Правительство Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования
«Санкт-Петербургский государственный университет»

Кафедра Системного Программирования

Шубин Сергей Владимирович

Автоматическая идентификация живописного полотна по его фотографии

Бакалаврская работа

Допущена к защите.
Зав. кафедрой:
д. ф.-м. н., профессор Терехов А. Н.

Научный руководитель:
к. ф.-м. н. Вахитов А.Т.

Рецензент:
Николаев А.И.

Санкт-Петербург
2015

SAINT-PETERSBURG STATE UNIVERSITY

Chair of Software Engineering

Shubin Sergey

Automatic identification of a painting by its photo

Bachelor's Thesis

Admitted for defence.
Head of the chair:
Professor Andrey Terekhov

Scientific supervisor:
Ph. D. Alexander Vakhitov

Reviewer:
Alexander Nikolaev

Saint-Petersburg
2015

Оглавление

1. Введение **4**

2. Обзор **5**

2.1. Глобальные дескрипторы 5

2.1.1. Гистограммы 5

2.1.2. Matrix of Brightness Variation 6

2.1.3. Вейвлет-преобразование 7

2.1.4. Проблемы глобальных дескрипторов 8

2.2. Локальные дескрипторы 8

2.2.1. SIFT 8

2.2.2. SURF 9

2.3. Поиск в дескрипторах 9

3. Алгоритм поиска **11**

3.1. Построение дескриптора изображения 11

3.2. Идентификация 12

3.3. Locality Sensitive Hashing 13

4. Тестирование **16**

Заключение **21**

Список литературы **22**

1. Введение

Задача идентификации изображения – это важная задача, которая часто встает на практике. На данный момент хорошо исследованы методы поиска дубликатов и почти-дубликатов изображений в Интернете с ограниченными изменениями оригинала [17, 7, 11, 16, 8]. Например, достаточно эффективным является метод на основе вейвлет-преобразования Добеши [7], который строит компактные дескрипторы для изображений (под дескриптором в данной работе понимается идентификатор изображения в совокупности с массивом чисел, которые как-то описывают изображение). Несмотря на то, что методы на глобальных дескрипторах хорошо себя рекомендуют при поиске слабо измененных изображений, они слабо устойчивы к классу искажений, которые возникают при съемке. Примеры таких искажений: изменение угла, под которым направлена камера на плоскость полотна, нелинейные шумы в цветовых диапазонах, изменения освещения и тому подобные.

Таким образом, на практике в задаче идентификации по изображении появляется другой подкласс задач, где изображения меняются под воздействием несовершенства съемочного оборудования. Одной из таких задач является идентификация живописного полотна по его фотографии. В этом случае не получается использовать глобальные дескрипторы, так как при выделении полотна на фотографии границы не могут быть определены точно, и полотна фотографируются не под прямым углом к плоскости полотна, отчего теряется соответствие областей.

Из-за данных ограничений были выбраны методы на базе выделения особых точек и построения дескриптора локальной области около точки. Итоговый дескриптор изображения будет составлен из дескрипторов особых точек, которые были детектированы на нем. Несмотря на то, что при выделении особых точек мы не можем добиться полного совпадения множеств, поиск построен таким образом, что достигается идентификация полотна по фотографии с близкой к 100% точностью. В данной работе не рассматриваются задачи определения нахождения оригинала в базе или поиск всех оригиналов, считается, что на каждую тестовую фотографию в базе имеется ровно одна соответствующая запись.

2. Обзор

Существующие подходы к идентификации изображения можно условно разделить на два типа — глобальные и локальные.

2.1. Глобальные дескрипторы

2.1.1. Гистограммы

Одним из самых простых глобальных дескрипторов является гистограмма яркостей [14]. Если считать, что яркость пикселя изображения находится в промежутке $[0..n-1]$, то гистограммой будет n-мерный вектор, каждый элемент которого — это количество пикселей в изображении с данной яркостью.

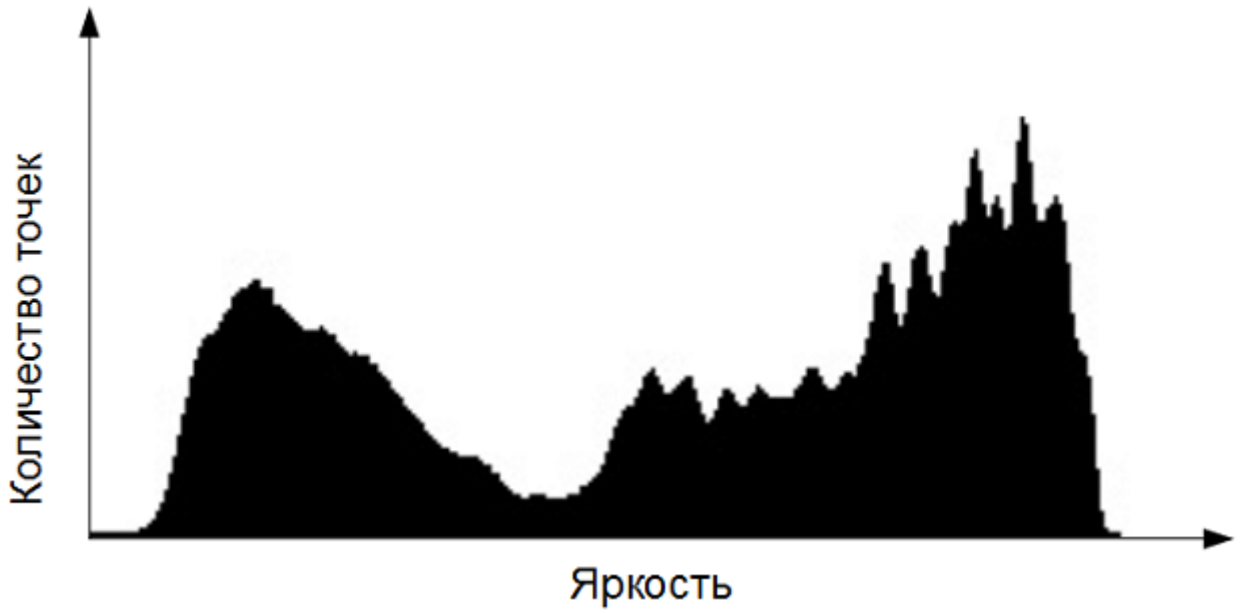

Рис. 1: Пример гистограммы яркостей для изображения.

