Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное
учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Институт естественных и точных наук
Факультет математики, механики и компьютерных технологий
Кафедра прикладной математики и программирования
Направление подготовки: 01.03.02 Прикладная математика и информатика

РАБОТА ПРОВЕРЕНА
Рецензент

_____/
«____»_____20___г.

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой, д.ф.-м.н., доцент
_____/А.А. Замышляева
«____»_____20___ г.

Определение идентичности двух изображений с учетом искажающих факторов

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУрГУ–01.03.02.2019.54.ПЗ ВКР

Руководитель работы
Доцент кафедры ПМиП, к.т.н.
_____/Т.Ю. Оленчикова
«____»_____2019 г.
Автор работы
Студент группы ЕТ-412
_____/ А.В. Ивашкин
«____»_____2019 г.
Нормоконтролер, ассистент
_____/Н.С. Мидоночева
«____»_____2019 г.

Челябинск
2019

АННОТАЦИЯ

Ивашкин А.В. Определение идентичности двух изображений с учетом искажающих факторов. – Челябинск: ЮУрГУ, ЕТ-412, 46 с., 34 ил., библиогр. список – 16 наим., 1 прил.

Целью данной работы является определение идентичности двух изображений с учетом искажающих факторов в виде компьютерной программы.

В первом разделе были рассмотрены методы и программы, применяющиеся для определения идентичности изображений. Приводится обоснование выбранного метода определения идентичности изображений.

Во втором разделе описана математическая модель распознавания идентичности изображений и искажений изображения.

Третий раздел посвящен разработке алгоритмов распознавателя идентичности изображений, а также разработке и описанию пользовательского интерфейса.

В четвертом разделе описаны результаты работы алгоритма на экспериментальных данных, а также результаты работы программы на тестовых изображениях.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Планово-предупредительный ремонт (ППР) является одним из важнейших этапов эксплуатации оборудования. Его сущность состоит в том, чтобы содержать оборудование в работоспособном состоянии. После отработки оборудованием определенного времени, нужно провести профилактический смотр и, если нужно, произвести ремонт, а также сделать фотографии оборудования до ППР и после.

Рабочие не всегда бывают добросовестными и не всегда выполняют эту работу качественно и в срок. Поэтому некоторые личности ищут способы обмануть работодателя, чтобы проделанная работа выглядела отлично, хотя не является таковой. Они не производят ППР, а берут фотографии прошлых лет и изменяют ее в графическом редакторе. Таких нарушителей можно выявить с помощью одного из самых популярных направлений компьютерных технологий – компьютерного зрения.

Программные решения, использующие компьютерное зрение, давно внедрены в нашу жизнь. Они выполняют различные задачи, и для каждой задачи существует множество способов решения.

Цель данной работы – предложить один из способов реализации системы проверки на идентичность двух изображений. Одно является оригиналом, а другое было изменено в графическом редакторе, т.е. искажено. Программа разрабатывается в соответствии с заказом от Челябинского информационно-вычислительного центра и в дальнейшем может быть внедрена в рабочий процесс, что позволит отследить недобросовестных сотрудников и повысить безопасность работы Южно-Уральской железной дороги.

Для достижения поставленной цели необходимо решить следующие задачи:

1) изучить и проанализировать методы, применяющиеся для распознавания идентичности изображений;

2) разработать математическую модель проверки идентичности изображений с учетом искажающих факторов;

3) разработать компьютерную программу, реализующую разработанную модель;

4) проверить работу программы на экспериментальных данных.

Содержание работы по главам.

В первой главе разобраны методы распознавания идентичности изображений:

– метод цветовых гистограмм;

– перцептивные хэш-алгоритмы;

– SIFT;

– SURF;

– ORB.

Наиболее подходящим методом для решения поставленной задачи является метод ORB, он был использован и модифицирован в дальнейшем.

Во второй главе разобрана математическая модель составных частей алгоритма ORB: FAST и BRIEF. А также приведены формулы для искажений изображения: перенос, сдвиг по оси и , масштабирование, вращение, изменение яркости и перевод изображения в градации серого.

В третьей главе разработаны алгоритм распознавателя и интерфейс программы.

В четвертой главе разобраны исходные данные, а также проведена оценка влияния искажений на распознавание.

Проверенные искажения:

– увеличение в центр;

– увеличение в левый верхний угол;

– смещение вниз;

– смещение вправо;

– поворот;

– изменение яркости.

Произведена оценка качества работы алгоритма и проверка работы программы.

В результате проделанной работы разобраны методы определения идентичности изображений, выбран и модифицирован метод ORB.

Произведена проверка влияния искажений на распознавание на тестовом изображении, т.е. при каком проценте искажения распознавание прекращается.

Общая оценка качества работы алгоритма на предложенных экспериментальных данных равна 99,9%.

Разработана программа, которая производит поиск дубликата или искаженного дубликата выбранного изображения в выбранном каталоге и подкаталогах и выводит на экран миниатюру найденного дубликата и путь к файлу.

1 МЕТОДЫ РАСПОЗНАВАНИЯ ИДЕНТИЧНОСТИ ИЗОБРАЖЕНИЙ

1.1 Обзор методов распознавания идентичности изображений

1.1.1 Метод цветовых гистограмм

Этот метод является очень популярным [1]. Он используется во многих задачах компьютерного зрения, таких как поиск похожих изображений, поиск конкретного объекта на изображении, а также классификация. Его популярность можно объяснить несколькими факторами, а именно устойчивостью к переносам, масштабированию и частичному перекрытию объектов, а также простотой вычисления и высокой скоростью получения.

Идея метода цветовых гистограмм заключается в следующем [2]: все множество цветов разбивается на набор непересекающихся, полностью покрывающих его подмножеств; для изображения формируется гистограмма, отражающая долю каждого подмножества цветов в цветовой гамме изображения.

Строится она очень просто [3]. Сначала представим RGB палитру в виде трехмерного куба, каждая ось которого отвечает за свой цвет. При таком рассмотрении любой цвет из RGB палитры будет лежать где-то внутри куба. Например, для рисунка 1.1 получится такой куб (рисунок 1.2).

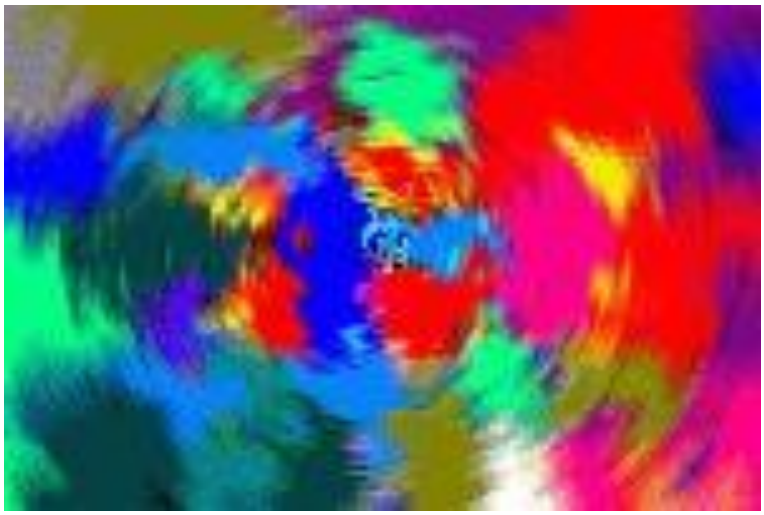

Рисунок 1.1 – Пример изображения

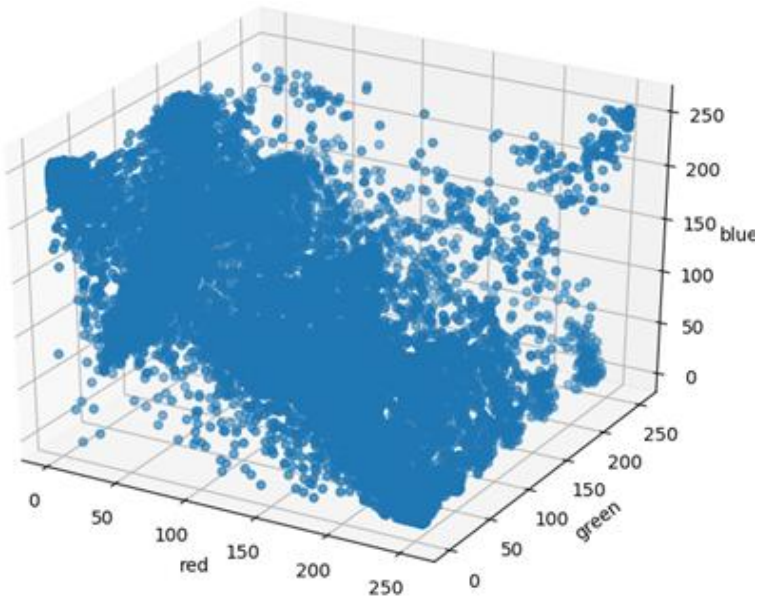

Рисунок 1.2 – Представление всех цветов
изображения в виде точек внутри куба

Чтобы построить гистограмму нужно поделить все стороны на 4 равные части. Получится, что куб делится на 64 прямоугольных параллелепипеда. Гистограмма изображения отражает распределение точек RGB-пространства, соответствующих цветам пикселей изображения, по параллелепипедам.

Можно построить гистограмму иным способом – составить гистограммы для каждого цвета и соединить их все на одном графике. Получится гистограмма, состоящая из всех цветов.

Для сравнения гистограмм вводится понятие расстояния между ними. Чтобы понять, насколько два изображения схожи по их цветовым гистограммам и , можно использовать различные функции расстояния, разберем некоторые наиболее популярные из них [4].

Пересечение гистограмм:

(1.1)

Расстояние Чебышева:

(1.2)

Метрика хи-квадрат:

(1.3)

Корреляционный метод:

(1.4)

где ;

– число столбцов гистограммы.

Цветовые гистограммы содержат информацию только о цвете изображения, поэтому совершенно разные изображения с похожей цветовой палитрой могут быть распознаны как одинаковые. Поэтому этот метод не подходит для решения поставленной задачи.

1.1.2 Перцептивные хэш-алгоритмы

В этом методе формируется индивидуальный «отпечаток» изображения [5]. Он и является перцептивным хэшем. Эти «отпечатки» подлежат сравнению, для определения идентичности.

Существует несколько различных методов формирования перцептивного хэша изображения, но все они делятся на основные этапы:

– предварительная обработка изображения. Это нужно для того, чтобы было легче обрабатывать изображение для построения хэша;

– непосредственно, вычисления. На этом этапе из обработанного изображения строится матрица (или вектор);

– формирование хэша. Из полученной матрицы берутся некоторые (возможно все) коэффициенты и преобразуются в хэш.

Вычисленные значения хэшей потом сравниваются с помощью функций, которые вычисляют «расстояние» между двумя хэшами.

Как уже было сказано выше, сформировать перцептивный хэш можно несколькими способами, разберем один из них, а именно «Simple Hash» [6]. Его суть заключается в отображении среднего значения низких частот.

Высокие частоты в изображениях обеспечивают детализацию, а низкие – структуру. Поэтому в начале алгоритма нам надо будет избавиться от высоких частот. Самый простой способ сделать это – уменьшить размер изображения.

Алгоритм «Simple Hash».

1. Уменьшить размер (рисунок 1.3). Изображение чаще всего уменьшается до 8 8 пикселей.

2. Убрать цвет (рисунок 1.4). Полученное изображение переводится в градации серого. Таким образом, хэш уменьшается втрое, т. е. вместо формата RGB теперь только градация серого.

3. Вычислить среднее значение цвета для всех пикселей в полученном изображении.

4. Построить цепочку битов (рисунок 1.5). Каждый пиксель заменяется на 1 или 0, в зависимости от того, меньше или больше среднего.

5. Построить сам хэш, т. е. перевести все биты в одно значение, вне зависимости от порядка, если он будет соблюден в дальнейшем.

Для сравнения двух изображений нужно построить хэш для каждого изображения и посчитать количество разных битов (расстояние Хемминга). Если расстояние равно 0, то изображения, скорее всего, идентичны, либо очень похожи.

Полученный хэш устойчив к манипуляциям с изображением, а именно сжатию, растягиванию, изменению яркости, контрастности, масштаба.

Однако существуют и недостатки – если изображение повернуть на достаточный угол, то полученный хэш будет существенно отличаться от первоначального. Поэтому этот метод не подходит для решения поставленной задачи.

