МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Институт математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование)

02.03.03 Математическое обеспечение и администрирование информационных систем
(код и наименование направления подготовки, специальности)

Технология программирования
(направленность (профиль) / специализация)

# ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему «Разработка приложения для сравнительного анализа дескрипторов особых точек изображений»

Студент Д.В. Лопатин
(И.О. Фамилия) (личная подпись)

Руководитель к.т.н, доцент, В.С. Климов
(ученая степень, звание, И.О. Фамилия)

Консультант М.В. Дайнеко

Тольятти 2020

# АННОТАЦИЯ

Тема данной бакалаврской работы: «Разработка приложения для сравнительного анализа дескрипторов особых точек изображений».

Работа посвящена проблеме сравнения дескрипторов особых точек изображений.

Объектом исследования является сравнительный анализ дескрипторов особых точек изображений. Предметом исследования является обнаружение особых точек изображений и последующая их обработка с помощью дескрипторов особых точек.

Целью работы является сравнение эффективности работы дескрипторов особых точек путём подсчета точности и скорости их работы.

Во введении дается краткий обзор проблемы сравнения дескрипторов особых точек, описываются цели и задачи работы.

В первой главе проводится анализ состояния вопроса. Осуществляется краткий обзор исследуемой области, проводится анализ недостатков существующих дескрипторов особых точек.

Во второй главе проводится анализ алгоритмов дескрипторов, и проектирование программного обеспечения для сравнительного анализа дескрипторов особых точек изображений.

В третьей главе проводится программная реализация предложенных решений. Производится подведение результатов работы дескрипторов особых точек, и описываются возможности по улучшению программы.

В заключении приводятся выводы по работе дескрипторов особых точек на разных типах изображений, а также предлагаются решения по масштабированию программного обеспечения.

Бакалаврская работа содержит пояснительную записку объемом 45 страниц, включая 34 рисунка, 2 таблицы и 9 формул, а также список литературы из 22 источников.

## ABSTRACT

The present graduation work is devoted to developing an application for comparative analysis of special pixel descriptors.

The research is dedicated to comparing the special pixel descriptors.

The graduation work consists of an explanatory note, 34 figures, 2 tables, 9 formulae and a list of 22 references.

The object of the research is the comparative analysis of the special pixel descriptors.

The subject of the research is the detection of the special pixels and their subsequent processing by using the special pixel descriptors.

The aim of the investigation is to compare the performance of the special pixel descriptors by calculating the accuracy and speed of their work.

The introduction provides a brief overview of the issue of comparing the special pixel descriptors, as well as sets the goals and objectives of the work.

The first chapter analyzes the state of the issue, gives a brief review of the studied area. In this chapter, the analysis of the existing special pixel descriptors shortcomings is carried out as well.

In the second chapter, the descriptor algorithms analysis is conducted. The chapter also deals with the software development for the comparative analysis of the special pixel descriptors.

The third chapter reveals the software implementation of the proposed solutions. In this chapter, the results of the special pixel descriptors operation are summarized and the opportunities for improving the software are described.

The graduation work presents some conclusions relating to the special pixel descriptors operation in case of different types of images and proposes the solutions for scaling the software.

**СОДЕРЖАНИЕ**

# ВВЕДЕНИЕ

Основной задачей компьютерного зрения является распознавание различных объектов на видеопотоках и изображениях. Первым этапом в распознавании объектов является нахождение особых точек изображения, являющихся своеобразными метками для последующих этапов распознавания. Особыми точками являются места резкого изменения яркости изображения: углы зданий, края силуэтов, границы объектов.

Обработка и сравнение особых точек изображений с помощью дескрипторов позволяют решать задачи распознавания и отслеживания объектов, стабилизации видео, генерации панорам, а также создавать трёхмерные реконструкции объектов с помощью дополненной реальности.

Новизна работы заключатся в использовании нового алгоритма сравнения дескрипторов особых точек.

Целью работы является сравнение эффективности работы дескрипторов особых точек на различных типах изображений путём подсчета точности и скорости их работы.

Объектом исследования является сравнительный анализ дескрипторов особых точек изображений.

Предметом исследования является обнаружение особых точек изображений и последующая их обработка с помощью дескрипторов особых точек.

# 1 АНАЛИЗ СОСТОЯНИЯ ВОПРОСА

## 1.1 Обзор предметной области

В наше время цифровая обработка изображений играет важную роль практически в любой технической сфере деятельности человека. С каждым годом обработка изображений становится доступна большему количеству конечных пользователей, позволяя открывать новые области применения данной технологии [1]. С другой стороны, возрастающее количество графической информации увеличивает время её обработки, хранения и изменения. Оптимизация обработки изображений является одной из ключевых задач новых исследований.

Для сравнения и идентификации изображений строятся специальный характеристики, основанные на найденных особых точках изображения.

Характеристика (feature) - это часть информации, которая важна для решения вычислительной задачи, связанной с определенным приложением [2]. Объектами характеристики могут быть конкретные структуры изображения, такие как точки или края. Характеристика также может быть результатом операции обнаружения характерных черт, примененных к изображению [4].

Однако каждая точка будет вносить собственный вклад в характеристику изображения, независимо от того, полезен этот вклад или нет [3]. Из-за этого характеристику строят по так называемым особым точкам (ключевым). Пример особых точек представлен на рисунке 1.1.

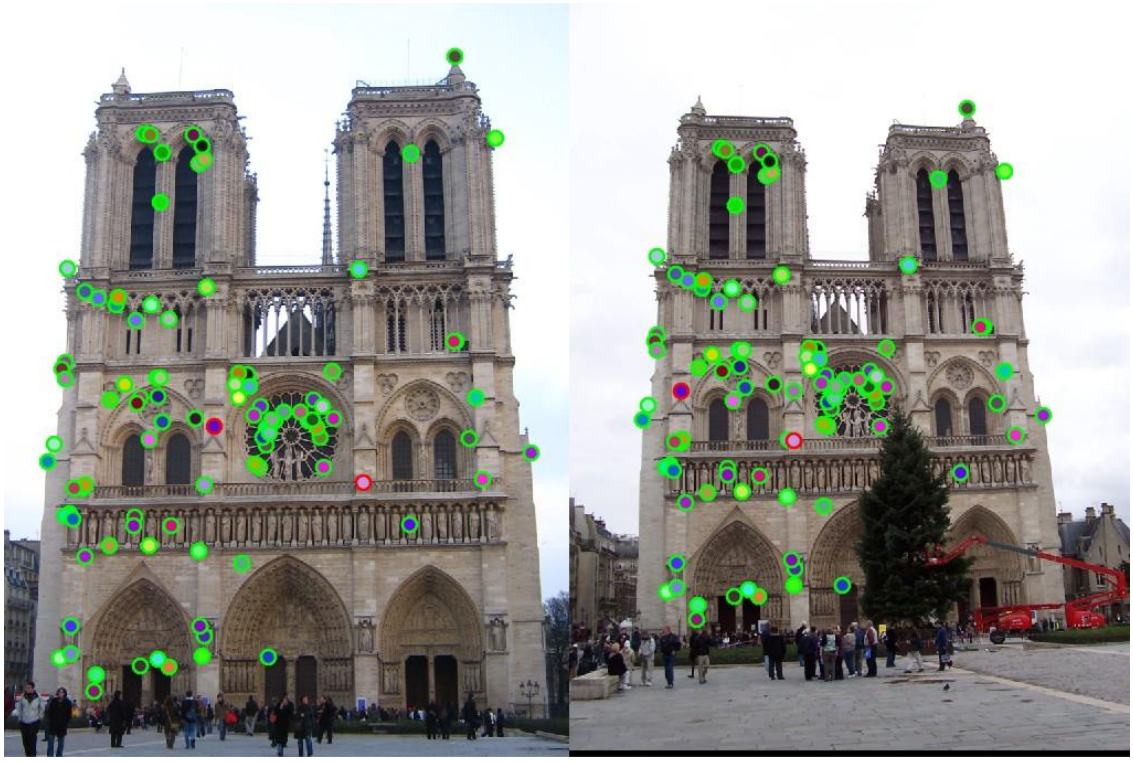

Рисунок 1.1 – Особые точки на изображении

Каждая особая точка имеет ряд свойств:

- определенность – степень отличия конкретной особой точки от соседних точек;
- устойчивость – изменение яркости, контрастности и цветовой гаммы не должны влиять на место особой точки на объекте или сцене;
- инвариантность – особые точки должны обладать устойчивостью к повороту, изменению масштаба изображения и смене ракурса съемки;
- стабильность – зашумленность изображения, не превышающая определенный порог, не должна влиять на работу детектора;
- интерпретируемость – особые точки должны быть представлены в формате, пригодном для дальнейшей работы [6].

Для нахождения ключевых точек используются специальные методы - детекторы. Однако детекторы могут определить лишь координаты особых точек, которые отличаются в каждом изображении [7].

Поэтому в дополнении к детекторам используются дескрипторы – специальные идентификаторы особых точек, выделяющие одну конкретную точку от множества остальных. Таким образом, при сравнении изображений сначала с помощью детекторов находятся и выделяются особые точки. После этого находится дескрипторы для каждой точки. В итоге, находятся соответствующие друг другу особые точки, по которым и сравнивается изображение.

Дескрипторы особых точек имеют широкую область применения. В основном они нужны для решения следующих задач:

- распознавание и отслеживание объектов;
- роботизированная навигация;
- калибровка камер;
- стабилизация видео;
- генерация панорам;
- трёхмерная реконструкция объектов в системах дополненной реальности.

Следующим этапом анализа изображения является сопоставление признаков изображений или самих изображений [5].

После того как элементы и их дескрипторы извлечены из двух или более изображений, устанавливаются некоторые предварительные совпадения между этими изображениями. Пример сопоставления особых точек представлен на рисунке 1.2.

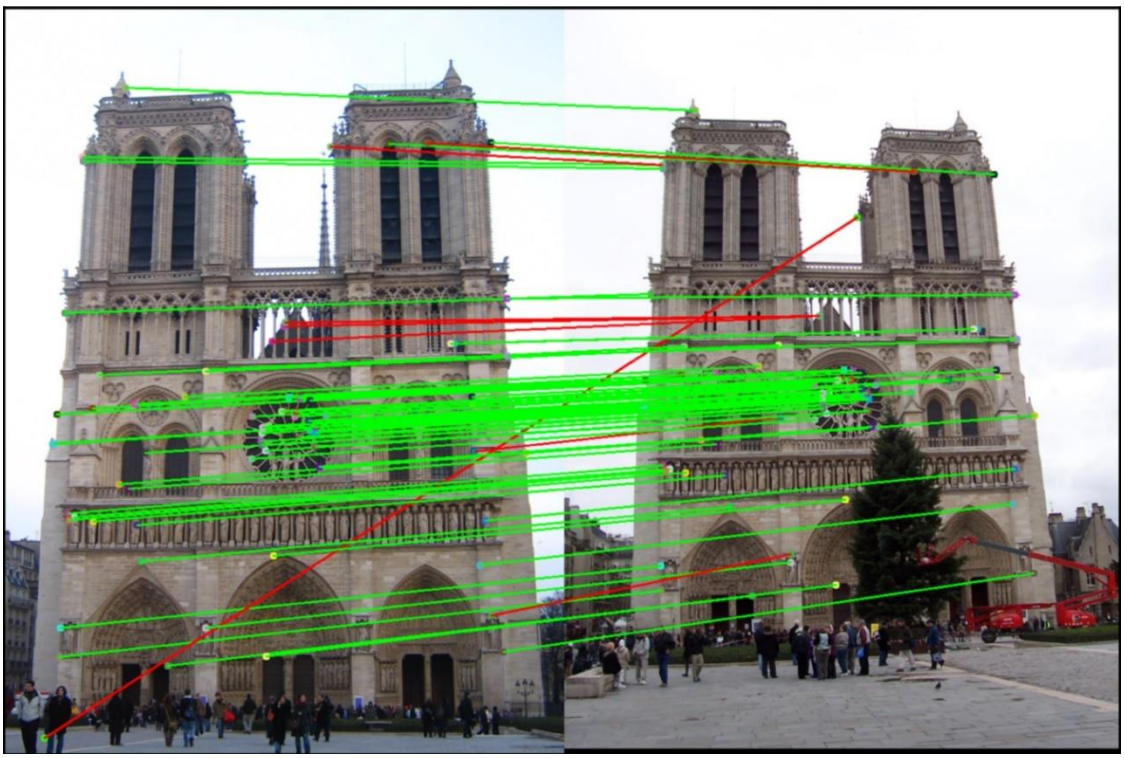

Рисунок 1.2 – Сопоставление особых точек изображений

Как правило, эффективность методов сопоставления, основанных на особых точках, зависит как от свойств базовых особых точек, так и от выбора связанных дескрипторов изображений [9]. Таким образом, детекторы и дескрипторы, соответствующие содержанию изображений, должны корректно использоваться в приложениях [8]. Например, если изображение содержит клетки бактерий, следует использовать детектор капли, а не детектор угла. Но если это изображение города, сделанное с высоты птичьего полета, необходимо использовать угловой датчик для поиска искусственных сооружений [11].