Для улучшения различаемости дескрипторов могут использоваться различные формулы для вычисления яркости в точке, например, можно дать больший вес для зеленого цвета, так как человек к нему наиболее чувствителен [12]. Можно усложнить метод построением сразу трех гистограмм для цветовых каналов – красного, зеленого и синего, однако чаще всего задаются специальные формулы для яркости, позволяющие строить максимально различающиеся гистограммы для различных изображений. Часто применяются различные нормализации яркостей или цветов, которые позволяют получать сходные гистограммы даже с учетом изменения цветового баланса на вариациях изображения [13]. Также можно разделить изображение на блоки и считать гистограмму в каждом блоке отдельно, при этом, конечно, длина дескриптора увеличится, однако это позволяет провести идентификацию даже при наличии добавленных объектов (как пример, копирайты или текст), так как они изменят один или два

Рис. 2: Пример характерного засвета на полотне при фотографировании.

блока из многих. Этот метод прост, быстр, и, в принципе, позволяет искать оригинал изображения при отсутствии искажений, однако в случае с реальными фотографиями и картинами гистограммный метод не работает, так как на фотографиях присутствует сильные засветы от сопутствующего освещения и масляного материала красок, который очень сильно отражает свет. Кроме того, камеры нелинейно меняют баланс цвета. Даже если предположить, что библиотека сможет выровнять яркости, то остаются засветы, которые занимают от 30% изображения и неравномерно меняют яркость из-за мазков на картине, как видно на рис. 2.

2.1.2. Matrix of Brightness Variation

Другой «простой» подход — это построить матрицу варьирования яркостей (Matrix of Brightness Variation, или MBV) [1]. Если $I(x, y)$ – это матрица яркостей изображения, то MBV задается следующим образом:

$$M(x, y) = \left[sgn \frac{\partial I}{\partial x} \; sgn \frac{\partial I}{\partial y} \right]_{(x,y)} \tag{1}$$

, где $M(x, y)$ - это значение матрицы в точке (x, y), $I(x, y)$ - это значение яркости в точке (x, y).

Данный метод стабильнее гистограмм в том смысле, что он основан на изменении яр-

кости, но не на абсолютных их значениях, поэтому при равномерных изменениях цветов дескриптор не изменится. Несмотря на то, что на синтетических тестах точность высока, данный подход не работает на реальных фотографиях из-за неравномерности засветов, которая была подробно описана в предыдущем пункте (как указано в пункте 6 [1]). Таким образом, этот метод также не подходит для решения задачи.

2.1.3. Вейвлет-преобразование

Еще один распространенный метод основан на вейвлет-преобразованиях [7]. Его идея в том, что при использовании специальной вейвлет-функции возможно получить уменьшенные в два раза по ширине и высоте матрицы, одна из которых — текстурные детали, а три другие — вертикальные, горизонтальные и объединенные пики яркостей.

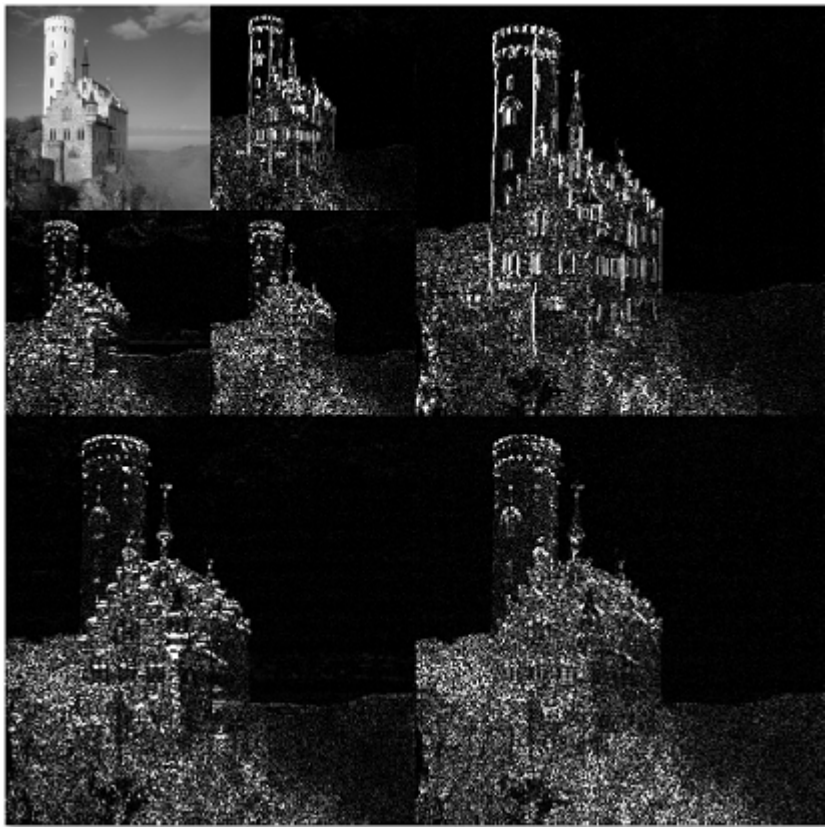

Рис. 3: Пример двумерного вейвлет-преобразования изображения.