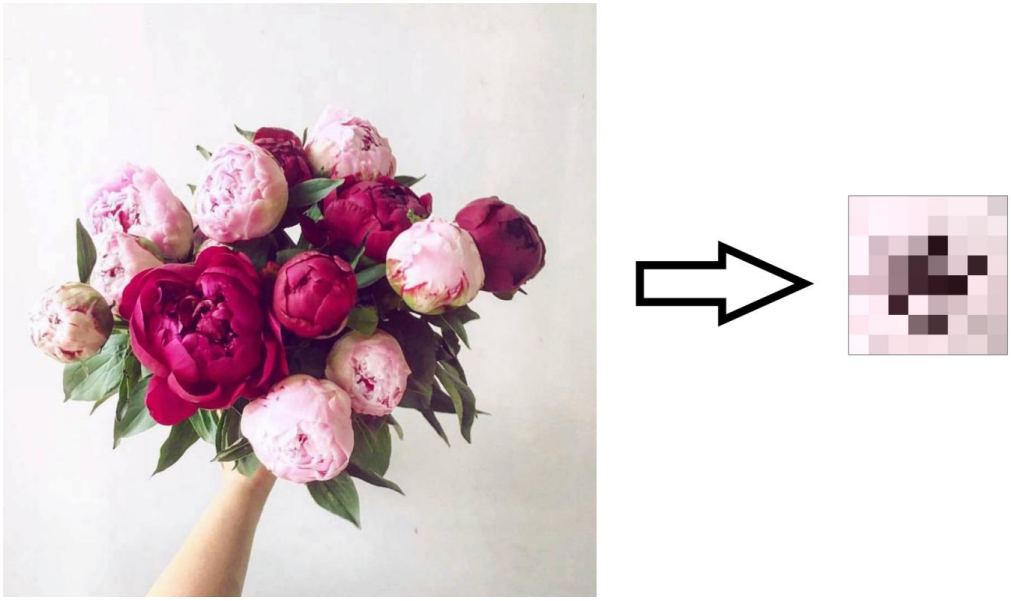

Рисунок 1.3 – Уменьшение размера

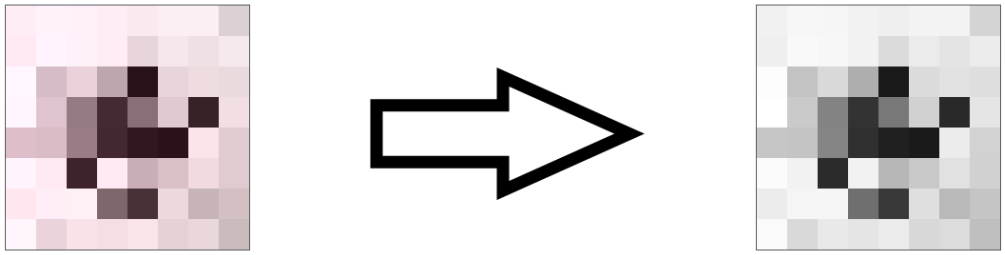

Рисунок 1.4 – Перевод изображения в градации серого

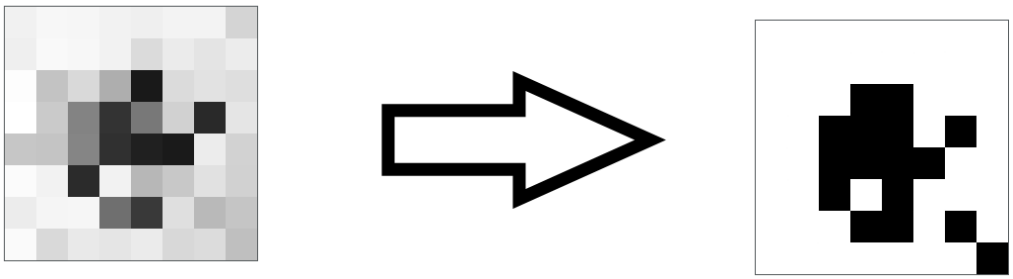

Рисунок 1.5 – Перевод изображения в черно-белый формат

1.1.3 SIFT

SIFT (Scale Invariant Feature Transform) – масштабно-инвариантная трансформация признаков [7]. Этот метод преобразует данные изображения в масштабно-инвариантные координаты относительно локальных объектов. Важным аспектом этого метода является то, что он генерирует большое количество объектов, которые плотно покрывают изображение во всем

диапазоне масштабов и координат. Ниже приведены основные этапы вычислений, используемые для создания набора функций изображения.

1. Обнаружение экстремумов в масштабном пространстве. Выполняется поиск по всем масштабам и координатам изображений. Он эффективно реализован с помощью функции разности Гауссов (difference-of-Gaussian) для определения потенциальных точек интереса, которые не зависят от масштаба и ориентации.

2. Локализация ключевых точек. В каждом подходящем месте детальная модель подходит для определения координаты и масштаба. Ключевые точки выбираются исходя из показателей их устойчивости.

3. Назначение ориентации. Одна или несколько ориентаций назначаются для каждой ключевой точки на основе локальных направлений градиента изображения. Все будущие операции выполняются с данными изображения, которые были преобразованы относительно назначенной ориентации, масштаба и координаты для каждой функции, тем самым обеспечивая неизменность этих преобразований.

4. Дескриптор ключевой точки. Локальные градиенты изображения измеряются в выбранном масштабе в области вокруг каждой ключевой точки. Они преобразуются в представление, которое учитывает значительные уровни локального искажения формы и изменения освещенности.

Этот метод инвариантен к смещению, повороту, масштабу и изменению яркости. Это именно те искажения, которые нужны для решения данной задачи.

1.1.4 SURF

SURF (Speeded-Up Robust Features) – предложенный в 2006 году алгоритм, который основан на тех же шагах, что и SIFT, но использует отличающуюся схему работы, что обеспечивает уменьшение времени работы [8].

Чтобы работать с этим алгоритмом, нужно представить изображение в интегральном формате. Этот формат предлагает использовать матрицу, чья размерность равна размерам изображения, а ее элементы вычисляются по формуле:

(1.5)

где – яркость пикселя изображения;

– координаты пикселя на матрице.

(1.6)

где – функция изменения градиента яркости.

(1.7)

Для поиска ключевых точек используется матрица Гессе (формула 1.6). Если ее определитель (формула 1.7) достигает своего экстремума, то градиент яркости максимально изменяется. Этот подход нужен для обнаружения контрастных переходов и краев. С помощью гессиана достигается инвариантность к повороту изображения, потому что оригинальное и повернутое изображение будут иметь одинаковый набор особых точек.

Результат работы – описание каждой особой точки в виде вектора, состоящего из чисел.

Этот метод инвариантен к смещению, повороту, масштабу, но при сильном изменении яркости он показывает результаты хуже.

1.1.5 ORB

ORB (Oriented FAST and Rotated BRIEF) – предложенная в 2011 году альтернатива SIFT и SURF, которая не уступает в качестве, но превосходит их в скорости вычисления [9].

В этом методе для нахождения особых точек используется алгоритм FAST (Features from Accelerated Segment Test) [10], который быстро детектирует локальные особенности изображения.

После определения особых точек определяется центроид и угол ориентации, которые используются в дескрипторе.

Для описания особых точек используется дескриптор BRIEF (Binary Robust Independent Elementary Features) [11]. В результате работы этого алгоритма каждая точка описывается в виде бинарного вектора.

Чтобы увеличить время, относительно SURF, метод ORB использует детектор углов Харриса, чтобы отсеять некоторое количество особых точек.

Для решения задачи будет выбран этот метод, потому что он инвариантен к необходимым искажениям и имеет меньшее время работы, чем SIFT и SURF.

1.2 Обзор программного обеспечения для распознавания идентичности изображений

1.2.1 Image Comparer

Программа может находить похожие и идентичные изображения из каталога, можно настроить порог схожести изображений. Если установить порог в 100%, то будут отображены только идентичные изображения. Из достоинств можно выделить неплохой функционал, интерфейс на русском языке, что немаловажно для российского потребителя. Из недостатков можно отметить неочевидный интерфейс, который может смутить неопытного пользователя, нужно сделать много неочевидных действий, чтобы программа начала работать так, как следует. Программа зависает, если работает с

большим количеством изображений. После некоторого количества обработанных изображений программа начинает работать не так, как работала до этого, перестает выполнять возложенные на нее функции. А также, программа очень плохо распознает повернутые изображения, что неприемлемо для решения задачи. Интерфейс представлен на рисунке 1.6 [12].

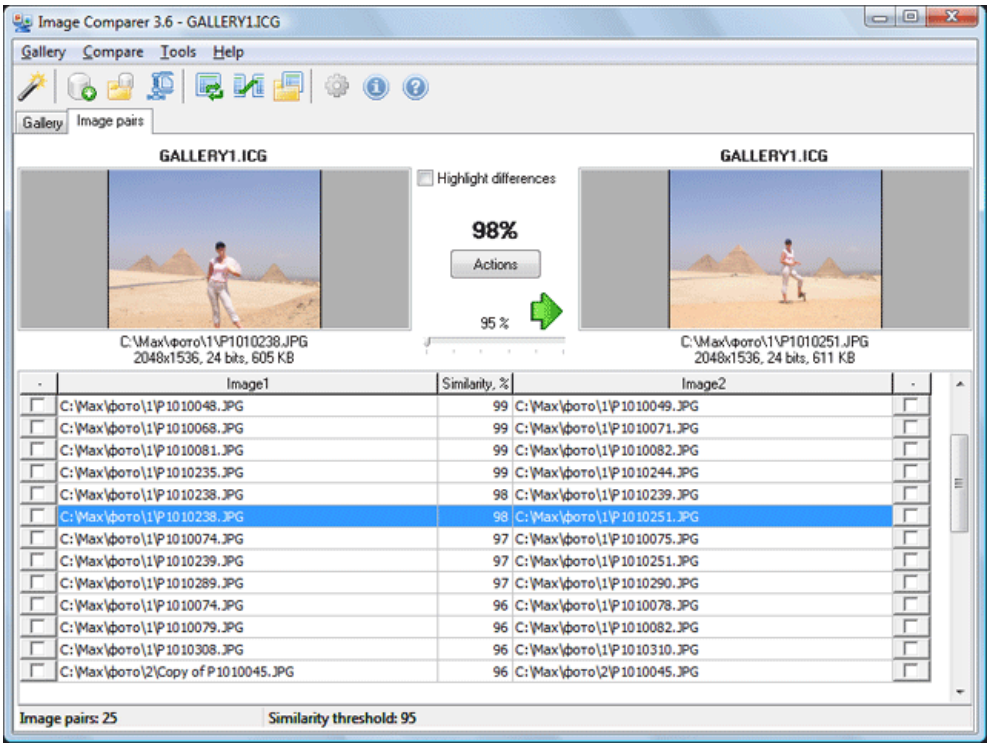

Рисунок 1.6 – Интерфейс программы «Image Comparer»

1.2.2 VisiPics

Это бесплатная программа, которая может находить и сразу удалять похожие и идентичные изображения, степень похожести пользователь также может регулировать, как и в предыдущей программе. У этого приложения имеется ряд преимуществ над Image Comparer, а именно бесплатное распространение, а также поиск в подкаталогах заданного каталога. Из недостатков можно выделить отсутствие русского языка, неочевидный интерфейс, а также плохая работа с зашумленными изображениями. Интерфейс представлен на рисунке 1.7 [13].

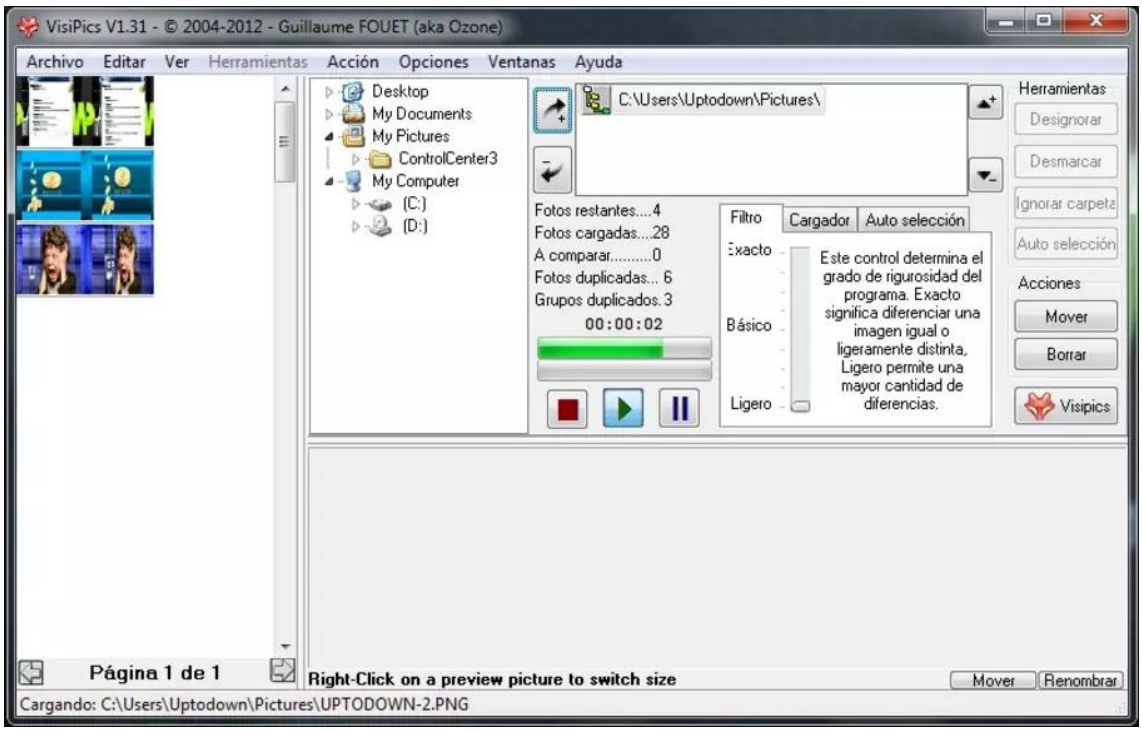

Рисунок 1.7 – Интерфейс программы «VisiPics»

