**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ**

**И ИНФОРМАТИКИ**

**Кафедра многопроцессорных систем и сетей**

Павлова Маргарита Валерьевна

**Web-сервис для анализа изображений**

Отчет по преддипломной практике

студентки 5 курса 1 группы

«Допустить к защите» **Руководитель практики**

с предварительной оценкой Рафеенко Екатерина Дмитриевна

Руководитель практики доцент, кандидат физико-

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ математических наук

«\_\_»\_\_\_\_\_\_\_\_\_\_\_ 2016 г.

Минск, 2016

**ОГЛАВЛЕНИЕ**

[ВВЕДЕНИЕ 4](#_Toc443424022)

[1 АЛГОРИТМ ПОИСКА ПОХОЖИХ ИЗОБРАЖЕНИЙ НА ОСНОВЕ ЦВЕТА И ТЕКСТУРЫ 8](#_Toc443424023)

[1.1 Описание 8](#_Toc443424024)

[1.2 Результаты 10](#_Toc443424025)

[2 АЛГОРИТМЫ ПОИСКА ПОХОЖИХ ИЗОБРАЖЕНИЙ С ИСПОЛЬЗОВАНИЕМ МОДЕЛИ ИНДЕКСА КАЧЕСТВА ИЗОБРАЖЕНИЯ 11](#_Toc443424026)

[2.1 Описание 11](#_Toc443424027)

[2.2 Результаты 13](#_Toc443424028)

[3 АЛГОРИТМ ПОИСКА ПОХОЖИХ ИЗОБРАЖЕНИЙ С ИСПОЛЬЗОВАНИЕМ AHASH 14](#_Toc443424029)

[3.1 Описание 14](#_Toc443424030)

[3.2 Результаты 17](#_Toc443424031)

[4 АЛГОРИТМ ПОИСКА ПОХОЖИХ ИЗОБРАЖЕНИЙ С ИСПОЛЬЗОВАНИЕМ PHASH 18](#_Toc443424032)

[4.1 Описание 18](#_Toc443424033)

[4.2 Результаты 19](#_Toc443424034)

[ЗАКЛЮЧЕНИЕ 20](#_Toc443424035)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 21](#_Toc443424036)

**РЕФЕРАТ**

Отчет по преддипломной практике, 19 страниц, 4 рисунка, 4 источника.

**Ключевые слова**: ПОИСК ИЗОБРАЖЕНИЙ ПО СОДЕРЖАНИЮ, БАЗА ДАННЫХ, ЦВЕТОВАЯ ХАРАКТЕРИСТИКА, ТЕКСТУРА, МАТРИЦА КОВАРИАЦИИ, РАССТОЯНИЕ ЕВКЛИДА, **РАССТОЯНИЙ БХАТТАЧАРЙЯ,**  СРЕДНЕЕ ИЗОБРАЖЕНИЯ, ДИСПЕРСИЯ, ПЕРЦЕПТУАЛЬНЫЙ ХЭШ, РАСТОЯНИЕ ХЭММИНГА.

**Объект исследования:** технологии для разработки WEB-приложений.

**Цель работы:** разработка программной системы для управления цветочными заказами с использованием фреймворков Spring и AngularJS.

**Методы работы:** исследование эффективных решений для создания WEB-приложения.

**В результате** проведенной работы были исследованы технологии для разработки WEB-приложения.

# ВВЕДЕНИЕ

Роль цифровых данных значительно возросла в последние 10 лет в связи с увеличивающимся распространением цифровой техники. Возрастает количество и доступность цифровых фотоаппаратов, сканеров, настольных издательских систем, настольных баз данных. Постоянно увеличивается пропускная способность каналов сети Интернет. Все это приводит к значительному росту количества обрабатываемых данных, однако алгоритмы обработки и поиска не успевают обеспечить адекватную реакцию на столь значительное увеличение объема доступных цифровых материалов. Особенно сильно это ощущается в области средств и алгоритмов поиска в мультимедийных базах данных.

На данный момент поиск в большей части мультимедийных баз производится с применением принципов и алгоритмов, разработанных несколько десятков лет назад для другого типа данных, что приводит к неудовлетворительным результатам поиска.

Большая часть имеющихся на данный момент алгоритмов не учитывает особенности восприятия человеком изображений, и базируется на разметке баз изображений текстовыми тэгами и набором атрибутов, по которым в дальнейшем можно производить поиск с использованием классических алгоритмов поиска строки.

Восприятие человеком информации является трудноформализуемой задачей, и результаты поиска с использованием существующих алгоритмов поиска слабо коррелируют с тем, что хотел бы видеть пользователь, осуществляющий поиск.