Итоговым дескриптором будет выступать наименьшая матрица яркостей, объединенная с матрицами вертикальных, горизонтальных и объединенных пиков яркостей, которые будут получены при последнем применении вейвлет-преобразования. Данный алгоритм хорош тем, что позволяет получать дескрипторы любого достаточно малого размера, однако он требует предварительного изменения размера изображения, так как длина дескриптора явно зависит от ширины и высоты изображения, и при поиске среди вейвлет-дескрипторов необходим одинаковый их размер. Другим относительным недостатком является достаточно большая вычислительная сложность вейвлет-преобразования в сравнении с предыдущими.

2.1.4. Проблемы глобальных дескрипторов

Все глобальные методы построения дескриптора страдают от общих проблем: изменение ракурса камеры и засветы отдельных участков фотографии. Так как дескриптор строится по всему изображению целиком, важно, чтобы на двух сходных тестовых фотографиях соответствующие области были в одинаковых местах, однако фотограф не обязательно будет фотографировать с одной и той же точки под одним и тем же углом к плоскости полотна, из-за чего одна фотография будет смещена относительно другой.

Частично эта проблема может быть решена предобработкой, как, например, выделение углов полотна и разворот изображения во фронтальную плоскость. Однако далеко не всегда получается с достаточной точностью определить границы полотна, отчего будут присутствовать сдвиги, которые будут добавлять отклонение в дескриптор относительно оригинального. Проблема с засветами серьезней, так как из-за особенностей строения полотна и освещения музея засветы получаются относительно большими и нелинейно изменяющими цвета, из-за чего очень затруднительно их локализовать и убрать. Блочный подход к построению дескриптора позволяет сгладить данную проблему, однако требуется соблюдать баланс между точностью и количеством блоков – если блоков взять слишком много, то итоговый дескриптор будет занимать большое количество памяти и будет сложно организовать эффективный поиск. Однако, если блоков будет мало, то слишком большой процент будет испорчен засветами (так как даже при засвете небольшого участка блока дескриптор этого блока, скорее всего, будет безнадежно испорчен). Более того, из-за неточного определения границы одни и те же блоки на сходных изображениях могут выделять немного отличающиеся участки изображения, что приводит к увеличению различия между дескрипторами этих блоков.

2.2. Локальные дескрипторы

Другой подход к построению дескриптора изображения — выделение некоторого количества точек, про которые можно сказать, что они, скорее всего, будут определены как на оригинальном изображении, так и на его вариациях, и описать локальную область вокруг каждой из точек [17, 3, 4]. При выборе достаточно большого количества точек какая-то их часть будет находиться в не испорченных областях изображения, и их можно будет корректно сопоставить с точками из оригинала.

2.2.1. SIFT

Самым известным алгоритмом подобного сорта является Scale Invariant Feature Transform (SIFT) [3]. Поиск особых точек выполняется детектором на основе разностей гауссовых размытий изображения. Для каждой определенной точки считается угол градиента яркости, который вычисляется по следующей формуле:

$$\text{atan2}\ (L(x, y+1) - L(x, y-1), L(x+1, y) - L(x-1, y)),$$

где $L(x, y)$ - это яркость пиксела в смазанном изображении. Все полученные особые точки сортируются по норме доминирующего градиента, и при малом его значении точка отбрасываются как нестабильная. Для дальнейшей обработки берется несколько точек с максимальными значениями нормы градиента, которые являются наиболее стабильными для идентификации. В качестве дескриптора используется HOG дескриптор [5] локальных областей, задаваемый следующим образом: берется квадратная область на 16x16 пикселей вокруг особой точки, которая поворачивается на угол доминирующего градиента (делается это для получения независимости дескриптора от поворота), делится на 16 блоков размера 4x4 пикселя, в каждом блоке происходит нормализация, после чего считается HOG дескриптор каждого блока с 8 опорными значениями. Таким образом, на одну точку генерируется 128 значений SIFT дескриптора, с учетом угла доминирующего градиента итоговый дескриптор составляет 129 целых чисел (угол для простоты округляется до ближайшего целого градуса).

2.2.2. SURF

Другим известным алгоритмом является Speeded Up Robust Features (SURF) [4]. Метод позиционируется как улучшенный вариант SIFT метода с увеличенной скоростью детектирования и построения дескриптора. Основные стадии алгоритма совпадают с SIFT: детектируются особые точки, считается угол доминирующего градиента, и считается дескрипор локальной области. Для детектирования особых точек считается определитель матрицы Гессиана и определяются локальные максимумы. Аналогично с SIFT, отбрасываются по пороговому значению определителя нестабильные точки и для дальнейшей обработки берутся наиболее стабильные с максимальными значениями определителя. Вместо HOG дескриптора локальных областей используются значения вейвлет-преобразования Хаара. Как указано в [4], SURF быстрее SIFT, однако в условиях данной задачи при его использовании теряется точность идентификации.

2.3. Поиск в дескрипторах

Полученные наборы дескрипторов особых точек можно обрабатывать одним из следующих способов: или задать n классов эквивалентности локальных дескрипторов точек, или же делать поиск среди точек. Первый способ называется Bag of Words (BoW) [6], и при его использовании длина дескриптора изображения будет равна количеству классов эквивалентности. Данный метод хорош при правильном выборе классов, однако в поставленной задаче он не работает по двум причинам:

1. Искажения в изображениях слишком сильные, и для двух вариаций одного и того же полотна множества совпадающих особых точек может иметь пересечение меньше 50%, из-за чего дескриптор меняется очень сильно;

2. Дескрипторы состоят из 128 чисел, и достаточно сложно выбрать классы так, чтобы

они адекватно разделяли множество особых точек, так как малое число классов может не учитывать многие различия между точками, а слишком большое число классов потребует большого количества памяти для хранения итогового дескриптора, и затраты вычислительных мощностей будут критически большими.

Поэтому было решено остановиться на поиске среди точек. Схема идентификации основана на работах [17, 9] и состоит в следующем:

1. На обрабатываемом изображении детектируются особые точки;
2. Для каждой задетектированной особой точки строится дескриптор;
3. Для каждой точки из идентифицируемого изображения производится поиск ближайшей точки из базы;
4. В качестве результата возвращается информация про изображение из базы с наибольшим количеством ближайших точек с идентифицируемым изображением.