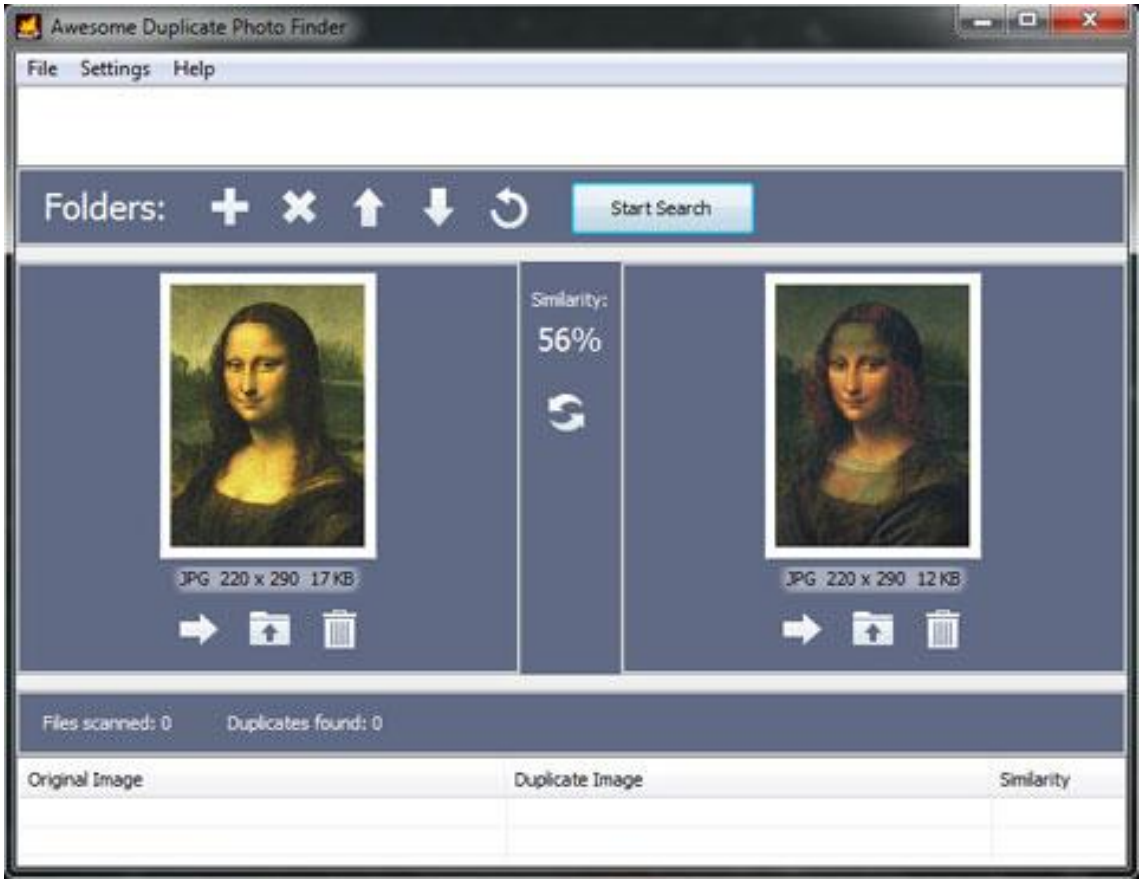

Рисунок 1.8 – Интерфейс программы «Duplicate Photo Finder»

1.2.3 Duplicate Photo Finder

Это одна из самых простых программ из списка. Выполняет точно такие же функции, что и Image Comparer, только с гораздо более понятным интерфейсом для неопытного пользователя. Недостатки: отсутствие русского языка, работа далеко не со всеми форматами изображений, очень плохая работа с обрезанными изображениями. Интерфейс представлен на рисунке 1.8 [14].

1.2.4 Similar Images Finder

Это платное приложение, которое поддерживает 29 форматов изображения. Выполняет аналогичные функции, что и предыдущие программы. Однако есть существенный недостаток – при поиске дубликатов приложение показывает и отличающиеся друг от друга изображения. Интерфейс представлен на рисунке 1.9 [15].

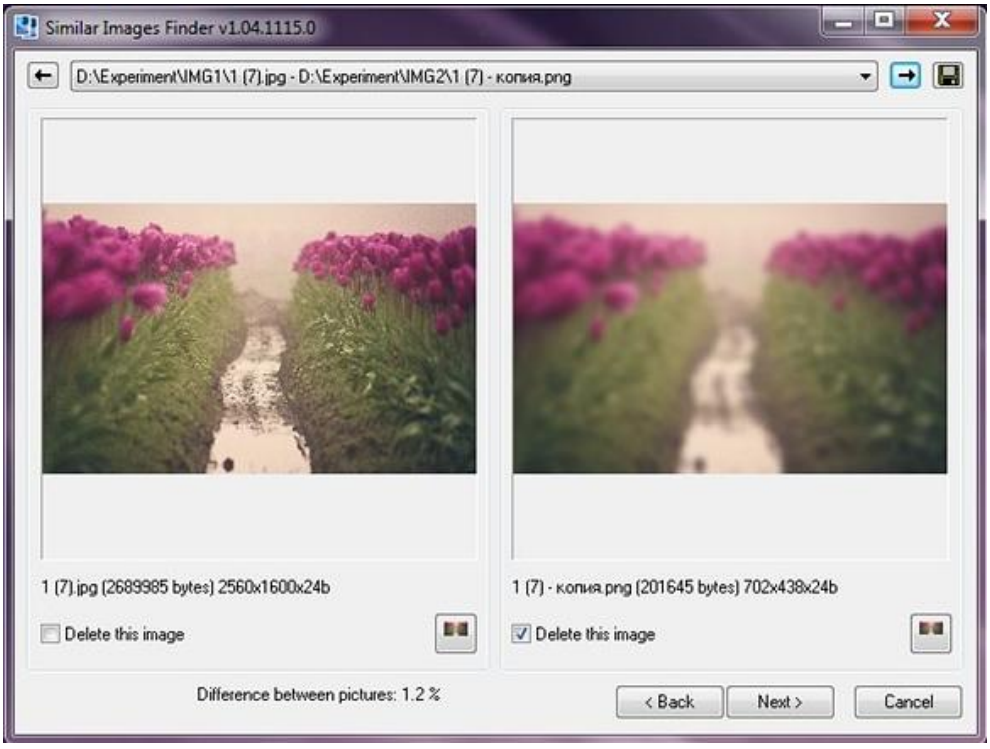

Рисунок 1.9 – Интерфейс программы «Similar Images Finder»

1.2.5 AntiDupl

Это одно из самых лучших решений для поиска похожих и идентичных изображений. Отличает его от остальных возможность выбора алгоритма сравнивания изображений, поддержка достаточно большого количества форматов, а также наличие русского языка в интерфейсе приложения. Программа распространяется на бесплатной основе. Недостатки: выбор из всего двух алгоритмов сравнения, оба из которых показывают довольно сильную погрешность в работе. Интерфейс представлен на рисунке 1.10 [16].

В конечном итоге, после разбора существующего программного обеспечения, не нашлось универсального приложения, которое будет точно решать поставленную задачу.

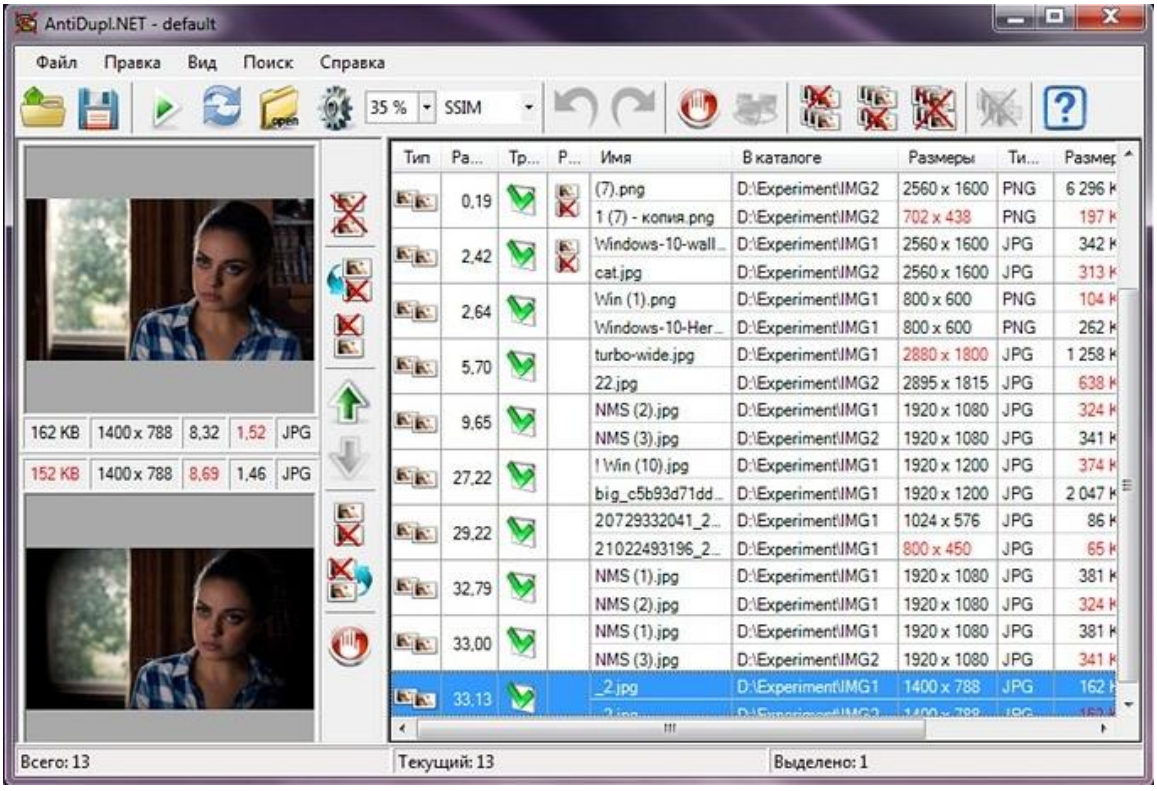

Рисунок 1.10 – Интерфейс программы «AntiDupl»

1.3 Постановка задачи

Необходимо исследовать алгоритмы и реализовать приложение, которое будет искать идентичные заданному изображению в подкаталогах, с

учетом искажающих факторов, а именно поворот, сдвиг, масштабирование, смена яркости или контрастности, наложение шума.

1.4 Выводы по разделу

В данной главе рассмотрены основные методы решения поставленной задачи, а именно метод цветовых гистограмм, перцептивные хэш-алгоритмы и методы SIFT, SURF и ORB.

Метод цветовых гистограмм не подходит для решения, потому что в этом методе оценивается только цветовой диапазон изображений. Этот метод будет устойчив к повороту, но изменение контрастности или обрезка изображения нанесет существенный ущерб распознаваемости, поэтому этот метод не подходит.

Метод перцептивных хэш-алгоритмов также не решает поставленную задачу, т. к. при повороте или обрезки фотографии хэш будет существенно отличаться от изображения-оригинала, что, в свою очередь, не будет решать задачу.

Для решения поставленной задачи подходят методы SIFT, SURF и ORB, но для решения задачи будет использоваться метод ORB, т. к. он работает быстрее метода SIFT и SURF.

В данной главе рассмотрены множество программ, которые разработаны для поиска дубликатов изображений, но каждая из них не подходит по тем или иным причинам. Большинство представленных программ не работает с повернутыми и зашумленными изображениями, а также все представленные программы имеют избыточный функционал, вследствие чего имеют непонятный рядовому пользователю интерфейс, в котором нужно разбираться некоторое время.

2 МАТЕМАТИЧЕСКАЯ МОДЕЛЬ РАСПОЗНАВАНИЯ ИДЕНТИЧНОСТИ ИЗОБРАЖЕНИЙ С УЧЕТОМ ИСКАЖЕНИЙ

2.1 Математическая модель распознавания идентичности изображений

2.1.1 Модель поиска особых точек

Для принятия решения о том, считать ли заданную точку C особой или нет, в этом методе рассматривается яркость пикселей на окружности с центром в точке C и радиусом R (рисунок 2.1) [10].

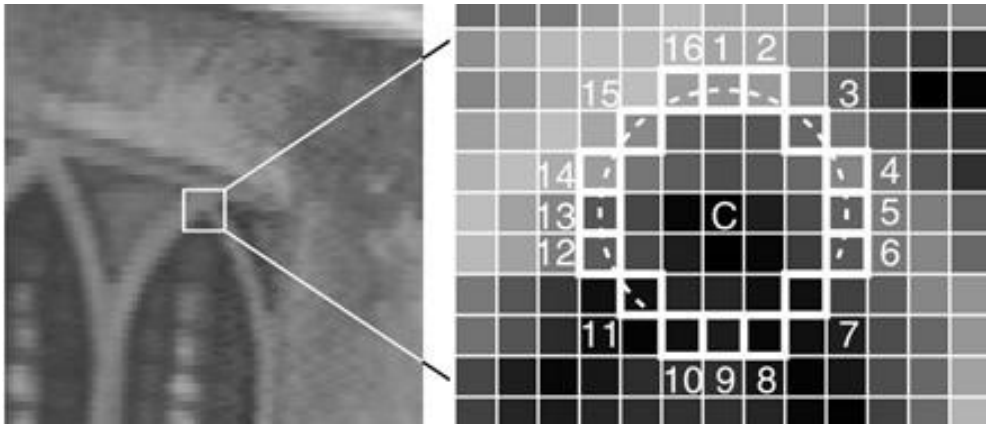

Рисунок 2.1 – Окружность с центром в C и радиусом 3

Сравнивая яркости пикселей, можно получить один из трех исходов:

(2.1)

(2.2)

(2.3)

где – яркость пикселей;

– некоторый заранее фиксированный порог по яркости.

Точка является особой, если на круге существует 3/4 от возможных пикселей, которые удовлетворяют формулам 2.1 и 2.2. Для того чтобы ускорить процесс, можно проверить только четыре пикселя с номерами 1, 5, 9, 13 (рисунок 2.1). Если среди них есть минимум 3 пикселя светлее или

темнее центра, то выполняется полная проверка, иначе – точка сразу становится не особой.