Зачастую, с увеличением размера базы изображений традиционные алгоритмы исчерпывают свои возможности. Например, для ускорения поиска можно просматривать уменьшенные копии изображений. Поиск изображения среди сотен уменьшенных копий еще является посильной задачей для пользователя, однако увеличение порядка размера базы до тысяч изображений делает данную задачу практически невозможной. Поэтому при существенном расширении базы данных поиск методом полного перебора изображений теряет свою эффективность.

Одной из распространенных стратегий поиска является индексация по ключевым словам и атрибутам. Данный подход имеет свои преимущества, однако выдвигает ряд требований, исполнение которых порой становится чрезвычайно затруднительным.

Первой причиной является, собственно необходимость разметки базы изображений ключевыми словами, которая является весьма затратной процедурой. Второй причиной является то, что некоторые визуальные аспекты изображения в принципе являются трудноописуемыми или неописуемыми вообще. Далее, пользователю может быть неизвестно, какие аспекты внесены в индекс, а какие нет. Стоит отметить, что некоторые классы изображений (например, текстуры) вообще с большим трудом поддается разметке ключевыми словами. Третьей причиной является неоднозначность восприятия изображения и, как следствие из этого, поиск по базе изображений, размеченной тэгами одним человеком для другого значительно менее эффективен.

Изложенные выше соображения привели к росту исследований в области контекстного поиска изображений. Контекстный поиск изображений имеет ряд других названий, среди которых «поиск по содержимому», «поиск по образцу», «поиск по эскизу», «поиск по методу подобия». Можно заметить, что контекстный поиск является альтернативой традиционным методам поиска и может применяться совместно с ними.

Поиск изображений похожих на заданное изображение находит применение во многих областях, включая архитектуру, телевидение, средства мультимедиа, графический дизайн, историю искусств, криминологию, геологию, медицину.

Дизайнер может потребовать найти изображения, похожие на заданное, для использования в качестве альтернативы или на замену.

В криминологии данные алгоритмы востребованы при поиске фотографии по фотороботу. Также для правоохранительных органов актуальным является поиск фрагмента или целого изображения в записи камер наблюдения. Как правило, при этом используется фотография грубого разрешения.

Архитектор может использовать данный метод поиска для отбора решений, которые являются удачными для данной местности, а также для поиска аналогов задумываемого или исполненного строения.

Данная задача также является актуальной и в домашней сфере. Увеличившаяся доступность цифровой аппаратуры привела к значительному росту домашних видео- и фотоархивов.

Контекстный поиск призван качественно улучшить результаты поиска в цифровых библиотеках. Во всех упомянутых областях совершенствование технологии формирования изображения является важной и общепризнанной проблемой.

За последние 10 лет в свете усиления интереса к данной проблеме появилось несколько систем контекстного поиска изображений. Стоит упомянуть наиболее известные системы, базирующиеся на поиске с использованием ключевых слов:

1. Yandex (https://yandex.by/images);

2. NBCi (http://www.snap.com);

3. Yahoo! Image Search (http://images.search.yahoo.com);

4. Tineye (https://www.tineye.com);

5. Bing (http://www.bing.com);

6. Google Image Search (http://images.google.com);

7. PICSearch (http://www.picsearch.com/).

Наиболее старой и известной системой является система контекстного поиска QBIC (Query By Image Content), разработанная компанией IBM. Некоторые исследователи отмечают, что данная система достаточно успешна в работе с цветом и текстурой, однако недостаточно точна при работе с формой изображения.

Большинство данных систем являются результатом исследований и скорее полигоном для испытания идей исследователя, нежели законченным продуктом. Преимущественно этим обосновывается то, что алгоритмы, лежащие в основе данных систем, обыгрывают один или два аспекта контекстного поиска. Большая часть этих систем использует цветовые и текстурные особенности, небольшая часть использует пространственные особенности, т.е. местоположение (в некоторых случаях распределение) особенности на картинке.

Поиски по цветовым особенностям, как правило, дают неплохие результаты, несколько хуже обстоит дело с поиском по текстурным особенностям. Исключением являются случаи присутствия в базе некой превалирующей текстуры. Поиск по форме изображения на данный момент дает плохие результаты в большинстве данных систем.

Таким образов, в данной работе будут рассмотрены алгоритмы поиска подобных изображений, выполнен их анализ и модификация для улучшений качества решения.