Так как в дескрипторах сохраняется угол доминирующего градиента точки и считается, что пользователь не будет снимать картины под большим углом, то при поиске ближайшей точки есть возможность отбросить большую часть базы по углу градиента. Тем не менее, даже с такой оптимизацией идентификация одной фотографии требует много времени, так как приемлимая точность идентификации достигается при количестве точек порядка 50-ти, и при размере базы в 15 000 изображений в базе дескрипторов хранится 750 000 дескрипторов особых точек. Из-за этого приходится использовать хеширование дескрипторов для уменьшения множества точек, которые требуется обработать. Стоит заметить, что необходим особый класс хеш-функциях, которые дают одинаковые или сходные значения для похожих дескрипторов, что позволит обрабатывать только неким образом похожие друг на друга дескрипторы. Таким образом, стандартные известные хеш-функции, примером которых является MD5, не подходят. Конкретно в данной работе используется Locality Sensitive Hashing [2], который позволяет гибко настраивать чувствительность хеш-функции к изменению исходных данных. Метод заключается в следующем - рассматривая дескриптор как битовую строку (считается, что все значения дескриптора приводятся к целочисленному типу) задается некоторое количество специальных хеш-функций, каждая из которых выделяет некоторое подмножество бит из битовой строки и объединяет их, получая некоторое другое число, которое будет являться искомым значением хеша для данного дескриптора. После чего поиск ближайшего дескриптора будет производиться только среди тех особых точек из базы, у которых хеш хотя бы одной хеш-функции будет равен хешу обрабатываемой точки.

3. Алгоритм поиска

3.1. Построение дескриптора изображения

Общая схема построения дескриптора изображения выглядит следующим образом:

1. Изменение размера изображения до 256x256 пикселей;
2. Поиск особых точек;
3. Выбор n самых стабильных точек из найденного набора;
4. Построение локального дескриптора каждой точки;
5. Сохранение всех локальных дескрипторов как общего дескриптора изображения.

Остановимся на каждом пункте подробнее. Уменьшением размера изображения мы добиваемся ускорения детектирования особых точек, при этом теряются только нестабильные точки (в любом случае не вошедшие бы в дескриптор). Однако это накладывает огрничение на распознаваемые изображения – на них полотно должно иметь те же или близкие пропорции, что и на образце в базе, иначе даже локальные области вокруг точки будут изменены, что испортит дескриптор изображения. Конечно, можно использовать любой другой размер изображения, но этот размер является балансом между точностью идентификации и скоростью обработки: Так как и SIFT, и SURF строят дескриптор длины 128 целых чисел, то они

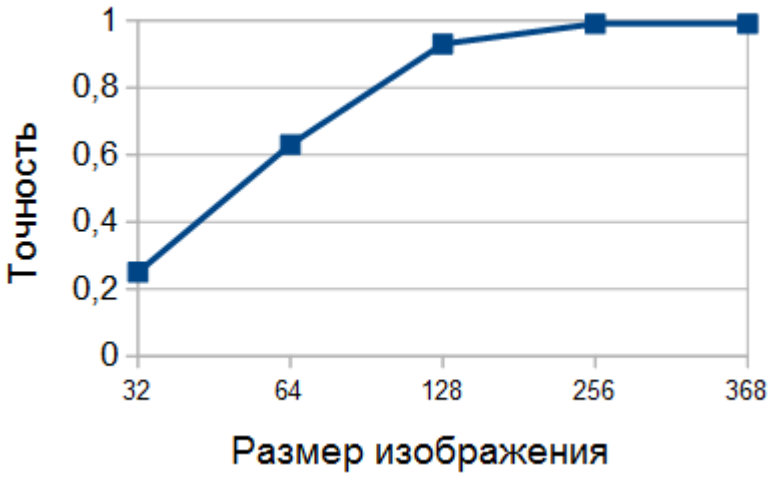

Рис. 4: График точности идентификации при различных изменениях размера изображения

обрабатываются сходным образом: выделяется некоторое заранее определенное количество точек, для каждой строится дескриптор длины 128, к которому приписывается округленный до целого угол доминирующего градиента. После чего полученный дескриптор особой точки вместе с информацией про исходное изображение сохраняется в базе.

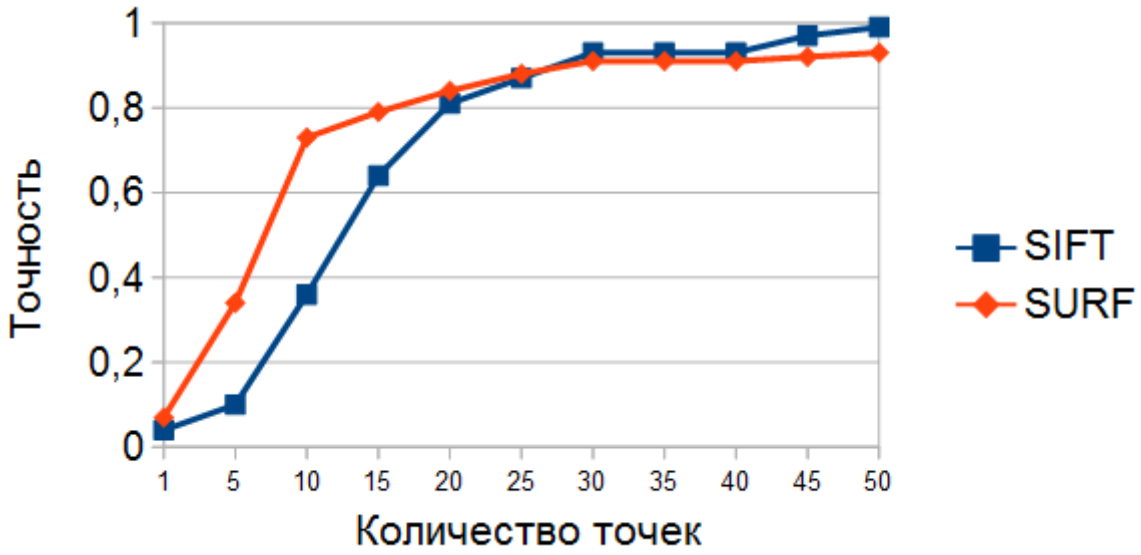

Рис. 5: График точности идентификации при различных количествах выделямых особых точек