Определение моментов яркости:

(2.4)

где – координаты пикселя;

– интенсивность пикселя.

Имея моменты, можно получить центроид ориентации:

(2.5)

Угол ориентации:

(2.6)

В данном виде алгоритм будет очень долго обрабатывать изображения, поэтому он будет модифицирован. Введем ограничение в 1000 ключевых точек, для данной задачи этого будет достаточно.

2.1.2 Модель определения дескрипторов особых точек

(2.7)

где – интенсивность в точке ;

– интенсивность в точке .

(2.8)

где .

Алгоритм:

– выбирается область, центрированная относительно центроида ориентации (формула 2.5) особой точки;

– из области выбирается некоторым образом множество пар пикселей и , для которых строится набор бинарных тестов (формула 2.7);

– для каждой области выбирается множество, содержащее 256 пар точек, которые однозначно определяют набор бинарных тестов. На основании этих тестов строится бинарная строка (формула 2.8).

2.2 Математическая модель искажения изображения

2.2.1 Аффинные преобразования

Общий вид аффинных преобразований в матричном виде имеет вид:

(2.9)

где – новые координаты пикселя;

– старые координаты пикселя;

– коэффициенты изменения.

Формула переноса:

(2.10)

где – количество пикселей по оси и оси переноса.

Сдвиг по оси :

(2.11)

где – количество пикселей сдвига по оси .

Сдвиг по оси :

(2.12)

где – количество пикселей сдвига по оси .

Масштабирование:

(2.13)

где – количество пикселей по оси и оси масштабирования.

Вращение:

(2.14)

где – угол поворота.

2.2.2 Изменение яркости

Если изображение представлено в цветовой модели RGB, то формула изменения яркости будет иметь вид:

(2.15)

где – новые значения RGB пикселя;

– старые значения RGB пикселя;

– коэффициент яркости. Если , то изображение осветляется, если , то изображение затемняется.

2.2.3 Перевод изображения в градации серого

При переводе изображения в градации серого, каждый пиксель теряет свой цвет, зато приобретает интенсивность.

(2.16)

где – интенсивность пикселя;

– старые значения RGB пикселя.

2.3 Выводы по разделу

В данном разделе математически разобраны методы поиска особых точек FAST (Features from Accelerated Segment Test) и определения дескрипторов особых точек BRIEF (Binary Robust Independent Elementary Features) из которых состоит алгоритм ORB. Он был модифицирован для увеличения скорости работы.

В данном разделе приведены основные формулы искажения изображения.

3 РАЗРАБОТКА АЛГОРИТМОВ И ПРИЛОЖЕНИЯ

3.1 Алгоритм работы распознавателя идентичности изображений

Чтобы начать использовать алгоритм распознавания, нужно сначала подготовить изображение-оригинал и потенциально искаженное изображение к поиску особых точек. Предобработка заключается в переводе изображения в градации серого (grayscale) по формуле 2.16.

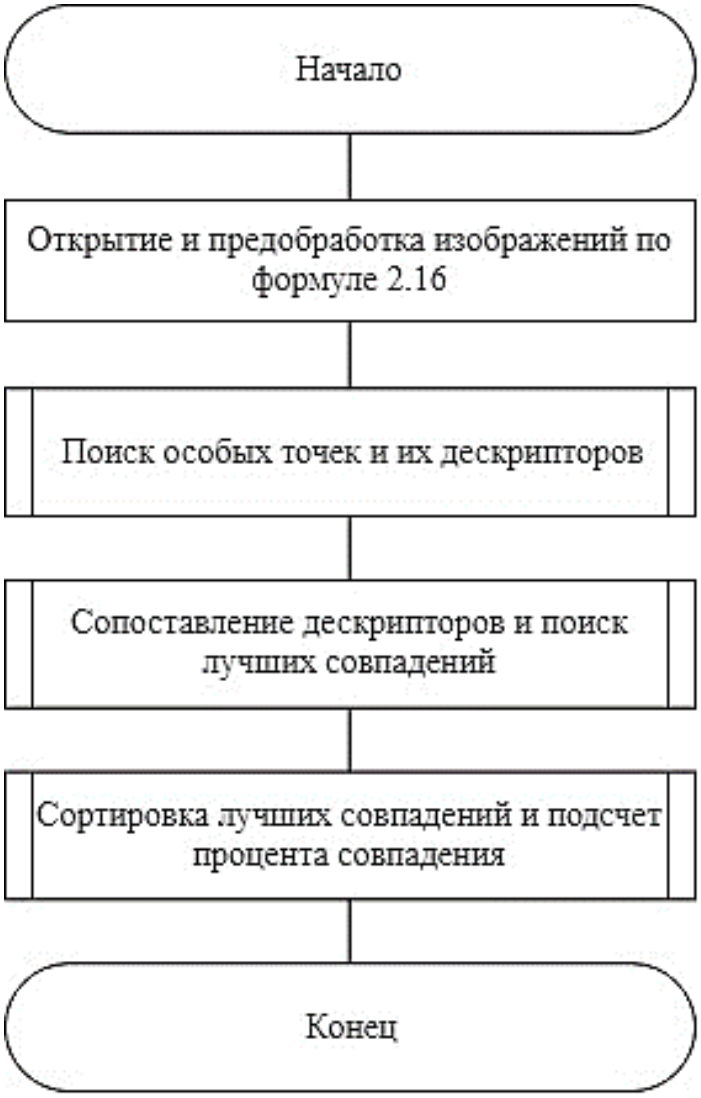

Рисунок 3.1 – Алгоритм работы распознавателя идентичности изображения

Далее происходит поиск особых точек и дескрипторов с помощью алгоритма ORB. Устанавливается ограничение в 1000 особых точек, чтобы уменьшить время работы алгоритма. После этого с помощью алгоритма Brute-Force Matcher из библиотеки OpenCV, который берет дескриптор

одного изображения и сопоставляет его со всеми дескрипторами другого изображения с использованием вычисления расстояния Хемминга, происходит поиск лучших совпадений, т.е. пар, между которыми будет наименьшее расстояние. Потом происходит сортировка этих расстояний по возрастанию с помощью стандартной функции sorted, которая определена в языке Python, и подсчет количества расстояний, которые прошли порог. Алгоритм представлен на рисунке 3.1.

Рисунок 3.2 – Алгоритм поиска особых точек и дескрипторов

3.1.1 Вспомогательный алгоритм поиска особых точек и дескрипторов

Происходит поиск особых точек из всех точек выбранного изображения. После нахождения особой точки определяется ее дескриптор. Если было найдено 1000 особых точек, тогда их поиск прекращается.

3.2 Описание интерфейса программы

После запуска программы появляется главное окно, содержащее всплывающие меню: «Файл», «Справка» и кнопки: «Выбрать каталог», «Выбрать изображение», «Найти» (рисунок 3.3).

При нажатии на всплывающее меню «Файл» появятся подпункты меню: «Выбрать изображение», «Выбрать каталог поиска», «Очистить все», «Выход» (рисунок 3.4).

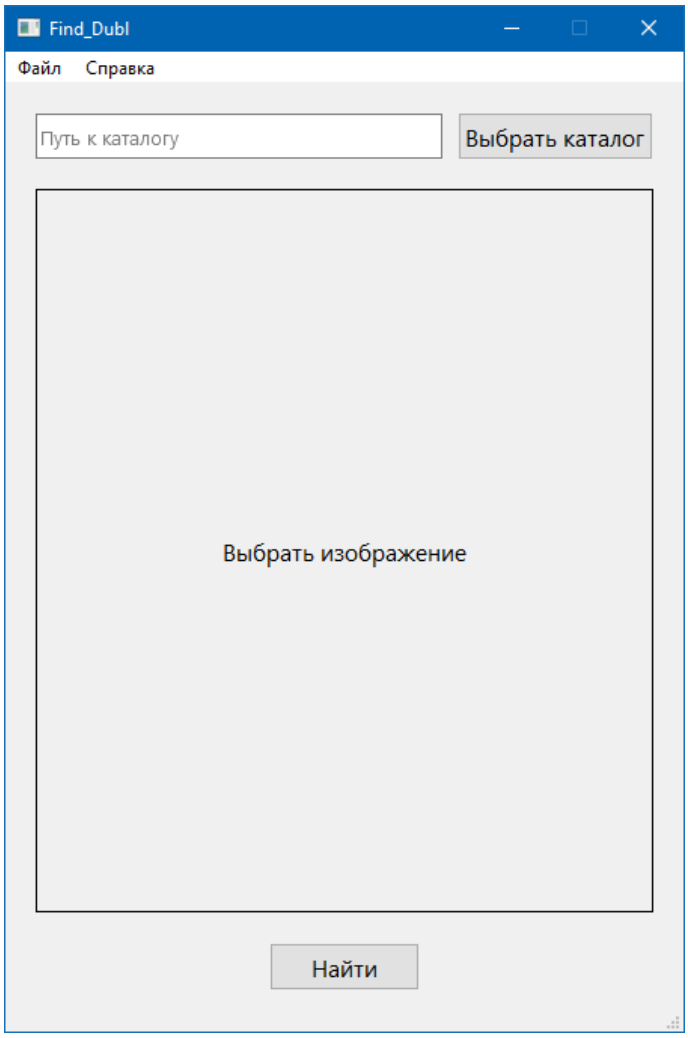

Рисунок 3.3 – Главное окно программы

При нажатии на подпункт «Выход» произойдет выход из программы.

При нажатии на подпункт «Очистить все» поле «Путь к каталогу» и место для изображения главного окна будут очищены.

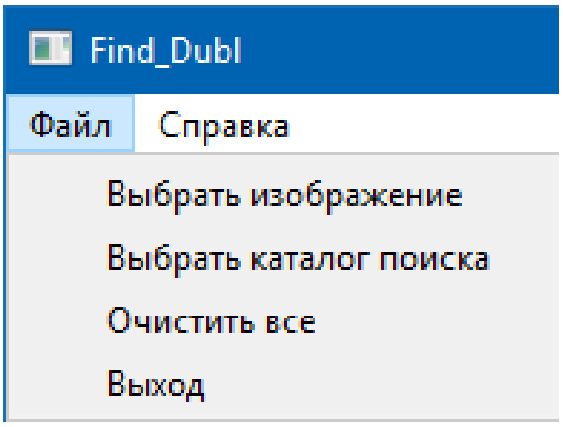

Рисунок 3.4 – Всплывающее меню «Файл»

При нажатии на подпункт «Выбрать каталог поиска», либо на кнопку на главном экране «Выбрать каталог», можно будет выбрать каталог, в котором будет происходить поиск. Выбранный каталог будет показан в поле главного окна «Путь к каталогу». А также пользователь может сам написать путь в поле главного окна «Путь к каталогу» (рисунок 3.5).

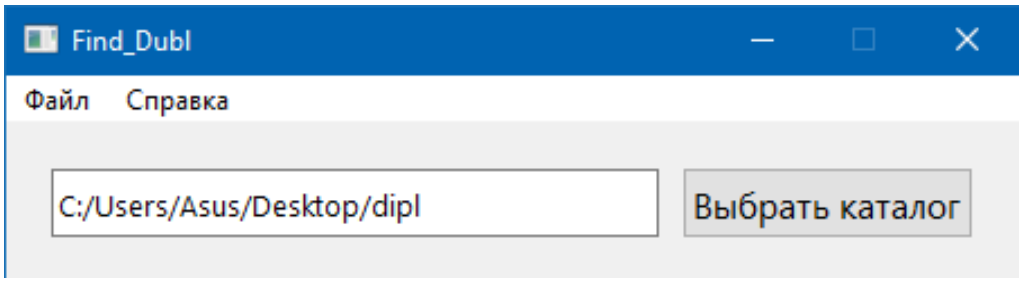

Рисунок 3.5 – Выбор каталога

При нажатии на подпункт меню «Выбрать изображение», либо на кнопку на главном экране с таким же названием, можно будет выбрать изображение, чей дубликат нужно найти. Выбранное изображение будет отображено в центре главного экрана (рисунок 3.6).

При нажатии на всплывающее меню «Справка» появятся подпункты меню: «Справка», «О программе» (рисунок 3.7).

При нажатии на подпункт «Справка» можно узнать, как пользоваться программой.

При нажатии на подпункт «О программе» можно узнать информацию о разработчике и программе.

Рисунок 3.6 – Выбор изображения

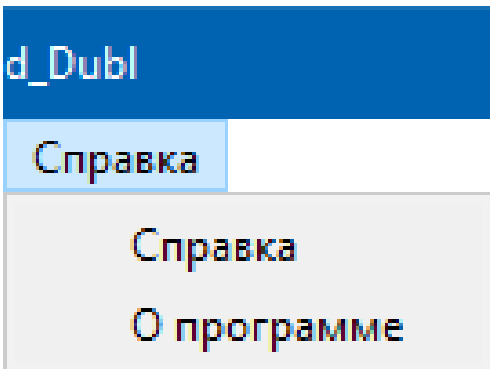

Рисунок 3.7 – Всплывающее меню «Справка»

При нажатии на кнопку главного экрана «Найти», если не выбран каталог поиска или изображение, то будет показано соответствующее предупреждение (рисунок 3.8), если же и каталог, и изображение выбраны, то начнется поиск дубликатов (рисунок 3.9).