1. АЛГОРИТМ ПОИСКА ПОХОЖИХ ИЗОБРАЖЕНИЙ НА ОСНОВЕ ЦВЕТА И ТЕКСТУРЫ

# 1.1 Описание

**Одним из методов поиска похожих изображений является извлечение из каждого изображения несколько ключевых характеристик, которые будут использоваться для вычисления подобия между изображениями. Эти характеристики описывают содержимое изображения, и поэтому они должны быть тщательно выбраны в соответствии с контекстом. Таким образом, в данном разделе будет рассмотрен поиск похожих изображений в терминах цвета и текстуры.**

**Цветовая характеристика представлена с помощью среднего цвета**  и матрицы ковариации . Пусть

|  |  |
| --- | --- |
|  | (1) |

**где – число пикселей изображения, – цвет -ого пикселя в канале . Предполагая использование оригинального RGB (Red, Green, Blue) цветового пространства, матрица ковариаций будет:**

|  |  |
| --- | --- |
|  | (2) |

**Текстурная характеристика извлечена из обобщенной смежной матрицы. Предположим, изображение содержит столбцов и**  **строк и** **уровней серого цвета. Пусть представляют столбцы, – строки, и – множество уровней серого цвета. Смежная матрица – матрица размерности , где**

|  |  |
| --- | --- |
|  | (3) |

**т.е. количество смежных пар серого уровня и , для которых расстояние Пары пикселей и имеют расстояние и ориентацию**

**Обобщенная смежная матрица это расширение смежной матрицы для мультиспектральных изображений, т.е. изображений, представленных n цветовыми каналами. Пусть и два цветовых канала. Обобщенная ковариационная матрица будет:**

|  |  |
| --- | --- |
|  | (4) |

**Например, в случае цветного изображения, закодированного в трех каналах (RGB), имеется шесть смежных матриц: (RR), (GG), (BB), которые так же, как и смежная матрица серого уровня, вычисляются по одному каналу, и (RG), (RB), (GB), которые учитывают корреляцию между каналами.**

**Система поиска основана на мере сходства между тестовым изображением и изображением из базы данных.**

**Для цветовой характеристики, основанной на матрице ковариаций, мера расстояний Бхаттачарйя используется в следующем виде:**

|  |  |
| --- | --- |
|  | (5) |

**где и вектора средних цветовых характеристик, и – ковариационные матрицы изображений и , - определитель матрицы.**

**Расстояние Евклида используется как расстояние между текстурными характеристиками:**

|  |  |
| --- | --- |
|  | (6) |

**Объединяя два расстояния в функцию согласующегося подобия, получаем:**

|  |  |
| --- | --- |
|  | (7) |

**где**  и веса факторов, выбираемые экспериментально.

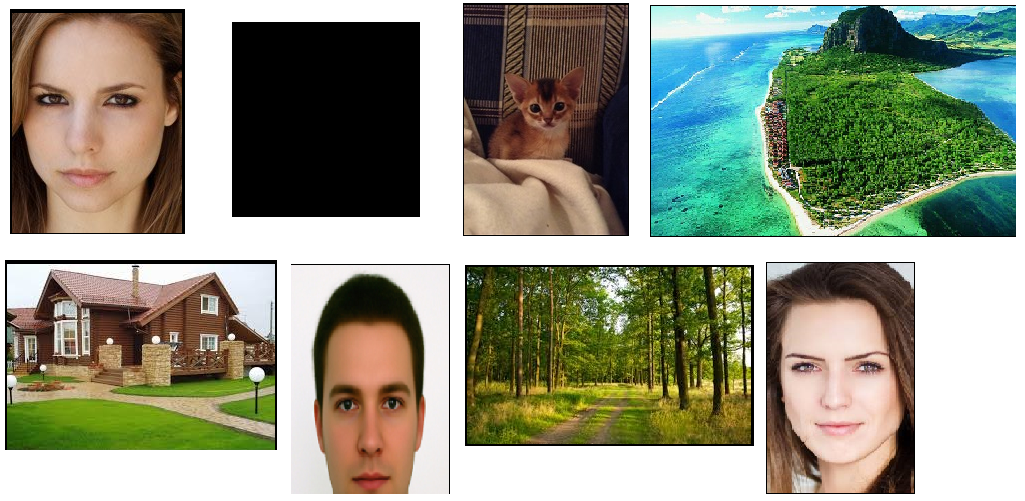
# 1.2 Результаты

**Рассмотрим тестовое изображение (рисунок 1.1):**



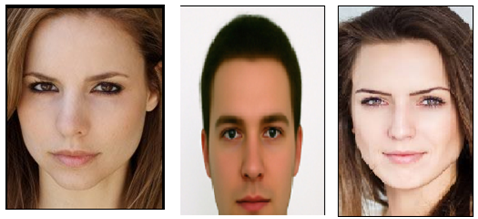
**Рисунок 1.1 – Тестовое изображение**

**и базу изображений (рисунок 1.2):**



**Рисунок 1.2 – База изображений**

**При реализации алгоритма поиска похожих изображений на основе цвета и текстуры были выбраны образы, помеченные как схожие с тестовым изображением. Выбранные образы изображены на рисунке 1.3. Как можно заметить, результат алгоритма корректен: лица, находящиеся в схожем положении, были выбраны; черный квадрат, котенок, остров, дом и лес выбраны не были.**



