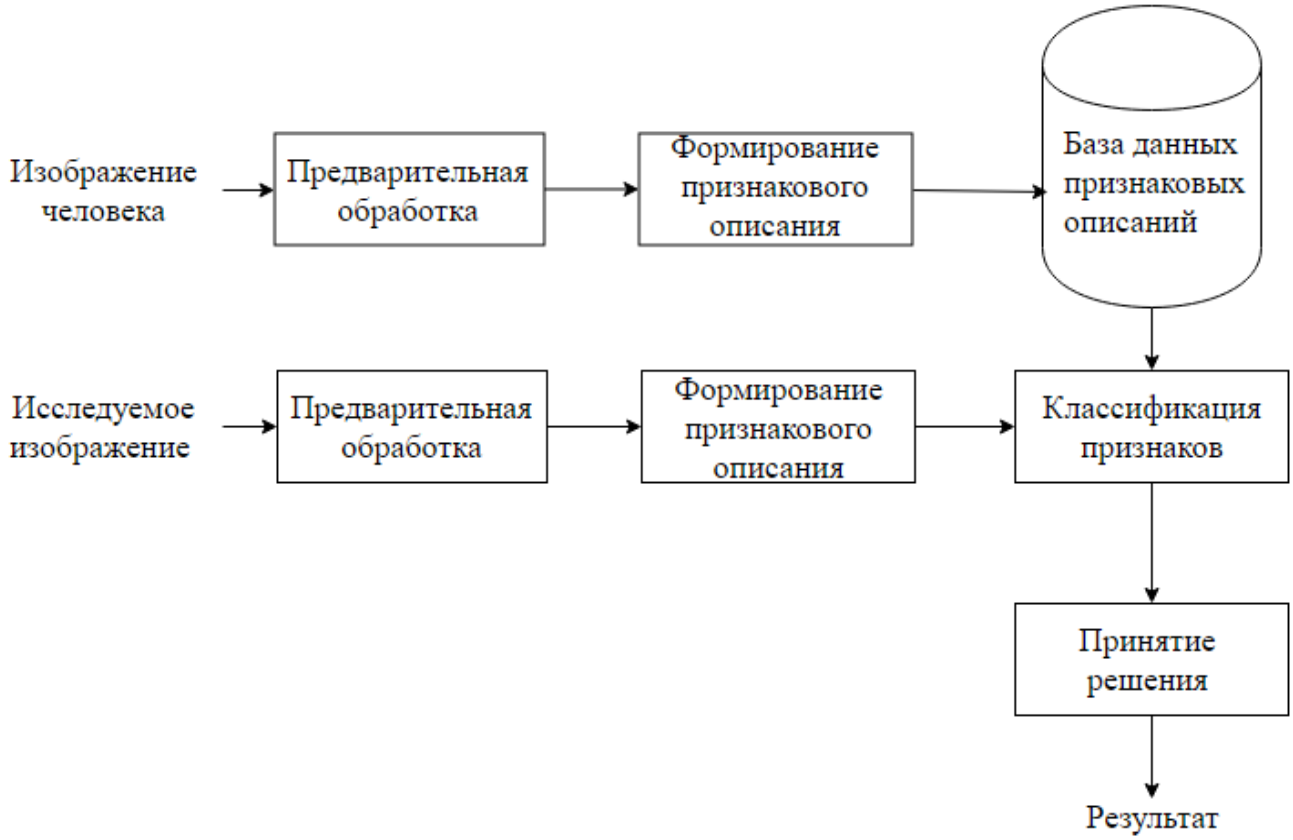

Рисунок. 3.2. Архитектура системы идентификации человека по фотографии

Клиентская программа состоит из двух модулей: графический интерфейс, который был создан при использовании библиотеки Tkinker, а так же из основной программы идентификации, написанная на Python 3.6. (рис. 3.3).