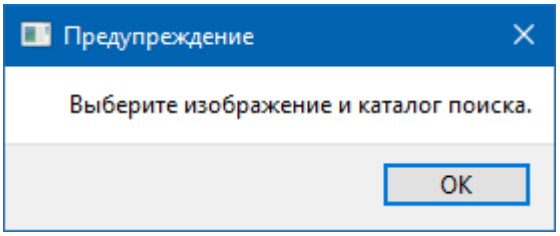

Рисунок 3.8 – Предупреждение

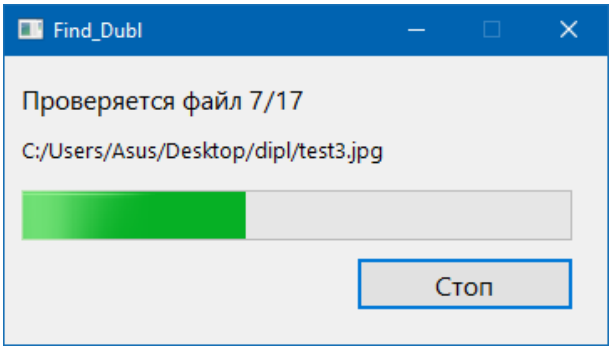

Рисунок 3.9 – Процесс поиска

Во время поиска можно нажать кнопку «Стоп», тогда поиск будет остановлен (рисунок 3.10), и при нажатии «ОК» можно будет увидеть результат проверки до того момента, пока не была произведена остановка (рисунки 3.12, 3.13).

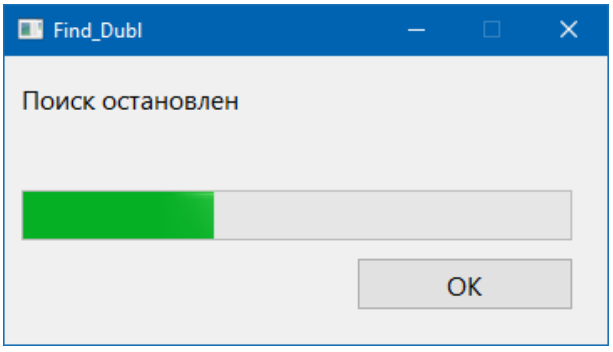

Рисунок 3.10 – Остановка поиска

Если же поиск не останавливать, то как только все файлы будут проверены, появится соответствующая надпись (рисунок 3.11). При нажатии

кнопки «ОК» можно будет увидеть результат проверки (рисунки 3.12, 3.13). Результатом проверки является путь к изображению-дубликату и его миниатюра.

При нажатии кнопки «ОК» результирующего окна, оно закроется.

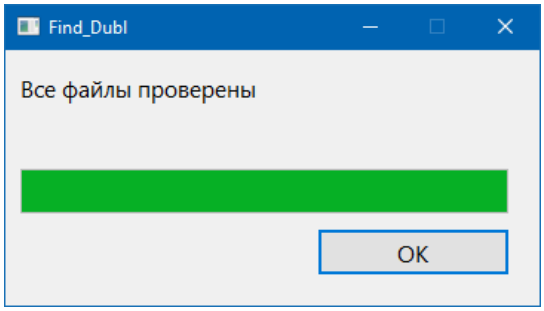

Рисунок 3.11 – Конец поиска

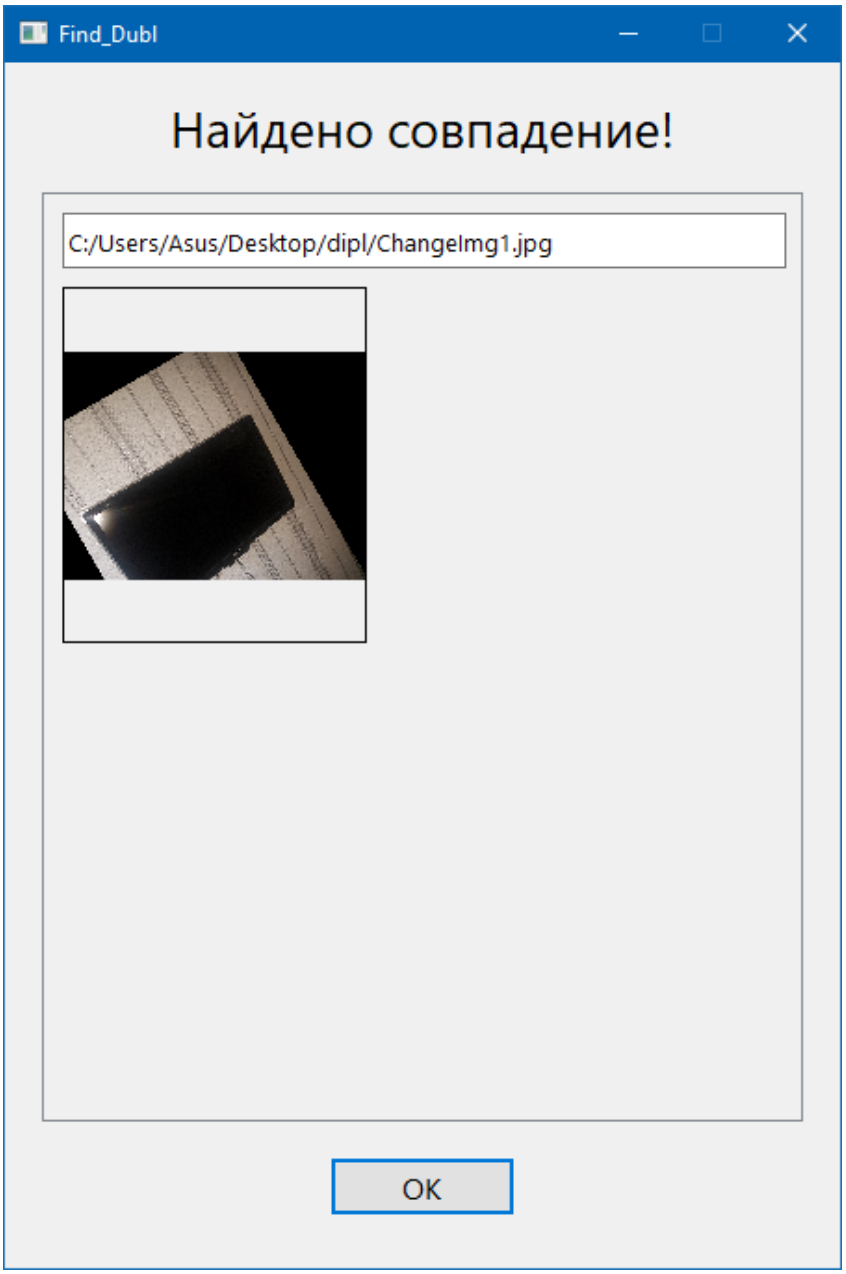

Рисунок 3.12 – Положительный результат поиска

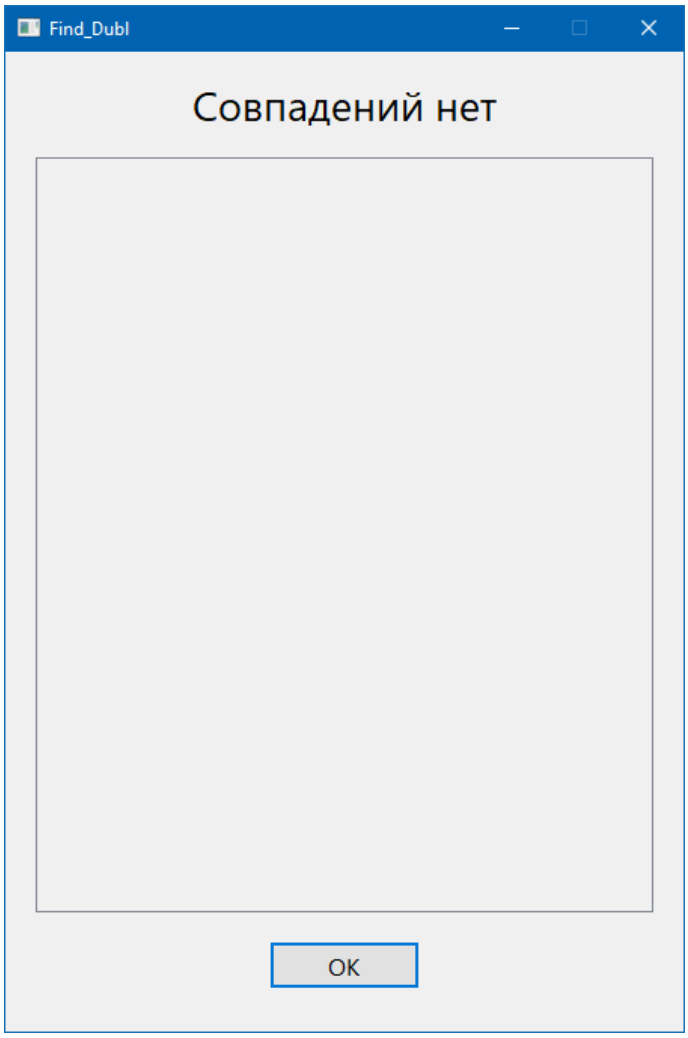

Рисунок 3.13 – Отрицательный результат поиска

3.3 Выводы по разделу

В данном разделе представлены алгоритмы распознавателя идентичности изображений.

В данном разделе разработан интерфейс программы, которая позволяет пользователю выбрать изображение и каталог и найти искаженные дубликаты выбранного изображения, которые будут выведены в список, в котором присутствуют миниатюра и путь к файлу.

4 МЕТОДИКА ЭКСПЕРИМЕНТАЛЬНОГО ИССЛЕДОВАНИЯ РЕЗУЛЬТАТОВ РАБОТЫ ПРОГРАММЫ

4.1 Исходные данные

Исходными данными является реальная база фотографий объектов до и после планово-предупредительных ремонтов (ППР), которые разбиты по годам, станциям и объектам. Фотографии имеют размер пикселей, где и , чаще всего, различаются.

4.2 Критерии оценки искажений

Чтобы проверить, насколько искажения влияют на распознавание, была взята реальная фотография объекта (рисунок 4.1), которая была искажена с помощью специально написанной программы, которая выполняет изменение изображения в зависимости с выбранным искажением и сравнивает его с оригинальным изображением.

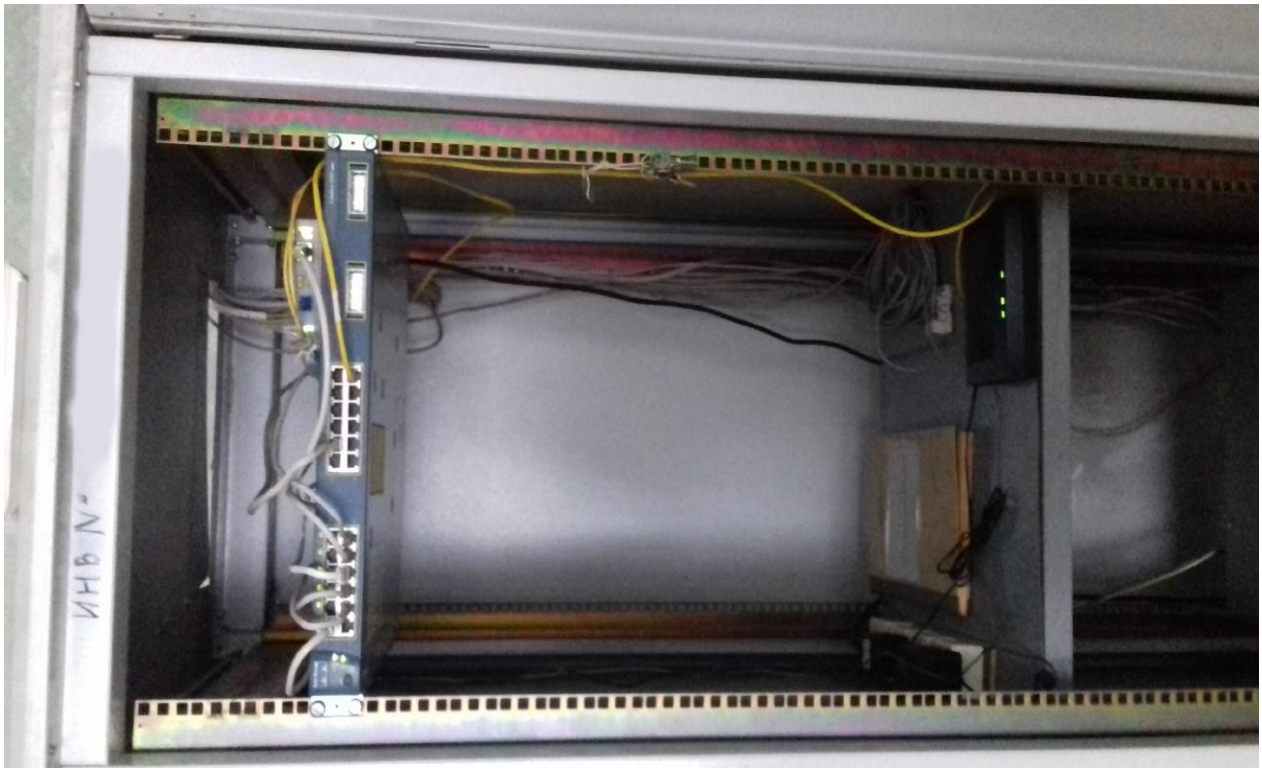

Рисунок 4.1 – Фотография объекта

4.2.1 Увеличение в центр

Изображение постепенно увеличивалось к центру, и чем больше оно увеличивалось, тем меньше особых точек совпадало с оригинальным изображением (рисунок 4.2). Для данного теста распознавание прекратилось при 50% увеличении.