Итак, алгоритм нахождения и сопоставления особых точек выглядит следующим образом:

- найти набор отличительных ключевых точек;
- определить область вокруг каждой ключевой точки;
- извлечь и нормализовать содержимое региона;
- вычислить локальный дескриптор из нормализованной области;
- сопоставить локальные дескрипторы.

Из всего перечисленного можно сделать вывод, что дескрипторы особых точек используются в довольно широком спектре разнообразных задач самой разной направленности и сложности. Необходимо использовать различные детекторы особых точек в соответствии с поставленными задачами.

Таким образом, было описаны понятия дескрипторов, детекторов и ключевых точек. Были проанализированы существующие области применения дескрипторов особых точек.

### 1.2 Постановка задачи

Отталкиваясь от обзора классических задач, для выполнения которых применяются дескрипторы особых точек изображения, можно сделать вывод, что довольно часто дескрипторы используются бездумно, без чёткого понимания направленности конкретного дескриптора. Например, для распознавания людей на изображениях может использоваться дескриптор, который хорошо распознаёт прямые линии (здания, искусственные объекты) и совершенно не подходит для распознавания людей [12]. Это приводит к значительному увеличению времени обработки и понижению производительности программы. Также необходимо учитывать влияние различных искажений (размытые участки изображений, изменение масштаба) на работу дескриптора.

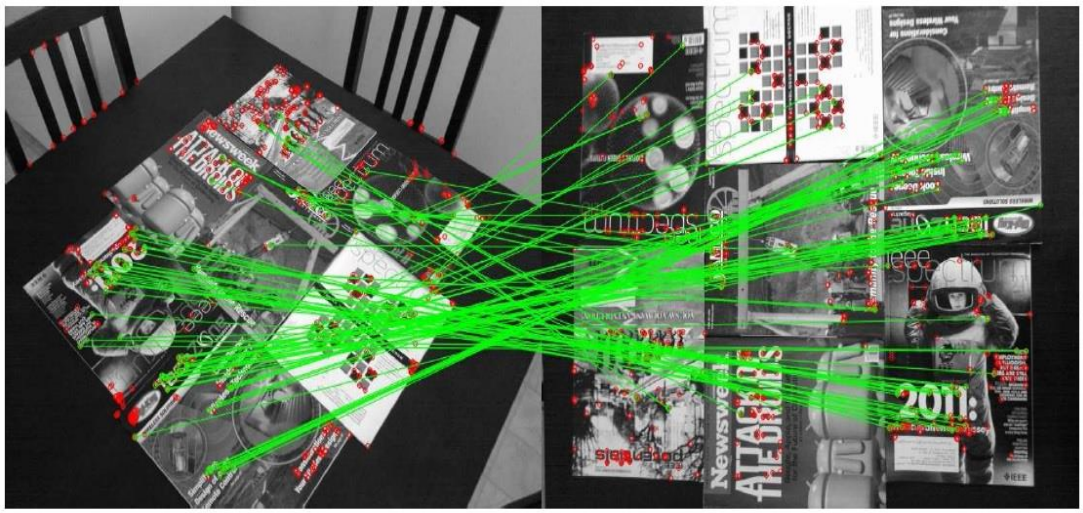

Рисунок 1.3 – Пример работы дескриптора ORB

Как видно из рисунка 1.3, один из самых популярных дескриптор ORB неплохо справляется с задачей распознавания объектов на изображении, сделанном с другого ракурса. Однако для этой задачи есть оптимальные, но малоизвестные дескрипторы, справляющиеся с задачей намного лучше.

Таким образом, необходимо реализовать программу, выполняющую следующие задачи:

- получать на вход изображение;
- давать пользователю возможность выбора необходимого дескриптора для обработки изображения;
- обрабатывать изображение согласно выбранному дескриптору и выводить обработанное изображение на экран;
- выводить различную информацию об обработке изображения.

В данном разделе был проведен краткий обзор особых точек и их дескрипторов. Были рассмотрены основные области применения дескрипторов особых точек и алгоритм их работы. Исходя из полученных данных, была сформулирована задача, выполнение которой подразумевает разрабатываемое программное обеспечение.

# 2 ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ СРАВНИТЕЛЬНОГО АНАЛИЗА ДЕСКРИПТОРОВ

## 2.1 Анализ дескриптора BRIEF

После определения ключевых точек, необходимо вычислять дескриптор для каждой из них. Дескрипторы объектов кодируют необходимую информацию в серию чисел и действуют как своего рода числовой «отпечаток», который можно использовать для различения одного объекта от другого [10]. Определенная окрестность вокруг пикселя (ключевая точка) называется патчем (patch), представляющим собой квадрат ширины и высоты пикселя. Пример патча представлен на рисунке 2.1.

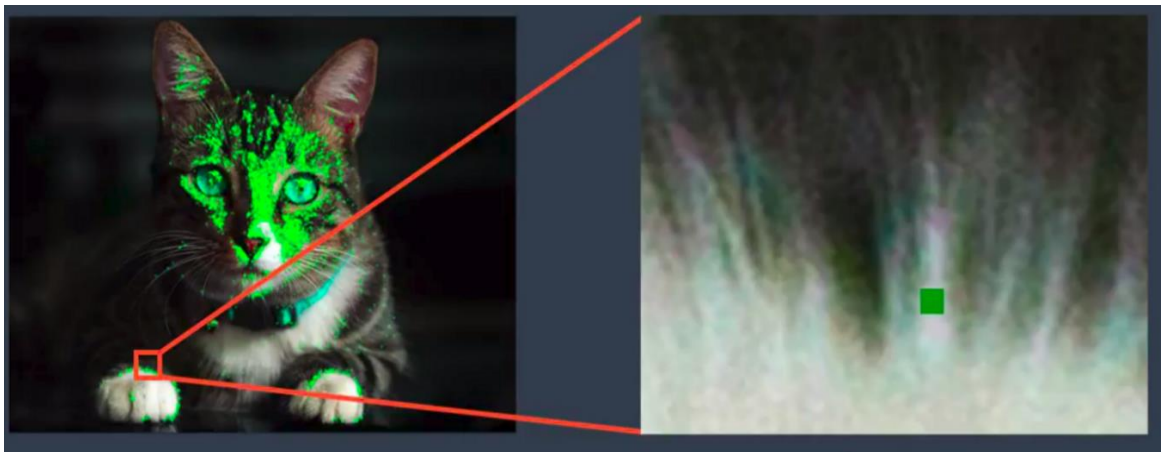

Рисунок 2.1 – Пример патча на изображении

Патчи изображений могут быть эффективно классифицированы на основе сравнительно небольшого числа парных сравнений интенсивности. BRIEF преобразовывает патчи изображений в двоичный вектор. Этот вектор двоичных объектов также известен как дескриптор, который содержит только 1 и 0. Вкратце, каждая ключевая точка описывается вектором объектов, который представляет собой строку длиной 128–512 бит [16]. На рисунке 2.2 представлен пример дескрипторов ключевых точек.

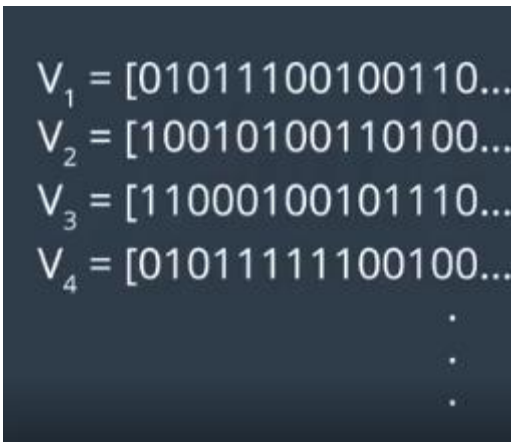


Рисунок 2.2 – Дескрипторы ключевых точек

BRIEF обрабатывает изображения на попиксельно, поэтому он очень чувствителен к шуму. Путем предварительного сглаживания патча эта чувствительность может быть уменьшена, что повышает стабильность и повторяемость дескрипторов. BRIEF использует ядро Гаусса для сглаживания изображения.

$$\tau(p,x,y)=\begin{cases}1, p(x)<p(y)\\0, p(x)\geq p(y)\end{cases} \quad (1)$$

где p(x) – интенсивность пикселя в точке с координатами x = (u, v) в патче. В итоге получается множество бинарных тестов. Важным является выбор пикселей в области для бинарного теста, всего существует пять методов определения векторов x и y.

Визуализация всех пяти методов представлена на рисунке 2.3.

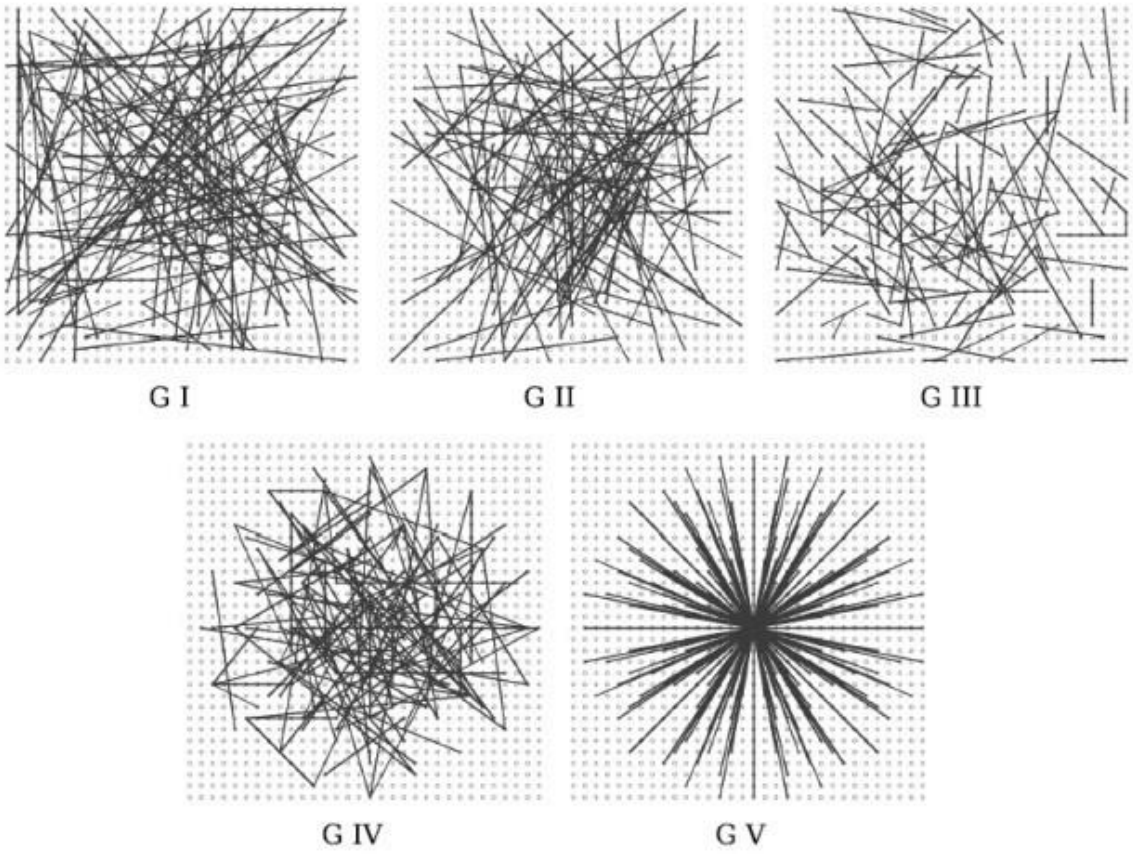


Рисунок 2.3 – Методы определения векторов x и y

По результатам сравнения в [15] точность распознавания в пяти перечисленных случаях примерно одинаковая.

