Введение

Панорама (от греч. πᾶν «все» и ὅραμα «видение») – это любое представление пространства с широким углом и большой глубиной обзора, будь то рисунок, картина, фотография, видео, карта или трехмерная модель. Понятие впервые стало использоваться в живописи, для описания особого типа картин. Сегодня панорамы встречаются повсеместно как средства наглядного отображения больших пространств со множеством деталей. Существует особый вид панорам, охватывающих все возможные углы обзора из определенной точки, т.е. позволяющих создавать виртуальные трехмерные сцены. В том или ином виде, панорамы используются для научных, развлекательных и рекламных целей.

Подавляющее большинство панорам имеют вид фотографий. Современные объективы фотоаппаратов жестко ограничены в плане углов обзора, и часто неспособны охватить всю требуемую картину. Для решения этой проблемы при съемке получается набор фотографий с различным углом поворота камеры, затем при пост-обработке они объединяются в панораму.

Процесс пост-обработки существенно варьируется по сложности. При использовании специальной аппаратуры и техники съемки, генерация панорамы может происходить «на лету», с помощью самого фотографирующего устройства. Такой метод применяется повсеместно, однако имеет ряд существенных недостатков:

- дороговизна аппаратуры для качественной съемки;
- практическое отсутствие настроек процесса генерации, что в некоторых условиях может приводить к неизбежности дефектов в получаемой панораме.

Более сложная пост-обработка происходит вне фотоаппарата, в удобной и настраиваемой среде. Такой метод ослабляет зависимость качества получаемой панорамы от исходных фотографий, позволяя использовать в этой роли произвольные изображения - как полученные непосредственно для этой цели, так и доступные из других источников. Это избавляет от необходимости в специальном процессе съемки, делая создание панорам легкодоступным процессом. Такой подход предоставляет большой простор для творчества, однако ставит новую проблему: ручная подгонка и редактирование изображений может быть весьма трудозатратным процессом. Он поддается автоматизации, но любое ограничение контроля пользователя над процессом увеличивает вероятность получения неудовлетворительного результата. Необходим компромисс, и используемые средства автоматизации должны позволять изменять баланс для конкретной ситуации. Создание таких средства является нетривиальной задачей, использующей знания нескольких научных дисциплин.

Назначение разработки данного программного средства – развитие научной области путем решения актуальных проблем и новой реализации существующих решений.

1 АНАЛИТИЧЕСКИЙ ОБЗОР

1.1 Область применения и теоретические основы

Автоматизация построения панорам относится к таким научным областям, как компьютерное зрение и обработка изображений. Компьютерное зрение – теория и технология создания машин, которые могут производить обнаружение, слежение и классификацию объектов [1]. Она относится к теории создания искусственных систем, которые получают информацию из изображений. Обработка изображений – форма обработки информации, для которой входные данные представлены двухмерным изображением. Хотя по определению эта область и является производной от области компьютерного зрения, вторая часто упоминается как отдельная, сосредотачивающаяся на обработке трехмерных сцен (работа же с панорамами подразумевает оперирование терминами как двух-, так и трехмерного пространства).

Всю область можно охарактеризовать как молодую, разнообразную и динамично развивающуюся. Интенсивное её изучение началось лишь в конце 1970-х гг., когда компьютеры смогли управлять обработкой больших наборов данных, какими являются изображения. И сейчас нет стандартной формулировки этой области, а многие методы и приложения всё ещё находятся на стадии фундаментальных исследований. В последнее время наблюдается повышение активности изучения области, ввиду всё большего применения её методов в коммерческих продуктах. Важнейшей прикладной областью, где используется компьютерное зрение, является медицина. Компьютерная обработка позволяет отобразить данные, получаемые с помощью микроскопии, рентгенографии и томографии, в наиболее оптимальном для исследования экспертами виде. В промышленности

Цель компьютерного зрения – имитация биологического зрения, чрезвычайно сложного и многоаспектного явления. Так, процесс человеческого видения зависит как от биологических особенностей субъекта, так и от его психики, всего накопленного опыта и инстинктов. Точное воспроизведение подобных явлений с помощью искусственных артефактов практически невозможно. В то же время, попытки автоматизации зрительного процесса уже дали ощутимые плоды, что говорит о перспективности дальнейших исследований в этой области.

Существует два способа хранения цифровых изображений: векторный и растровый. Векторная графика представляет набор элементарных геометрических объектов, иным словом, описывается с помощью математических функций. Подобное представление обычно подразумевает предрасположенность изображений к семантическому (содержательному, смысловому) анализу. С другой стороны, растровая графика оперирует лишь упорядоченным множеством атомарных единиц графической информации – пикселей. Так как понятие изображения подразумевает двухмерность, пиксели

объединены в матрицу размерности x*y. Каждый пиксель, помимо уникальных координат в матрице, характеризуется также значениями таких свойств, как интенсивность и цвет.

Любое изображение, полученное из реального мира, имеет изначально растровый характер. Биологическое зрение связано с хорошо развитой нейронной сетью смотрящего, которая с высокой скоростью и точностью объединяет поступающую с сетчатки глаза информацию в узнаваемые образы. Вычислительная техника по своей сложности еще очень далека от органического мозга, поэтому компьютерное восприятие столь больших объемов данных имеет весьма ограниченный характер.

Однако, одно из преимуществ современной вычислительной техники над человеческих мозгом – скорость вычислений. Статистический анализ изображений используется повсеместно для имитации и замены семантических операций над ними.

К числу таких операций относится воссоздание изображений по их сегментам. Тривиальная для человека задача соотнесения частей целого может решаться автоматически с помощью сложного математического механизма. Статистический аспект механизма придает его работе некоторую долю погрешности, однако постоянное развитие технологий и расширение баз данных постепенно сводит эту погрешность к минимуму.

1.2 Склейка изображений

Задача склейки нескольких изображений в общее их представление – одна из старейших в области компьютерного зрения [2]. Уже несколько десятилетий специальные алгоритмы используются для составления карт и картин высокого разрешения (наиболее известное их применение – обработка спутниковых и телескопных фотографий). За такой срок было разработано множество различных подходов. К примеру, изначально изображения соотносились друг с другом по их пиксельному сравнению и минимизации различий. Сегменты вручную выстраивались в необходимом порядке, а вычислительные машины подгоняли их один под другой (см. рисунок 1.1). Позднее был разработан более быстрый подход, оперирующий уже не каждым отдельным пикселем, но множеством локальным особенностей изображения. Появилась возможность генерации панорам по неупорядоченному набору изображений.

Определение процесса синтеза панорамы состоит в решении следующих задач [3]:

1) Описание отношений координат пикселей одного изображения по отношению к другому;

2) Нахождение и оценка корректности вариантов относительного расположения сегментов;

3) Совместное позиционирование всех сегментов;

4) Выбор поверхности или холста, на котором будет располагаться скомпонованное сегменты;

5) Сочетание сегментов в виде цельной панорамы.

Рисунок 1.1 – Склейка сегментов в панораму

Общий алгоритм генерации или синтеза панорам на основании множества сегментов состоит из трех этапов, каждый из которых имеет специфические проблемы и методы их решения:

1) Регистрация.

2) Калибровка.

3) Сочетание.

Регистрация подразумевает анализ исходных сегментов панорамы. Этот этап пропускается при использовании метода прямой подгонки. Калибровка – соотнесение сегментов друг с другом. Сочетание – объединение сегментов в одно изображение и ликвидация признаков склейки.

1.2.1 Модели трансформаций

Одной из фундаментальных операций при склейке изображений является их пространственное преобразование. Исходные сегменты представляют собой проекции некоторой сцены на плоскость объектива в разных позициях, углах наклона и поворота. Сегменты не хранят подобную информацию, однако её наличие является необходимым условием для воссоздания сцены в виде панорамы.

В этом контексте задачу объединения можно выразить как нахождение такого преобразования для каждого сегмента, которое позволит корректно разместить его на панораме. Для практического оперирования понятием преобразования, необходимо определить используемую математическую модель (*motion model*). В общем случае она отображается как матрица параметров размерностью 3 x 3 и используется в составе оператора, определяющего однозначное соответствие пары точек в двухмерном пространстве координат. Расчеты с использованием всех восьми параметров (девятый всегда принимается равным единице) могут требовать значительных вычислительных затрат, поэтому зачастую матрицу ограничивают, приспосабливая для конкретных типов преобразований [4].

Двухмерные перемещения используются при обработке видеоданных, когда разница между положением объектов на двух соседних кадрах невелика. Эта модель оперирует всего двумя параметрами.

Двухмерное евклидово преобразование – трехпараметровая модель, добавляющая к перемещению эффект вращения. Используется при работе с плоскими вращениями, например, при частичном сканировании большого изображения на малом сканере.

Масштабируемое вращение, или преобразование подобия, прибавляет четвертый параметр. Эффективно при склейке сегментов, полученных с помощью с большим фокусным расстоянием. Важно отметить, что данная модель сохраняет углы на изображении неизменными.

Аффинное преобразование - шестипараметровая модель, использующая верхние два ряда универсальной матрицы. Модель не влияет на свойство параллельности объектов.

Гомографическое, или перспективное, преобразование – самое общее из двумерных, определяющееся восемью параметрами. Оно сохраняет прямоту линий на изображении и считается достаточным для эффективной обработки фотографий. Однако, определение параметров гомоморфной модели заключается в решении системы уравнений с восемью неизвестными, что может пагубно отразиться на скорости работы вычислительной системы. Поэтому, в зависимости от конкретной ситуации рекомендуется использовать модель с наименьшим количеством параметров.

Все двухмерные преобразования координат наглядно продемонстрированы на рисунке 1.2.