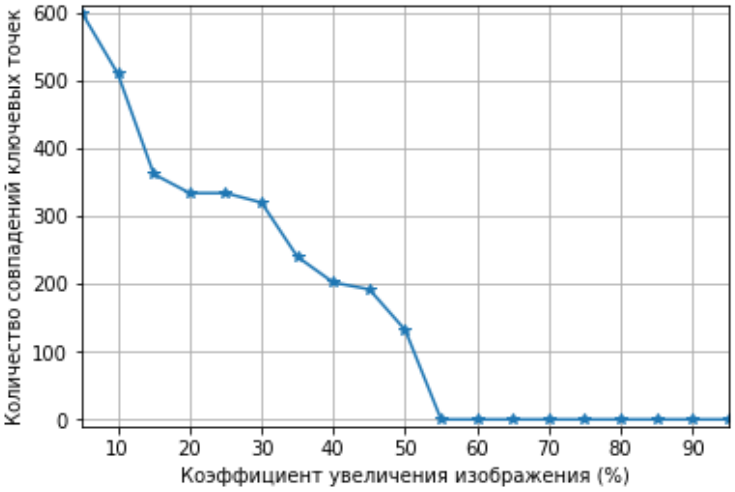

Рисунок 4.2 – Увеличение к центру

4.2.2 Увеличение в левый верхний угол

Изображение постепенно увеличивалось к левому верхнему углу, и чем больше оно увеличивалось, тем меньше особых точек совпадало с оригинальным изображением (рисунок 4.3). Для данного теста распознавание прекратилось при 80% увеличении.

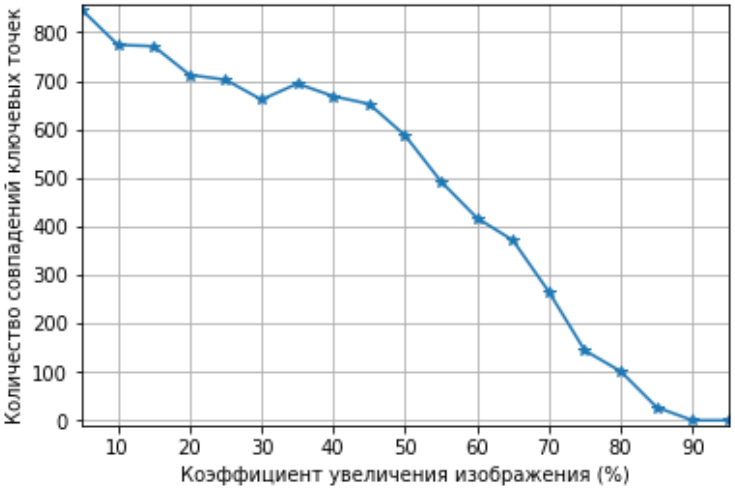

Рисунок 4.3 – Увеличение к левому верхнему углу

4.2.3 Смещение вниз

Изображение постепенно смещалось вниз, и чем больше оно смещалось, тем меньше ключевых точек совпадало с оригинальным изображением (рисунок 4.4). Для данного теста распознавание прекратилось при 70% смещении.

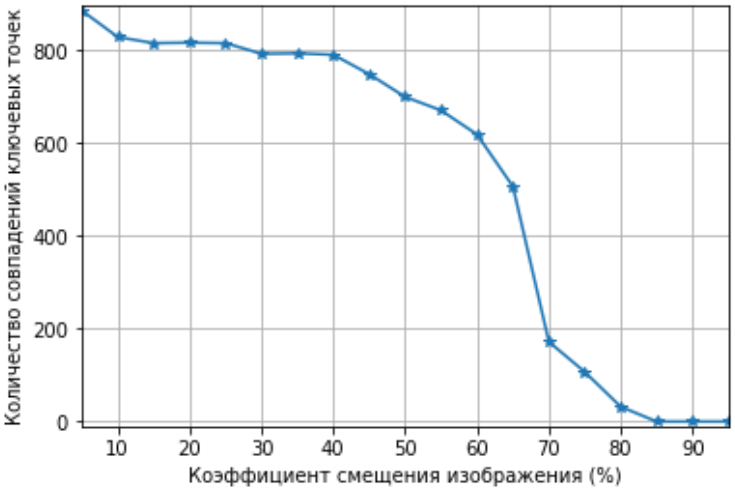

Рисунок 4.4 – Смещение вниз

4.2.4 Смещение вправо

Изображение постепенно смещалось вправо, и чем больше оно смещалось, тем меньше ключевых точек совпадало с оригинальным изображением (рисунок 4.5). Для данного теста распознавание прекратилось при 85% смещении.

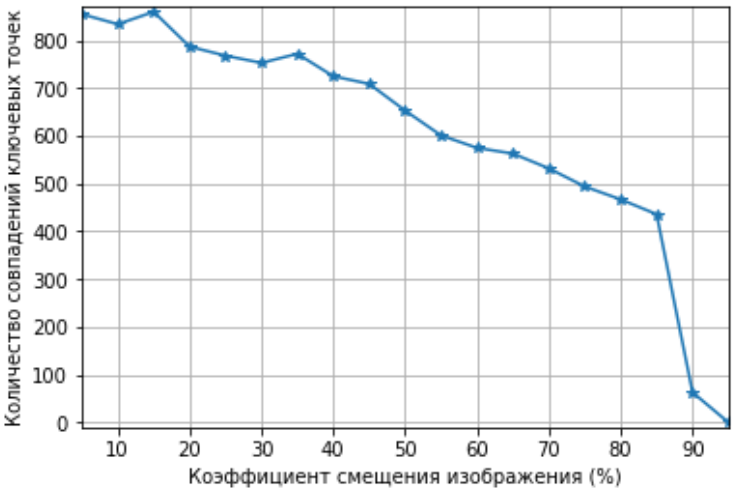

Рисунок 4.5 – Смещение вправо

4.2.5 Поворот

Изображение постепенно поворачивалось вокруг своего центра. Во время полного поворота, изображение всегда распознавалось (рисунок 4.6).

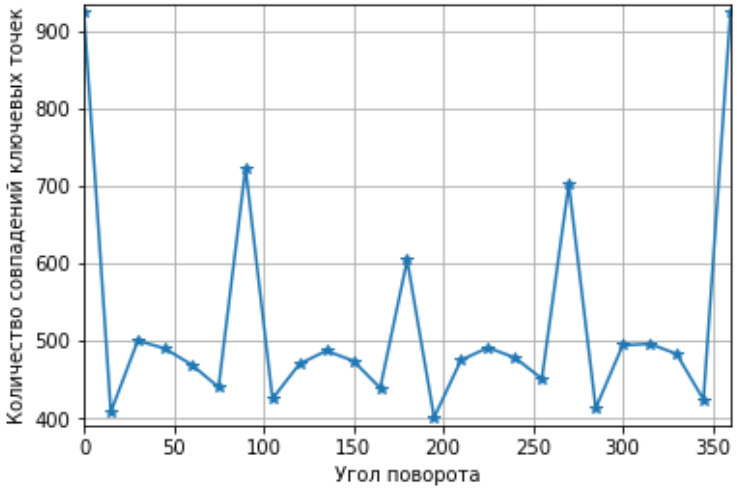

Рисунок 4.6 – Поворот

4.2.6 Яркость

У изображения изначально была уменьшена яркость, и она постепенно увеличивалась. Для данного теста уменьшение яркости особо не повлияло на распознавание, однако увеличение яркости существенно повлияло на него (рисунок 4.7).

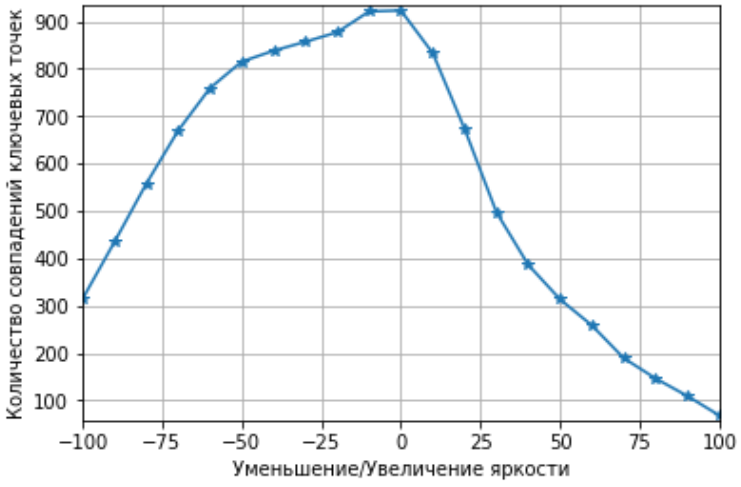

Рисунок 4.7 – Яркость

4.3 Оценка качества работы алгоритма

Было взято 100 случайных изображений из базы и 100 искажений этих изображений, т.е. у каждого оригинала есть свое искажение.

Для оценки качества была взята формула:

—— , (4.1)

где – сумма правильных ответов;

– сумма всех тестов;

Сначала нужно проверить, как алгоритм справляется с неискаженными изображениями. Для этого каждое неискаженное изображение сравнивалось с каждым неискаженным, кроме себя самого. Получается тестов. Из них только 4 теста были неправильными. Значит, оценка качества работы алгоритма с неискаженными изображениями равна 99,9%.

Чтобы проверить работу алгоритма с искаженными изображениями, нужно каждое искаженное изображение сравнить со всеми неискаженными. Тест считается успешным, если алгоритм определил изображение-оригинал по искаженному изображению. Всего тестов , из них только 11 тестов были неправильными, значит, оценка качества работы алгоритма с искаженными изображениями равна 99,8%.

Суммарная оценка качества работы алгоритма равна 99,9%, что является доверительным показателем.

4.4 Проверка работы программы на экспериментальных данных

4.4.1 Первый пример работы программы

На рисунке 4.8 представлена работа программы, слева выбрано изображение, которое является уникальным в выбранном каталоге, справа результат. Как и ожидалось, совпадений нет.

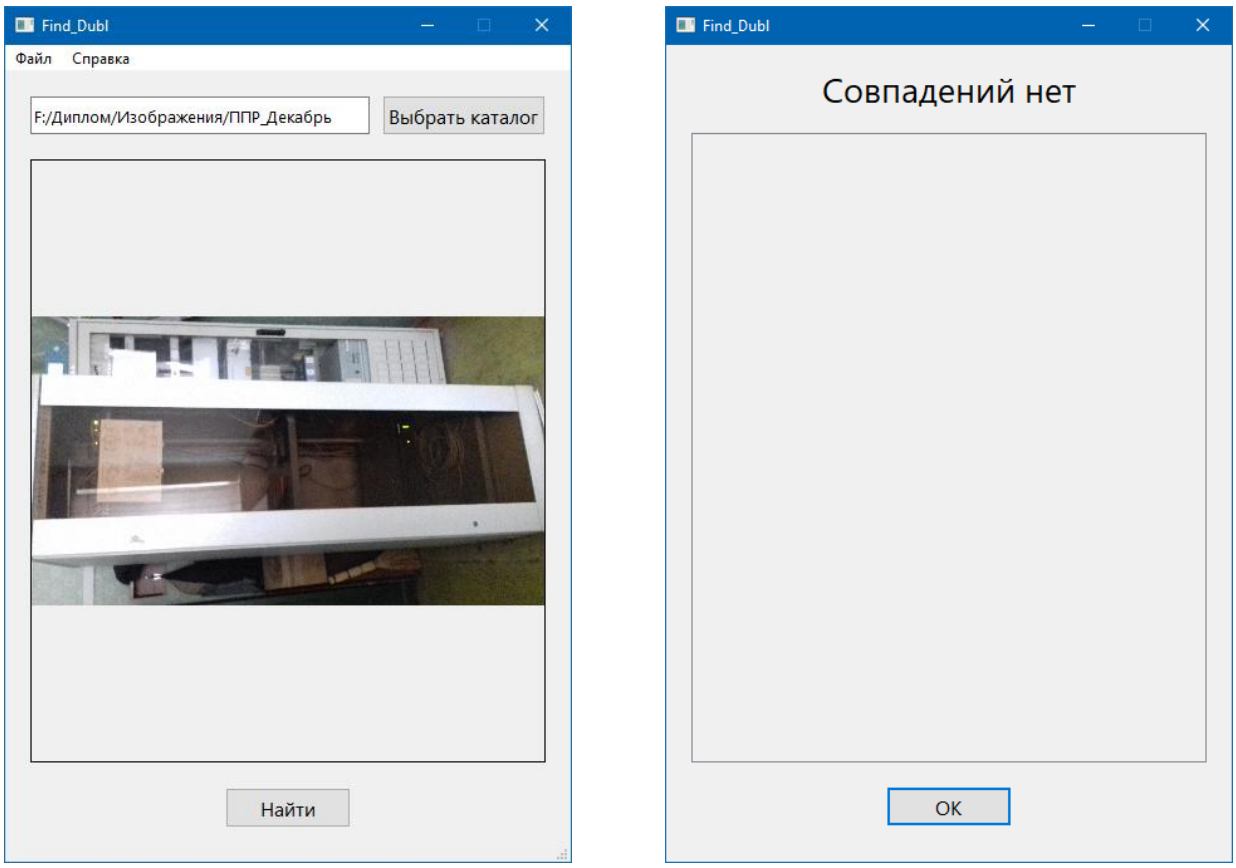

Рисунок 4.8 – Пример работы программы

4.4.2 Второй пример работы программы

На рисунке 4.9 слева изображена искаженная фотография, а справа оригинал. На рисунке 4.10 представлена работа программы слева выбрано изображение, которое является искаженным, а его оригинал находится в выбранном каталоге. Совпадение найдено.