Таким образом, BRIEF дескриптор особой точки определяется как nd-мерная битовая строка, определяемая по формуле 2:

$$f_{nd=} \sum 2^{i-1} \tau(p, x, y) \quad (2)$$

Построение и сопоставление особых точек для BRIEF намного быстрее, чем для других современных дескрипторов, если не требуется инвариантность к большим поворотам в плоскости.

Основными проблемами метода BRIEF является неэффективный выбор точек для расчёта дескриптора и плохая устойчивость к вращению при распознавании изображений.

## 2.2 Анализ дескриптора ORB

ORB является улучшенной версией дескриптора BRIEF и решает некоторые его недостатки [14]. Для нахождения особых точек он использует детектор FAST, который строит деревья принятия решений для классификации пикселей на изображении.

Основные особенности ORB:

- добавление быстрого и точного компонента ориентации в FAST;
- эффективный расчет ориентированных характеристик BRIEF;
- анализ дисперсии и корреляции, ориентированных BRIEF признаков;
- метод обучения для декорреляции функций BRIEF при вращательной инвариантности, приводящий к повышению производительности.

Для множества бинарных тестов размера n, с координатами (x, y) строится матрица S размерности 2xn:

$$S = \begin{pmatrix} x_1 \dots x_n \\ y_1 \dots y_n \end{pmatrix} \quad (3)$$

Функции FAST не имеют компонента ориентации и функций нескольких масштабов. Так что алгоритм ORB использует многомасштабную пирамиду изображений. Пирамида изображения - это многомасштабное представление одного изображения, которое состоит из последовательностей изображений, каждый из которых является версией изначального изображения с урезанным разрешением. Каждый уровень в пирамиде содержит уменьшенную версию изображения, чем предыдущий уровень. Как только ORB создал пирамиду изображения, он использует алгоритм FAST для определения ключевых точек на изображении. Путем обнаружения ключевых точек на каждом уровне пирамиды ORB эффективно находит ключевые точки на разных масштабах одного изображения. Пример пирамиды представлен на рисунке 2.4.

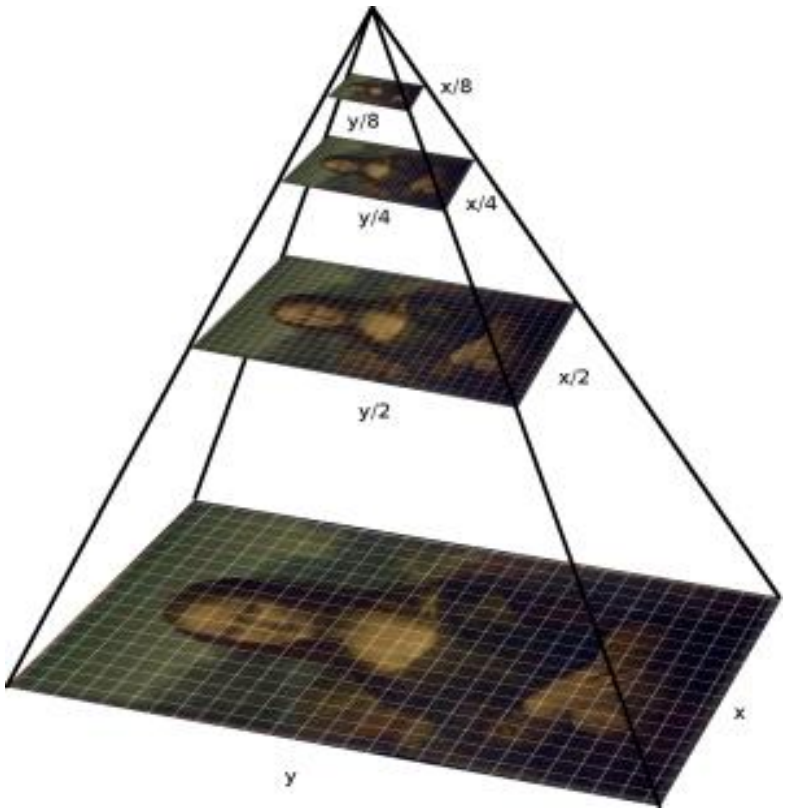


Рисунок 2.4 – Пирамида изображения

После нахождения ключевых точек ORB назначает ориентацию каждой ключевой точке, например, влево или вправо, в зависимости от того, как уровни интенсивности меняются вокруг этой ключевой точки. Для обнаружения изменения интенсивности ORB использует центроид интенсивности. Центроид интенсивности предполагает, что интенсивность угла смещена относительно его центра, и этот вектор может использоваться для определения ориентации.

В методе ORB для расчёта ориентации угла используются координаты центра тяжести, вычисляемые через моменты патча m:

$$m_{pq=\sum_{x,y} x^p y^q I(x,y)} \quad (4)$$

Тогда ориентация угла будет задаваться вектором, начало которого будет в центральной точке, а конец – в центре тяжести. Пример поворота особых точек представлен на рисунке 2.5.

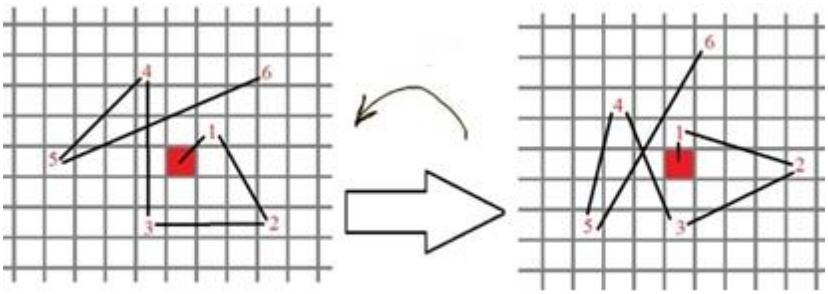


Рисунок 2.5 – Разворот последовательности точек на 30 градусов

В [20] рассматривается определение «качества» дескрипторов на основе их дальнейшего сравнения. В некоторых случаях дескриптор в однотонной области будет коррелировать с другими дескрипторами в этой точке, являясь плохо различимым. Из-за этого оценка эффективности дескриптора происходит через параметры среднего арифметического и дисперсии.

В дескрипторе ORB присутствует большая дисперсия и среднее значение около 0,5. Однако, после нахождения ориентации ключевой точки, дисперсия дескриптор значительно уменьшается. Дескриптор ORB решает эту проблему путём сравнения всех возможных бинарных тестов. На основе этих сравнений находятся максимальные и минимальные значения дисперсий, а также значения, близкие к 0,5. Для сопоставления бинарных тестов используется метод понижения размерности многомерных данных LSH (Locality-sensitive hashing).

Итак, основные усилия по повышению эффективности рассмотренных дескрипторов необходимо сконцентрировать по двум направлениям: вычислительная оптимизация и ускорение SIFT и повышение качества ORB.

### 2.3 Анализ дескриптора SIFT

Основными преимуществами SIFT являются:

- характеристики локальны, поэтому устойчивы к окклюзии и беспорядку (без предварительной сегментации);
- отдельные характеристики могут быть сопоставлены с большой базой данных объектов;
- многие характеристики могут быть созданы даже для небольших объектов;
- дескриптор близок к производительности в реальном времени;
- дескриптор легко может быть масштабирован на широкий спектр различных типов объектов.

Объекты реального мира имеют смысл только в определенном масштабе. Человек легко может увидеть сахарный кубик на столе или машину на дороге, но не сможет увидеть Млечный путь или структуру атома. Эта многомасштабная природа объектов довольно распространена в природе. Масштаб пространства (scale space) пытается воспроизвести эту концепцию на цифровых изображениях.

Масштаб пространства изображения - это функция $L(x, y, \sigma)$, которая получается из свертки гауссова ядра (размытие) в разных масштабах с входным изображением. Масштаб пространства разделен на октавы, а количество октав и масштаб зависит от размера исходного изображения. Таким образом, мы генерируем несколько октав исходного изображения. Размер изображения каждой октавы равен половине предыдущего. Пример масштаба пространства представлен на рисунке 2.6.

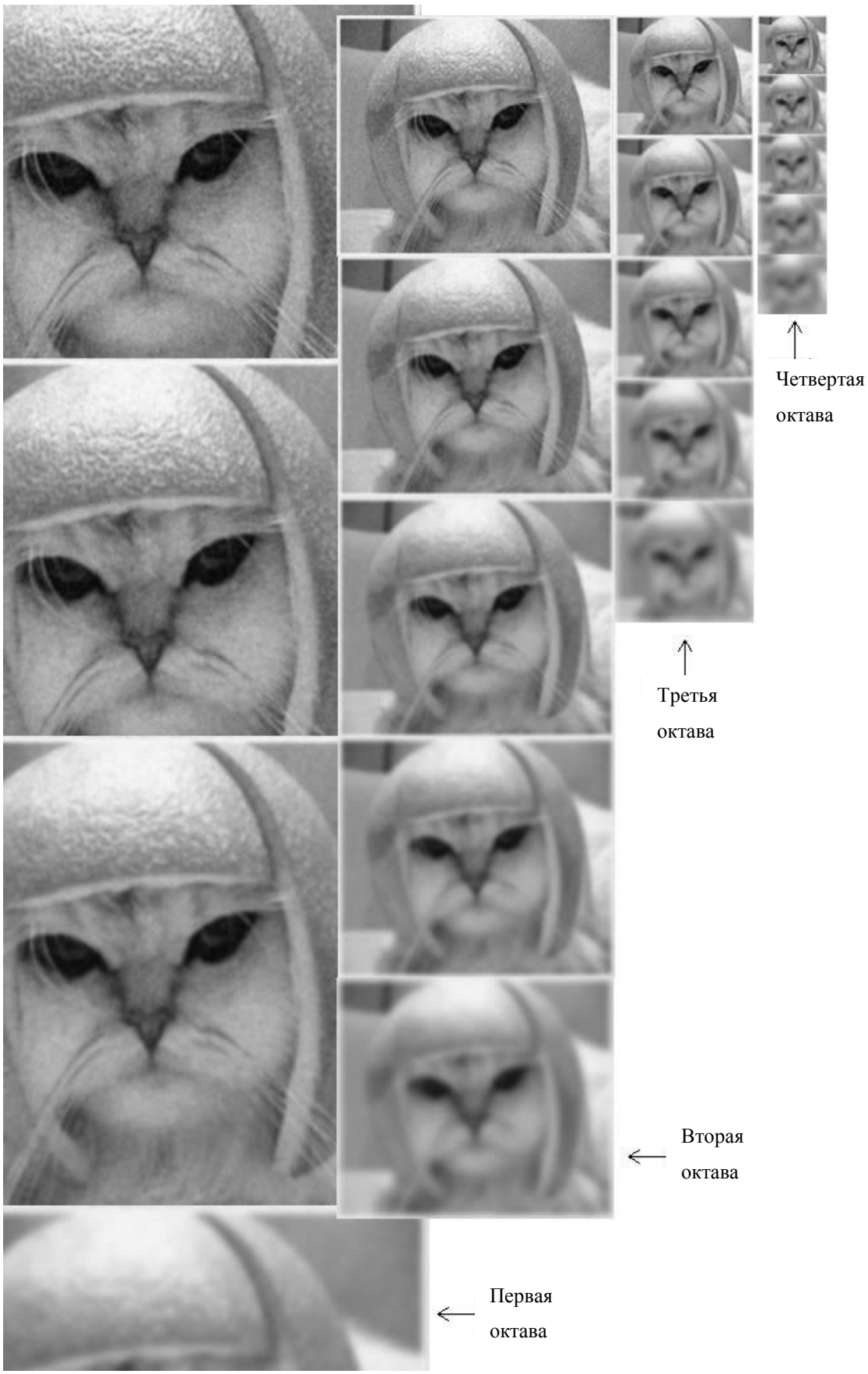


Рисунок 2.6 – Масштаб пространства изображения

Ключевой особенностью метода SIFT является построение пирамиды гауссианов и разностей гауссианов. Изображение размывается с помощью фильтра Гаусса, который соответствует следующе формуле:

$$L(x, y, o) = G(x, y, o) * I(x, y) \quad (5)$$

В формуле (5) L(x,y,o) – это значение гауссиана в точке с координатами (x,y) и радиусом размытия o, G(x,y,o) – гауссово ядро, I(x,y) – значение исходного изображения. Между гауссовым ядром и значением исходного изображения происходит операция свёртки, а не перемножения.