Существую и более сложные модели – такие, как преобразование в трехмерной плоскости, использующее матрицу размерностью 4 на 4. Подобные модели не принято использовать при склейке изображений из-за чрезмерной сложности вычислений, поэтому гомоморфная модель используется как универсальная, но с рядом допущений:

- исходные фотографии сделаны неперемещающейся камерой с вращением по фиксированной оси;

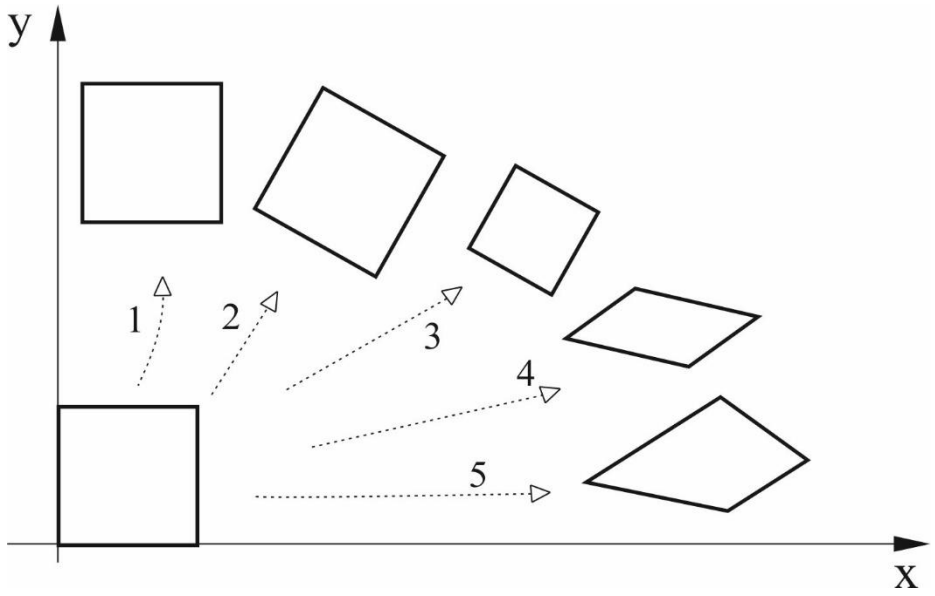

Рисунок 1.2 – Двухмерные преобразования:
1 – перемещение, 2 – евклидово, 3 – подобие, 4 – аффинное, 5 – перспективное

- фотографируемая сцена считается плоской, что приемлемо при достаточном удалении фотокамеры от объектов сцены.

За определением используемой модели преобразований следует выбор метода нахождения параметров этой модели. Процедура нахождения называется выравниванием, и для неё существуют два принципиально разных подхода.

1.2.2 Выравнивание сегментов

Исходные данные: два изображения (сегмента), для которых необходимо определить преобразование, объединяющее их в одно изображение. Прямое выравнивание заключается в последовательном применении различных преобразований и выбора оптимального, по некоторому критерию (метрика ошибок). Простейший пример – полный поиск, применяющий все возможные варианты преобразования. Очевидным недостатком такой реализации является большое время работы. Способы оптимизации:

- иерархический поиск на основе пирамид изображений;
- использование преобразования Фурье для ускорения вычислений;
- инкрементный поиск с разложением изображения в ряд Тейлора.

Прямое выравнивание основано на вычислениях в пиксельном масштабе, когда каждый пиксель является компонентом итерирования. В противовес этому низкоуровневому подходу существуют более абстрактные, самый известный из которых – использование *локальных особенностей*. Он заключается в оперировании не всем множеством пикселей, а лишь

небольшим набором элементов, относящихся к объектам на изображении. Если сегменты относятся к одной сцене, есть вероятность наличия одного объекта на нескольких из них, и каждый сегмент будет иметь характерную этого для объекта особенность. Также, если у объекта имеется несколько особенностей, их относительное расположение не должно существенно меняться от сегмента к сегменту. Таким образом, эти особенности относятся в больше степени к фотографируемой сцене, чем к фотографиям (см. рисунок 1.3). Различие между представлением одной особенности на разных сегментах выражается в форме оператора преобразования, который затем может быть применен к каждому пикселю сегмента при его вставке в панораму.

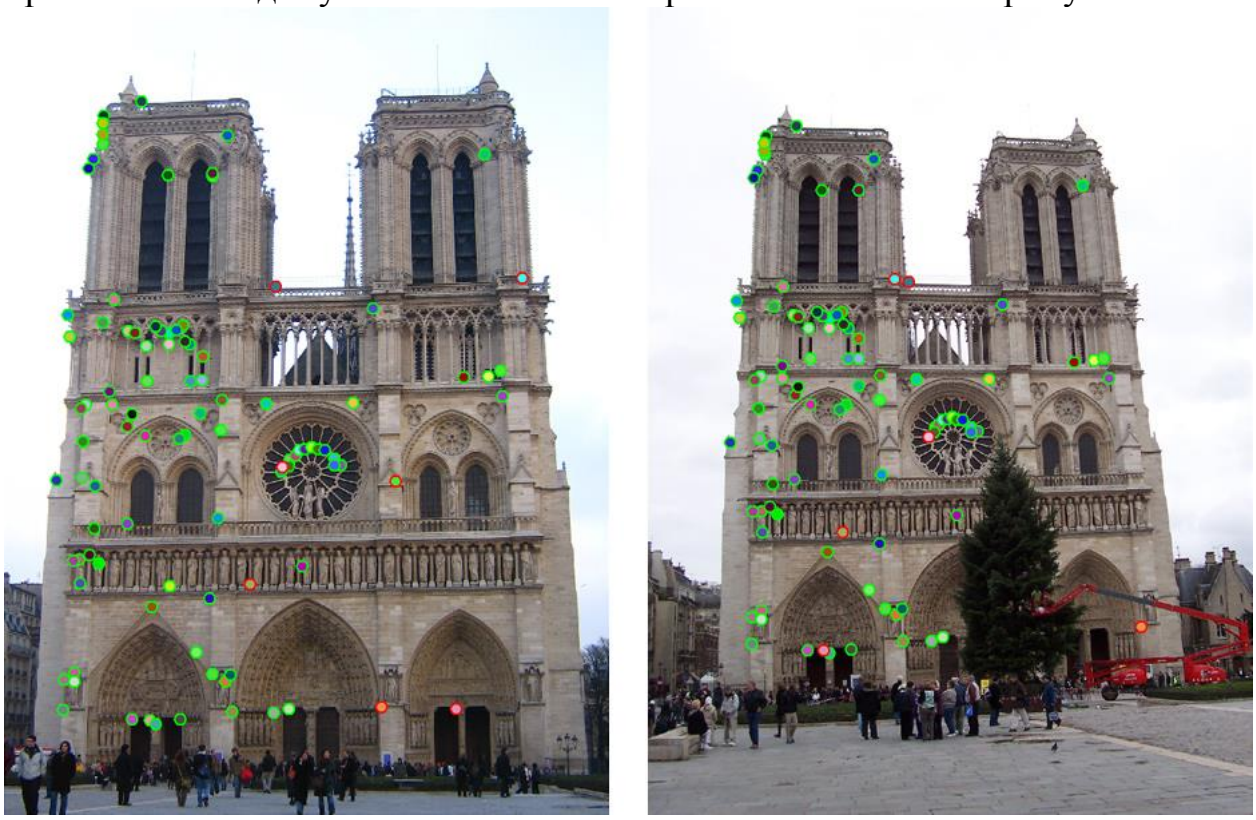

Рисунок 1.3 – Совпадение ключевых точек на различных изображениях

Миколайчик и Туителаарс [5] определили набор необходимых свойств, которыми должна обладать особая точка:

- повторяемость – особая точка находится в одном и том же месте в изображаемой сцене или объекте, несмотря на изменения точки обзора и освещенности;
- различаемость – окрестности особых точек должны иметь существенные отличия друг от друга, достаточные для сопоставления этих точек;
- локальность – особая точка должна занимать небольшую область изображения для минимизации геометрических и фотометрических искажений;

- количество – число и плотность особых точек должны быть достаточными для обнаружения малых объектов (варьируется в зависимости от области применения);

- точность – обнаруженные точки должны точно локализовываться, в исходном или ином масштабе;

- эффективность – время обнаружения особых точек должно быть в допустимых для приложения пределах.

Задача поиска локальных особенностей на изображении является нетривиальной. Необходимым свойством особенности является стойкость к преобразованиям. Стандартом сегодня является независимость, в той или иной мере, ко всем двумерным трансформациям вплоть до аффинного преобразования. Это обеспечивается подходящим дескриптором особенности.

Существует ряд комплексных методик для поиска и описания локальных особенностей. Наиболее популярные из них будут рассмотрены далее в подразделе 1.4.

Выравнивание изображений при помощи особых точек может быть дополнено прямым выравниванием на конечном этапе, когда преобразования сегментов вычислены, но содержат некоторую долю погрешности. Т.е. пиксельное выравнивание эффективно при уточнении значений преобразований на ограниченном интервале значений.

1.2.3 RANSAC

После описания особенностей каждого сегмента выполняется процедура их сравнения, имеющая целью нахождение совпадений и вычисление соответствующих им преобразований. Идентичные дескрипторы локальной особенности на практике встречаются крайне редко, поэтому сравнение допускает определенную величину отклонения. Это порождает новую проблему - нахождение ложных совпадений. Для определения достоверных пар связанных сегментов используются статистические алгоритмы, к примеру, RANSAC.

RANSAC (Random Sample Consensus) – статистический метод оценки параметров модели на основе случайных выборок. Предложен Фишлером и Боллесом в 1981 г. Алгоритм предназначен для построения и оценки моделей по выборке данных, содержащей выбросы. Точные методы (типа метода наименьших квадратов) в таком случае приведут к выведению некорректной модели, так как она будет удовлетворять всем исходным данным, независимо от их корректности. Статистический метод основан на предположении, что количество некорректных данных значительно ниже, и они могут быть расценены как «выбросы». Так, RANSAC выбирает из исходных данные некоторые наборы ограниченной величины и строит модель («гипотезу») на их основе. Каждая полученная таким образом модель оценивается с применим всего набора исходных данных методом подсчета совпадений. Выбирается та гипотеза, которую подтверждает наибольшая доля данных (см. рисунок 1.4).