Рисунок 4.9 – Тестируемые изображения

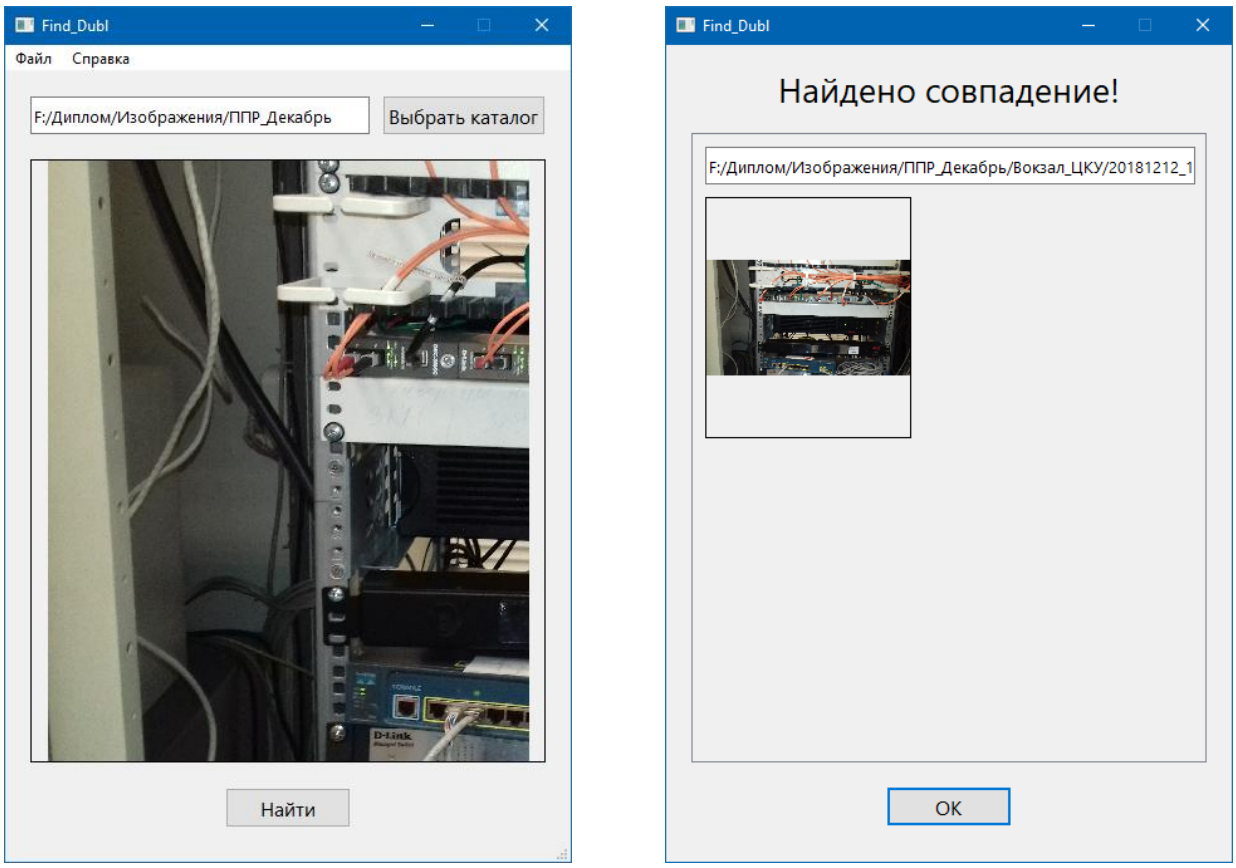

Рисунок 4.10 – Пример работы программы

4.5 Выводы по разделу

В данном разделе разобрано, как именно искажения влияют на распознавание оригинального изображения по его искаженному дубликату. По результатам исследования сделан вывод, что для правильного распознавания достаточно совпадения 100 особых точек. Проведена оценка работы алгоритма из 19900 тестов, которые показали, что с вероятностью 99,925% алгоритм срабатывает верно, что является довольно высоким показателем. А также проведена проверка работы программы на 2-х экспериментальных изображениях, оба теста были пройдены верно.

ЗАКЛЮЧЕНИЕ

Данная работа посвящена определению идентичности двух изображений с учетом искажающих факторов. За основу взят алгоритм, основанный на поиске особых точек, ORB. Разработана программа, которая реализует данный алгоритм.

Разработанная программа позволяет пользователю выбрать изображение и каталог, где нужно найти дубликат. Затем, после подтверждения, будет происходить поиск и после его окончания можно посмотреть результаты поиска в виде списка с миниатюрой дубликата и пути к файлу.

В результате работы были решены следующие задачи:

1) изучены и проанализированы методы, применяющиеся для распознавания идентичности изображений;

2) разработана математическая модель проверки идентичности изображений;

3) разработана компьютерная программа, реализующая разработанную модель;

4) проведена проверка работы программы на экспериментальных данных.

Разработанная программа имеет ряд преимуществ над программами поиска дубликатов:

– поиск повернутых изображений;

– поиск зашумленных изображений;

– поиск изображений, искаженных несколькими искажениями одновременно;

– время работы существенно меньше, чем у некоторых конкурентов;

– простой и понятный интерфейс.

В дальнейшем планируется продолжение работы и решение следующих задач:

– увеличение скорости работы;

– добавление других методов распознавания идентичности изображений, чтобы увеличить точность распознавания;

– возможность пользователю выбирать между методами;

– возможность пользователю выбирать новый режим работы – поиск всех искаженных дубликатов на компьютере.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1 Огневой, Г.Д. Методы и алгоритмы поиска изображений / Г.Д. Огневой // Информационные технологии в образовании, науке и производстве – 2014. – 34 с.

2 Сравнение изображений. Image retrieval. – Дата обновления: 14.03.2010. URL: http://ccv-dinalt.blogspot.com/ (дата обращения: 09.02.2019).

3 Rubner, Y.A metric for distributions with applications to image databases / Y. Rubner, C. Tomasi, L.J. Guibas // Proceedings ICCV – 1998. – P. 138–142.

4 Парасич, А.В. Методы на основе цветовых гистограмм в задачах обработки изображений / А.В. Парасич, В.А. Парасич // Nauka-rastudent.ru. – URL: http://naukarastudent.ru/18/2742 – 2015. – № 06 (18). – P. 28–30.

5 Looks Like It. – Дата обновления: 26.05.2011. URL: www.hackerfactor.com/blog/index.php?/archives/432-Looks-Like-It.html (дата обращения: 10.02. 2019).

6 Как бороться с репостами или пара слов о перцептивных хэшах. – Дата обновления: 18.09.2014. URL: https://habr.com/ru/post/237307 (дата обращения: 10.02.2019).

7 Lowe, D. Distinctive Image Features from Scale-Invariant Keypoints / D. Lowe // International Journal of Computer Vision – 2004. – № 60 (2) – P. 91–110.

8 Bay, H. SURF: Speeded Up Robust Features / H. Bay, A. Ess, T. Tuytelaars, L. Van Gool // Computer Vision and Image Understanding (CVIU) – 2008. – V. 110, №3. – P. 346–359.

9 Rublee, E. ORB: an efficient alternative to SIFT or SURF / E. Rublee, V. Rabaud, K. Ronolige, G. Bradski // IEEE International Conferemce on Computer Vision – 2011. – P. 10.

10 Drummond, T. Fusing Points and Lines for High Performance Tracking / T. Drummond, E. Rosten // Proceeding of the Tenth IEEE International Conference on Computer Vision – 2005. – P. 54.

11 Calonder, M. BRIEF: Binary Robust Independent Elementary Features / M. Calonder, V. Lepetit, C. Strecha, P. Fua // In European Conference on Computer Vision – 2010. – P. 5–15.

12 Image Comparer official page. – Дата обновления: 01.02.2019. URL: ht tp://www.imagecomparer.com/rus (дата обращения: 11.02.2019).

13 VisiPics official page. – Дата обновления: 15.04.2015. URL: http://ww w.visipics.info/index.php?title=Main_Page (дата обращения: 11.02.2019).

14 Dublicate Photo Finder. – Дата обновления: 17.06.2018. URL: https://w ww.duplicate-finder.com/photo.html (дата обращения: 11.02.2019).

15 Поиск дубликатов фотографий. – Дата обновления: 23.01.2019. URL: http://www.itrew.ru/programs/poisk-dublikatov-fotografiy-sravnenie-6-programm. html (дата обращения: 11.02.2019).

16 AntiDupl official page. – Дата обновления: 09.01.2018. URL: http://anti dupl.sourceforge.net/english/index.html (дата обращения: 12.02.2019).