Далее рассчитывается разность гауссиан. Из каждого пикселя изображения вычитается пиксель изображения с другой степенью размытия:

$$D(x, y, o) = L(x, y, ko) - L(x, y, o) \quad (6)$$

Далее масштабируемые пространства делятся на октавы. После этого достраиваются ещё два гауссиана, выходящие за пределы октавы [17].

В то же время будет строиться пирамида разностей гауссианов, в которой будет на одно изображение меньше, чем в пирамиде гауссианов. Создание пирамид представлено на рисунке 2.7.

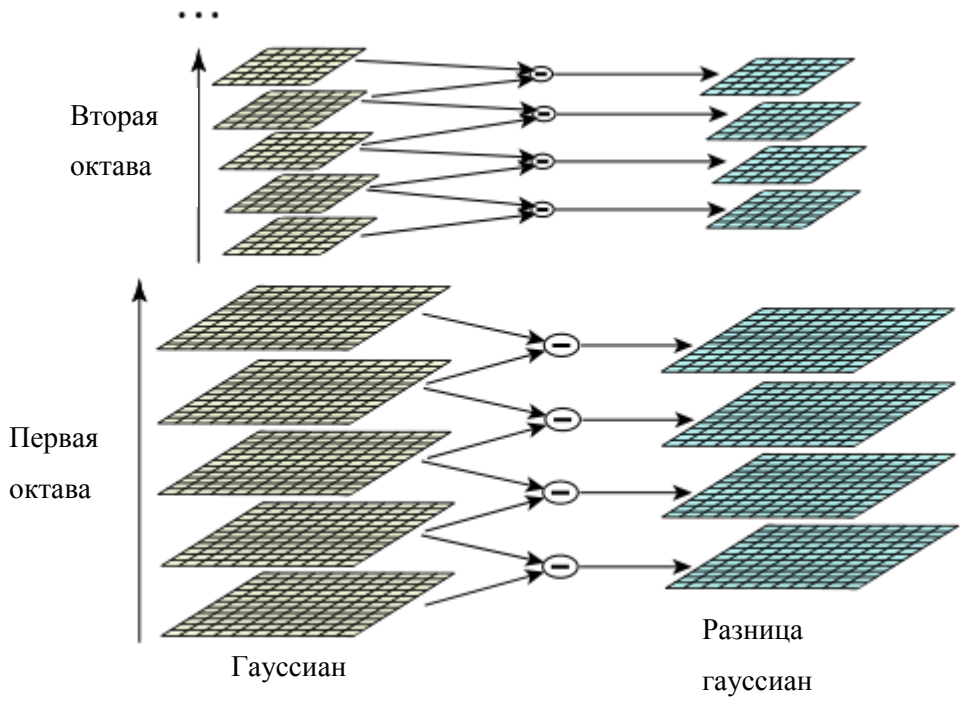


Рисунок 2.7 – Построение пирамид гауссианов и разностей гауссианов

Следующим шагом является определение особых точек. Если значение в точке с крестиком существенно отличается от значений точек с кругом, то такая точка будет являться локальным экстремумом разности гауссианов, то есть считаться особенной. Пример особой точки представлен на рисунке 2.8. Экстремум помечен крестиком и является особой точкой.

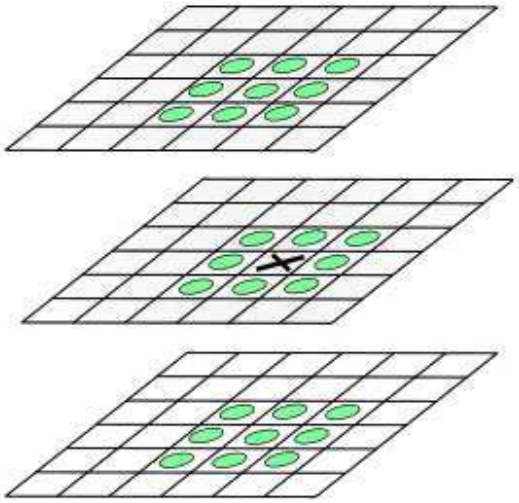

Рисунок 2.8 – Определение особой точки по локальному экстремуму

После этого ключевые точки проверяются на стабильность, используя масштаб, при котором была обнаружена ключевая точка (он совпадает с масштабом размытого изображения). Далее каждой ключевой точке назначается ориентация, чтобы сделать её устойчивой к вращению. Пример особой точки изображения представлен на рисунке 2.9.

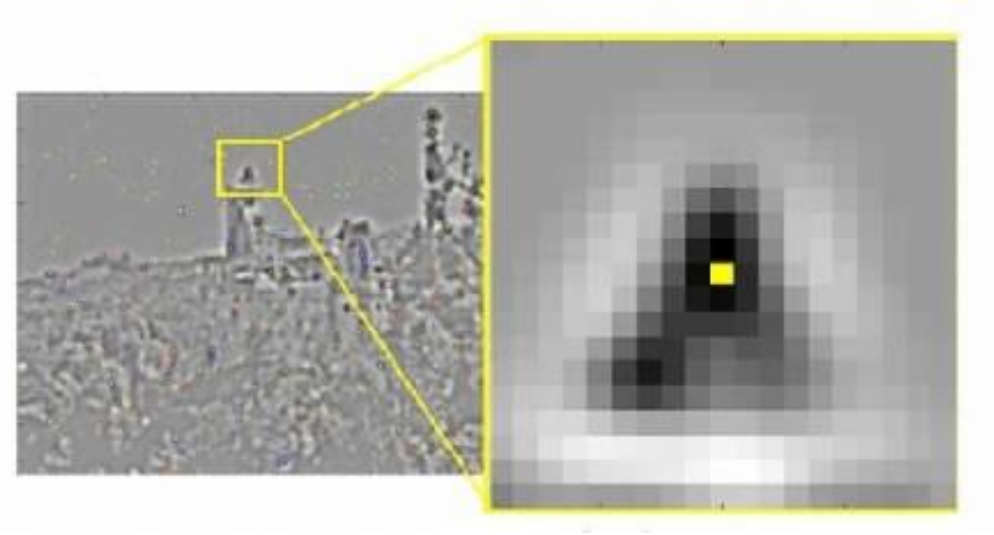

Рисунок 2.9 – Пример особой точки изображения

Вокруг расположения ключевой точки берется окрестность в зависимости от масштаба, и в этой области вычисляются величина и направление градиента. Создается гистограмма ориентации с 36 ячейками, охватывающими 360 градусов. Например, направление градиента в определенной точке (в области сбора ориентации) составляет 18,759 градуса, тогда оно войдет в корзину 10–19 градусов. И количество, которое добавляется в корзину, пропорционально величине градиента в этой точке. После повторения этой операции для всех пикселей вокруг ключевой точки, гистограмма будет иметь пик в некоторой точке.

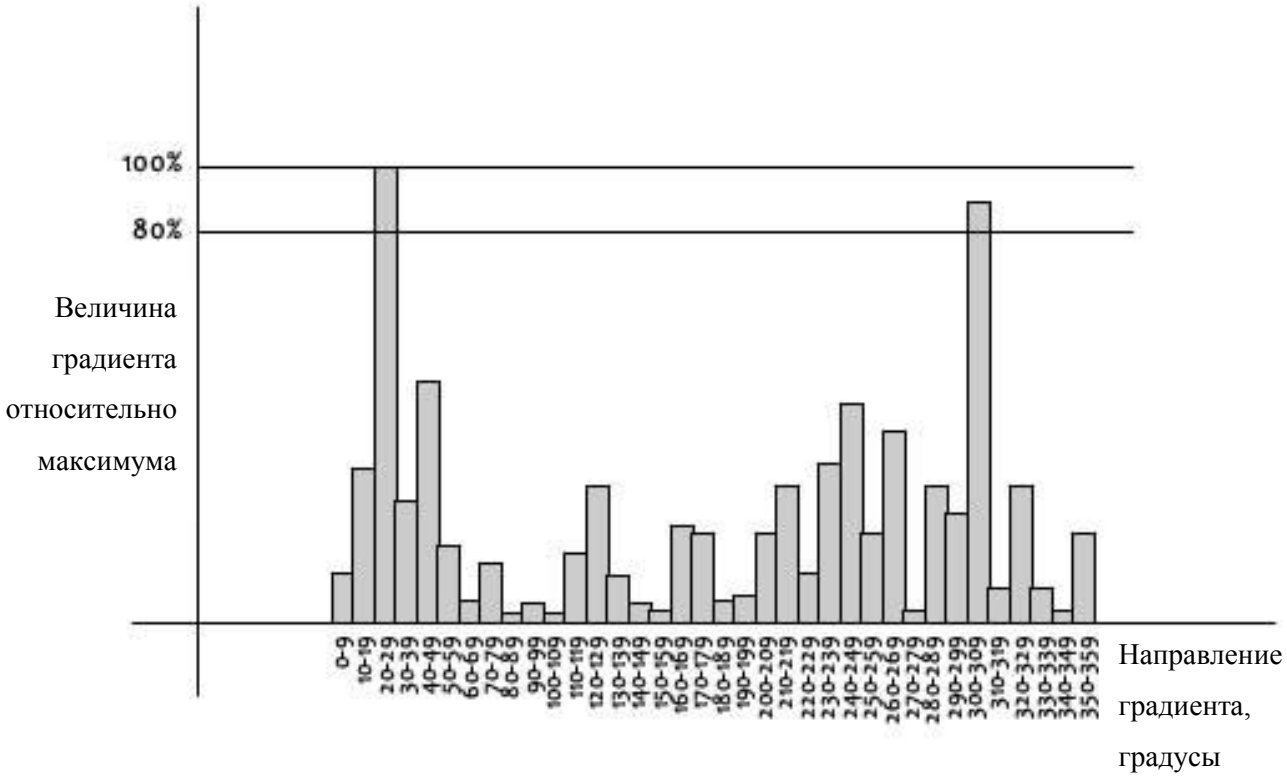


Рисунок 2.10 – Гистограмма ориентации

На примере из рисунка 2.10 взят самый высокий пик в гистограмме, и любой пик выше 80% от его значения также считается для расчета ориентации.

Такой подход помогает создать ключевые точки с одинаковым расположением и масштабом, но в разных направлениях и способствует стабильности соответствия ключевых точек.

Дескриптор SIFT определяется на гауссиане, максимально приближенном по масштабу к особой точке. Работа дескриптора представлена на рисунке 2.11.

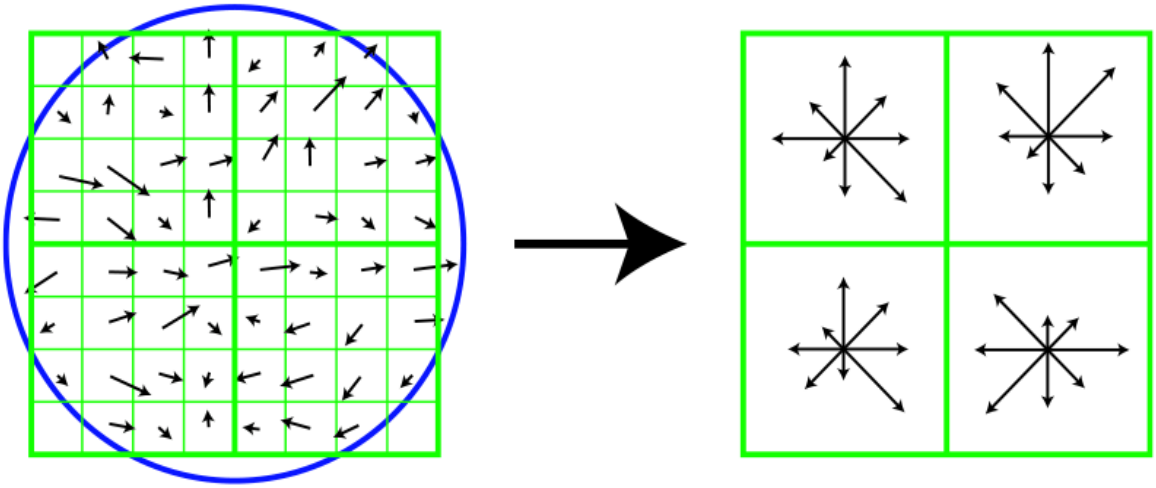

Рисунок 2.11 – Схематичное представление градиентов изображения (слева) и полученного на их основе дескриптора (справа)

Как видно из рисунка 2.11, каждому градиенту в окне дескриптора можно присвоить три координаты: x,y,n. На основе этих градиентов и строится дескриптор.