Как видно, для слишком маленького количества особых точек нельзя гарантировать высокий шанс на идентификацию, поэтому было решено остановиться на 50-ти. Если брать больше, то точность не повышается, однако увеличиваются расходы на поиск и хранение, что нецелесообразно. На этой стадии проходит первая оптимизация – все точки классифицируются по углу градиента, который вычисляется на этапе построения дескриптора изображения, с шагом в пять градусов. Таким образом, все точки попадают в один из 72 классов эквивалентности, и в будущем при поиске будут обрабатываться только те точки из базы, которые находятся либо в таком же классе, что и целевая особая точка, либо в одном из двух соседних. Данная оптимизация позволяет обрабатывать только 5% базы, что дает существенное ускорение идентификации.

3.2. Идентификация

Поиск проходит в несколько стадий:

1. Построение дескриптора изображения;
2. Для каждого локального дескриптора точки:
   (a) Поиск ближайшей точки;
   (b) Сохранение идентификатора дескриптора связанной точки из базы.
3. Поиск дескриптора с наибольшим пересечением по особым точкам.

Процесс построения дескриптора подробно описан в предыдущем пункте, при этом для идентифицируемого изображения он не сохраняется в базе. Особые точки хранятся в базе с

классификацией по углу градиента, что позволяет уменьшить количество обрабатываемых точек. Далее ищется ближайшая по метрике точка из этой части базы для исходной особой точки идентифицируемого изображения, и связанный идентификатор изображения сохраняется. Таким образом, для каждой особой точки определяется изображение из базы, в котором находится наиболее похожая особая точка. После завершения обработки изображения в памяти хранится набор идентификаторов оригинальных сканов, по которым находится скан, в котором находится наибольшее количество похожих особых точек, который возвращается как результат. Так как выбирается достаточно большое количество точек, то с большой вероятностью найдутся такие точки, которые находятся на неиспорченных частях изображения и для которых получится определить ближайшую точку из базы, при этом для точек, которые находятся на испорченных частях изображения ближайшими точками будут точки из различных сканов базы, и оригинальный скан будет иметь наибольшее пересечение по особым точкам.

3.3. Locality Sensitive Hashing

Наивная реализация поиска с перебором базы работает крайне медленно (на 2.9 GHz процессоре идентификация одного изображения с базой размера 15 000 изображений может занимать до 5-ти минут даже с учетом фильтрации по углу), так как при этом для каждой особой точки из идентифицируемого изображения ближайшая ищется среди в среднем 40 000 точек из базы, поэтому было решено использовать Locality Sensitive Hashing [2]. Общая схема алгоритма:

1. Создается n хеш функций специального вида;
2. Для каждой особой точки из идентифицируемого изображения:
   (a) Для каждой хеш функции:
      i. Считается хеш особой точки;
      ii. Среди всех особых точек с таким же хешом ищется ближайшая по метрике;
      iii. (Если хеш функция не первая) Найденная лучшая точка сравнивается с предыдущей лучшей точкой и выбирается ближайшая по метрике;
   (b) Связанный с ближайшей особой точкой идентификатор сохраняется;
3. Изображение с наибольшим пересечением по особым точкам возвращается как ответ.

Хеш функция строится следующим образом — дескриптор особой точки представляется как битовая строка (так как все числа в дескрипторе целые, то результирующая битовая строка будет конкатенацией битовых представлений целых чисел), далее из этой битовой строки выделяется m бит, которые образуют некоторое число N, возможно, достаточно большое.

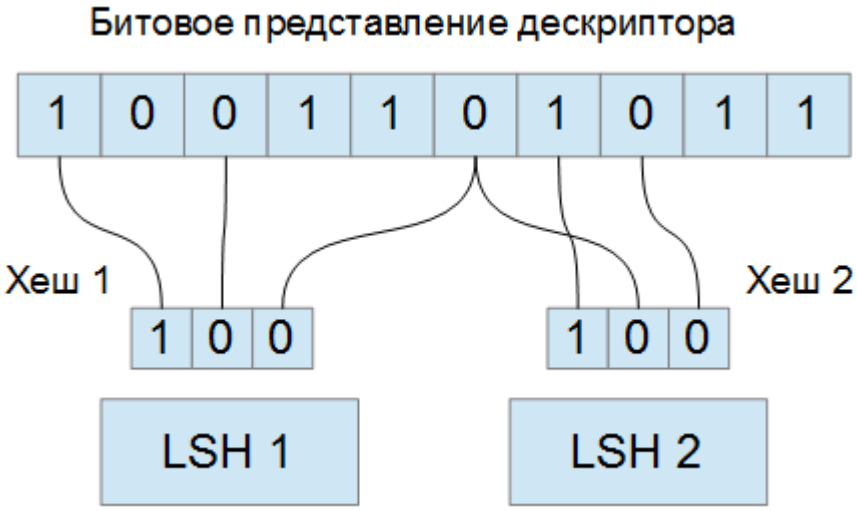

Рис. 6: Схема построения хешей

При идентификации будут обрабатываться только те особые точки, чьи дескрипторы имеют совпадающий хеш. Так как число может быть очень велико, то является разумным использвоание hash map для ускорения обработки. В зависимости от выбора маски чувствительность хеш функции может быть как крайне высокой (если выбирать нижние биты для числа из дескриптора), так и низкой (при выборе старших бит).