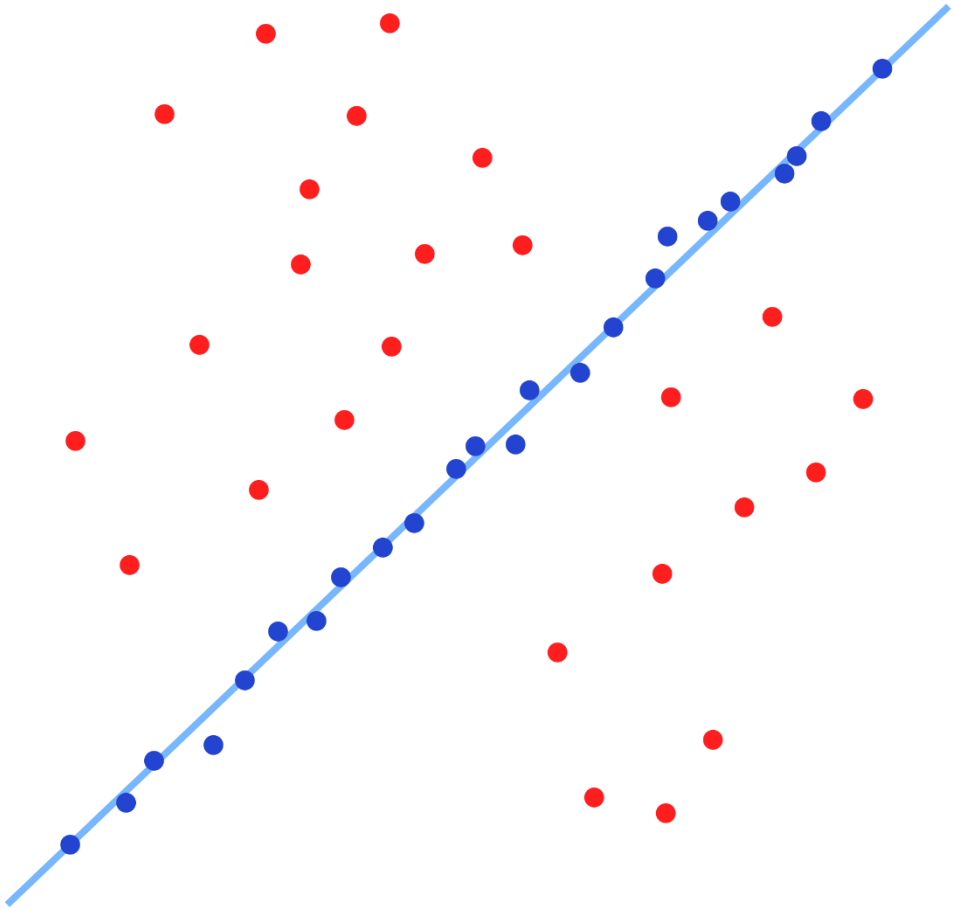

Рисунок 1.4 – Применение алгоритма RANSAC для построения прямой по набору точек, содержащему сдвиги и выбросы

Метод RANSAC способен дать надежную оценку параметров модели, то есть с достаточно высокой точностью, даже если в исходном наборе данных присутствует значительное количество выбросов. Недостатком метода является отсутствие верхней границы времени, необходимого для вычисления параметров модели. Если использовать в качестве некоторой границы времени максимальное число итераций, полученное решение может быть неоптимальным, а также существует очень малая вероятность того, что ни одна модель не будет соответствовать исходным данным. Точная модель может быть определена с некоторой вероятностью, которая становится тем больше, чем большее количество итераций используется. Другим недостатков RANSAC является то, что для выполнения алгоритма необходимо задавать конкретные пороговые значения. Наконец, методом RANSAC можно определить только одну модель для определенного набора данных. Соответственно, если данные предполагают наличие нескольких наличий, может быть не найдена ни одна.

1.2.4 Глобальная регистрация

В предыдущих подразделах было определено, как вычисляются парные преобразования, объединяющие один сегмент с другим. Однако, цель процесса – единое изображение, совмещающее в себе множество сегментов. То есть, необходима глобальная система, в которой каждому сегменту будет соответствовать определенное пространственное преобразование, безотносительно других сегментов.

Простейшее решение – последовательно добавлять сегменты на панораму, выравнивая каждый новый с ранее установленными и определяя их перекрывающиеся области. В случае, если строится панорама с углом обзора 360, аккумулированная ошибка неизбежно приведет к образованию пробела между двумя концами изображения. Более универсальная альтернатива такого метода – одновременное выравнивание всех сегментов с равномерным распределением общей ошибки. Такой способ называется методом наименьших квадратов и заключается в нахождении такой системы расположений всех сегментов, которая минимизирует суммарный квадрат разницы между текущим положением пикселей и предполагаемым. Предполагаемая позиция вычисляется преобразованиями координат точки между трехмерным и двумерным пространствами.

1.2.5 Ретуширование

Когда все сегменты расположены на панораме, необходима дополнительная обработка для гарантии целостного восприятия получаемого изображения. В общем случае после подгонки сегментов остаются видны дефекты, обусловленные качеством исходных изображений или спецификой применяемых алгоритмов (см. рисунок 1.5). Наиболее часто встречаемые дефекты: видимые швы (из-за разницы в яркости сегментов), нечеткость (из-за ошибок при регистрации сегментов) и «призраки» (движущиеся объекты на фотографиях). Четкое изображение достигается путем сокрытия лишних пикселей и размытия. Для выполнения этих действий разработано множество методик.

Медианный фильтр и сглаживание (*feathering*) решают проблемы шума и локальных дефектов. Одним из направлений развития этих фильтров является центрированное взвешивание пикселей. Так, в случае панорам многие проблемы решаются путем фокусировки сглаживающего фильтра на центр изображения (то есть, чем ближе пиксель находится к центру, тем большую роль он играет при корректировке интенсивности других пикселей).

Рисунок 1.5 – Видимые дефекты при склейке панорам

При перекрытии одной области несколькими сегментами их границы могут быть слишком заметны, возникает отчетливый «шов». Существует несколько методов решения проблемы. Один из таких – нахождение идеального шва, на котором границы двух сегментов имеют наименьшее различие. Такой шов прокладывается из одного конца стыка в другой, путем оптимального выбора каждого следующего шага (пикселя).

Для ликвидации «призраков» Uyttendaele разработал алгоритм [3], заключающийся в сканировании области перекрытий на наличие регионов с наибольшей разницей между сегментами и заменой его тем или иным вариантом (в зависимости от конкретной реализации алгоритма).

Существует большое множество других алгоритмов, но практически все они предоставляют лишь частичное решение проблем. Дальнейшие улучшения требуют разработки специальных алгоритмов или оптимизации процесса склейки.

1.3 Методы нахождения и описания особых точек

1.3.1 SIFT

SIFT (Scale-Invariant Feature Transform) – алгоритм из области компьютерного зрения для детектирования и описания локальных особенностей изображения. Опубликован Д. Лоуи в 1999 г. Применяется для распознавания объектов, в робототехнике, трехмерном моделировании, распознавании жестов и отслеживании объектов на видео. SIFT способен точно идентифицировать объекты в сложных ситуациях, например, при перекрытии. Дескриптор SIFT полностью инвариантен к вращению и масштабированию, частично – к аффинным преобразованиям и смене освещения.

Особые точки определяются путем построения пирамиды масштабов для каждого изображения и нахождения локальных экстремумов интенсивности их пикселей (разница гауссиан). Затем они последовательно уточняются по направлению градиента разницы гауссиан. При помощи местных градиентов интенсивности определяется «направленность» особой точки. В конечном виде дескриптор имеет вид вектора.

При последовательном итерировании по исходным изображениям, для каждого определяется набор дескрипторов и сравнивается (по метрике евклидова расстояния) с ранее найденными наборами. Из совпадающих пар выделяется подмножество, характеризуемое одинаковым преобразованием позиции, масштаба и поворота. Нахождение таких подмножеств зачастую оптимизируется с помощью преобразования Хафа. Выделяются кластеры мощностью более трёх пар и так далее, до тех пор, пока не будет достигнута заданная достоверность совпадений.

Хотя метод SIFT подвержен статистическим ошибкам, точность детектирования относительно высока, что обусловило его распространенность.

1.3.1 SURF

SURF (Speeded-Up Robust Features) – алгоритм детектирования особенностей изображения, разработанный Г. Бэем в 2006 г. Применяется для распознавания объектов и трехмерной реконструкции. Метод частично основан на SIFT.

Упор делается на повсеместном использовании интегральных изображений (см. рисунок 1.6). Интегральным представлением изображения называется матрица той же размерностью, в которой каждая ячейка содержит сумму яркости соответствующего ей пикселя и значений всех ячеек слева и сверху от неё. Предварительный расчет такой матрицы позволяет значительно ускорить ряд вычислений при работе с изображением.

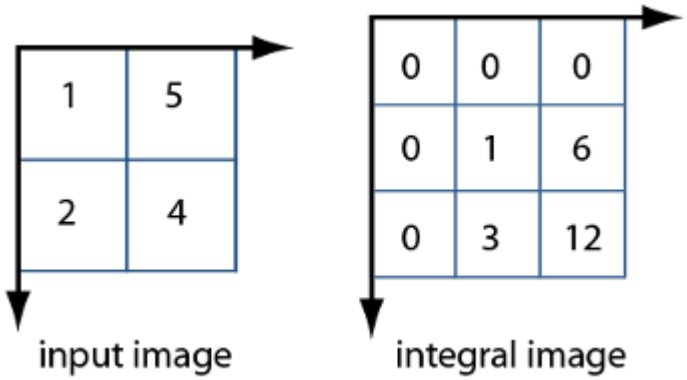

Рисунок 1.6 – Интегральное изображение

В SURF при детектировании локальных особенностей используется лапласиан гауссиана (LoG) и его гессиан. Но фильтр не используется в классическом виде, а заменяется приближенным линейным фильтром (box filter), легко вычисляемым при помощи интегрального изображения. Для описания ориентаций используется отклик вейвлета Хаара, который также может быть рассчитан из интегральной матрицы.

В общем случае SURF работает в несколько раз быстрее, чем SIFT. Однако, его применение неэффективно при обработке фотографий из разных точек съемки, а также при различных уровнях освещения.