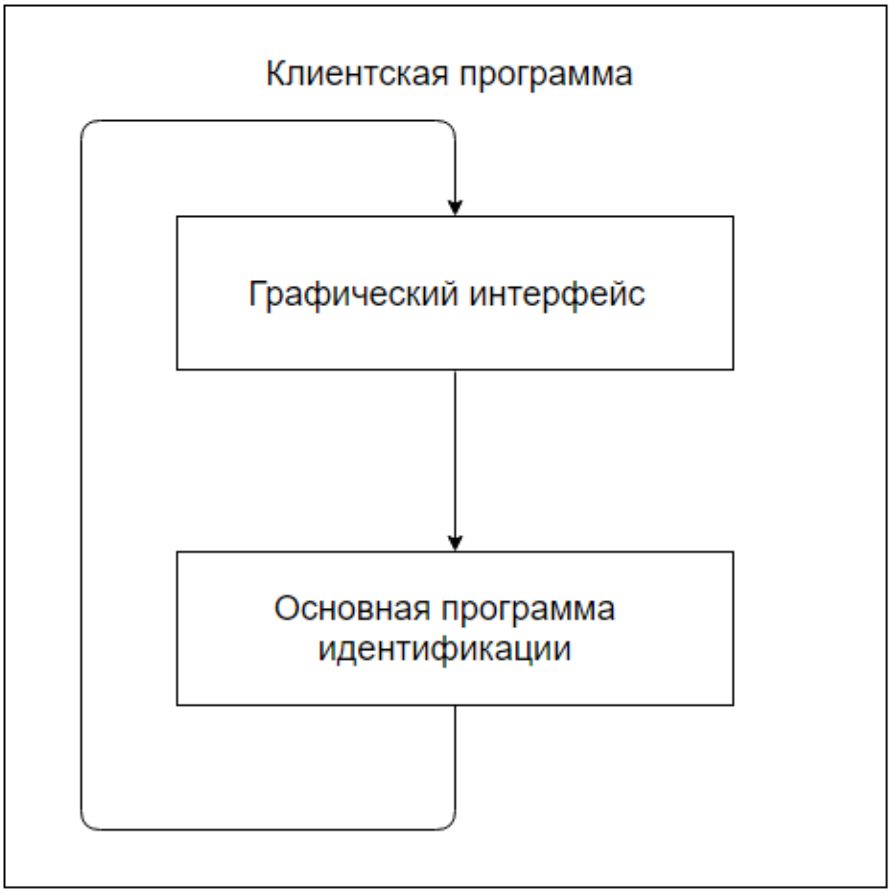

Рисунок 3.3. Алгоритм работы клиентской программы

Пользователь средствами клиентской программы может выбрать три варианта работы с программой:

1) Добавить человека. Пользователь может внести в базу данных программы нового человека и загрузить его фотографии, в случае, если человек с таким именем уже присутствует в базе, то программа сообщит пользователю об этом.

2) Добавить фото. Пользователь может загрузить в базу данных новые фотографии уже внесенного в базу данных человека, в случае, если такого человека нет в базе, программа сообщит пользователю об этом.

3) Сравнить. Пользователь может загрузить любую фотографию и провести идентификацию с внесенными в базу данных людьми, в результате, программа выведет загруженное изображение и идентифицированное изображение.

Во всех, вышеупомянутых случаях, пользователю предоставляется возможность вручную выделить лица на изображении.

3.2 Разработка алгоритма формирования признакового описания

Последовательность действий по вычислению признакового описания см. рис. 3.4:

Рисунок 3.4. Алгоритм формирования признакового описания

3.2 Разработка алгоритма принятия решения

Для проведения идентификации, необходимо провести классификацию признаковых описаний. Для каждого человека в базе данных присутствует несколько фотографий формирующие целый список из массива признаковых описаний. С помощью классификатора, все признаки классифицируются, таким образом, признаковые описания исследуемого изображения будут отнесены к одному из классов. На основе этого мы получим результат. Рассмотрим алгоритм по принятию решения об идентификации человека см. рис. 3.5:

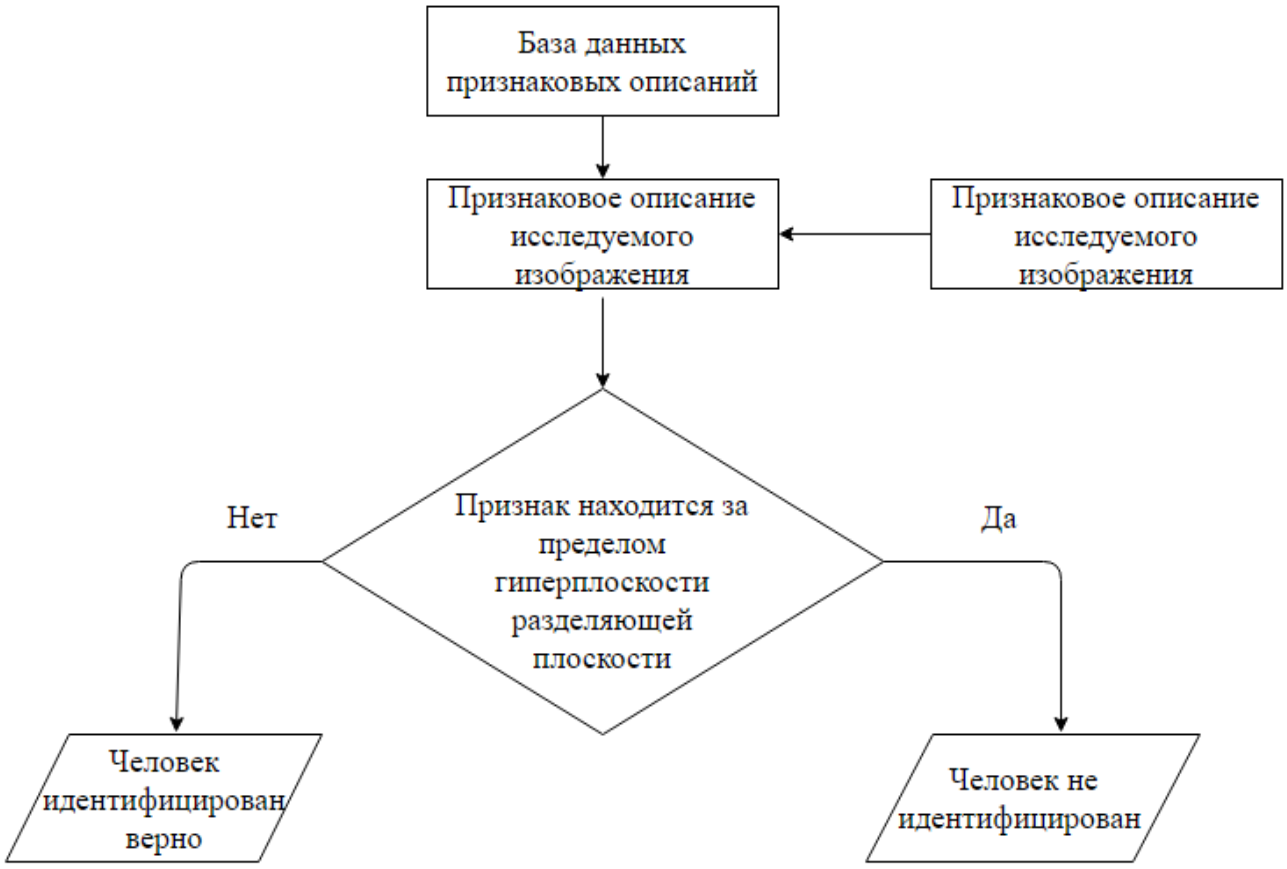

Рисунок 3.5. Алгоритм принятия решения

4. Разработка программных средств

4.1.Разработка интерфейса пользователя системы

Разработка интерфейса пользователя проводилась на языке Python. Использовалась библиотека Tkinker. Рассмотрим примененные классы и методы данной библиотеки:

1) Button - простая кнопка, используемая для выполнения команды или другой операции.

2) Canvas - структурированная графика. Этот виджет может использоваться для рисования графиков, создания графических редакторов и для реализации пользовательских виджетов.

3)Checkbutton - редставляет переменную, которая может иметь два разных значения. Нажатие кнопки переключает между значениями.

4) Entry - поле ввода текста.

5)Frame - виджет контейнера. Фрейм может иметь границу и фон и используется для группировки других виджетов при создании приложения или макета диалога.

6) Label - отображает текст или изображение.

7) Listbox - отображает список альтернатив.

8) Menu - панель меню. Используется для реализации раскрывающихся меню и всплывающих меню.

9) Menubutton - меню. Используется для реализации выпадающих меню.

10) Message - отображает текст. Подобно виджету ярлыков, но может автоматически переносить текст в заданную ширину или соотношение сторон.

11) Radiobutton - представляет одно значение переменной, которое может иметь одно из нескольких значений. Нажатие кнопки устанавливает переменную в это значение и очищает все другие radiobutton, связанные с одной и той же переменной.

12) Scale - позволяет установить числовое значение, перетащив «ползунок».

13) Scrollbar - полоса прокрутки.

14) Text - форматированный текстовый дисплей. Позволяет отображать и редактировать текст с различными стилями и атрибутами. Также поддерживает встроенные изображения и окна.

15) Toplevel - виджет контейнера отображается как отдельное окно верхнего уровня

При разработке интерфейса программы использовались следующие элементы: Button, Label, Entry . Программная реализация GUI представлена на рис. 4.1.

```
def main():
    root = Tk()
    root.title( ' Идентификация ' )
    root.configure(background="light gray")
    lable1 = Label( root ,
                text = '    Эта программа предназначена для проведения идентификации по фото. \n '
                       '    Пожалуйста выберите один из 3х вариантов ' ,
                font = ' arial 11 ' )
    lable1.grid( row = 1 , column = 0 , columnspan = 3 )
    button1 = Button(root,text='Сравнить',background="light gray", font='arial 14',command=button_1)
    button1.grid(row=3,column=0)
    button2=Button(root,text='Добавить фото',background="light gray",font='arial 14',command=button_2)
    button2.grid(row=3,column=1)
    button3=Button(root,text='Добавить человека',background="light gray",font='arial 14',command=button_3)
    button3.grid(row=3,column=2)
    button5 = Button(root, text='Выход', background="light gray", fg= "red", font='arial 14',command=root.quit)
    button5.grid(row=4, column = 1)
    root.mainloop()
```

Рисунок 4.1. Программная реализация интерфейса

Рассмотрим программные составляющие интерфейса:

1) root – переменная базового класса Tkinter приложения;

2) root.geometry задает параметры окна приложения;

3) label1 элементы применяются для сообщения пользователю информации;

4) button1 – кнопка, по нажатию которой пользователь может загрузить изображение для проведения идентификации;

5) button2 – кнопка, по нажатию которой пользователь может добавить изображение в базу;

6) button3 – кнопка, по нажатию которой пользователь может добавить нового человека в базу.

С помощью функции «mainLoop()» запускается цикл обработки событий.