**Рисунок 1.3 – Изображения, схожие с тестовым изображением**

1. АЛГОРИТМЫ ПОИСКА ПОХОЖИХ ИЗОБРАЖЕНИЙ С ИСПОЛЬЗОВАНИЕМ МОДЕЛИ ИНДЕКСА КАЧЕСТВА ИЗОБРАЖЕНИЯ

# 2.1 Описание

**Для обработки изображений модель индекса качества делает акцент на 3 фактора: потерю корреляции, искажение яркости и контрастность искажений.**

**В 2003 году был предложен метод на основе CDESSO, который характеризовал сложность цвета, цветовые различия между соседними пикселями для 24-битного полноцветного изображения. Для начала этот метод уменьшал цветовое пространство изображения. Он разделял все пиксели изображения на 64 кластера. Каждый кластер имел своё собственное хранилище для записи разницы между соседними пикселями. Каждый пиксель изображения был вставлен в ближайший кластер. Затем сканировалось уменьшенное изображение, и вычислялась разность между каждыми двумя соседними пикселями. Разность добавляется к соответствующему кластеру текущего пикселя.**

**Для поиска изображений сравнивалась, используя Евклидово расстояние, кластеры тестового изображения и кластеры изображений из базы данных. После проведения испытаний был выявлено, что алгоритм обеспечивает не только высокую степень точности, но и не изменяет своего качества при вращении изображения.**

**Способ поиска изображения CDESSO, а также другие цветовые методы, основанные на CBIR, используют Евклидово расстояние, чтобы оценить разницу между двумя изображениями. Однако такие способы измерения, как MSE, PSNR и Евклидово расстояние, не могут быть применены при искажении изображения окружающей средой. Поэтому был предложен математически определенный универсальный показатель качества изображения.**

**Пусть изображение из базы данных, где – -ый пиксель изображения , и – тестовое изображение. Расстояние между изображениями и измеряется с помощью:**

|  |  |
| --- | --- |
|  | (8) |

**где – среднее изображения , определенное**

|  |  |
| --- | --- |
|  | (9) |

**– среднее изображения , определенное**, – дисперсия **,**  – дисперсия **,**  – корреляция между изображением  **и изображением , определяемая**

|  |  |
| --- | --- |
|  | (10) |

Диапазон от до . Если значение близко к , тогда можно сказать, что изображение из базы данных и тестируемое изображение схожи. Если значение равно , тогда изображение из базы данных и тестируемое изображение абсолютны различны.

Большинство цифровых изображений представлены в цветовом пространстве RGB. Таким образом, каждый пиксель может быть интерпретирован в 3 измерениях: красном, зеленом и синем. Однако пространство RGB не очень удобно для анализа изображений. Следовательно, первый шаг предложенного метода состоит в преобразовании цветового 3-размерного пространства в 1-размерное серое пространство. Пусть  **и а есть, соответственно, изображение из базы данных и тестовое изображение в 1-размерном сером пространстве. Применим уравнение (8) для расчета индекса качества между изображениями и . Изображение с большим показателем индекса качества – изображение, которое наиболее схоже с тестовым. Поскольку диапазон от до , то когда равно можно сказать, что изображения и одинаковы. В модели индекса качества значение вычисляется с помощью среднего, дисперсии и корреляции и . Среднее и дисперсия могут быть вычислены независимо. Но корреляцию нужно вычислять динамически, что требует больше времени. Поэтому формулу показателя качества (8) можно преобразовать. Новая формула может отфильтровать несовпадающие изображения заранее, чтобы ускорить поиск схожих изображений. В соответствии с определением модели, индекс качества является комбинацией трех факторов: потери корреляции, искажение яркости и контраст искажений. Измененное уравнение индекса качества:**

|  |  |
| --- | --- |
|  | (11) |

**Где**  – коэффициент корреляции, – искажение

яркости, – контраст искажения. Диапазон от до ; диапазон

от до ; диапазон от до . Можно заметить, что только значение касается каждого пикселя  **и** , а величины  **, ,**  можно вычислить заранее. Пусть . Тогда индекс качества может быть представлен . Учитывая диапазон значений , очевидно, что любое значение , за исключением , уменьшает значение . Следовательно, можно заранее определить порог , чтобы отфильтровать неподходящие изображения. Другими словами, сравнивать необходимо те изображения, для которых .

# 2.2 Результаты

Реализация алгоритма при использовании тестового изображения (рисунок 1.1) и базы изображений (рисунок 1.2) дала следующие результаты: изображения, представленные на рисунке 2.1, были выбраны как схожие; изображения на рисунке 2.2 не были выбраны.