1.3.3 Детекторы углов

Широкое применение среди детекторов особых точек находят детекторы углов. Углы – особые точки, которые формируются из двух или более граней, определяющих границы между объектами или их частями. Иначе говоря, угол – точка, в окрестности которой интенсивность изменяется относительно центра в двух доминирующих градиентах. Градиент – векторная величина, показывающая направление наискорейшего возрастания функции (здесь – функции интенсивности пикселей на изображении). Так как изображение дискретно, вектор градиента определяется через частные производные по осям x и y через изменения интенсивностей соседних точек. Большинство детекторов рассматривают угловатость, зависящую от производной второго порядка, поэтому в общем случае чувствительность к шуму высока.

Детектор Харриса (1988) – наиболее эффективный детектор L-связных углов. Характеризуется анизотропией по всем направлениям, следовательно, и инвариантностью к поворотам. Недостаток детектора – сильная чувствительность к шуму и изменениям масштаба.

FAST (Features from Accelerated Segment Test) – другой распространенный детектор углов, введенный Е. Ростеном и Т. Драммондом в 2005 г. Алгоритм рассматривает относительную интенсивность пикселей на фиксированной окружности. В 2008 г. алгоритм дополнен и назван FAST-ER (FAST with Enhanced Repeatability). Главным новшеством стало введение

машинного обучения, сделавшего найденные особые точки повторяемыми (одинаково распознаваемыми на разных изображениях).

Детекторы углов определяют лишь алгоритмы поиска особых точек, но не способ их описания. Поэтому нередко встречаются комбинации типа детектор углов Харриса и SIFT, сочетающие быстроту детектирования углов с инвариантностью дескриптора.

1.4 Способы отображения панорам

Всё поле зрения может быть представлено как сферическая поверхность. Это свойство относится как к биологическому зрению, так и к фотоаппаратам. Из него следует, что при проецировании сферического изображение на плоскую поверхность неизбежны искажения. При малых углах обзора эти искажения могут быть практически незаметны, однако при больших – к примеру, на панорамах – они значительны (см. рисунок 1.7), что может вызывать неудобство восприятия таких изображений. Решением проблемы является применение различных видов проецирования на двухмерную плоскость.

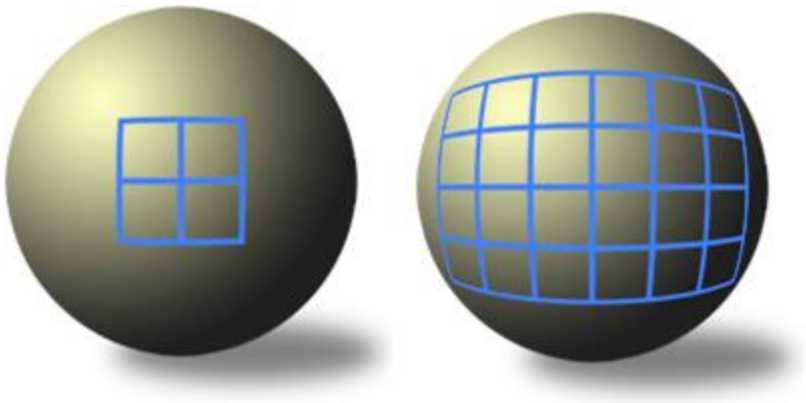

Рисунок 1.7 – Искажения при различных углах обзора

Метод равных прямоугольников заключается в нанесении на сферу координатной сетки в терминах параллелей и меридиан (взаимно ортогональных). Сеть проецируется на своё двухмерное представление, на котором все ячейки имеют равные размеры и форму. Эта проекция имеет горизонтальную длину, в два раза большую, чем её высота. Она характеризуется горизонтальным искажением, сильнее всего проявляющимся на полюсах сферы (верх и низ изображения).

Цилиндрическая проекция схожа с проекцией равных прямоугольников, однако она вносит дополнительное вертикальное искажение, нулевое на экваторе и бесконечно большое на полюсах (верхние и нижние ячейки сети имеют всего три грани). Это позволяет улучшить способность к сохранению размеров и пропорций объектов и изображении, однако вносит дополнительное искажение прямых линий и не позволяет оперировать

большими вертикальными углами обзора. Цилиндрический метод взят за стандарт в панорамных камерах.

Метод «рыбий глаз» (fisheye) создает проекции, в которых расстояние от центра сетки пропорционально покрываемому углу обзора, что придает сходство с отражением на металлической сфере. Горизонтальный и вертикальный углы обзора ограничены 180. Характерная особенность проекции – усиление искажений по мере отдаления от центра изображения. Такой вид проекции редко используется для отображения панорам, однако чрезвычайно полезен при их создании, так как позволяет сократить количество необходимых сегментов буквально до нескольких штук.

Проекция Меркатора – компромисс между цилиндрической и равными прямоугольниками. Характеризуется небольшим вертикальным растяжением и способностью охватывать большие вертикальные углы. Недостаток, по сравнению с цилиндрической проекцией, - большая степень искажения прямых линий. Наиболее широко известное применение проекции Меркатора – плоская карта Земли.

Синусоидальная проекция имеет специфическую особенность – она сохраняет площади при переносе на двухмерную плоскость. Проекции широт являются идеально прямыми линиями. При этом вся имеющаяся информация о сферическом представлении сохраняемся, что позволяет делать обратное проецирование.

Стереографическая проекция схожа с «рыбьим глазом», но характеризуется лучшей сохранностью перспективных свойства. Это достигается путем растяжения объектов в сторону от точки перспективы.

Сравнительные параметры различных видов проекции сферической поверхности даны в таблице 1.1.

Таблица 1.1 – Сравнение различных видов проекции сферической поверхности на двухмерную плоскость

Тип проекции	Поле зрения		Сохранение прямых линий	
	Горизонтальное	Вертикальное	Горизонтальных	Вертикальных
Прямолинейная	<120	<120	Да	Да
Цилиндрическая	120-360	<120	Нет	Да
Проекция Меркатора	120-360	<150	Нет	Да

Продолжение таблицы 1.1

Прямые прямоугольники	120-360	120-180	Нет	Да
«Рыбий глаз»	<180	<180	Нет	Нет
Синусоидальная	360	180	Да	Нет

1.5 Примеры реализации

1.5.1 Photoshop

В популярном средстве для профессиональной обработки изображений Photoshop имеется специальный компонент для объединения изображений – Photomerge. Одним из его частных применений является создание панорам.

Достоинства:

- высокая степень интеграции с другими средствами Photoshop.

Недостатки:

- громоздкость (привязка к Photoshop);
- малый набор настроек;
- ограничения на входные изображения: неизменная яркость и наклон снимков, большая доля перекрытий.

1.5.2 Image Composite Editor

Image Composite Editor (ICE) – программа для компиляции панорамных изображений, созданная Microsoft Research Computational Photography Group. На основе набора перекрывающихся фотографий одной сцены, сделанных из одной точки, приложение собирает панораму в высоком разрешении. В качестве основы могут также использоваться видеоданные. Возможен трехмерный обзор полученных панорам и их сохранение в популярных форматах (JPEG, TIFF, PSD/PSB).

Достоинства:

- бесплатность;
- малый размер и высокая скорость работы;
- минималистичный стилизованный интерфейс (см. рисунок 1.8);
- поддержка множества форматов для экспорта результата, популярных и специфичных;
- возможность преобразования панорам в трехмерные сцены.

Недостатки:

- относительно небольшой набор настроек;

- высокие требования к количеству и степени перекрываемости входных изображений.

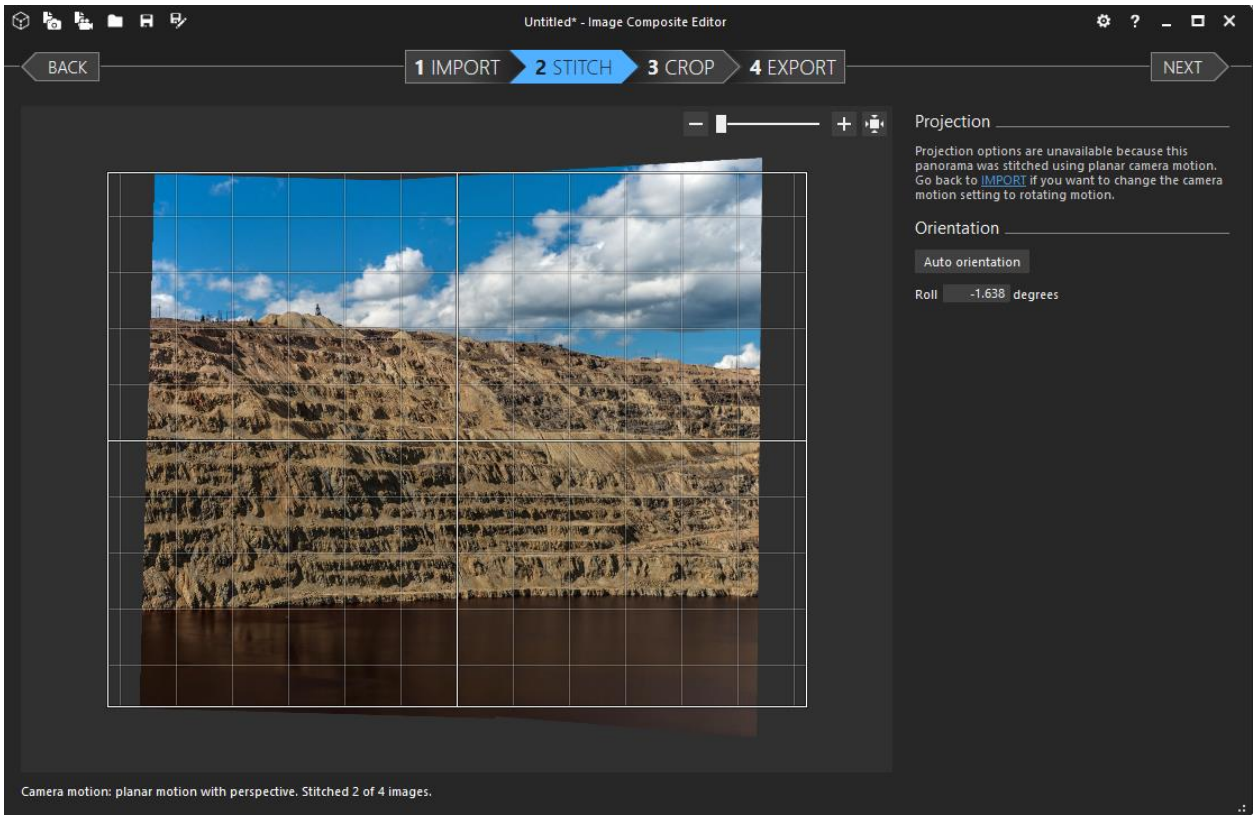

Рисунок 1.8 – Пример использования Image Composite Editor

1.5.3 PTGui

PTGui (Graphical User Interface for Panorama Tools) – специализированное средство для склейки изображений. Изначально разработанное как дополнение к Panorama Tools, ныне является самостоятельным приложением. Предназначается для профессиональной деятельности, поэтому предлагает широкий набор опций и настроек. Среди специфических функций – возможность ручного задания особых точек и их соответствий на разных изображениях (см. рисунок 1.9).