Обработчик кнопки «Сравнить» – функция «button_1()», см. рис. 4.2.

```
def button_1():
    res1=getlendopname()[:]
    ind=int(classification(openfile()))
    print (ind)
    result = res1[ind]
    messagebox.showinfo("Информация", "Формирование признака завершено")
    top1 = Toplevel()
    lable4 = Label(top1,
                   text="Результат идентификации: " + result,
                   font=' arial 15 '
                   , fg='green')
    lable4.grid(row=1, column=0,columnspan = 2 )
    button9 = Button(top1, text='Закрыть',
                     font='arial 14',fg= "red", command=top1.destroy)
    button9.grid(row=3, column=1)
    image1 = Image.open("test.jpg")
    photo1 = ImageTk.PhotoImage(image1)
    image2 = Image.open(str(result) + ".jpg")
    photo2 = ImageTk.PhotoImage(image2)
    label5 = Label(top1,image=photo1)
    label5.image = photo1
    label5.grid(row=2, column=0)
    label6 = Label(top1,image=photo2)
    label6.image = photo2
    label6.grid(row=2, column=1)
    top1.mainloop()
```

Рисунок 4.2. Обработчик кнопки «Сравнить»

Обработчик кнопки «Добавить фото» – функция «button_2()», см. рис. 4.3.

```
def button_2():
    top = Toplevel()
    mas=[]
    mystring = StringVar()
    lable2 = Label(top,
                   text='Введите имя человека,\n чье фото желаете добавить',
                   font=' arial 11 ')
    lable2.grid(row=1, column=0,sticky=W)
    text1 = Entry(top,textvariable = mystring)
    text1.grid(row=2, column=0)

    def cliced2():
        #name.append(mystring.get())
        if openNameFile().count(mystring.get())==0:
            messagebox.showerror("Ошибка", "Такого человека нет в базе!")
            top.destroy
        else:
            MassAppend(mystring.get(),openfile())
            messagebox.showinfo("Информация", "Формирование признака завершено")
            mas.clear()
    button6 = Button(top, text='ok', font='arial 14', command=cliced2)
    button6.grid(row=3, column=0)
    button4 = Button(top, text='Закрыть', font='arial 14',command=top.destroy)
    button4.grid(row=3, column=1)
    top.mainloop()
```

Рисунок 4.3. Обработчик кнопки «Добавить фото»

Обработчик кнопки «Добавить человека» – функция «button_3()», см. рис. 4.4.

```
def button_3():
    top3 = Toplevel()
    mystring3 = StringVar()
    lable3 = Label(top3,
                   text='Введите имя человека,\n которого желаете добавить',
                   font=' arial 11 ')
    lable3.grid(row=1, column=0,sticky=W)
    text2 = Entry(top3,textvariable = mystring3)
    text2.grid(row=2, column=0)

    def cliced3():
        if openNameFile().count(mystring3.get())!=0:
            messagebox.showerror("Ошибка", "Такой человек уже есть в базе!")
            top3.destroy
        else:
            appendNameFile(mystring3.get())
            MassAppend(mystring3.get(),openfile())
            messagebox.showinfo("Информация", "Формирование признака завершено")
            print(openNameFile())
    button7 = Button(top3, text='ok', font='arial 14', command=cliced3)
    button7.grid(row=3, column=0)
    button8 = Button(top3, text='Закрыть', font='arial 14',command=top3.destroy)
    button8.grid(row=3, column=1)
```

Рисунок 4.4. Обработчик кнопки «Добавить человека»

Также в программе для взаимодействия с пользователем предназначены диалоговые окошки, сообщающие о следующих событиях:

1) Человек присутствует в базе;
2) Человек отсутствует в базе;
3) Формирование признака завершено.

Данные окошки реализованы с помощью модуля «tkinter.messagebox» следующим способом, см. рис. 4.5:

```
messagebox.showinfo("Информация", "Формирование признака завершено")
```

Рисунок 4.5. Пример реализации диалогового окна

4.2. Программная реализация модулей системы

4.2.1 Модуль предварительной обработки изображения.

С помощью функции «norm_face» обрезаем фотографию, нажав на левый верхний угол и на правый нижний угол желаемого изображения (см. рис. 4.6.):

```
def norm_face(SourseImage, namefile):
    im = Image.open(SourseImage)
    imshow(im)
    print("Please click 2 points")
    X = ginput(2)
    print("you clicked:", X)
    show()
    x1 = int(X[0][0])
    y1 = int(X[0][1])
    x2 = int(X[1][0])
    y2 = int(X[1][1])
    box = (x1, y1, x2, y2)
    region = im.crop(box)
    X.clear()
    print(namefile)
    region.save(str(namefile) + '.jpg')
```

Рисунок 4.6. Реализация метода «norm_face»

С помощью функции «gray_scale» создаем матрицу каналов изображений. Переводим изображения в градации серого путем сложения RGB – каналов (см. рис. 4.7.):

```
def gray_scale(source_name):
    source = Image.open(source_name)
    result = Image.new('L', source.size)
    for x in range(source.size[0]):
        for y in range(source.size[1]):
            r, g, b = source.getpixel((x, y))
            gray = round((r + g + b) / 3)
            result.putpixel((x, y), (gray))
    return result
```

Рисунок 4.7. Реализация метода «gray_scale»

Функция «image_to_mussiv» позволяет привести изображение к матричному виду см. рис. 4.8.

```
def image_to_massive(source):
    width = source.size[0]  # Определяем ширину.
    height = source.size[1]  # Определяем высоту.
    massive = np.zeros((height, width))
    for x in range(width):
        for y in range(height):
            gray = source.getpixel((x, y))
            massive[y][x] = gray
    return massive
```