Рисунок 2.1 – Изображения, схожие с тестовым изображением

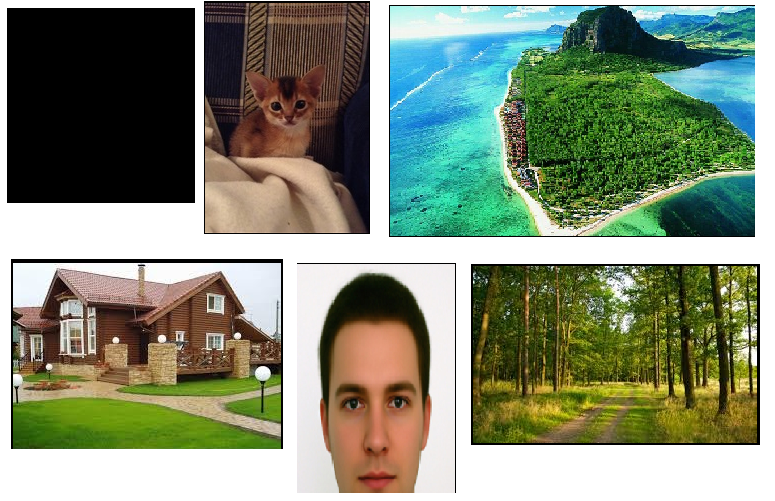


Рисунок 2.2 – Изображения, не схожие с тестовым изображением

1. АЛГОРИТМ ПОИСКА ПОХОЖИХ ИЗОБРАЖЕНИЙ С ИСПОЛЬЗОВАНИЕМ AHASH

# 3.1 Описание

**Хэш-функции (или функции свертки) представляют собой функции преобразования, которые позволяют получить укороченный вариант исходных данных. Классические криптографические хэш-алгоритмы (например, MD5 или SHA-1) работают таким образом, чтобы для различных исходных данных, как мало бы они не отличались друг от друга, в результате получались максимально отличные хэш-значения. В отличие от классических хэш-функций, перцептуальные хэш-алгоритмы генерируют хеши, предназначенные для сравнения исходных данных. Хэш-значения таких алгоритмов тем ближе, чем более схожи были исходные данные.**

**Таким образом, задача сравнения изображений сводится к вычислению хэш-значений этих изображений, и вычислению расстояния Хэмминга между ними. Чем меньше расстояние Хэмминга, тем более похожи данные изображения (рисунок 3.1). Расстояние 0, например, означает, что изображения скорее всего являются полностью идентичными. И напротив, чем более высокая дистанция, тем больше изображения отличаются друг от друга (к примеру, дистанция 10 и больше для хэша размером 64 бита означает, что это, скорее всего, разные картинки).**



**Рисунок 3.1 – Расстояние Хэмминга для изображений**

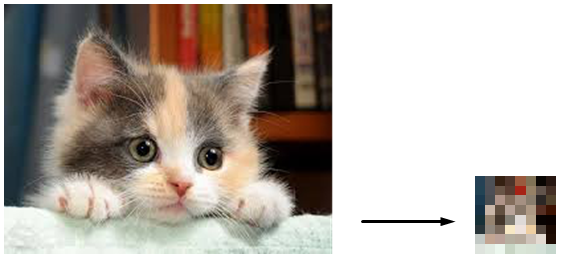
**Существуют различные алгоритмы вычисления перцептуальных хешей, которые различаются между собой чувствительностью к определенным типам искажений: изменению размера изображения, изменению соотношения сторон, цветовых характеристик (яркость, контраст, гамма), наложения водяных знаков и так далее. Одним из них является aHash (Average Hash). Данный хэш-алгоритм является очень быстрым, не чувствителен к масштабированию исходного изображения, сжатию или растяжению, изменению яркости или контрастности. aHash основан на среднем значении, и, как следствие, является чувствительным к операциям, изменяющим среднее значение (например, изменение уровней или цветового баланса изображения).**

**Для построения aHash выполняются следующие шаги:**

1. **Уменьшение размера изображения.**

**Вначале исходное изображение уменьшается до размера пикселей, что даст хэш размером 64 бита. Размер изображения влияет на точность сравнения, и скорость работы алгоритма. Чем больше изображение, тем более высокая точность сравнения будет получена, что потребует больше времени для расчета.**

**Масштабирование может выполняться без соблюдения пропорций, таким образом, полученный хэш будет соответствовать всем вариантам изображения с любым отношением сторон.**

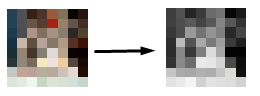


**Рисунок 3.2 – Масштабирование изображения**

**Если рассматривать изображение как дискретный сигнал, то в таком сигнале высокие частоты обеспечивают детализацию изображения, а низкие частоты показывают его структуру. Уменьшение размера изображения удаляет высокие частоты, и, таким образом, полученное изображение состоит преимущественно из низких частот, сохраняя общую структуру изображения.**

1. **Перевод в градации серого цвета.**

**Данный шаг позволяет в 3 раза уменьшить размер хэша, за счет уменьшения количества компонент с 3-х значений RGB до одного уровня серого.**