Дескриптор SIFT является неплохим выбором для обработки изображений, однако он имеет ряд недостатков. Точки и их дескрипторы не всегда соответствуют необходимым требованиям, а решение не всегда может быть найдено. В случае если изображение состоит из повторяющихся фрагментов (например, стена), то этот дескриптор будет работать некорректно.

### 2.4 Анализ дескриптора SURF

Работа дескриптора SURF состоит их двух этапов: выявление особенностей (feature extraction) и их описание (feature description).

Подход для обнаружения точек интереса использует очень простое приближение матрицы Гессе на основе интегральных изображений.

Интегральное изображение или таблица суммированных площадей была введена в 1984 году. Интегральное изображение используется в качестве быстрого и эффективного способа вычисления суммы значений пикселей в данном изображении, либо вычисления прямоугольного подмножества сетки данного изображения. Оно также используется для расчета средней интенсивности в пределах данного изображения [18].

Интегральные изображения позволяют быстро вычислять свёрточные фильтры. Запись интегрального изображения $I_\varepsilon$ (x) в местоположении x = (x, y)$^T$ представляет сумму всех пикселей во входном изображении I в прямоугольной области, образованной началом координат и x.

$$I_\varepsilon(x) = \sum_{i=0}^{i\leq x} \sum_{j=0}^{j\leq y} I(i,j) \qquad (7)$$

При вычислении $I_\varepsilon$ требуется всего четыре суммирования для вычисления суммы интенсивностей по любой вертикальной прямоугольной области, независимо от её размера.

Далее изображение отфильтровывается по гауссову ядру. Для точки X = (x, y) в местоположении x и масштабе σ матрица Гессе H (x, σ) определяется по формуле 8:

$$H(x,\sigma) = \begin{bmatrix} L_{xx}(x,\sigma) & L_{xy}(x,\sigma) \\ L_{xy}(x,\sigma) & L_{yy}(x,\sigma) \end{bmatrix} \qquad (8)$$

В формуле 8 $L_{xx}$(x, σ) - свертка гауссовой производной второго порядка с изображением I в точке x, аналогично для $L_{xy}$(x, σ) и $L_{yy}$(x, σ). Гауссианы оптимальны для пространственно-масштабного анализа, но на практике их нужно дискретизировать и обрезать [19]. Это приводит к появлению искажений при поворотах изображения. Эта недостаток всех детекторов на основе гауссиана. Тем не менее, эти детекторы работают хорошо, и

небольшое снижение производительности не перевешивает преимущество быстрых сверток за счёт дискретизации и кадрирования.

Для вычисления определителя матрицы Гессе, необходимо применить свертку с ядром Гаусса, а затем с производной второго порядка. После успеха Лоу [21] с аппроксимациями типа LoG, SURF пытается усовершенствовать эту идею, используя аппроксимацию (как свертки, так и производной второго порядка) с помощью блочных фильтров. Приближенные производные Гаусса второго порядка могут быть рассчитаны с низкими вычислительными затратами, используя интегральные изображения с разными размерами. Это является одной из причин быстроты дескриптора SURF.

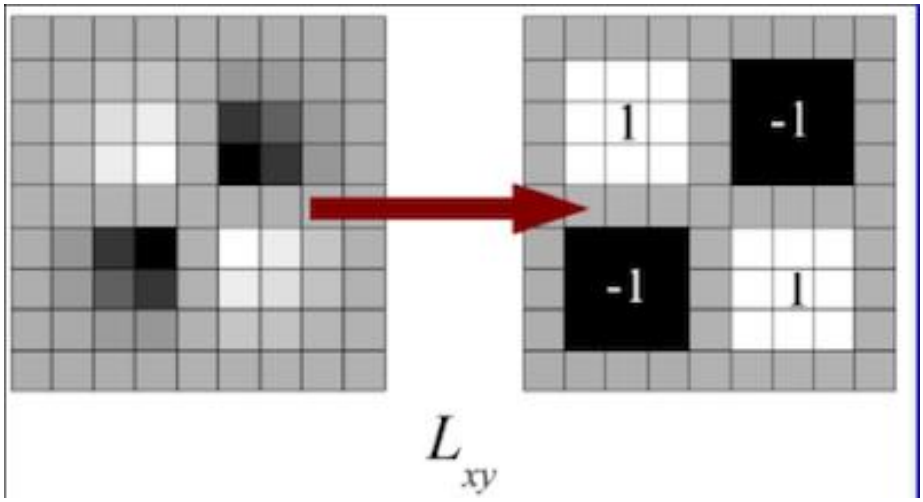


Рисунок 2.12 – Гауссова частная производная второго порядка по xy

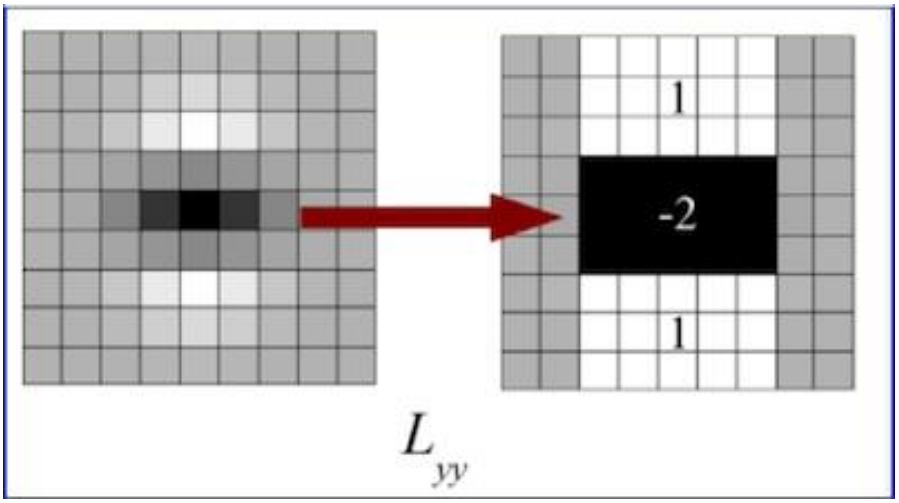


Рисунок 2.13 – Гауссова частная производная второго порядка по y

Рамочные фильтры 9 × 9 на рисунках 2.12 и 2.13 являются приближениями для гауссовых производных второго порядка с σ = 1,2. Обозначив эти приближения через $D_{xx}$, $D_{yy}$ и $D_{xy}$, можно представить определитель гессиана (приближенный) по формуле 9:

$$\det(H_{approx}) = D_{xx}D_{yy} - (\omega D_{xy})^2 \quad (9)$$

Масштаб пространства обычно реализуется в виде пирамиды. Изображения многократно сглаживаются гауссовым методом и впоследствии подвергаются дополнительной выборке для достижения более высокого уровня пирамиды.

Чтобы локализовать точки интереса на изображении и в масштабе, применяется не максимальное подавление в окрестности 3 × 3 × 3. Как видно на рисунке 2.14, вместо итеративного уменьшения размера изображения (слева), дескриптор использует интегральные изображения для масштабирования фильтр с постоянной нагрузкой (справа). Это позволяет сохранить производительность при обработке высококачественных изображений.

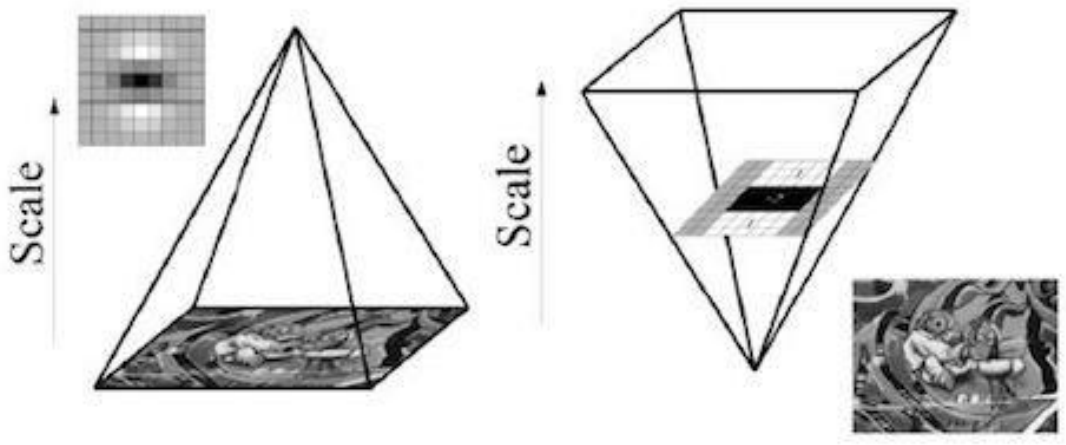


Рисунок 2.14 – Итеративное уменьшение размера изображения (слева) против масштабирования фильтра (справа)

Чтобы быть инвариантным к вращению, SURF пытается определить ориентацию для точек интереса. Для достижения этого дескриптор вычисляет сумму вертикальных и горизонтальных вейвлет-откликов в области сканирования, меняя ориентацию сканирования (добавляя π / 3) и пересчитываем их, пока не найдет ориентацию с наибольшим значением суммы. Эта ориентация будет являться основной ориентацией дескриптора.

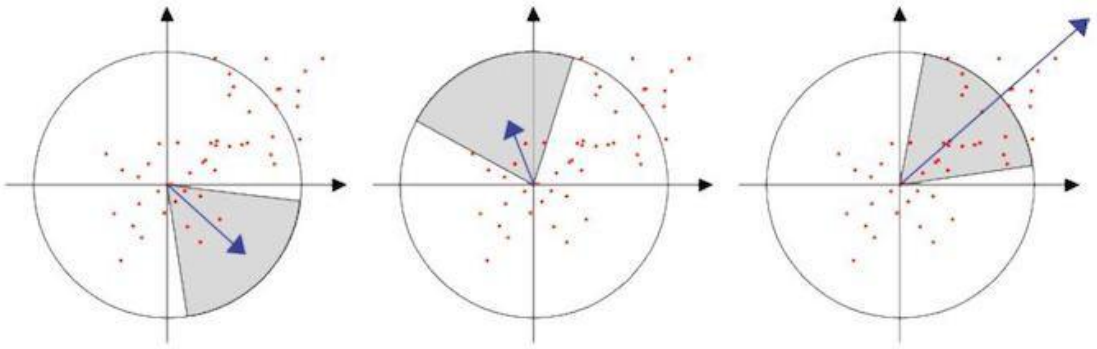

Рисунок 2.15 – Определение ориентации точек интереса

Последним этапом является извлечение дескриптора.

Вокруг ключевой точки строится специальная область с ранее найденной ориентацией. Затем эта область делится на более мелкие субрегионы, в которых находятся значения функции. Значения функций суммируются, образуя необходимый дескриптор.

Пример этого процесса представлен на рисунке 2.16.

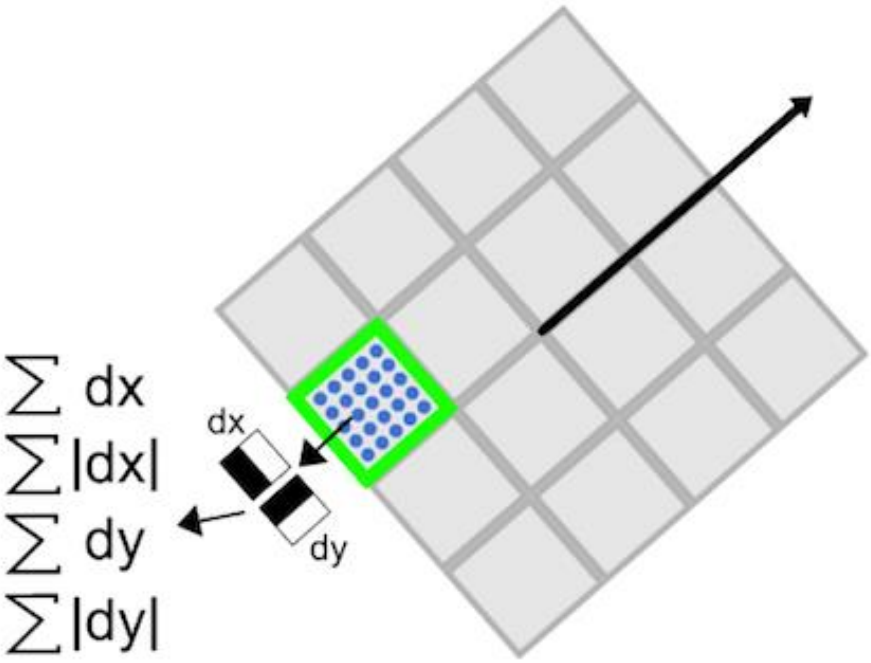


Рисунок 2.16 – Процесс извлечения дескриптора

В результате получается вектор дескриптора длиной 64 для всех субрегионов размером 4 × 4. В SIFT дескриптор представляет собой вектор длиной 128, поэтому SURF с 64-значной длиной дескриптора работает быстрее, чем SIFT.