Рисунок 4.8. Реализация метода «image_to_mussiv»

Это позволяет нам провести нормирование: используя функцию «minimum» мы вычитаем из каждого элемента минимальный, а с помощью функции «maximum» делим каждый элемент на максимальный см. рис. 4.9.

```
def minimum(source):
    minimum = 255
    for x in range(source.shape[0]):
        for y in range(source.shape[1]):
            if (source[x][y] < minimum):
                minimum = source[x][y]
    for x in range(source.shape[0]):
        for y in range(source.shape[1]):
            source[x][y] = source[x][y] - minimum
            if (source[x][y] < 0):
                source[x][y] = 0
    return source

def maximum(source):
    maximum = 0
    for x in range(source.shape[0]):
        for y in range(source.shape[1]):
            if (source[x][y] > maximum):
                maximum = source[x][y]
    for x in range(source.shape[0]):
        for y in range(source.shape[1]):
            source[x][y] = round(source[x][y] / maximum, 1)
    return source
```

Рисунок 4.9. Функции для обработки минимального и максимального элемента

4.2.2 Модуль формирования признаков.

После проведенных операций над матрицей, производим *Q* преобразование «q_preobr» и получаем «матрицу визуальных масс» см. рис. 4.10.

```
def q_preobr(source):
    q = 4
    N = int(source.shape[1] / 4)   # stolb
    M = int(source.shape[0] / 4)   # stroka
    result = np.zeros((q, q))
    for i in range(q):  # stolb
        for j in range(q):  # stroka
            sum = 0
            for y in range(i * M, (i + 1) * M):
                for x in range(j * N, (j + 1) * N):
                    sum = sum + source[y][x]
            result[i][j] = sum
    return result
```

Рисунок 4.10. Реализация метода «q_preobr»

Далее производим « *U*- преобразование» с помощью функций «m_create» и «m_mass_create» таким образом, получаем матрицы спектральных коэффициентов см. рис. 4.11.

```
def m_create(source, f):
    res = 0
    for x in range(source.shape[0]):
        for y in range(source.shape[1]):
            res = res + source[x][y] * f[x][y]
    return round(res, 1)

def m_mass_create(source):
    mass = []
    mass.append(m_create(source, F0))
    mass.append(m_create(source, F1))
    mass.append(m_create(source, F2))
    mass.append(m_create(source, F3))
    mass.append(m_create(source, F4))
    mass.append(m_create(source, F5))
    mass.append(m_create(source, F6))
    mass.append(m_create(source, F7))
    mass.append(m_create(source, F8))
    mass.append(m_create(source, F9))
    mass.append(m_create(source, F10))
    mass.append(m_create(source, F11))
    mass.append(m_create(source, F12))
    mass.append(m_create(source, F13))
    mass.append(m_create(source, F14))
    mass.append(m_create(source, F15))
    return mass
```

Рисунок 4.11. Формирование спектральных коэффициентов

Работа с базой данных происходит при помощи функций «openXCoefFile()» и «appendXCoefFile()» для признаковых описаний (см. рис. 4.12) и «appendNameFile()» и «openNameFile()» для имен (см. рис. 4.13).

```
def openXCoefFile(namefile):
    with open(str(namefile)+'.pickle', 'rb') as f:
        try:
            coefXdata = pickle.load(f)
        except EOFError:
            coefXdata = list()
    npCoef=np.array(coefXdata, dtype=float64)
    return npCoef

def appendXCoefFile(namefile, coef):# photo one human
    with open(str(namefile)+'.pickle', 'ab') as f:
        pickle.dump(coef, f)
```

Рисунок 4.12. Реализация методов «openXCoefFile()» и «appendXCoefFile()»

```
def openfile():
    op = askopenfilename()   # окно выбс
    return op

def openNameFile():# Y
    f = open('name.txt', 'a')
    f.close()
    f = open('name.txt', 'r')
    name = [line.strip() for line in f]
    f.close()
    return name
```

Рисунок 4.13. Реализация методов «appendNameFile()» и «openNameFile()»

4.2.3 Модуль принятия решения

Согласно алгоритму, загружаем признаки из базы данных, в которой находятся все признаковые описания. Далее обучаем классификатор на основании этих признаков. Формируем признаковое описание исследуемого

изображения и определяем его положение относительно гипперплоскости разделяющие образованные классы. Классификация признаков реализована в функции «classification()» см. рис. 4.14:

```
def classification(ImageSourse):
    c = (getclassific()[:])
    y=[]
    for i in range(len(openNameFile())):
        y.append(i)
    print("X")
    print(c)
    print("Y")
    print(y)
    clf =SVC()
    clf.fit(c, y)
    SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='auto', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
    norm_face(ImageSourse, "test")
    gray1 = gray_scale(ImageSourse)
    norm_mass = norma(gray1)
    q4 = m_mass_create(q_preobr(norm_mass))

    print( "ver" + str(clf.decision_function([q4])))
    pred = clf.predict([q4])
    print(pred)
    y.clear()
    return pred
```

Рисунок 4.14. Реализация метода «classification()»

5.Тестирование системы

5.1 Описание набора данных

Для проведения качественной идентификации лучше использовать фотографии, сделанные с одного ракурса, с одинаковой выдержкой и освещенностью, чтобы избежать погрешностей. Разработанная система позволяет сравнивать только одно изображение со всеми. Рекомендуемое количество фотографий одного человека более 10, для большего обхвата диапазона коэффициентов.