**Рисунок 3.3 – Перевод изображения в градации серого**

1. **Вычисление среднего значения уменьшенного изображения в градации серого цвета.**

**Далее вычисляется среднее значение по всем точкам изображения:**

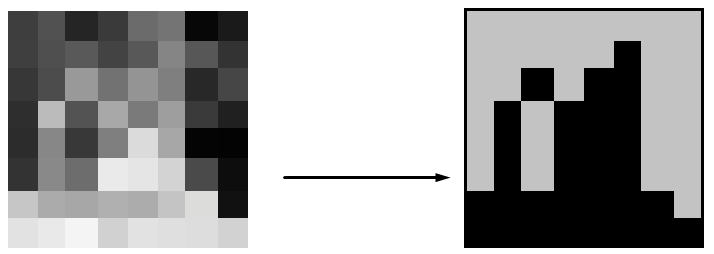
|  |  |
| --- | --- |
|  | (12) |

где и – размеры уменьшенного изображение, – значение пикселя в ячейке .

**4. Упрощение изображения, при котором каждый пиксель заменяется значением 0, если он меньше рассчитанного на 3-м шаге среднего значения, и 1 если больше.**

|  |  |
| --- | --- |
|  | (13) |

**Из данного изображения получается цепочка бит (считыванием изображения построчно), из которой в свою очередь и строится хэш-значение:**



**Рисунок 3.4 – Вычисление хэш-значения изображения**

**Значение расстояния Хэмминга между хэш-значениями определит меру схожести двух изображений.**

# 3.2 Результаты

**Реализация алгоритма с использованием aHash на тестовом изображении (рисунок 1.1) и базе изображений (рисунок 1.2) дало тот же результат, что и алгоритм с использованием модели индекса качества изображений (смотреть рисунок 2.1) . Из этого можно сделать вывод, что если для системы время является критическим параметром, то лучше всего использовать aHash, т.е. вычислить хэш-значение изображение можно заранее. В том время как модель индекса качества изображений требует вычисление параметров по запросу.**

1. АЛГОРИТМ ПОИСКА ПОХОЖИХ ИЗОБРАЖЕНИЙ С ИСПОЛЬЗОВАНИЕМ PHASH

# 4.1 Описание

**Как было сказано выше, aHash не чувствителен к изменению масштаба изображения, изменению яркости или контрастности, но является чувствительным к изменению цветового баланса изображения, так как это влияет на получаемое среднее значение. Данный недостаток исправляется следующим алгоритмом pHash.**

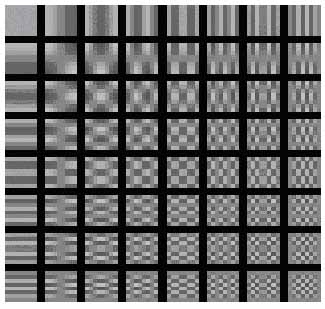
**pHash во многом повторяет шаги aHash, но при этом добавляет еще один этап, на котором выполняется дискретное косинусное преобразование (DCT), которое позволяет разделить изображение на части разной степени важности.**

**Первые два шага алгоритма идентичны aHash, за тем исключением, что размер изображения выбирается больше (например,). В данном случае данный шаг предназначен не для удаления высоких частот (что будет сделано позже), а для упрощения алгоритма дискретного косинусного преобразования.**

**На третьем шаге выполняется DCT-преобразование, которое разбивает изображение на набор базовых частот. Для двумерных матриц данное преобразование выглядит следующим образом:**

|  |  |
| --- | --- |
|  | (14) |

**где , — размер входной матрицы, — значение матрицы (интенсивность пикселя) в строке и колонке , — DCT-коэффициент в строке и колонке матрицы DCT. Данные коэффициенты могут рассматриваться как весовые коэффициенты базисных функций. Например, для матрицы с размером элементов существует 64 базовые функции, что продемонстрировано на изображении:**



**Рисунок 4.1 – Базовые функции изображения**

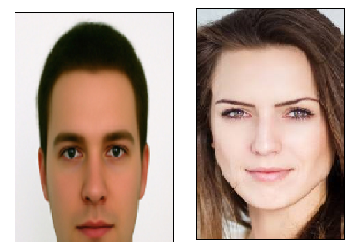
**Выходной массив DCT содержит целые числа в диапазоне . Для большинства изображений значимыми являются низкие частоты, которые будут расположены в левом верхнем углу DCT-матрицы. После расчета DCT-матрица сокращается, отбрасыванием незначимых (высоких) частот, в результате чего получается блок меньшего размера ( или ), и так же, как и в случае aHash-алгоритма, вычисляется среднее значение:**

**Дальнейшие шаги полностью соответствуют алгоритму aHash: значения получившийся матрицы сокращаются до значений 1 или 0 в зависимости от значения каждого пикселя (больше или меньше среднего значения), и на основе получившегося изображения строится хэш.**

**Как и в aHash, значения pHash можно сравнивать между собой с помощью алгоритма расстояния Хэмминга. Такой вариант уже выдержит гамма-коррекцию или изменение гистограммы изображения.**

# 4.2 Результаты

Реализация алгоритма pHash с использованием тестового изображения 1.1 и базы изображений 1.2 выявила в качестве похожих изображения, представленные на рисунке 4.1.