Таким образом, в данном разделе составлена математическая модель всех дескрипторов, необходимых для реализации программного обеспечения. Рассмотрены основные особенности и принципы работы каждого из дескрипторов .

Следующим шагом является написание программы на основе поставленных задач и сформированного математического аппарата.

# 3 РЕАЛИЗАЦИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ СРАВНИТЕЛЬНОГО АНАЛИЗА ДЕСКРИПТОРОВ

## 3.1 Описание программного обеспечения

Для создания программного обеспечения использовался язык программирования Java и среда разработки IntelliJ IDEA.

Для разработки приложения использовались следующие средства разработки и библиотеки:

- Объектно-ориентированный язык программирования Java [13];
- библиотека Swing, которая используется для создания графического интерфейса пользователя;
- библиотека компьютерного зрения OpenCV;
- паттерн mediator, использующийся для упрощения графического интерфейса;
- интегрированная среда разработки приложений – IntelliJ IDEA.

На рисунке 3.1 представлен проект приложения в среде IntelliJ IDEA

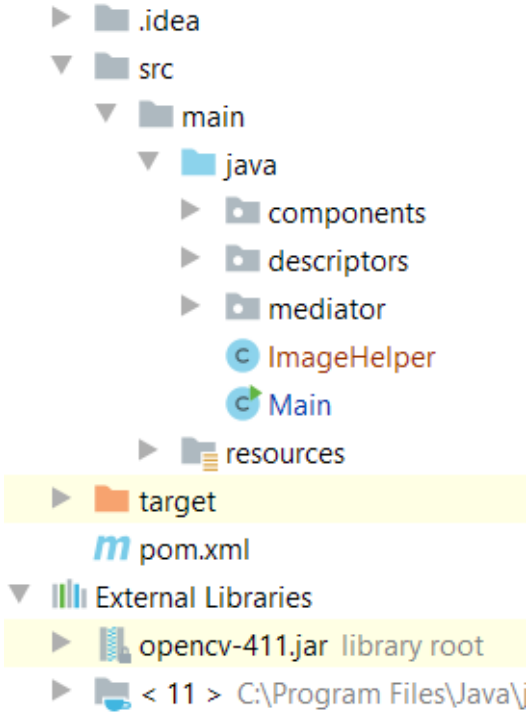


Рисунок 3.1 – Проект приложения в среде IntelliJ IDEA

Корневая папка проекта содержит:

- файл «Main.java», являющийся точкой входа при запуске приложения и содержащий методы для инициализации компонентов медиатора приложения, а также для отрисовки графического интерфейса;
- файл «ImageHelper.java», который отвечает за обработку входных изображений;
- папку «resources», которая содержит необходимые наборы изображений;
- папка «components», которая содержит файлы элементов графического интерфейса;
- папка «mediator», которая содержит реализацию паттерна посредника;
- папка «descriptors», которая содержит файлы дескрипторов особых точек. В данных файлах описываются дескрипторы особых точек, рассмотренные в предыдущей главе.

Также в проекте используются библиотеки Java 11 и OpenCV версии 4.1.1.

Для улучшения восприятия кода используется паттерн «посредник». Это поведенческий паттерн разработки. Паттерн отвечает за оптимизацию и скорость общения нескольких классов между собой, благодаря перемещению связей между классами в один специальный класс посредника [22].

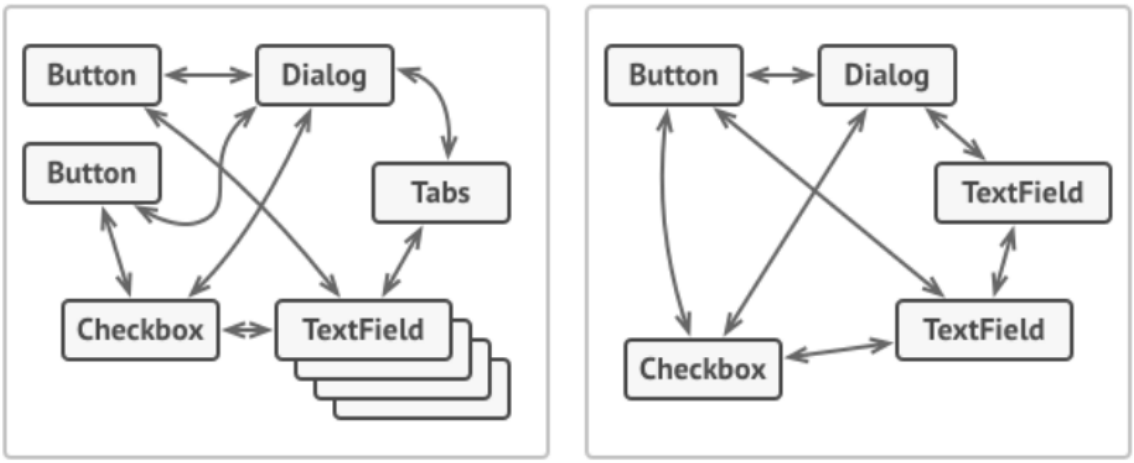


Рисунок 3.2 – Связи между классами до использования паттерна

Как видно из рисунка 3.2, до использования паттерна посредника связи между классами весьма запутанны и хаотичны. Каждый класс связан с несколькими другими классами, что значительно усложняет разработку. При масштабировании приложения это может сыграть критическую роль. Так как количество элементов интерфейса будет только возрастать, связность между ними возрастёт экспоненциально. На рисунке 3.3 представлены связи между классами после применения паттерна.

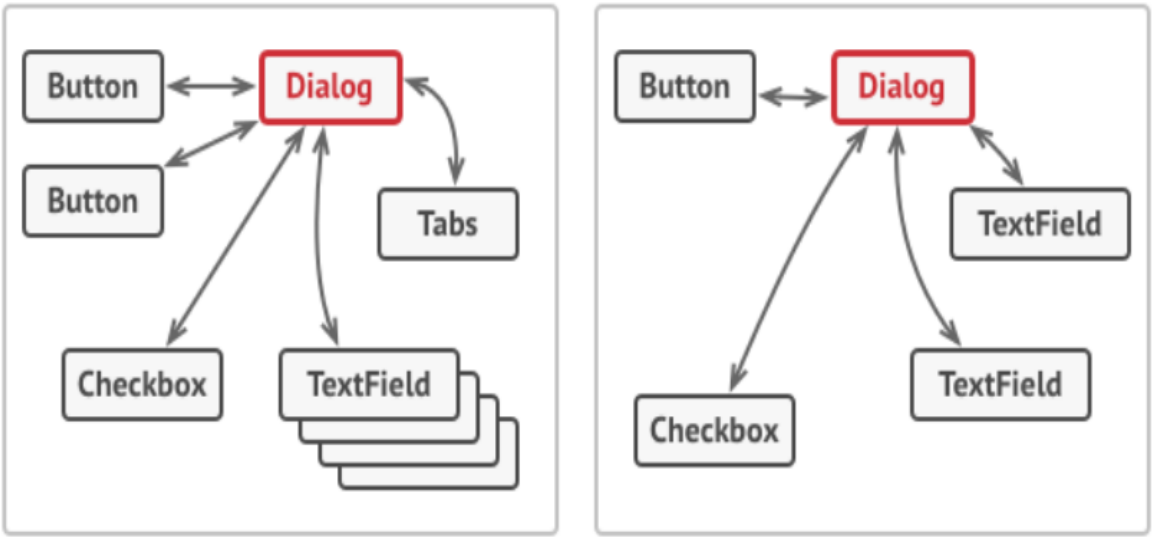


Рисунок 3.3 – Связи между классами после применения паттерна

Как видно из рисунка 3.3, связность между классами значительно уменьшилась. Для этого был создан специальный класс, являющийся посредником между остальными классами графического интерфейса. Благодаря этому, все компоненты системы работают друг с другом через этот класс-посредник.

На рисунке 3.4 представлена структура паттерна «посредник» с пояснениями.

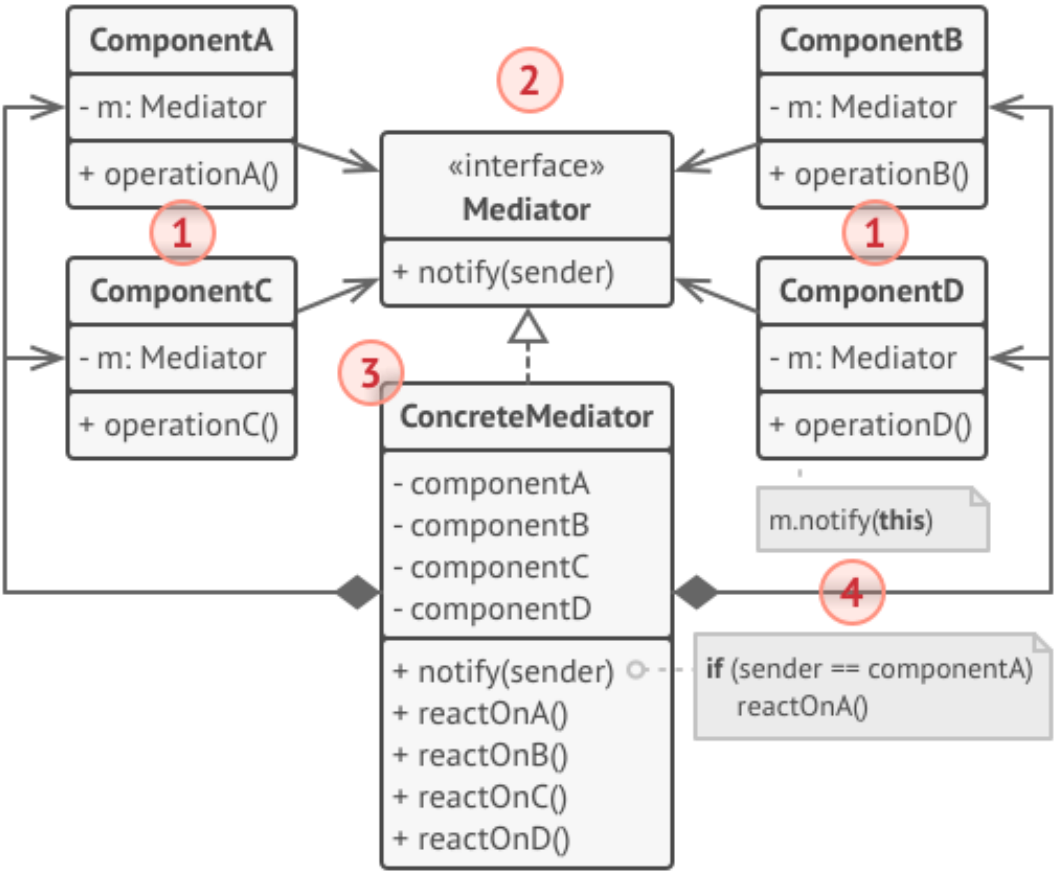


Рисунок 3.4 – Структура паттерна «посредник»

1. Компоненты — это разнородные объекты, содержащие бизнес-логику программы. У каждого компонента есть ссылка на объект посредника..

2. Интерфейс паттерна, использующийся для обмена информации между различными компонентами системы. Как правило, содержит один метод уведомления о различных событиях. В этот метод передаются детали определённого события: ссылку на компонент-уведомитель, и любые другие данные.

3. Реализация интерфейса посредника.

4. Все компоненты общаются через класс посредника, а не напрямую. Когда происходит важное событие, компонент оповещает своего посредника, а посредник сам решает кому и что передавать.

### 3.2 Реализация программного обеспечения

После полного проектирования приложения можно приступить к написанию алгоритмов дескрипторов. На рисунке 3.5 представлен код алгоритма ORB.

```
public Mat run(Mat input) {
    long time = System.currentTimeMillis();
    int hessianThreshold = 400;
    int nOctaves = 4, nOctaveLayers = 3;
    MatOfKeyPoint keypoints = new MatOfKeyPoint();
    Mat descriptor = new Mat();
    Mat output = new Mat();
    ORB orb = ORB.create(hessianThreshold, nOctaves, nOctaveLayers);
    orb.detectAndCompute(input, new Mat(), keypoints, descriptor);
    Features2d.drawKeypoints(input, keypoints, output);
    time = System.currentTimeMillis() - time;
    System.out.println("Processing time in ms = " + time);
    System.out.println("Number of keypoints = " + keypoints.toList().toArray().length);
    return output;
}
```

Рисунок 3.5 – Код алгоритма ORB

Функция принимает на вход необработанное изображение и помечает на нём особые точки. Также подсчитывается время выполнения кода и количество найденных особых точек.