Ниже приведены примеры изображений:

Рисунок 15.1. Тестовое изображение №1

Рисунок 5.2 Тестовое изображение №2

Рисунок 5.3. Тестовое изображение №3

Рисунок 5.4. Тестовое изображение №4

5.2 Описание методики тестирования

Проверим работоспособность программы. Для этого запустим пользовательский интерфейс, загрузим тестовые изображения, просмотрим результат идентификации. Пользовательский интерфейс представлен на рис. 5.5.

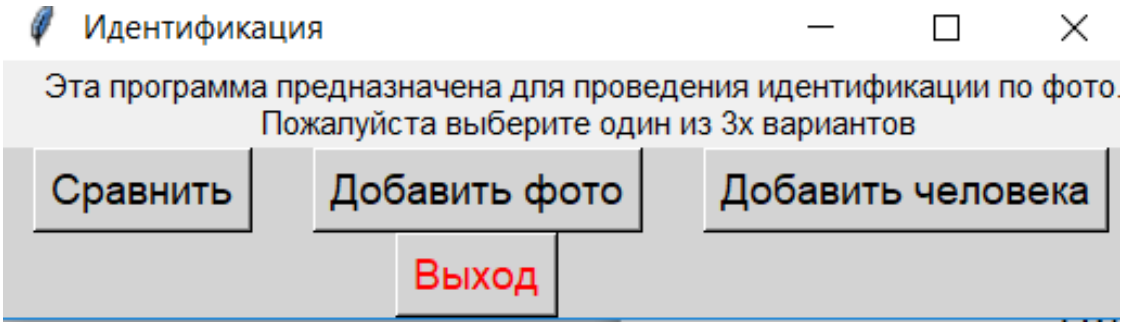

Рисунок. 16.5. Пользовательский интерфейс

Порядок работы системы идентификации изображений:

1) Необходимо добавить человека, введя его имя в поле ввода текста, нажав соответствующую кнопку «Добавить человека» см. рис. 5.6.

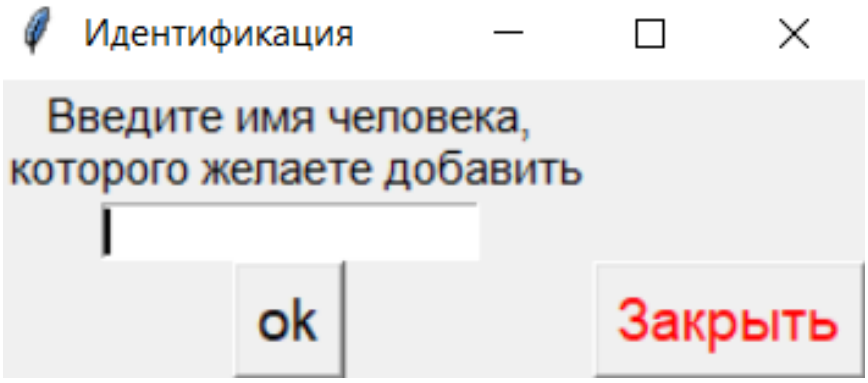

Рисунок 5.6 Интерфейс кнопки «Добавить человека»

2) Если такой человек уже присутствует в базе, система выведет соответствующее сообщение см. рис. 5.7. Иначе откроется окно выбора изображения см. рис. 5.8.