В ходе исследования было выяснено, что оптимальной стратегией является выбор старших варьирующихся бит у некоторых чисел из дескриптора с одновременным игнорированием тех точек, у которых все биты будут нулевыми (таких точек намного больше, нежели остальных, и их игнорирование не понижает точности идентификации, однако при этом они очень сильно замедляют скорость обработки). Так как выделяются только стабильные точки, которые имеют сильное изменение в яркости в своей области, то дескриптор будет иметь пики в некоторых своих значениях, и с большой вероятностью какая-нибудь из хеш функций будет иметь для данного дескриптора ненулевое значение, и особая точка будет идентифицирована корректно.

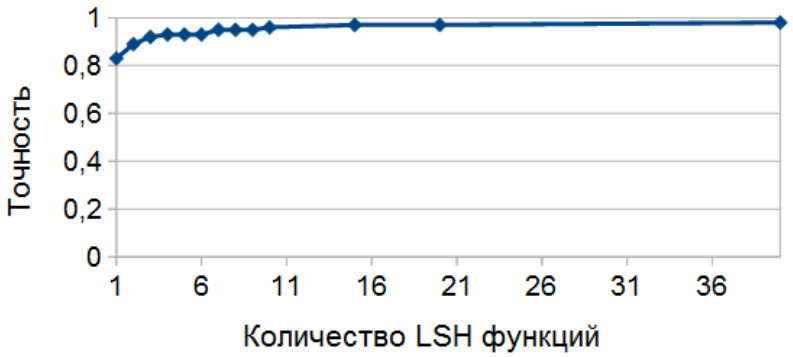

Рис. 7: График изменения точности идентификации в зависимости от количества LSH фунций

Не смотря на то, что уже при 15-ти хеш-функцях достигается практически максимальная точность (как видно на рис. 7), использование большего количества функций оправдано, так как при этом увеличивается количество верно сопоставленных особых точек даже для уже корректно идентифицированных изображений, что повышает устойчивость идентифицирования для других, возможно менее качественных фотографий.

4. Тестирование

Для проверки точности метода использовалась база фотографий живописных полотен из 17 000 элементов, полученная с сайта http://gallerix.ru [дата просмотра: 04.04.2015], в качестве тестовой базы использовались фотографии, сделанные автором, а также фотографии некоторых картин из базы, не вошедшие в идентификационный набор. Предварительно тестовые изображения обрезались по раме, но при этом не использовались поворот и растяжение полученного изображения. Полученная база для идентификации имела суммарный размер в 304 элемента для 25 различных картин, где для каждой картины было от десяти до пятнадцати фотографий, сделанных под разными ракурсами, на которой была достигнута точность в 98% корректных распознаваний. Под точностью подразумевается отношение верно идентифицированных изображений к общему количеству изображений для идентификации. Изображение считается верно идентифицированным, если первая картина из полученного результата есть оригинальный скан сфотографированной картины.

Обрезание изображения по раме настоятельно рекомендуется, так как из-за изменения размера изображения полотно меняет пропорции и дескрипторы точек сильно меняются, что вносит искажения. Также при наличии объектов между камерой и картиной особые точки могут детектироваться неправильно, так как имеются сильные перепады яркостей между зоной полотна и зоной объекта. Несмотря на то, что большая часть картин написаны маслом на холсте и в помещении присутствуют сильные источники света, которые могут засветить до 30% изображения, практически всегда удавалось определить оригинал изображения:

Рис. 8: Примеры изображений, для которых оригинал был корректно идентифицирован.

Однако, при слишком сильном засвете гарантировать корректность распознавания не удается, что видно на следующем примере:

Рис. 9: Пример изображения с засветом большей части изображения

Метод показал определенную стойкость к смазу, однако слишком сильный смаз портит изображение достаточно, чтобы библиотека не смогла достаточно точно определить и сопоставить особые точки:

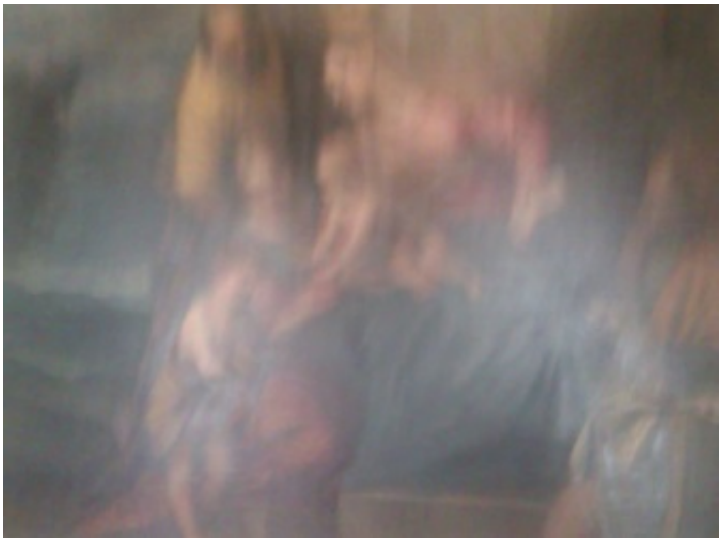

Рис. 10: Пример изображения с сильным смазом

Наивная реализация с перебор в среднем требовала до 5-ти минут машинного времени на процессоре с частотой 2.9 GHz, тогда как метод с LSH требует в среднем 3 секунды на одну идентификацию для SIFT дескриптора и в среднем 1.8 секунды для SURF дескриптора:

	Точность	Ср., мс	Мин., мс	Макс., мс	Ср. кв. откл., мс
SIFT	300/304 (98.6%)	250000	220000	270000	6000
SIFT + LSH	297/304 (97.7%)	3000	900	7000	830
SURF + LSH	282/304 (92.8%)	1800	500	3100	380

Как видно, SURF метод быстрее SIFT где-то в два раза, однако его точность ниже.