Достоинства:

- высокая степень настраиваемости исходных и результирующих данных, а также процесса склейки;
- встроенный редактор изображений (входных и выходных).

Недостатки:

- ограничения на входные изображения: одинаковые размеры и ориентации, яркость, большая доля перекрытий;
- низкое удобство использования относительно других рассмотренных приложений;

- относительно низкое качество панорамы при настройках «по умолчанию».

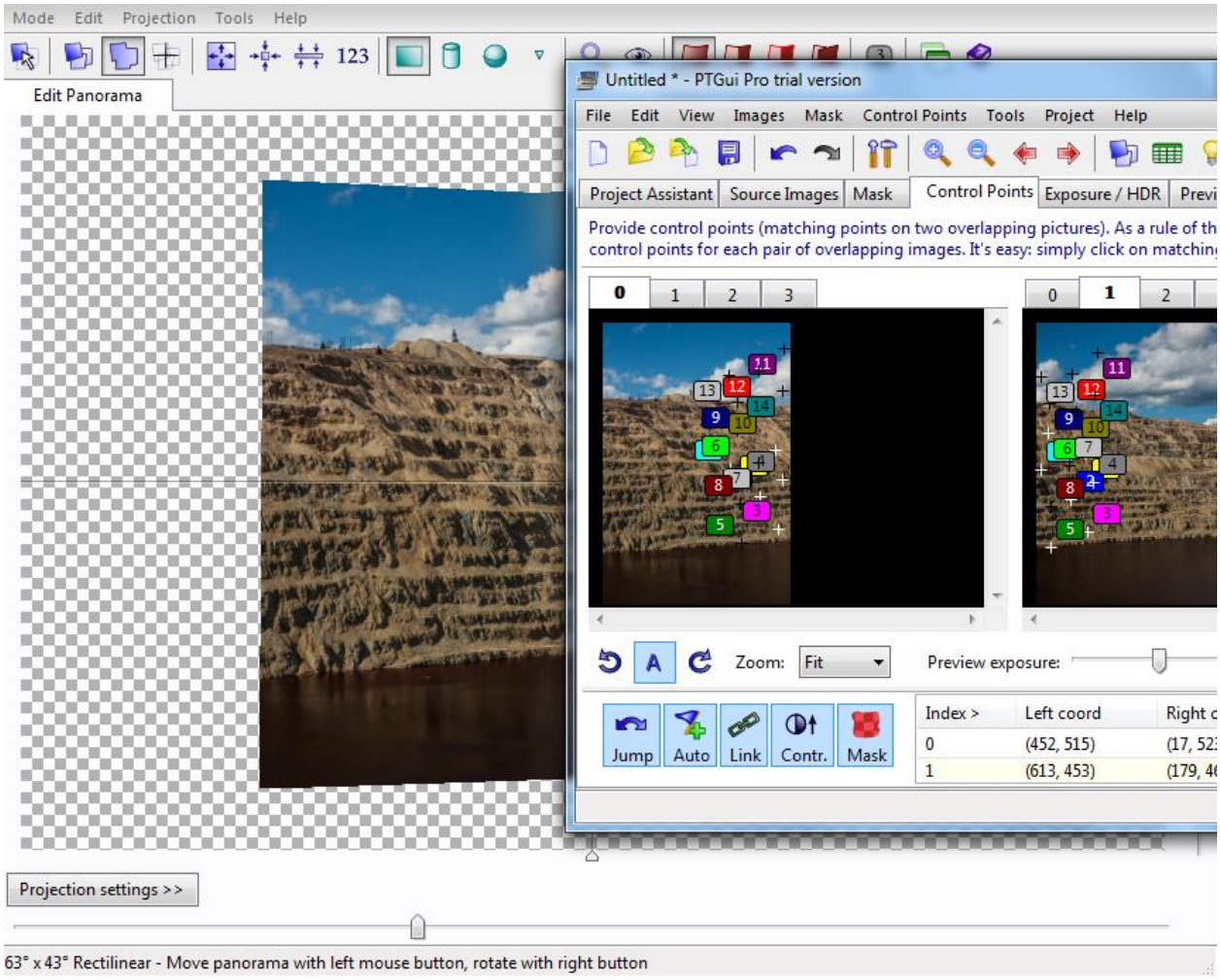

Рисунок 1.9 – Пример использования PTGui

1.5.4 Autopano Pro и Autopano Giga

Профессиональные средства для создания панорам, различающиеся полнотой функционала (Pro – облегченная версия).

Достоинства:

- предварительное автоопределение принадлежности исходных изображений к панораме;
- встроенный редактор входных и выходных изображений, включающий гомоморфные преобразования;
- большое количество настроек;
- возможность преобразования панорам в трехмерные сцены и 3D-туры.

Недостатки:

- высокая стоимость;
- стандартные ограничения для исходных изображений.

1.5.5 Выводы

Все рассмотренные приложения имеют идентичный принцип использования и схожие требования к исходным изображениям:

- значительная доля перекрытий (рекомендуется 20-60%);
- фото сделаны из одной точки нахождения камеры методом поворота вокруг фиксированной оси;
- фото сделаны с одним и тем же уровнем яркости.

Следовательно, применяются схожие методы склейки и обработки изображений. Это говорит о потенциальной возможности полной или частичной ликвидации вышеперечисленных ограничений путем использования иных, возможно, экспериментальных, методов.

1.6 Постановка цели и задач дипломного проекта

Цель – разработка программного средства для автоматизированной генерации панорамных (широкоформатных) изображений из произвольных взаимосвязанных сегментов.

Задачи:

- разработка архитектуры ПС;
- разработка алгоритмов;
- выбор платформы для разработки программного средства;
- проектирование пользовательского интерфейса;
- создание базового проекта в среде программирования;
- создание пользовательского интерфейса;
- программирование и тестирование модулей;
- сборка и тестирование ПС.

2 МОДЕЛИ И СПЕЦИФИКАЦИЯ ТРЕБОВАНИЙ

2.1 Функциональная модель

Входные данные: набор частично перекрывающихся изображений, вводимые пользователем параметры.

Выходные данные: широкоформатное изображение - панорама.

С оглядкой на существующие прикладные реализации рассматриваемой задачи, определяются варианты использования программного средства (см. рисунок 2.1).

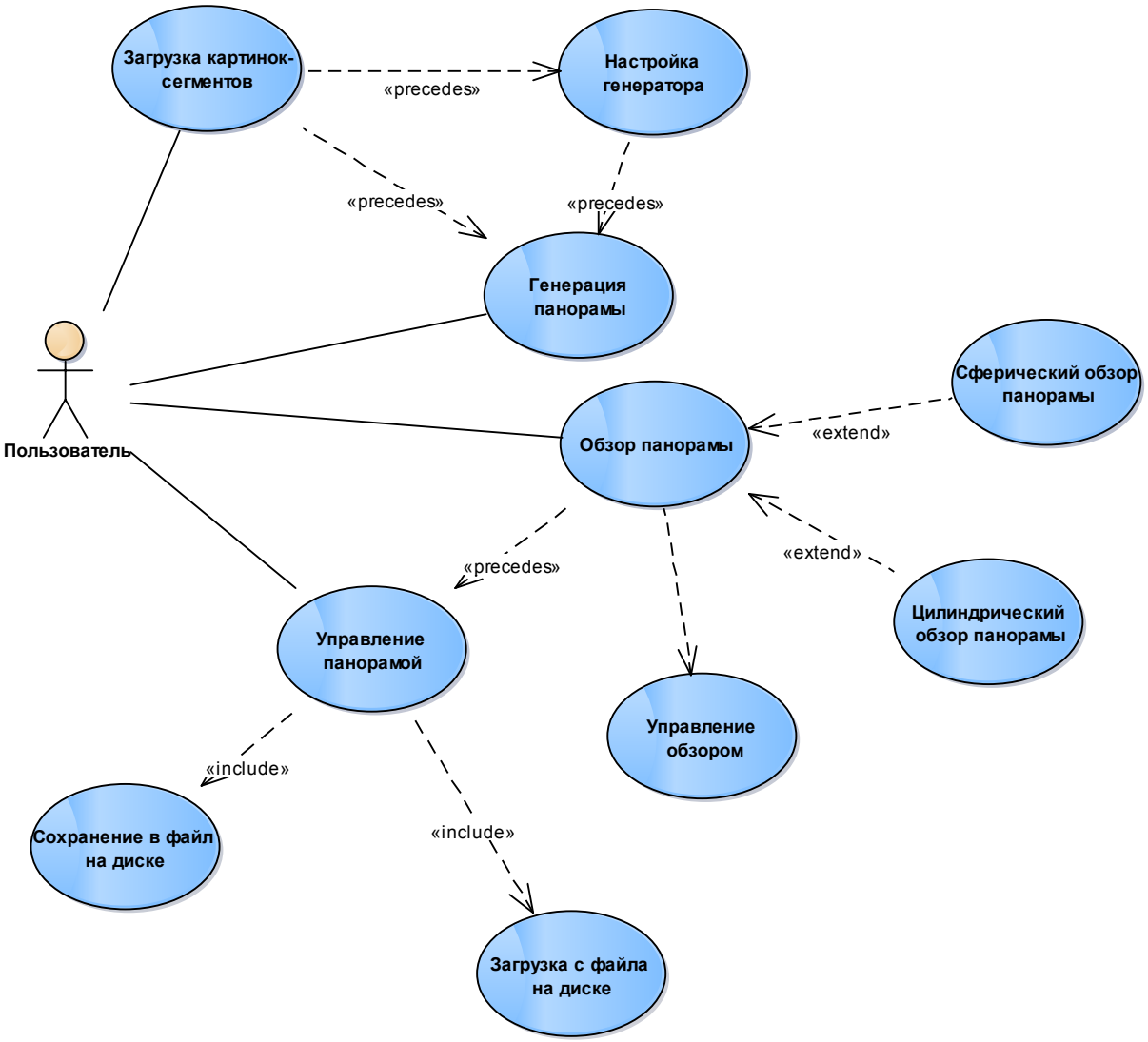

Рисунок 2.1 – Диаграмма вариантов использований

На основе вариантов использования для разрабатываемого программного средства формируется следующий набор функциональных требований:

1) Загрузка набора исходных изображений. Возможность выбора нескольких файлов, отображение уже загруженных, возможность добавления и удаления файлов из набора.