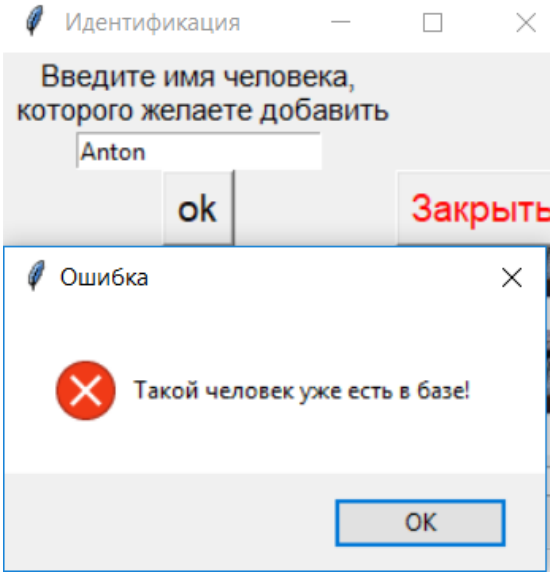

Рисунок 5.7 Информационное сообщение о наличие человека в базе

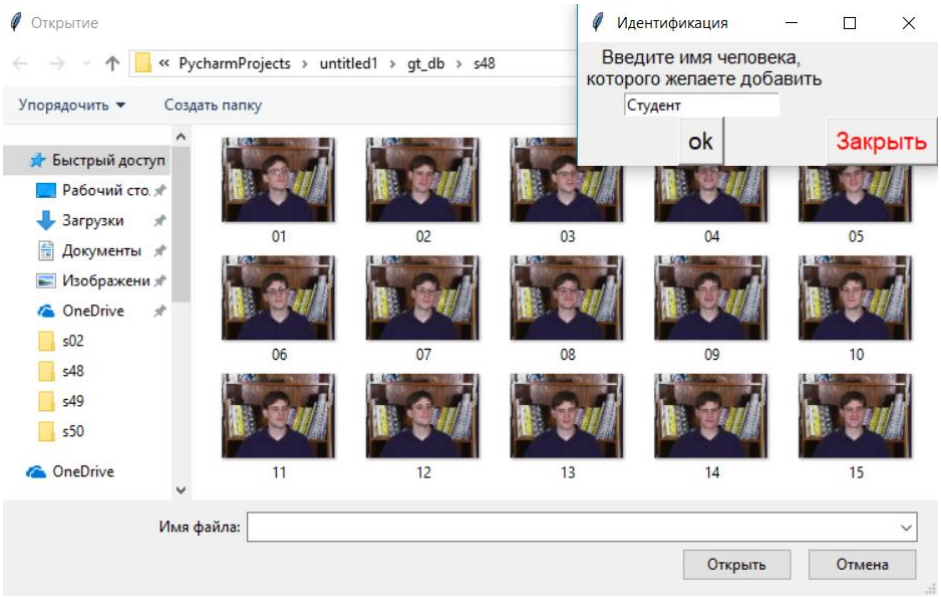

Рисунок 5.8 Окно выбора изображения при добавлении нового человека

3) Пользователь может добавить фото человека в базу, нажав кнопку «Добавить фото», и введя его имя в поле ввода текста см. рис. 5.9.

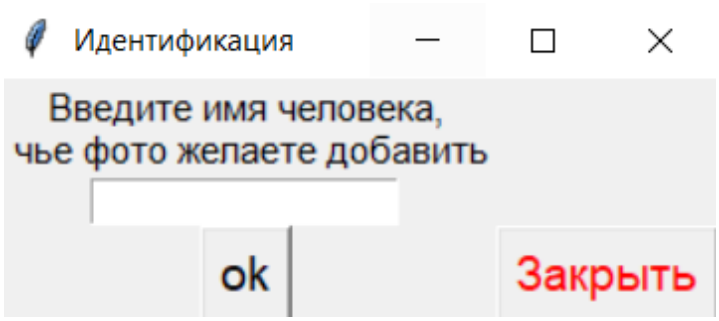

Рисунок 5.9 Интерфейс кнопки «Добавить фото»

4) Если такого человека нет в базе, система выведет соответствующее сообщение см. рис. 5.10. Иначе откроется окно выбора изображения см. рис. 5.8.

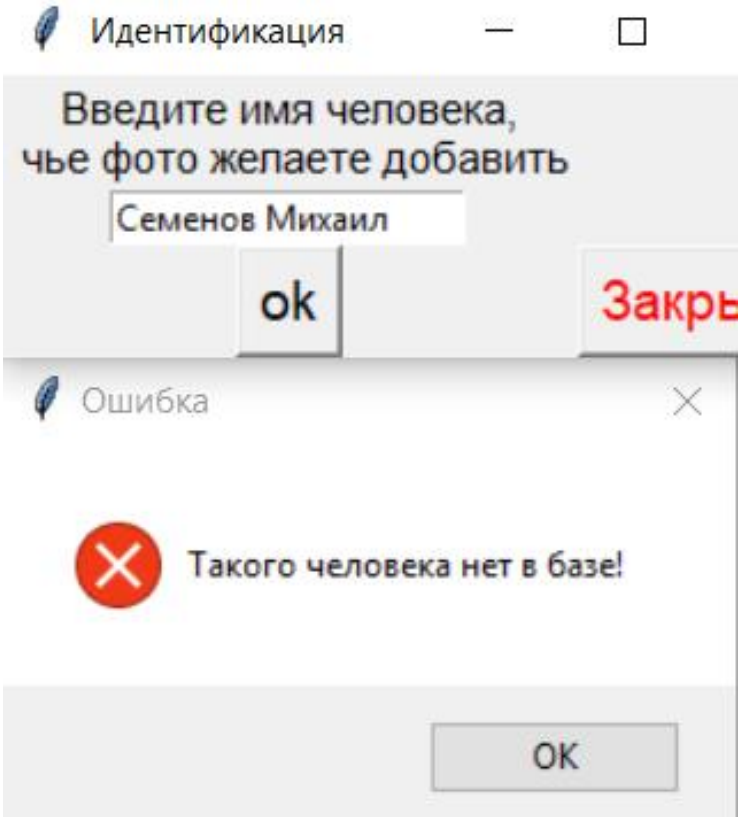

Рисунок 5.10 Информационное сообщение об отсутствие человека в базе

5) Для проведения идентификации необходимо нажать на кнопку «Сравнить». Будет открыто соответствующее окно выбора изображения см. рис. 5.8.