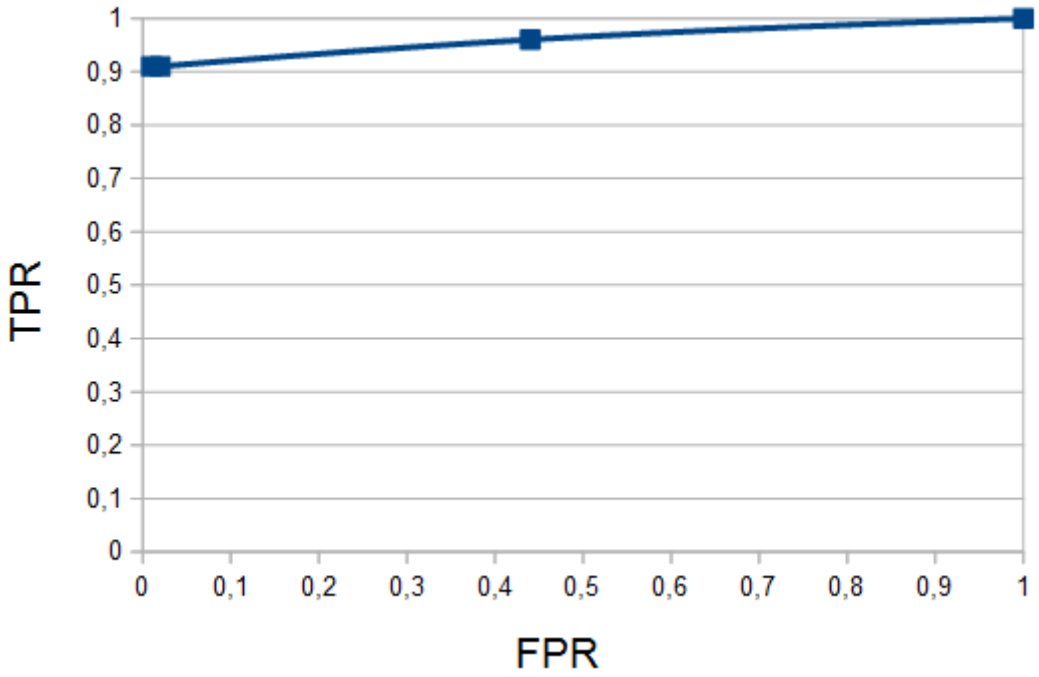

Рис. 11: ROC кривая для классификации “находится в базе” с отсечением по количеству точек (TPR - True Positive Rate, FPR - False Positive Rate)

Как видно из ROC-кривой, метод достаточно надежно определяет, находится ли изображение в базе. Это происходит потому, что если изображения в базе нет (как и его близких вариантов), то ближайший образец из базы будет иметь в районе двух общих точек (около 1% случаев трех точек, примерно 50% двух общих точек и примерно 50% одной общей точки), тогда как больше 90% вариаций имеющихся в базе изображений будут иметь 4 и более точек. Для каждой картины из идентифицируемой выборки можно было найти такую фотографию, которая будет иметь 5 или больше общих точек со своим аналогом в базе.