**Рисунок 4.2 – Изображения, выбранные в качестве похожих**

1. АЛГОРИТМ ПОИСКА ПОХОЖИХ ИЗОБРАЖЕНИЙ С ИСПОЛЬЗОВАНИЕМ ЦВЕТОВЫХ ГИСТОГРАММ

# 5.1 Описание

Цветовые гистограммы широко используются в различных задачах компьютерного зрения, таких как поиск объекта на изображении, поиск похожих изображений, классификация изображений. Популярность методов на основе гистограмм объясняется простотой их вычисления, устойчивостью к перемещениям и поворотам объекта интереса и изменениям положения камеры.

Цветовая гистограмма — это вектор, в каждой ячейке которого содержится число пикселей изображения или участка изображения, имеющих соответствующий цвет. Часто при построении гистограммы используется цветовое пространство RGB, однако программное обеспечение ряду причин предпочтительнее использовать цветовое пространство HSV.

В цветовом пространстве HSV координатами цвета являются: hue – цветовой тон (например, красный, зелёный или синий), saturation – насыщенность (чистота цвета), и value – яркость. Данная цветовая модель обладает рядом принципиальных преимуществ. В модели HSV информация о цвете и обработчик освещённости хранится в разных цветовых компонентах, что позволяет повысить инвариантность к освещению.

Гистограммы в цветовом пространстве HSV более удобны в использовании. Например, при эквализации гистограммы (преобразовании изображения, после которого его гистограмма содержит все возможные значения яркости, равномерно распределённые на всём диапазоне) требуется скорректировать только яркость пикселей изображения, оставив их цвет без изменений. Использование цветового пространства HSV позволяет добиться устойчивости к затенению изображения, так как затенение не оказывает влияния на компоненту hue.

Сходство гистограмм рассчитывается по формуле:

|  |  |
| --- | --- |
|  | (15) |

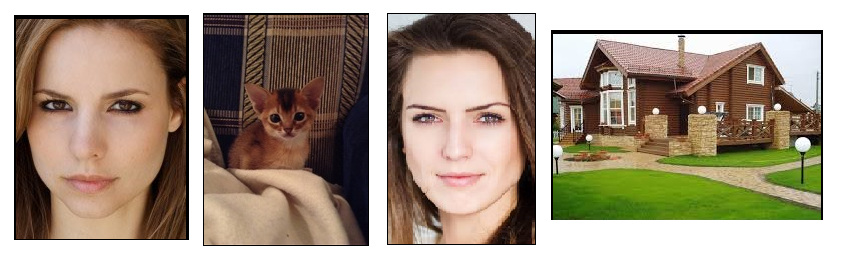
Возвращаемое значение лежит в интервале , и означает максимальное соответствие, 0 – отсутствие корреляции.

# 5.2 Результаты

На рисунке 5.1 представлены результаты реализации алгоритма с использованием цветовых гистограмм. Как можно заметить, некоторые выбранные изображения сильно отличаются от тестового изображения (рисунок 1.1). Это связано с тем, что гистограммы содержат только информацию о цвете, и не содержат информацию о пространственной конфигурации элементов сцены.

Для поиска заданных объектов на изображении с помощью гистограмм обычно применяется метод «скользящего окна». Вычисляется гистограмма исследуемого изображения внутри некоторого прямоугольника («окна»), которая сравнивается с гистограммой искомого объекта. Однако, искомый объект – не обязательно прямоугольной формы, следовательно, в «окно» попадут не только пиксели объекта, но и пиксели фона, что может существенно исказить гистограмму.

Также, если разные грани объекта имеют разный цвет, при повороте и аффинных преобразованиях пропорции этих цветов в гистограмме могут сильно измениться.



**Рисунок 5.1 – Изображения, выявленные в качестве схожих**

1. АЛГОРИТМ ПОИСКА ПОХОЖИХ ИЗОБРАЖЕНИЙ, ИСПОЛЬЗУЮЩИЙ ПИКСЕЛЬНОЕ СРАВНЕНИЕ

# 6.1 Описание

При сравнении похожих изображений первым встает вопрос: что считать мерой схожести изображений. Очевидно, что это величина имеет значение обратное различию изображений друг от друга. Следственно нужно выбрать некую метрику, характеризующую различие изображений друг от друга. Тогда схожими изображениями будут считаться изображения, отличие между которыми меньше некоторого порога. Для изображений с одинаковыми габаритами, обычно такой мерой различия служит среднеквадратическое отклонение пикселей одного изображения от другого.