5.3 Результат вычислительного эксперимента

Рассмотрим результаты работы системы на тестовых изображениях:

Рисунок 5.11 Результат идентификации тестового изображения №1

Рисунок 5.12 Результат идентификации тестового изображения №2

Рисунок 5.13 Результат идентификации тестового изображения №3

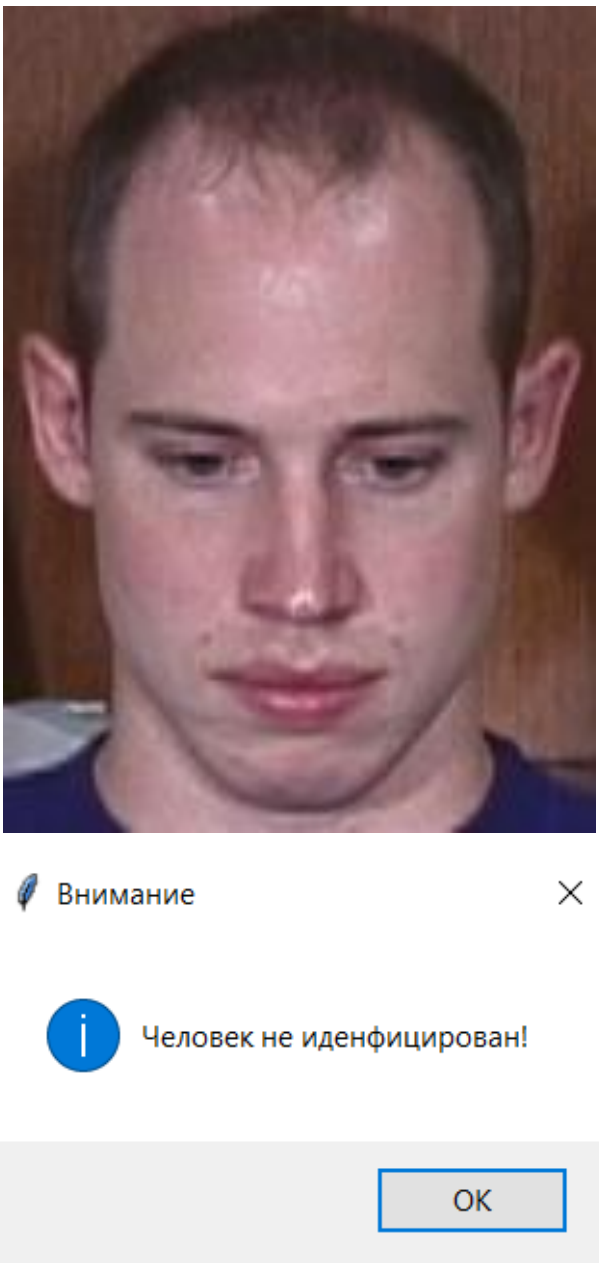

Рисунок 5.14 Результат идентификации тестового изображения №4

Первые 3 тестовых изображения были идентифицированы верно, а 4 нет, что соответствует истине, так как, такого человека нет в базе.

Заключение

В результате выполнения выпускной квалификационной работы была спроектирована и реализована программная система идентификации человека по фотографии. Общий алгоритм работы программы был основан на теории активного восприятия.

Разработанная система предназначена для идентификации человека по фотографии. Тестирование программы подтвердило ее работоспособность и возможность дальнейшего использования для решения данной задачи.

Список литературы

1. Kirby M. and Sirovich L. Application of the Karhunen-Loeve procedure for the characterization of human faces // IEEE Trans. Pattern Analysis and Machine Intelligence. – 1990. – № 12. – P. 103-108.
2. Bartlett M.S., Movellan J.R. and Sejnowski J. Face recognition by independent component analysis // IEEE. Trans. Neural Networks. – 2002. – № 13. – P. 1450-1464.
3. Belhumeur P.N., Hespanha J.P., Kriegman D.J. Eigenfaces vs. Fisherfaces: Recognition using Class Specific Linear Projection // Proc. of the 4th European Conference on Computer Vision. – Cambridge, UK. – 1996. – P. 45-58.
4. Lowe D.G. Distinctive image features from scale-invariant keypoints // International Journal of Computer Vision. – 2004. – № 60. – P. 91-110.
5. Sivic J., Everingham M. and Zisserman A. Person spotting: Video shot retrieval for face sets // Lecture Notes in Computer Science. – 2005. – P. 226-236.
6. Bicego M., Lagorio A., Grosso E. and Tistarelli M. On the use of SIFT features for face authentication // Proceedings of Conf. on Computer Vision and Pattern Recognition Workshop. – New York, NY, USA. – 2006. – 35 p.
7. Cootes T.F. and Taylor C.J. Statistical Models of Appearance for Computer Vision. – University of Manchester. – 2004. – 124 p.
8. Viola P., Jones M.J. Robust Real-Time Face Detection // International Journal of Computer Vision. – 2004. – Vol. 57, №. 2. – P. 137-154.
9. Вежневец А.П. Методы классификации с обучением по прецедентам в задаче распознавания объектов на изображениях // Труды конференции Graphicon-2006. – 2006. – С. 166-173.
10. Утробин В. А. Элементы теории активного восприятия изображений // НГТУ, 2001. - 64 с.