2) Генерация панорамы на основе набора исходных изображений. Применение настроек по умолчанию, возможность их изменения и сброса. Отображение текущего статуса в процессе генерации. Возможность перезапуска генерации с изменением исходных данных или настроек.

3) Графическое представление панорамы. Возможность управления и изменения параметров обзора.

4) Возможность редактирования панорамы. Стандартный набор фильтров и операций для улучшения качества изображения. Обрезка и дополнение краев.

5) Сохранение и загрузка панорамы в файл на диске.

2.2 Спецификация требований

2.2.1 Загрузка и сохранение изображений

Требования:

- поддержка популярных форматов изображений (JPEG, PNG, BMP) при загрузке сегментов и сохранении панорамы;
- поддержка кириллических символов в названии и в пути файлов;
- возможность сохранения текущего вида панорамы до или после изменения параметров её отображения.

2.2.2 Процесс генерации

Требования:

- невозможность запуска генерации при отсутствии или недостаточном количестве загруженных сегментов;
- отображение актуального состояния процесса, с периодом обновления текста не более минуты;
- наличие настроек по умолчанию;
- сохранение пользовательских настроек между сессиями.

2.2.3 Редактирование изображений

Требования:

- выбор инструментов из стандартного набора (цветовые, сглаживающие фильтры);
- возможность выполнения автоматических операций (корректировка яркостей, удаление «призраков»);
- возможность отмены нескольких последних операций.

2.2.4 Пользовательский интерфейс

Требования:
- язык интерфейса – русский;
- отображение приложения в перемещаемом окне;
- возможность сворачивания и разворачивания окна в любое время, независимо от статуса генерации.

2.2.5 Совместимость

Требуется совместимость с операционными системами Windows 7/8.

2.3 Модель предметной области

Разработанная модель предметной области представлена диаграммой на рисунке 2.2.

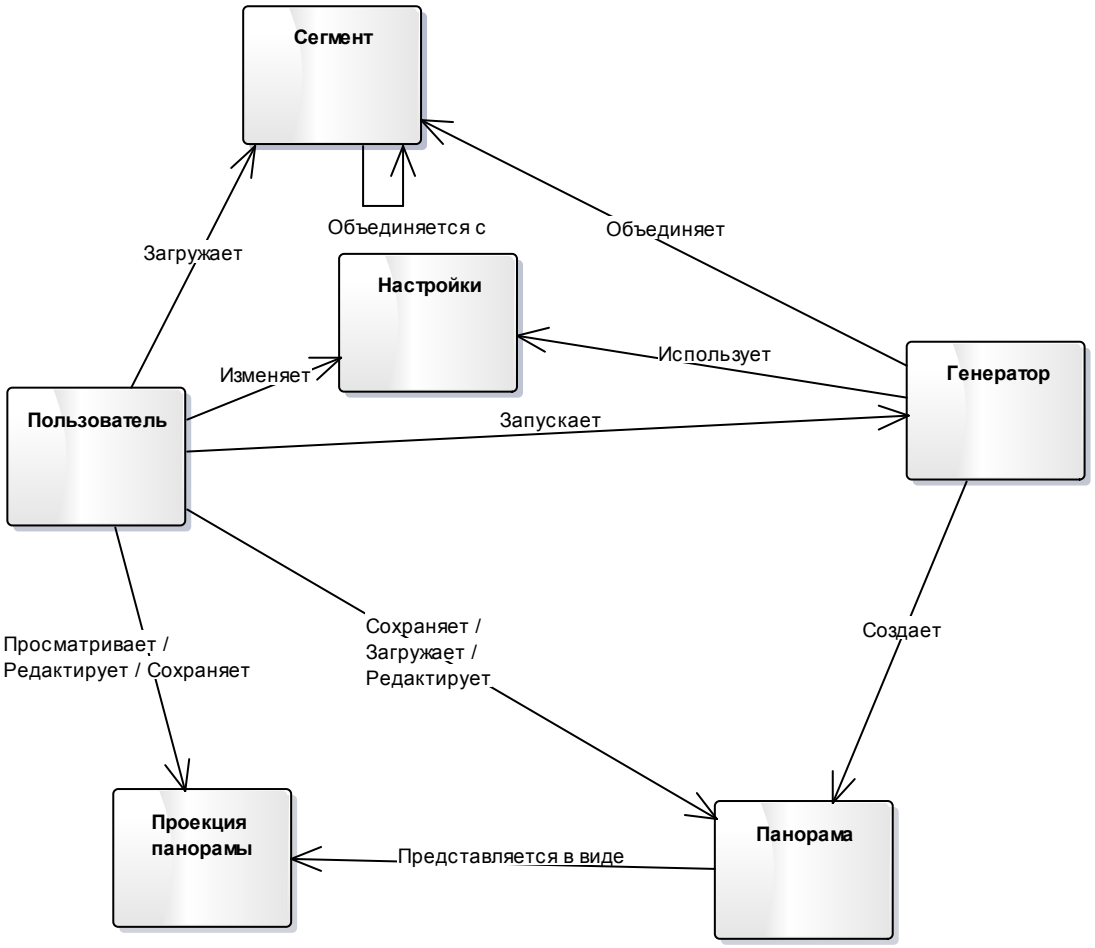

Рисунок 2.2 – Диаграмма модели предметной области

Модель предметной области основывается на требованиях к системе и представляет собой приближенный набор основных понятий и связи между ними. В данном случае объект «Пользователь» и его связи отображают функциональные требования. Его связь «сохраняет проекцию панорамы» означает экспорт в виде изображения, а «сохраняет панораму» - её сохранение в виде файла специального формата, предназначенного для дальнейшего переиспользования в данном программном средстве. Также разделены операции по редактированию самой панорамы и её представления. Первая влияет на все последующие представления панорамы, вторая – только на текущее.

Модель «Генератор» - главный активный компонент программного средства, может служить опорным примитивом при разработке архитектуры.

2.4 Математическая модель

2.4.1 Модель преобразования координат

Перспективное преобразование имеет вид

$$\boldsymbol{x}` \sim \boldsymbol{M}\boldsymbol{x} = \begin{bmatrix} m_0 & m_1 & m_2 \\ m_3 & m_4 & m_5 \\ m_6 & m_7 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \quad (2.1)$$

где $\boldsymbol{x} = (x, y, 1)$ и $\boldsymbol{x}` = (x`, y`, 1)$ – гомогенные и проекционные координаты, а ~ отображает равенство с точностью до масштаба. Операция может быть переписана как

$$x` = \frac{m_0 x + m_1 y + m_2}{m_6 x + m_7 y + 1}, \quad (2.2)$$

$$y` = \frac{m_3 x + m_4 y + m_5}{m_6 x + m_7 y + 1} \quad (2.3)$$

(в модели перемещения лишь параметры m_2 и m_5 используются).

Такое преобразование предполагает чистое вращение камеры (на фиксированной оси), что является справедливым приближением при съемке отдаленных объектов. При условии, что известно фокусное расстояние, модель сокращается до четырех неизвестных параметров.

2.4.2 Прямая подгонка сегментов

Прямое выравнивание изображения $I_1(\boldsymbol{x})$ по отношению к изображению $I_0(\boldsymbol{x})$ заключается в опробывании различных вариантов вектора смещения **u** и нахождении такого, которое соответствует минимальной сумме квадратов ошибок (sum of squared differences, SSD):

$$E_{SSD}(\boldsymbol{u}) = \sum_i [I_1(\boldsymbol{x_i} + \boldsymbol{u}) - I_0(\boldsymbol{x_i})]^2 = \sum_i {e_i}^2, \qquad (2.4)$$

где e_i – остаточная ошибка. В общем случае, модуль вектора $\boldsymbol{u}$ может быть дробной величиной, и из-за этого к дискретным значениям интенсивности картинки I должна быть применена некоторая интерполяция.

Интерполяция, или интерполирование – в вычислительной математике способ нахождения промежуточных значений величины по имеющемуся набору дискретных значений. Линейная - приближение линейной функции по двум её значениям, билинейная – функции двух аргументов. При обработке изображений лучшие результаты (гладкую поверхность) дает бикубическая интерполяция, представляющее функцию как полином третьего порядка (с двумя неизвестными).

2.4.3 Поиск особых точек

Среди универсальных детекторов особых точек решено использовать метод SIFT, как наиболее точный. Далее будет рассмотрено его применение на некотором изображении.

Первый этап – построение пирамиды масштабов изображения при помощи фильтра Гаусса. Это широко применяемый фильтр сглаживания, определяемый формулой

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}, \qquad (2.5)$$

где x и y –координаты текущего пикселя относительно сглаживаемого (ядра размытия), а σ – коэффициент, называемый стандартным отклонением. Из значений функции $G(x, y)$ образуется матрица свертки, налагаемая на окрестность изменяемого пикселя:

$$I`(x, y) = \sum_{dx,dy} I(x + dx, y + dy) * G(dx, dy). \qquad (2.6)$$

Для обеспечения максимальной точности матрица свертки должна покрывать всё изображение, однако, ввиду специфики функции Гаусса, вклад наиболее удаленных пиксели практически равен нулю в сравнении со вкладом ближайших, поэтому величину матрицы обычно ограничивают длиной стороны, равной 6σ.