ПРИЛОЖЕНИЕ 1

Исходный код программы

Файл int.py

```
# -*- coding: utf-8 -*-

from PyQt5 import QtCore, QtGui, QtWidgets
from cv2 import cv2
from PIL import Image
import os
import numpy as np

# ================================================================
class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        font = QtGui.QFont()
        font.setFamily("Segoe UI")
        font.setPointSize(12)
        font.setBold(False)
        font.setItalic(False)
        font.setWeight(50)

        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(450, 650)
        MainWindow.setMinimumSize(QtCore.QSize(450, 650))
        MainWindow.setMaximumSize(QtCore.QSize(450, 650))
        MainWindow.setSizeIncrement(QtCore.QSize(0, 0))
        MainWindow.setWindowTitle("Find_Dubl")
        MainWindow.setFont(font)
        MainWindow.setAnimated(True)

        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        MainWindow.setCentralWidget(self.centralwidget)

        self.lineEdit = QtWidgets.QLineEdit(self.centralwidget)
        self.lineEdit.setGeometry(QtCore.QRect(20, 20, 270, 30))
        font.setPointSize(10)
        self.lineEdit.setFont(font)
        font.setPointSize(12)
        self.lineEdit.setFocusPolicy(QtCore.Qt.ClickFocus)
```

```
self.lineEdit.setObjectName("lineEdit")
self.lineEdit.setPlaceholderText("Путь к каталогу")

self.lineEdit_2 = QtWidgets.QLineEdit(self.centralwidget)
self.lineEdit_2.setObjectName("lineEdit_2")
self.lineEdit_2.hide()

self.pushButton = QtWidgets.QPushButton(self.centralwidget)
self.pushButton.setGeometry(QtCore.QRect(300, 19, 130, 32))
self.pushButton.setFont(font)
self.pushButton.setObjectName("pushButton")
self.pushButton.clicked.connect(self.selectCatalog)
self.pushButton.setText("Выбрать каталог")

self.label = QtWidgets.QLabel(self.centralwidget)
self.label.setGeometry(QtCore.QRect(20, 70, 410, 480))
self.label.setFrameShape(QtWidgets.QFrame.Panel)
self.label.setAlignment(QtCore.Qt.AlignCenter)
self.label.setLineWidth(1)
self.label.setMidLineWidth(1)
self.label.setText("")
self.label.setObjectName("label")

self.pushButton_2 = QtWidgets.QPushButton(self.centralwidget)
self.pushButton_2.setGeometry(QtCore.QRect(20, 70, 410, 480))
self.pushButton_2.setFont(font)
self.pushButton_2.setAutoFillBackground(False)
self.pushButton_2.setInputMethodHints(QtCore.Qt.ImhNone)
self.pushButton_2.setCheckable(False)
self.pushButton_2.setAutoDefault(False)
self.pushButton_2.setFlat(True)
self.pushButton_2.setObjectName("pushButton_2")
self.pushButton_2.setText("Выбрать изображение")
self.pushButton_2.clicked.connect(self.selectImage)

self.pushButton_3 = QtWidgets.QPushButton(self.centralwidget)
self.pushButton_3.setGeometry(QtCore.QRect(175, 570, 100, 32))
self.pushButton_3.setFont(font)
self.pushButton_3.setObjectName("pushButton_3")
self.pushButton_3.setText("Найти")
self.pushButton_3.clicked.connect(self.find)

self.statusbar = QtWidgets.QStatusBar(MainWindow)
```

```
self.statusbar.setObjectName("statusbar")
MainWindow.setStatusBar(self.statusbar)

self.menuBar = QtWidgets.QMenuBar(MainWindow)
self.menuBar.setGeometry(QtCore.QRect(0, 0, 450, 21))
self.menuBar.setObjectName("menuBar")
MainWindow.setMenuBar(self.menuBar)

self.action = QtWidgets.QAction(MainWindow)
self.action.setObjectName("action")
self.action.setText("Выбрать изображение")
self.action.triggered.connect(self.selectImage)

self.action_2 = QtWidgets.QAction(MainWindow)
self.action_2.setObjectName("action_2")
self.action_2.setText("Выбрать каталог поиска")
self.action_2.triggered.connect(self.selectCatalog)

self.action_3 = QtWidgets.QAction(MainWindow)
self.action_3.setObjectName("action_3")
self.action_3.setText("Выход")
self.action_3.triggered.connect(self.exit)

self.action_4 = QtWidgets.QAction(MainWindow)
self.action_4.setObjectName("action_4")
self.action_4.setText("Справка")
self.action_4.triggered.connect(self.useDef1)

self.action_5 = QtWidgets.QAction(MainWindow)
self.action_5.setObjectName("action_5")
self.action_5.setText("О программе")
self.action_5.triggered.connect(self.useDef2)

self.action_6 = QtWidgets.QAction(MainWindow)
self.action_6.setObjectName("action_6")
self.action_6.setText("Очистить все")
self.action_6.triggered.connect(self.clearAll)

self.menu = QtWidgets.QMenu(self.menuBar)
self.menu.setObjectName("menu")
self.menu.setTitle("Файл")
self.menu.addAction(self.action) # Файл -> Выбрать изображение
self.menu.addAction(self.action_2) # Файл->Выбратькаталог поиска
```

```
            self.menu.addAction(self.action_6) # Файл -> Очистить все
            self.menu.addAction(self.action_3) # Файл -> Выход

            self.menu_2 = QtWidgets.QMenu(self.menuBar)
            self.menu_2.setObjectName("menu_2")
            self.menu_2.setTitle("Справка")
            self.menu_2.addAction(self.action_4) # Справка -> Справка
            self.menu_2.addAction(self.action_5) # Справка -> О программе
            self.menuBar.addAction(self.menu.menuAction()) # Файл
            self.menuBar.addAction(self.menu_2.menuAction()) # Справка

            QtCore.QMetaObject.connectSlotsByName(MainWindow)

        def selectCatalog(self): # Выбрать каталог

self.lineEdit.setText(str(QtWidgets.QFileDialog.getExistingDirectory()))

        def selectImage(self): # Выбрать изображение
            way = str(QtWidgets.QFileDialog.getOpenFileName()[0])
            img = QtGui.QPixmap(way)
            w, h = self.label.width(), self.label.height()
            w_img, h_img = img.width(), img.height()
            try:
                x = w / w_img if w_img > h_img else h / h_img
                self.lineEdit_2.setText(way)
                self.label.setPixmap(img.scaled(int(w_img*x), int(h_img*x)))
                self.pushButton_2.setText("")
            except:
                self.label.setPixmap(img)
                self.pushButton_2.setText("Выбрать изображение")

        def clearAll(self): # Очистить все
            img = QtGui.QPixmap("")
            self.label.setPixmap(img)
            self.pushButton_2.setText("Выбрать изображение")
            self.lineEdit_2.setText("")
            self.lineEdit.setText("")

        def exit(self): # Выход
            MainWindow.close()

        def useDef1(self):
            self.dialog = Def1(self)
```

```
            self.dialog.show()

        def useDef2(self):
            self.dialog = Def2(self)
            self.dialog.show()

        def errors(self, way, way_img): # Есть ли изображение и каталог
            msgBox = QtWidgets.QMessageBox()
            msgBox.setWindowTitle("Предупреждение")
            if way == "" and way_img == "":
                msgBox.setText("Выберите изображение и каталог поиска.")
                msgBox.exec()
                return False
            elif way_img == "":
                msgBox.setText("Выберите изображение.")
                msgBox.exec()
                return False
            elif way == "":
                msgBox.setText("Выберите каталог поиска.")
                msgBox.exec()
                return False
            return True

        def find(self): # Найти
            way = str(self.lineEdit.text())
            way_img = str(self.lineEdit_2.text())
            if self.errors(way, way_img):
                self.dialog = Find(self)
                self.dialog.show()

    # =================================================================
    class Find(QtWidgets.QDialog):
        def __init__(self, parent):
            super().__init__()
            font = QtGui.QFont()
            font.setFamily("Segoe UI")
            font.setPointSize(12)
            font.setBold(False)
            font.setItalic(False)
            font.setWeight(50)

            MainWindow.resize(360, 172)
            self.setMinimumSize(QtCore.QSize(360, 172))
```

```
self.setMaximumSize(QtCore.QSize(360, 172))
self.setSizeIncrement(QtCore.QSize(0, 0))
self.setFont(font)
self.setWindowFlags(QtCore.Qt.Window)
self.setWindowTitle("Find_Dubl")

self.centralwidget = QtWidgets.QWidget(self)
self.centralwidget.setObjectName("centralwidget")

self.label = QtWidgets.QLabel(self.centralwidget)
self.label.setGeometry(QtCore.QRect(10, 10, 340, 30))
self.label.setFont(font)
self.label.setObjectName("label")
self.label.setText("Проверяется файл 0/0")

self.label_2 = QtWidgets.QLabel(self.centralwidget)
self.label_2.setGeometry(QtCore.QRect(10, 40, 340, 30))
font.setPointSize(10)
self.label_2.setFont(font)
font.setPointSize(12)
self.label_2.setObjectName("label_2")
self.label_2.setText("")

self.pushButton = QtWidgets.QPushButton(self.centralwidget)
self.pushButton.setGeometry(QtCore.QRect(210, 120, 130, 32))
self.pushButton.setFont(font)
self.pushButton.setObjectName("pushButton")
self.pushButton.setText("Стоп")

self.pushButton_2 = QtWidgets.QPushButton(self.centralwidget)
self.pushButton_2.setGeometry(QtCore.QRect(210, 120, 130, 32))
self.pushButton_2.setFont(font)
self.pushButton_2.setObjectName("pushButton_2")
self.pushButton_2.setText("OK")
self.pushButton_2.hide()

self.progressBar = QtWidgets.QProgressBar(self.centralwidget)
self.progressBar.setGeometry(QtCore.QRect(10, 80, 340, 30))
font.setPointSize(1)
self.progressBar.setFont(font)
self.progressBar.setFormat("")
self.progressBar.setObjectName("progressBar")
self.progressBar.setMinimum(0)
```

```
            self.progressBar.setProperty("value", 0)

            self.way = str(parent.lineEdit.text())
            self.way_img = str(parent.lineEdit_2.text())
            self.catalog_i = self.catalog(self.way)
            self.progressBar.setMaximum(len(self.catalog_i))
            self.pushButton.clicked.connect(self.click)

            self.thread1 = Thread(self)

self.thread1.progressBar_signal.connect(self.progressBar.setValue)
            self.thread1.label_signal.connect(self.label.setText)
            self.thread1.label_2_signal.connect(self.label_2.setText)
            self.thread1.start()
            self.pushButton_2.clicked.connect(self.thread1.click_2)

        def click(self):
            if self.thread1.running is True:
                self.thread1.running = False
                print("Stop")

        def catalog(self, way):
            tree = os.walk(way)
            cat = []
            for d, _, files in tree:
                for f in files:
                    if f.endswith(".jpg") or f.endswith(".png") or
f.endswith(".bmp"):
                        p = (d + "/" + f).replace("\\", "/")
                        cat.append(p)
            return cat

    # ================================================================
    class Thread(QtCore.QThread):
        progressBar_signal = QtCore.pyqtSignal(int)
        label_signal = QtCore.pyqtSignal(str)
        label_2_signal = QtCore.pyqtSignal(str)
        def __init__(self, parent = None):
            super(Thread, self).__init__(parent)
            self.way = parent.way
            self.way_img = parent.way_img
            self.catalog = parent.catalog_i
            self.images = []
```

```
            self.parent = parent

        def click_2(self):
            self.parent.close()
            self.dialog = Result(self)
            self.dialog.show()

        def find(self, way, way_img): # Найти
            p = 70 # Порог
            N = 1000 # Количество ключевых точек
            D, K = 20, 100
            orb = cv2.ORB_create(nfeatures=N)

            img = np.array(Image.open(way_img))
            img1 = cv2.cvtColor(cv2.cvtColor(img, cv2.COLOR_RGB2BGR),
cv2.COLOR_BGR2GRAY)
            _, des1 = orb.detectAndCompute(img1, None)

            kf = 0 # Количество файлов прошло проверку
            for i in self.catalog:
                if self.running is True:
                    kf += 1
                    if i != way_img:
                        img = np.array(Image.open(i))
                        img2 = cv2.cvtColor(cv2.cvtColor(img,
cv2.COLOR_RGB2BGR), cv2.COLOR_BGR2GRAY)
                        _, des2 = orb.detectAndCompute(img2, None)

                        bf = cv2.BFMatcher(cv2.NORM_HAMMING,
crossCheck=True)
                        matches = bf.match(des1, des2)
                        matches = sorted(matches, key=lambda x: x.distance)
                        per = 0
                        for point in matches[:K]:
                            if point.distance < D:
                                per += 1
                        if per > p:
                            self.images.append(i)
                        #print(i, " -> ", per, "% ", ("Совпадение!" if per >
p else ""), EXIF.get_GPS(i), " ", EXIF.get_TIME(i), sep="")

                        self.progressBar_signal.emit(kf)
```

```
                        self.label_signal.emit("Проверяется файл "+ str(kf)
+ "/" + str(len(self.catalog)))
                        self.label_2_signal.emit(i)
                else:
                    self.label_signal.emit("Поиск остановлен")
                    break
            self.parent.pushButton.hide()
            self.parent.pushButton_2.show()
            self.label_2_signal.emit("")
            if len(self.catalog) == kf:
                self.label_signal.emit("Все файлы проверены")

        def run(self):
            self.running = True
            self.find(self.way, self.way_img)

    # ======================================================================
    class Result(QtWidgets.QDialog):
        def __init__(self, parent):
            super().__init__()
            font = QtGui.QFont()
            font.setFamily("Segoe UI")
            font.setPointSize(12)
            font.setBold(False)
            font.setItalic(False)
            font.setWeight(50)

            self.resize(450, 650)
            self.setMinimumSize(QtCore.QSize(450, 650))
            self.setMaximumSize(QtCore.QSize(450, 650))
            self.setSizeIncrement(QtCore.QSize(0, 0))
            self.setFont(font)
            self.setWindowFlags(QtCore.Qt.Window)
            self.setWindowTitle("Find_Dubl")

            self.centralwidget = QtWidgets.QWidget(self)
            self.centralwidget.setObjectName("centralwidget")

            self.pushButton_3 = QtWidgets.QPushButton(self.centralwidget)
            self.pushButton_3.setGeometry(QtCore.QRect(175, 590, 100, 32))
            self.pushButton_3.setFont(font)
            self.pushButton_3.setObjectName("pushButton_3")
            self.pushButton_3.setText("OK")
```

```
            self.pushButton_3.clicked.connect(self.exit)

            self.label_2 = QtWidgets.QLabel(self.centralwidget)
            self.label_2.setGeometry(QtCore.QRect(20, 19, 410, 30))
            font.setPointSize(20)
            self.label_2.setFont(font)
            self.label_2.setAlignment(QtCore.Qt.AlignCenter)
            self.label_2.setObjectName("label_2")
            if len(parent.images) != 0:
                self.label_2.setText("Найдено совпадение!")
            else:
                self.label_2.setText("Совпадений нет")

            self.scrollArea = QtWidgets.QScrollArea(self.centralwidget)
            self.scrollArea.setGeometry(QtCore.QRect(20, 70, 410, 501))
            self.scrollArea.setWidgetResizable(True)
            self.scrollArea.setObjectName("scrollArea")
            self.scrollAreaWidgetContents = QtWidgets.QWidget()
            self.scrollAreaWidgetContents.setGeometry(QtCore.QRect(0, 0,
408, 479))

self.scrollAreaWidgetContents.setObjectName("scrollAreaWidgetContents")
            self.scrollArea.setWidget(self.scrollAreaWidgetContents)

            for i in range(len(parent.images)):
                self.lineEdit_3 =
QtWidgets.QLineEdit(self.scrollAreaWidgetContents)
                font.setPointSize(10)
                self.lineEdit_3.setFont(font)
                font.setPointSize(12)
                self.new_img(parent.images[i], i)

        def exit(self): # Выход (ОК)
            self.destroy()

        def new_img(self, way_img, numb):
            p = 0 if numb == 0 else 10

            self.lineEdit_3.setGeometry(QtCore.QRect(10, p + 10 + numb*232,
390, 30))
            self.lineEdit_3.setText(way_img)

            self.label_5 = QtWidgets.QLabel(self.scrollAreaWidgetContents)
```

```
            self.label_5.setGeometry(QtCore.QRect(10, p + 50 + numb*232,
164, 192))
            self.label_5.setFrameShape(QtWidgets.QFrame.Panel)
            self.label_5.setText("")
            self.label_5.setAlignment(QtCore.Qt.AlignCenter)

            img = QtGui.QPixmap(way_img)
            w, h = self.label_5.width(), self.label_5.height()
            w_img, h_img = img.width(), img.height()
            x = w / w_img if w_img > h_img else h / h_img
            self.label_5.setPixmap(img.scaled(int(w_img*x), int(h_img*x)))

    # ================================================================
    class Def1(QtWidgets.QDialog):
        def __init__(self, parent):
            super().__init__()
            font = QtGui.QFont()
            font.setFamily("Segoe UI")
            font.setPointSize(12)
            font.setBold(False)
            font.setItalic(False)
            font.setWeight(50)

            self.resize(450, 650)
            self.setMinimumSize(QtCore.QSize(450, 650))
            self.setMaximumSize(QtCore.QSize(450, 650))
            self.setSizeIncrement(QtCore.QSize(0, 0))
            self.setFont(font)
            self.setWindowFlags(QtCore.Qt.Window)
            self.setWindowTitle("Find_Dubl")

            self.centralwidget = QtWidgets.QWidget(self)
            self.centralwidget.setObjectName("centralwidget")

    # ================================================================
    class Def2(QtWidgets.QDialog):
        def __init__(self, parent):
            super().__init__()
            font = QtGui.QFont()
            font.setFamily("Segoe UI")
            font.setPointSize(12)
            font.setBold(False)
            font.setItalic(False)
```

```
        font.setWeight(50)

        self.resize(450, 650)
        self.setMinimumSize(QtCore.QSize(450, 650))
        self.setMaximumSize(QtCore.QSize(450, 650))
        self.setSizeIncrement(QtCore.QSize(0, 0))
        self.setFont(font)
        self.setWindowFlags(QtCore.Qt.Window)
        self.setWindowTitle("Find_Dubl")

        self.centralwidget = QtWidgets.QWidget(self)
        self.centralwidget.setObjectName("centralwidget")

# ======================================================================
if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    MainWindow = QtWidgets.QMainWindow()
    ui = Ui_MainWindow()
    ui.setupUi(MainWindow)
    MainWindow.show()
    sys.exit(app.exec_())
```