Вопрос несколько усложняется, если нужно сравнить изображения разных размеров. Однако достаточно очевидным решением этой проблемы является приведение всех изображений к одинаковому размеру. Остается выбрать это размер. Если выбрать его слишком маленьким, то очевидно, что различия между изображениями тогда будут нивелироваться, и у нас будет много ложно положительных срабатываний. При слишком большом размере неоправданно повышается ресурсоёмкость алгоритма сравнения.

Таким образом, алгоритм поиска похожих изображений будет включать в себя следующие шаги:

1. Приведение всех изображений к одному размеру (возьмем для определенности размер );
2. Исключение цветовой информации (преобразование в серое изображение);
3. Нахождение среднеквадратической разности для каждой пары уменьшенных серых изображений;
4. Сравнение полученной среднеквадратической разности с некоторым порогом.

Порог определяет меру схожести изображений. Если выбрать порогом 0%, то алгоритм найдет только полностью идентичные изображения. При 5% пороге алгоритм сможет найти также визуально похожие изображения, которые могут различаться разрешениям, качеством сжатия, наличием мелких надписей, переворотом, при незначительном количестве ложных срабатываний. При пороге выше 10%, как правило, количество ложных срабатываний может превышать число положительных срабатываний.

# 6.2 Результаты

Выполнение алгоритма на тестовом изображении (рисунок 1.1) и базе изображений (рисунок 1.2) выбрало образы на рисунке 6.1 как схожие с тестовым.



**Рисунок 6.1 – Изображения, выявленные в качестве схожих**

Результат алгоритма не является корректным на основе визуального человеческого сравнения. Поэтому он не будет использоваться при построении web-сервиса.

ЗАКЛЮЧЕНИЕ

При выполнении работы были исследованы, реализованы и доработаны следующие алгоритмы:

1.

2.

3.

4.

Было выявлено, что анализ цвета и текстуры выявил максимальное количество похожих изображений. Таким образом, для достижения наилучшего качества приложения рекомендуется использовать приведенные алгоритмы вместе и выдавать пользователю данные, являющиеся объединением результатов алгоритмов.

В процессе исследования алгоритмов также были реализованы 2 метода поиска похожих изображений: пиксельное сравнение и анализ цветовой гистограммы. Но эти технологии не дали положительного результата, так как являлись длительными по времени и выявляли изображения, которые не были бы выбраны человеком как схожие.

Детерминированные алгоритмы поиска похожих изображений выдали хорошие результаты. Однако все они используют примитивные характеристики: цвет, текстура, яркость, резкость и тому подобное. Человек, глядя на картинку, воспринимает цельные образы, причем способность к такому восприятию во многом опирается на приобретенный жизненный опыт. У машины такой опыта нет, и это является причиной многих неточностей при поиске.

Пользовательский СВIR-поиск еще в самом начале пути. Большинство поисковиков открытого доступа находятся в стадии бета-версий. Экспериментальные машины поиска изображений занимают нишу специфических запросов и вряд ли готовы полностью заменить обычные поисковики и каталоги, ориентированные на использование ключевых слов. В то же время новейшие ‹контентные› технологии становятся прекрасным дополнением к «традиционным› способам индексации и поиска. Действительно: наиболее гибкими и удобными оказываются сервисы, объединяющие различные подходы к поиску: по ключевым словам, по тегам, по визуальным характеристикам изображений. Это, скорее всего, будет ведущей тенденцией ближайшего будущего. Возможности поиска изображений по косвенным признакам и проставленным пользователями тегам на универсальных поисковиках н фотохостингах все активнее дополняются технологиями контентного поиска.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Spring [Electronic resource]. – Mode of access : <http://www.spring.io>. – Date of access : 15.10.2014.
2. AngularJS [Electronic resource] / AngularJS – Superheroic JavaScript MVW Framework. – Mode of access : <https://www.angularjs.com>. – Date of access : 9.11.2014.
3. Hibernate [Electronic resource] / Hibernate. Everything data. - Hibernate. – Mode of access : <http://hibernate.org/>. – Date of access : 13.10.2014.
4. Walls C. Spring in action / C. Walls. – 3 edition. – New York : Manning Publications Co., 2011. – 400 p.
5. Nexmo [Electronic resource] / Nexmo – APIs for SMS, Voice and Phone Verification. - Nexmo. – Mode of access : <http://nexmo.org/>. – Date of access : 21.03.2015.
6. Google Maps [Electronic resource] / Google Maps API – Google Developers . – Google Maps. – Mode of access : <https://developers.google.com/maps>. – Date of access : 25.03.2015.