Для нахождения особых точек на основе исходного изображения строится пирамида гауссианов, а затем – пирамида разностей гауссианов (DoG, Difference of Gaussian). Гауссианом условно принято называть

изображение, к которому применен сглаживающий фильтр Гаусса. Пирамида гауссианов – многоступенчатое упорядоченное множество изображений, полученных с помощью последовательных применений фильтра Гаусса и масштабирований исходного изображения. Пирамида состоит из октав (см. рисунок 2.3), каждая определяется по некоторому значению масштаба. Все октавы включают в себя по N значимых изображений и 2 вспомогательных, каждое получено одно из другого с помощью размытия.

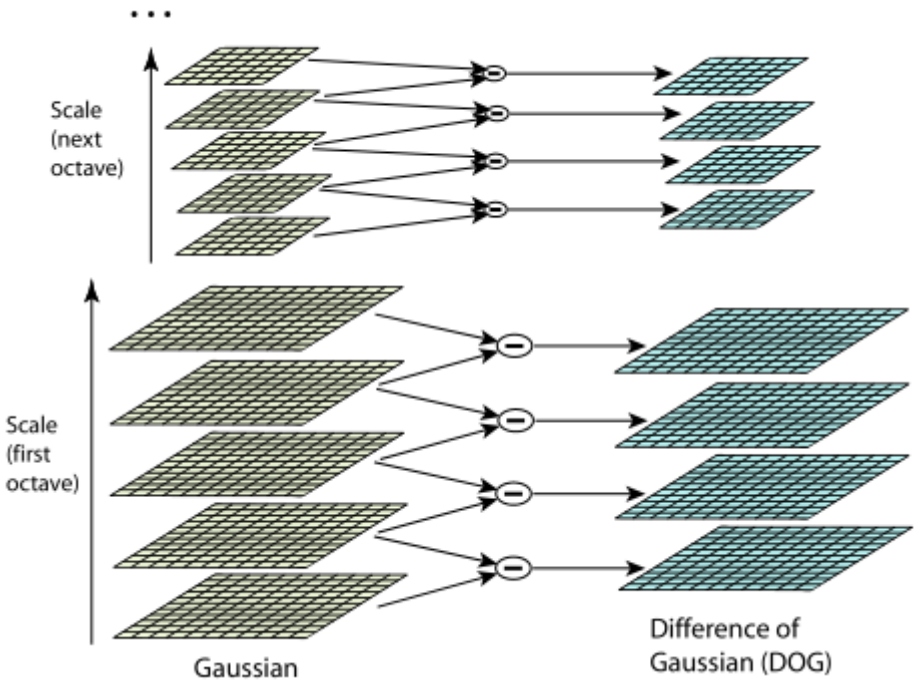

Рисунок 2.3 – Пирамида гауссианов и пирамида их разностей

Разность гауссианов – результат вычитания пиксельных значений одного гауссиана из другого, относящихся к одному изображению, но определяемых разным значением стандартного отклонения σ. Подобная операция позволяет локализовать градиенты интенсивности по их модулям, иначе говоря – разбивать изображение на частотные составляющие. Одним из применений разностей гауссианов является выделение флуктуаций (граней) на картинке. Также они позволяют определять особые точки, по их частотным свойствам.

Таким образом, построенная пирамида DoG служит для поиска локальных экстремумов значения разности. При этом пиксельная точка считается экстремумом (и потенциальной особой точкой), если 8 окружающих ее пикселей, а также 9 пикселей на изображениях выше и ниже по октаве, имеют меньшие (большие) значения разности. Дополнительные две изображения в каждой октаве пирамиды масштабов используются для полноты этой проверки.

Не все из найденных точки пригодны в роли ключевых. Для отбора каждая из них пропускается через ряд проверок. Предварительно следует

уточнить координаты каждой точки, так как при масштабировании они получают некоторое отклонение. Для этого применяется интерполяция функции DoG в виде квадратичного многочлена Тейлора с ядром в точке экстремума:

$$D(\boldsymbol{x}) = D + \frac{\partial D^t}{\partial \boldsymbol{x}}\boldsymbol{x} + 0.5\boldsymbol{x}^{\boldsymbol{T}}\frac{\partial^2 D}{\partial \boldsymbol{x}^{\boldsymbol{2}}}\boldsymbol{x}\,, \tag{2.7}$$

где $\boldsymbol{x} = (x, y, \sigma)$. Точное расположение экстремума находится далее путем приравнивания производной функции приближения D($\boldsymbol{x}$) к нулю. При расчете координат следует учесть, что функция D(x) оперирует не абсолютными координатами, а координатами относительно ядра – рассматриваемого экстремума. И полученные значения x и y будут представлять собой корректирующий сдвиг ядра $\widehat{\boldsymbol{x}}$. Если одна из его компонент превышает значение 0.5, значит, рассматриваемый экстремум – ложный, и следует перейти к рассмотрению другого, в направлении $\widehat{\boldsymbol{x}}$ (если координаты не превышают размер изображения). Если значение функции интерполяции D(**x**) в точке $\widehat{\boldsymbol{x}}$ меньше 0.03, точка характеризуется низкой величиной контраста, и также отбрасывается.

Следующая проверка ликвидирует точки на гранях объектов. Подобные точки являются ярко выраженными экстремумами функции DoG, однако весьма чувствительны к шумам. Граничные точки характеризуются тем, что взаимно перпендикулярные градиенты в них значительно различаются по величине. Это определяется при помощи матрицы Гессе:

$$\boldsymbol{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{yx} & D_{yy} \end{bmatrix}, \tag{2.8}$$

где

$$D_{xy} = \frac{\partial^2 D}{\partial x \partial y}\,. \tag{2.9}$$

Если определить неизвестные α и β через систему

$$\begin{aligned} Tr(\boldsymbol{H}) &= D_{xx} + D_{yy} = \alpha + \beta \\ Det(\boldsymbol{H}) &= D_{xx}D_{yy} - \left(D_{xy}\right)^2 = \alpha\beta \end{aligned} \tag{2.10}$$

и представить r как

$$r = \frac{\alpha}{\beta}\,, \tag{2.11}$$

то условие отбраковки рассматриваемого экстремума:

$$\frac{Tr(\boldsymbol{H})}{Det(\boldsymbol{H})} \geq \frac{(r+1)^2}{r} \qquad (2.12)$$

Оставшиеся уточненные экстремумы считаются особыми точками.

2.4.4 Построение дескрипторов

Дескриптор особой точки обеспечивает её стойкость к различным преобразованиям, например, к вращению и масштабированию. Дескрипторы SIFT зарекомендовали себя на практике и часто используются отдельно от прочих компонентов метода.

Для построения дескриптора необходима операция по вычислению компонент градиента произвольной точки:

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2}, \quad (2.13)$$

$$\theta(x,y) = tg^{-1}\frac{L(x+1,y) - L(x-1,y)}{L(x,y+1) - L(x,y-1)}\ , \qquad (2.14)$$

где *L(x,y)* – значение гауссиана с фиксированным значением σ, m – величина градиента, θ – его направление. Сперва определяется окрестность точки, образующая дескриптор. Она выбирается из гауссиана с масштабом σ`, в полтора раза большим, чем у ключевой точки, и имеет вид окружности радиусом $3\sigma$`. Она делится на 36 равных секторов (по 10^o). Градиент каждой точки внутри окружности суммируется с другими из того же сектора, таким образом строится гистограмма направлений. Общее направление может уточняться с помощью интерполяции лидирующего и соседних с ним и нахождения максимума. Если имеются несколько отстоящих от главного направлений, близких к нему по величине (более 80%), они используются для создания новой ключевой точки, с теми же координатами и другой направленностью. Результат всех этих вычислений – направление ключевой точки. Перед выполнением дальнейших операций всё окружение точки (изображение) ориентируется на это направление. Так достигается инвариантность дескриптора к вращению.

На окрестности ключевой точки (на масштабе, ближайшем к точному) формируется квадратная окрестность, состоящая из блоков пикселей размером 4 x 4. Размер окрестности варьируется, в данном примере принят равным 16 (т.е. область имеет стороны длиной 16 пикселей и вмещает 16 блоков 4 x 4).
На окрестности вычисляется дополнительный гауссиан – свертка с центром в ключевой точке. Затем для каждого блока в окрестности строится гистограмма направлений, по уже рассмотренному принципу, но на 8 направлений и с взвешиванием градиентов по соответствующим значениям новой свертки. Векторы гистограмм нормализуются по общей величине.

Результирующая совокупность шестнадцати гистограмм является дескриптором ключевой точки. Фактически он представляет собой массив 128

(4 x 4 x 8) чисел. Пример дескриптора типа 2 x 2 x 8 представлен на рисунке 2.4.