Далее этот метод выполняется при нажатии кнопки в графическом интерфейсе. Сначала вызывается метод обработки картинки, в котором на изображении отмечаются особые точки, и подсчитывается их количество. Далее вспомогательный класс ImageHelper создает окно для отображения обработанной картинки. Код события при нажатии кнопки представлен на рисунке 3.6.

```
@Override
protected void fireActionPerformed(ActionEvent actionEvent) {
    Orb orb = new Orb();
    Mat processedImage = orb.run(Imgcodecs.imread(IMAGE_PATH));
    ImageHelper imageHelper = new ImageHelper();
    imageHelper.addImage(processedImage);
}
```

Рисунок 3.6 – Код для обработки нажатия кнопки

Также стоит отметить реализацию паттерна посредника. Для этого был создан интерфейс Mediator, от которого наследуется класс Editor. На рисунке 3.7 представлена часть кода основного метода класса Editor, отвечающего за отрисовку основного окна программы.

```
public void createGUI() {
    JFrame mainFrame = new JFrame( title: "Comparator");
    mainFrame.setSize( width: 960,  height: 600);
    mainFrame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);

    JPanel left = new JPanel();
    left.setBorder(new LineBorder(Color.BLACK));
    left.setSize( width: 320,  height: 600);
    left.setLayout(new BoxLayout(left, BoxLayout.Y_AXIS));

    JPanel listPanel = new JPanel();
    list.setFixedCellWidth(260);
    listPanel.setSize( width: 320,  height: 470);
    JScrollPane scrollPane = new JScrollPane(list);
    scrollPane.setPreferredSize(new Dimension( width: 275,  height: 410));
    listPanel.add(scrollPane);
```

Рисунок 3.7 – Код метода отрисовки основного окна программы

Таким образом, было вкратце описана реализация алгоритмов дескрипторов и построения графического интерфейса программы. Теперь необходимо провести сравнение дескрипторов особых точек на разных типах изображений и сделать выводы по их работе.

### 3.3 Проведение сравнительного анализа дескрипторов

Используя реализованное программное обеспечение, необходимо провести эксперимент по сравнению производительности и точности дескрипторов особых точек.

Для начала рассмотрим работу программы. На рисунке 3.8 представлено основное окно программы до выбора обрабатываемой

картинки. В левой части присутствуют кнопки добавления и удаления картинок из списка.

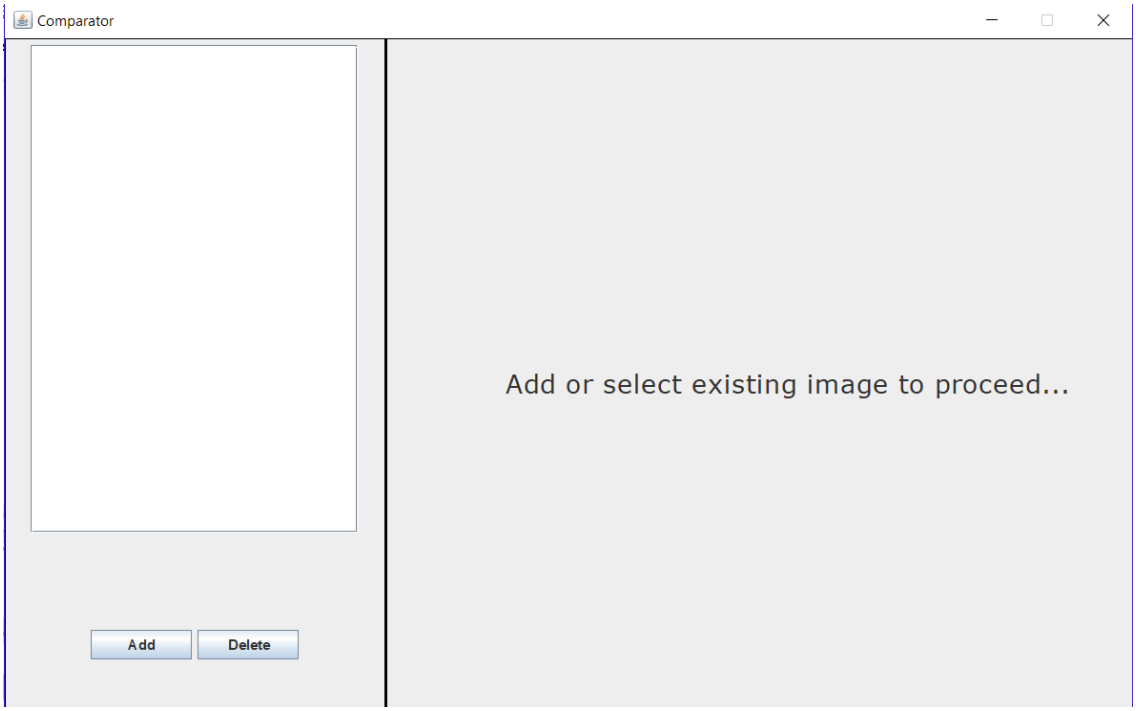


Рисунок 3.8 – Основное окно программы до выбора изображения

Далее необходимо нажать кнопку add и выбрать нужное изображение в файловой системе. После откроется доступ к правой части программы, которая содержит превью выбранного из списка изображения и доступные методы обработки изображения. Пример окна программы после выбора изображения представлен на рисунке 3.9.

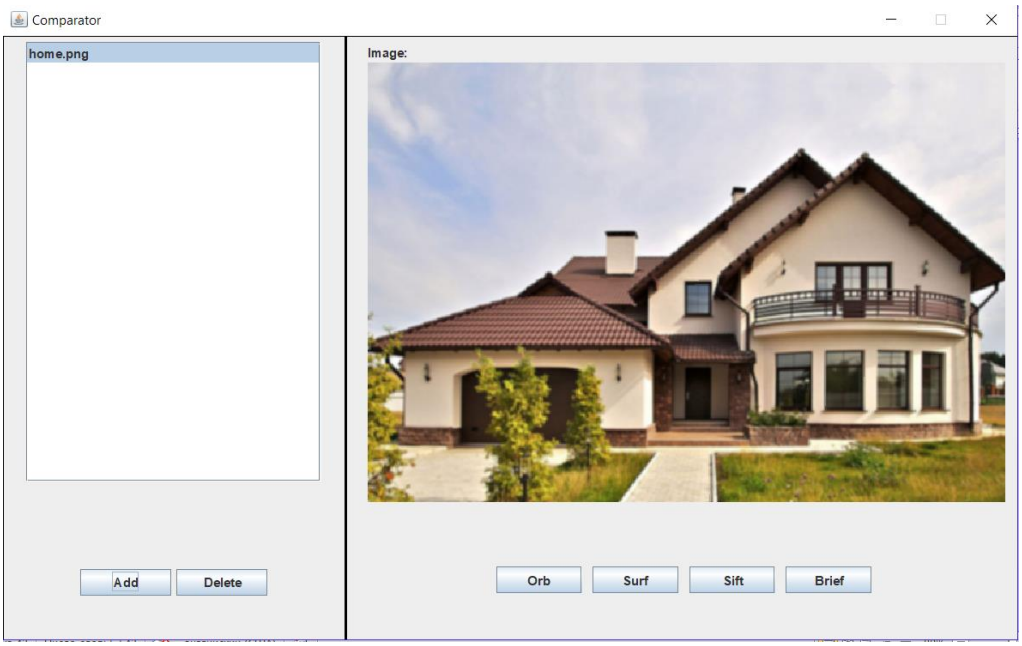

Рисунок 3.9 – Основное окно программы после выбора изображения

После нажатия на одну из кнопок дескрипторов произойдет обработка изображения с помощью выбранного дескриптора. Также программа подсчитает количество найденных особых точек и время обработки изображения. Результат обработки выведется в отдельное окно, а время и количество точек – в консоль. Пример обработанного изображения представлен на рисунке 3.10.

Рисунок 3.10 – Обработанное программой изображение

Теперь необходимо найти результаты работы дескрипторов на разных типах изображений и сравнить их. Для этого были взяты 3 датасета изображений. Первый датасет состоит из картинок с пейзажами. На этих типах изображений много кривых линий и мало углов. Пример данного датасета представлен на рисунке 3.11.

Рисунок 3.11 – Пример датасета пейзажей

Второй тип датасета – лица людей. Пример датасета с лицами представлен на рисунке 3.12.

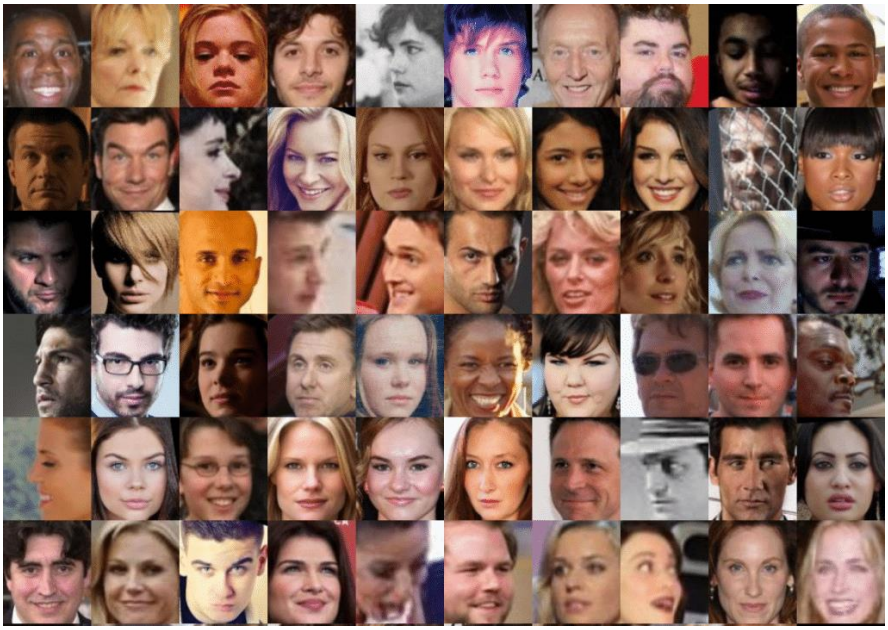

Рисунок 3.12 – Пример датасета лиц

Такие изображения имеют примерно одинаковое количество углов и прямых линий. Как правило, они имеют значительно меньшее количество особых точек.

Последним датасетом являются изображения документов. В них очень много букв, соответственно особых точек должно быть значительно больше. Пример датасета документов представлен на рисунке 3.13.


Рисунок 3.13 – Пример датасета документов

Таким образом, последним этапом является фиксирование и сравнение результатов обработки изображений. Для каждого датасета рассчитывается среднее время обработки изображения и среднее количество особых точек на изображении. Далее находится максимальное значение из среднего количества найденных особых точек. Остальные значения рассчитываются в процентах относительно максимального. Результаты представлены в таблице 3.1. Зелёным выделены лучшие дескрипторы на каждом наборе изображений.

Таблица 3.1 – Точность дескрипторов на разных датасетах

| **Дескриптор / Датасет** | **BRIEF** | **ORB** | **SIFT** | **SURF** |
|---|---|---|---|---|
| **Пейзажи** | 94% | 100% | 93% | 87% |
| **Лица** | 89% | 94% | 100% | 86% |
| **Документы** | 100% | 99% | 92% | 90% |

Как видно из таблицы 3.1, на каждом датасете дескрипторы ведут себя по-разному. На изображениях пейзажей большего всего особых точек нашёл дескриптор ORB, за которым следует BRIEF. Для лиц больше всего подходит дескриптор SIFT, а ORB также неплохо справляется с этой задачей. Самую высокую точность нахождения особых точек на документах показали

дескрипторы BRIEF и ORB. На рисунке 3.14 представлена визуализация точности дескрипторов в виде гистограммы.