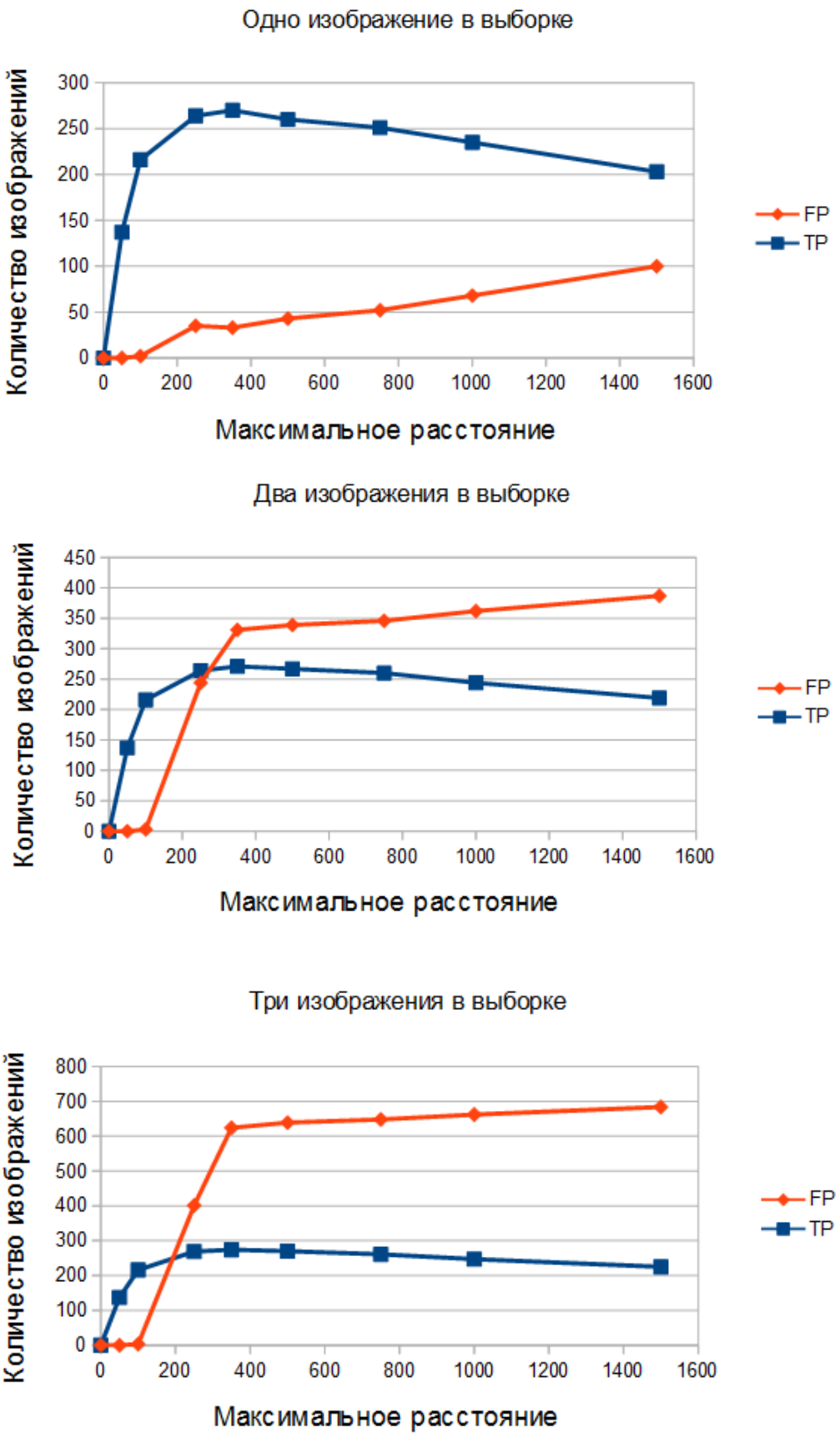

Рис. 12: Количество правильных/неправильных изображений в выборке при отсечении точек по расстоянию (TP - True Positive, FP - False Positive)

Как видно, если выбирать не наиближайшую точку, а множество точек с ограничением по расстоянию, то точность понижается до 90% (т.к. идентифицируется 304 изображения, но корректных идентификаций около 270), при этом увеличение выборки влияет на точность незначительно.

Заключение

Результаты данной работы:

1. Собран набор изображений для проверки эффективности методов идентификации изображения по реальной фотографии;
2. Реализована библиотека на языке C++ для обработки и поиска оригиналов в базе по фотографиям:
   (a) Реализован модуль для построения дескриптора изображения с использованием реализаций SIFT и SURF из OpenCV;
   (b) Реализован модуль для поиска оригинала изображения по полученному дескриптору;
   (c) Реализован метод локально чувствительного поиска для ускорения поиска в базе;
3. Были проведены тесты над собранным бенчмарком и показана эффективность данного метода для решения задачи идентификации живописного полотна по фотографии.

В будущем планируется исследование эффективности при использовании других дескрипторов локальных точек (таких как PCA-SIFT [17, 9], Local-Difference-Pattern [16], Quantified SIFT Descriptor [15]) и реализация методов более эффективного поиска, таких как min-hash [11, 10] и других. Также планируется реализация эффективной параллелизации идентификации.

Список литературы

[1] A. Goncharov, A. Melnichenko. Pseudometric Approach to Content Based Image Retrieval and Near Duplicates Detection // Evaluation of Methods of Data. — 2008. — P. 120–134.

[2] Aristides Gionis, Piotr Indyky, Rajeev Motwaniz. Similarity Search in High Dimensions via Hashing // Proceedings of the 25th VLDB Conference. — 1999. — P. 518–529.

[3] David G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints // International Journal of Computer Vision. — 2004. — Vol. 60(2). — P. 91–110.

[4] H. Bay, T. Tuytelaars, L. V. Gool. SURF: Speeded Up Robust Features // Lecture Notes in Computer Science. — 2006. — Vol. 3951. — P. 404–417.

[5] Intel. Histogram of Oriented Gradients (HOG) Descriptor // Intel, Developer zone. — 2012. — URL: https://software.intel.com/en-us/node/529070 (online; accessed: 08.05.2015).

[6] J. Philbin, O. Chum, M. Isard, J. Sivic. Object retrieval with large vocabularies and fast spatial matching // Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on. — 2007. — P. 1–8.

[7] James Ze Wang, Gio Wiederhold, Oscar Firschein, Sha Xin Wei. Content-based image indexing and searching using Daubechies' wavelets // International Journal on Digital Libraries. — 1998. — Vol. 1(4). — P. 311–328.

[8] Jun Jie Foo, Justin Zobel, Ranjan Sinha, S.M.M. Tahaghoghi. Detection of Near-duplicate Images for Web Search // CIVR '07 Proceedings of the 6th ACM international conference on Image and video retrieval. — 2007. — P. 557–564.

[9] Jun Jie Foo, Ranjan Sinha. Pruning SIFT for Scalable Near-Duplicate Image Matching // ADC '07 Proceedings of the eighteenth conference on Australasian database. — 2007. — Vol. 63. — P. 63–71.

[10] Ondrej Chum, James Philbin, Andrew Zisserman. Near Duplicate Image Detection: min-Hash and tf-idf Weighting. — 2008.

[11] Ondrej Chum, James Philbin, Michael Isard, Andrew Zisserman. Scalable Near Identical Image and Shot Detection. // CIVR '07 Proceedings of the 6th ACM international conference on Image and video retrieval. — 2007. — P. 549–556.

[12] R. Szeliski. Computer Vision: Algorithms and Applications. 2.3.2 Color. Color balance. — 2010. — PDF : http://szeliski.org/Book/.

[13] R. Szeliski. Computer Vision: Algorithms and Applications. 3.1.4 Histogram equalization. — 2010. — PDF : http://szeliski.org/Book/.

[14] Sebe, N, Lew, M.S. Color-based retrieval. // Color-based retrieval. — 2001. — Vol. 21. — P. 223–230.

[15] Wengang Zhou, Yijuan Lu, Houqiang Li, Yibing Song, Qi Tian. Spatial Coding for Large Scale Partial-Duplicate Web Image Searc // MM ’10 Proceedings of the international conference on Multimedia. — 2010. — P. 511–520.

[16] Xin Yang, Qiang Zhu, Kwang-Ting Cheng. Near-Duplicate Detection for Images and Videos // LS-MMRM ’09 Proceedings of the First ACM workshop on Large-scale multimedia retrieval and mining. — 2009. — P. 73–80.

[17] Yan Ke, Rahul Sukthankar, Larry Huston. Efficient near-duplicate detection and sub-image retrieval // MULTIMEDIA ’04 Proceedings of the 12th annual ACM international conference on Multimedia. — 2004. — P. 869–876.