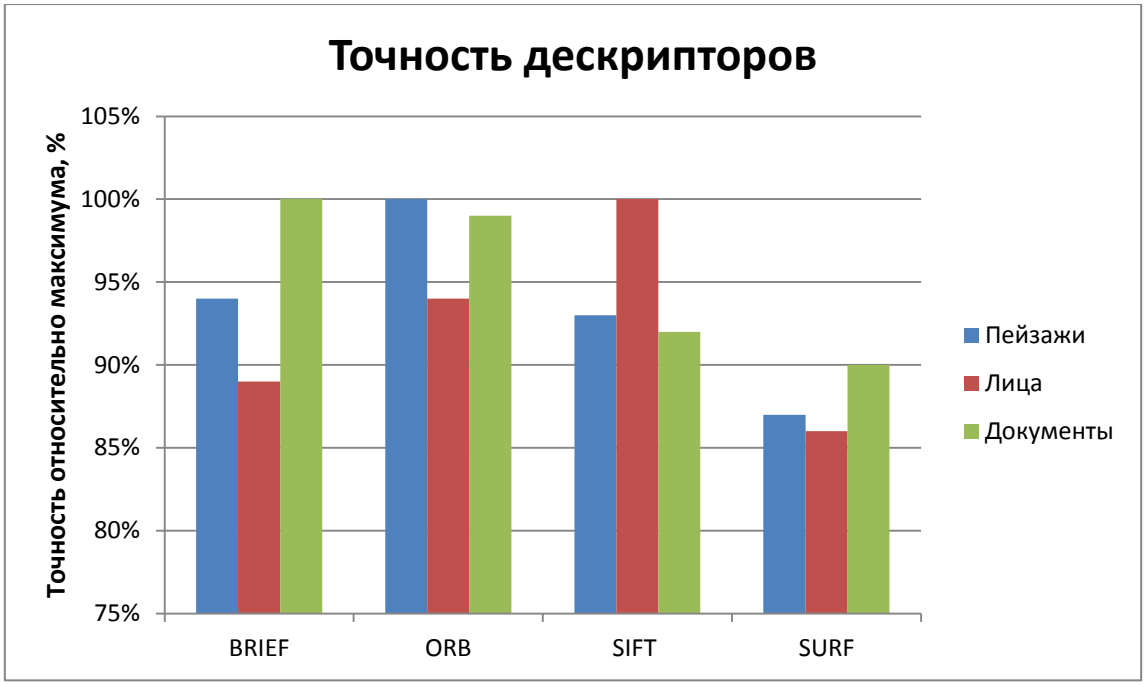


Рисунок 3.14 – Гистограмма точности дескрипторов

Если рассчитать среднее значение точности для каждого дескриптора, то ORB будет самым эффективным. Наименее точно показал себя дескриптор SURF, плохо справляющийся с выбранными типами изображений.

Последним шагом является сравнение скорости работы дескрипторов. На каждом наборе изображений была рассчитана средняя скорость обработки изображений для каждого дескриптора. Результаты представлены в таблице 3.2.

Таблица 3.2 – Средняя время обработки на разных датасетах, мс

| Дескриптор / Датасет | BRIEF | ORB | SIFT | SURF |
|---|---|---|---|---|
| **Пейзажи** | 370 | 341 | 528 | 437 |
| **Лица** | 266 | 252 | 319 | 280 |
| **Документы** | 218 | 188 | 275 | 262 |

Как видно из таблицы 3.2, наиболее быстрым дескриптором во всех случаях оказался ORB. Также дескриптор BRIEF показал неплохие результаты, а вот дескрипторы SIFT и SURF находили особые точки значительно медленнее первых двух. Это обусловлено вычислительной сложностью этих алгоритмов. На рисунке 3.15 представлено сравнение быстродействия дескрипторов на трёх датасетах.

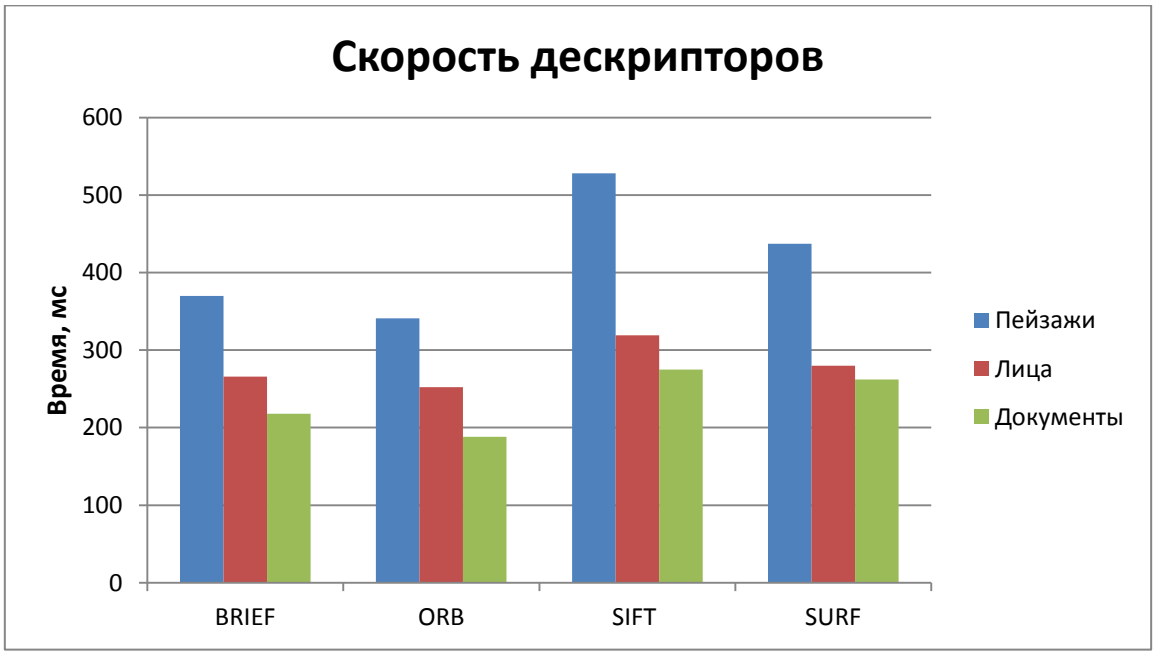


Рисунок 3.15 – Гистограмма быстродействия дескрипторов

Из наблюдений можно сделать следующие выводы:

1. Дескриптор ORB показал лучшие результаты, как в точности нахождения особых точек, так и в скорости их обработки.

2. Наиболее точным дескриптором для пейзажей стал дескриптор ORB, для лиц – SIFT, для документов – SURF.

3. Наиболее быстрыми из всех дескрипторов оказались ORB и BRIEF, а самыми медленными – SIFT и SURF.

4. Дескриптор SURF является самым неточным и медленным из всех сравниваемых дескрипторов.

5. Изображения пейзажей являются наиболее сложными для обработки и вычислений, а  изображения документов – наиболее быстрыми и лёгкими.

Итак, в данном разделе представлена разработка программного обеспечения. Описаны используемый паттерн и методология разработки. Также зафиксированы и проанализированы результаты тестирования на разных типах изображений для каждого из дескрипторов.

# ЗАКЛЮЧЕНИЕ

Данная бакалаврская работа посвящена исследованию дескрипторов особых точек изображений и их сравнительному анализу. Большое разнообразие и широкое применение дескрипторов особых точек в области компьютерного зрения делает актуальным нахождение наиболее точного и быстрого из дескрипторов. Целью работы являлось сравнение эффективности работы дескрипторов особых точек путём подсчета точности и скорости их работы.

Для этого были рассмотрены и проанализированы четыре дескриптора особых точек: BRIEF, ORB, SIFT и SURF. Эффективность дескрипторов была измерена с использованием наиболее часто встречающихся типов изображений: лиц, пейзажей и текста. В процессе разработки приложения были сформулированы математические аппараты каждого из дескрипторов, которые в дальнейшем были реализованы в программе.

Кроме этого, были формализованы требования к разрабатываемому программному продукту, описан язык программирования, с помощью которого разработаны алгоритмы программы, а также была описана работа данного программного обеспечения.

В результате работы были сделаны выводы по точности и скорости работы каждого из дескрипторов особых точек. Также был проведен сравнительный анализ дескрипторов для каждого набора изображений.

Данное программное обеспечение является легко масштабируемым. В дальнейшем его можно использовать для сравнения дескрипторов на других типах изображений, не включенных в работу, а также добавить реализации новых дескрипторов особых точек.

## СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ


1. Визильтер Ю. В. Обработка и анализ изображений в задачах машинного зрения: Курс лекций и практических занятий. – М.: Физматкнига, 2010. – 672 с.
2. Гонсалес Р. Цифровая обработка изображений / Р. Гонсалес, Р. Вудс. – Москва : Техносфера, 2012. – 1104 с.
3. И. С. Грузман Цифровая обработка изображений в информационных системах / Грузман И.С., Киричук В.С., Косых В.П., Перетягин Г.И., Спектор А.А.: Учебное пособие.- Новосибисрк: Изд-во НГТУ, 2002. - 352 с.
4. Использование GPU для решения задач компьютерного зрения в библиотеке OpenCV [Электронный ресурс] / Режим доступа: http://agora.guru.ru/hpc-h/files/Using_GPU_for_computer_vision_in_OpenCV_library.pdf
5. Клейнберг Дж., Тардос Е. Алгоритмы: разработка и применение. Классика Computers Science / Пер. с англ. Е. Матвеева. – СПб.: Питер, 2016. – 800 с.
6. Машинное зрение: понятия, задачи и области применения [Электронный ресурс] / Режим доступа: http://rusnauka.com/25_SSN_2009/Informatica/51050.doc.htm.
7. Н. Красильников Цифровая обработка 2D- и 3D-изображений / Красильников Н.: Отдельное издание. - БХВ-Петербург, 2016. - 608 с.
8. Паттерн посредник [Электронный ресурс] / Режим доступа: https://refactoring.guru/ru/design-patterns/mediator
9. Системы компьютерного зрения: современные задачи и методы [Электронный ресурс] / Режим доступа: http://controleng.ru/innovatsii/sistemy-komp-yuternogo-zreniya-sovremenny-e-zadachi-metody/.
10. Что такое машинное зрение и чем оно отличается от человеческого? [Электронный ресурс] / Режим доступа:

https://meduza.io/feature/2019/03/30/chto-takoe-mashinnoe-zrenie-i-chem-ono-otlichaetsya-ot-chelovecheskogo-seychas-ob-yasnim-ponyatno

11. A tutorial on binary descriptors [Электронный ресурс] / Режим доступа: https://gilscvblog.com/2013/10/04/a-tutorial-on-binary-descriptors-part-3-the-orb-descriptor/

12. Beyerer J. Automated Visual Inspection: Theory, Practice and Applications / J. Beyerer, P. Fernando, F. Christian. – Springer Berlin Heidelberg, 2016. – 798 p.

13. Calonder, M. BRIEF: binary robust independent elementary features / M. Calonder, V. Lepetit, C. Strecha, P. Fua // European Conference on Computer Vision, – 2010. – P. 778-792.

14. Feature detection and description [Электронный ресурс] / Режим доступа: https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_table_of_contents_feature2d/py_table_of_contents_feature2d.html

15. James L. Pro JavaFX 2/ L. James, G. Weiqi, C. Stephen, I. Dean, V. Johan. – Apress, 2012. – 641 p.

16. Just look at the image: viewpoint-specific surface normal prediction for improved multi-view reconstruction [Электронный ресурс] / Режим доступа: https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/Galliani_Just_Look_at_CVPR_2016_paper.pdf

17. Introduction To Feature Detection And Matching [Электронный ресурс] / Режим доступа: https://medium.com/data-breach/introduction-to-feature-detection-and-matching-65e27179885d

18. Introduction to SURF [Электронный ресурс] / Режим доступа: https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_surf_intro/py_surf_intro.html

19. ORB: an efficient alternative to SIFT or SURF / E. Rublee, V. Rabaud, K. Konolige, G. Bradski // IEEE International Conference on Computer Vision. – 2011. –Vol. 58, Issue 11. – P. 2564-2571.

20. ORB: Oriented FAST and Rotated BRIEF [Электронный ресурс] / Режим доступа: http://www.willowgarage.com/sites/default/files/orb_final.pdf

21. SIFT: Theory and practice [Электронный ресурс] / Режим доступа: https://aishack.in/tutorials/sift-scale-invariant-feature-transform-introduction/

22. SURF: Speeded Up Robust Features [Электронный ресурс] / Режим доступа: http://people.ee.ethz.ch/~surf/eccv06.pdf