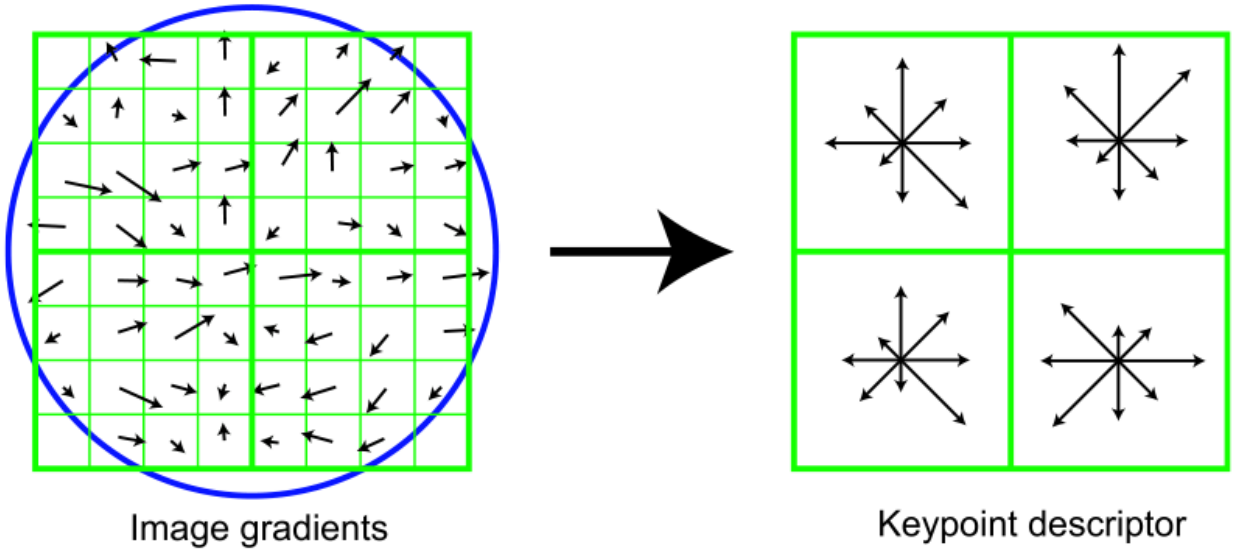

Рисунок 2.4 – Дескриптор особой точки в виде совокупности гистограмм направлений

2.4.5 Соотнесение особых точек

Имеется набор исходных сегментов с описанными особыми точками, необходимо соотнести их между собой оптимальным образом. Решение этой задачи должно быть выражено оператором преобразования координат, для которого необходимо определение восьми параметров (см. 2.2.1). Для вычисления четырех из них (перемещение, вращение, масштабирование) достаточно было бы одной пары соответствующих ключевых точек (по одной на каждом сегменте). Однако, для определения полного преобразования, необходимы четыре пары.

Нахождение соответствий между ключевыми точками может быть реализовано простым перебором с нахождением квадратов ошибок:

$$E_{SSD}(\boldsymbol{x_i}, \boldsymbol{y_j}) = \sum_k \left[\boldsymbol{x_{ik}} - \boldsymbol{y_{jk}}\right]^2 = \sum_k {e_k}^2. \qquad (2.15)$$

Выбираются пары со значением E_{SSD} меньше некоторого порога. Если таких оказывается слишком мало, изображения считаются несовместимыми, и рассматривается следующая пара сегментов.

Если количество ключевых точек велико, полный перебор может оказаться весьма длительным процессом. Применение метода поиска ближайших соседей с вейвлетной индексацией позволяет сравнивать ключевую точку только с теми точками, которые близки к ней по градиентным

характеристикам. Для этого строится трехмерная матрица, в которой каждая ключевая точка регистрируется по трем координатам - c_1, c_2 и c_3:

$$c_1 = \frac{\partial I}{\partial x}; c_2 = \frac{\partial I}{\partial y}; c_3 = \frac{\partial^2 I}{\partial x \partial y}. \quad (2.16)$$

Пространство каждой меры в матрицу разбивается на b частично перекрывающих друг друга интервалов (*bins*). Матрица используется при итерировании по ключевым точкам, для сокращения количества их комбинаций. Минус такого метода заключается в возможности ложного отсеивания.

Высока вероятность нахождения случайных пар ключевых точек, которые приведут к конфликту при вычислении трансформации. Во избежание таких ситуаций используется алгоритм RANSAC. Варианты преобразований рассматриваются как модели, а пары соответствующих ключевых точек – зашумленные исходные данные. Согласно алгоритму, среди пар выбираются случайные комбинации (в данном случае – по 4 пары), и по ним строятся модели. Модель проверяется на корректность путем пропускания через неё остальных данных и проверки совпадения значений. Так как исходные данные (координаты точек) обладают некоторой долей погрешности, для проверки используется диапазон значений отклонения, в пределах которого оно считается удовлетворительным (см. рисунок 2.5). Точный размер диапазона выбирается в результате проб и экспериментов.

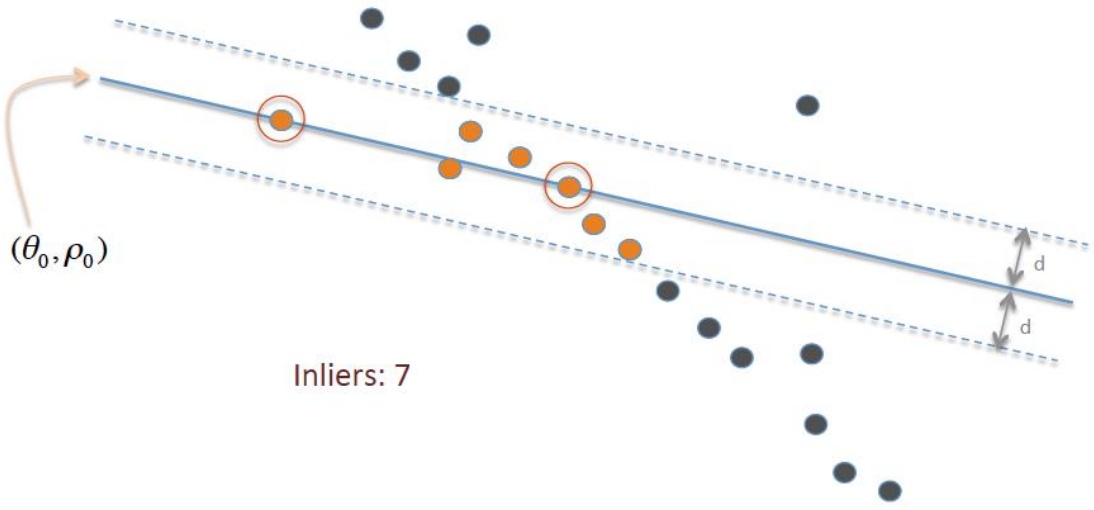

Рисунок 2.5 – Диапазон корректных значений в алгоритме RANSAC

Результат работы RANSAC – достоверный оператор преобразования координат, определяющий положение одного сегмента панорамы относительно другого. Параметры преобразования могут быть уточнены

После перебора всех комбинаций сегментов возникает совокупность относительных преобразований, которые необходимо представить в виде глобальной системы.

2.4.6 Корректировка

При склейке изображений зачастую образуются области перекрытия, не совпадающие на 100%. Существует два пути решения этого конфликта: смешение пикселей разных сегментов или проведение четкой границы между ними.

Простейшее решение при реализации последнего варианта – выбирать один из сегментов по какой-либо признаку и замещать его пикселями пиксели другого. Это создает риск возникновения видимых швов ввиду разности освещений или кривизны проекций. Чтобы шов был незаметен, его следует провести особым образом. Для этого используется алгоритм Graphcut [14]. Перекрываемая область представляется в виде связного графа пикселей (см. рисунок 2.6), в котором вес дуги равен разности между пикселями смежных сегментов, а именно:

$$M(s,t,\boldsymbol{A},\boldsymbol{B}) = \|\boldsymbol{A}(s) - \boldsymbol{B}(s)\| + \|\boldsymbol{A}(t) - \boldsymbol{B}(t)\|, \qquad (2.17)$$

где $\boldsymbol{A}$ и $\boldsymbol{B}$ - сегменты, t и s – смежные пиксели (вершины графа). В таком контексте нахождение шва становится задачей нахождения минимального среза, которая решается с помощью графового алгоритма поиска максимального потока. Две особые вершины обозначают грань области перекрытия. Вес их связей со смежными вершинами равен бесконечности, таким образом, они не участвуют в поиске. После нахождения оптимального среза, граф используется как маска для выбора пикселей из разных сегментов.

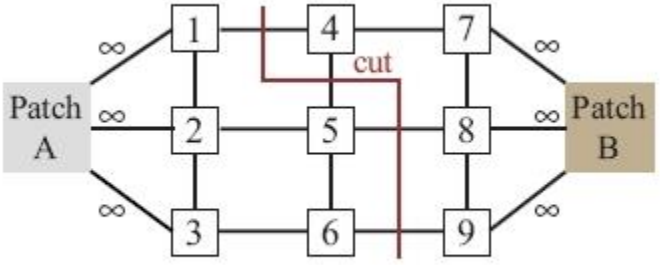

Рисунок 2.6 – Граф для определение оптимального шва на перекрываемой области

Смешение пикселей производится быстро путем замены интенсивностей средним значением конфликтующей пары. Результирующее

изображение будет хранить информацию обоих сегментов, что может быть полезным свойством, но порождает ряд проблем.

Одна из проблем – шум. Малый уровень шума легко ликвидируется с помощью медианного пространственного фильтра. Из упорядоченного набора интенсивностей всех пикселей в окрестности рассматриваемого выбирается среднее значение. Оно применяется как новая интенсивность пикселя. Размер окрестности варьируется в зависимости от требуемой мощности фильтра.

Другая проблема – «призраки». При создании панорамы на основе фотографий, следует учесть, что они, скорее всего, сделаны не одновременно. Если в кадр попадает движущийся объект, он будет присутствовать лишь на некоторых сегментам, что вызовет видимый дефект при их склеивании. Проведение шва зачастую исключает такую проблему, ей подвержено смешение пикселей. Для решения проблемы можно использовать алгоритм Uyttendaele [3]. Он заключается в поиске областей, характеризуемых большой разностью между пикселями двух сегментов. В результате последовательного прохода по области перекрытия определяется конфликтный регион (по наибольшим значениям разностей интенсивности). Он представляется в виде связного графа, в котором вершины – конфликтующие пиксели. В ходе досконального прохода по всем вершинам графа, соответствующие им пиксели панорамы получают смещение интенсивности (т.е. выбираются из одного сегмента). Таким образом можно либо удалить призрачный объект с картинки, либо сделать его более четким – в зависимости от того, пиксели какого сегмента будут выбраны.

3 Проектирование программного средства

3.1 Разработка архитектуры

Архитектура программного средства – его высокоуровневое структурное описание, определяющее роли основных компонентов системы и связи между ними. Она должна обеспечивать эффективное решение по реализации требований к системе, оптимальным образом отражать варианты использования.

3.1.1 Среда работы программного средства

Фундаментальные требования к разрабатываемому программному средству были поставлены в разделе 2.1. Из них следует, что ключевыми операциями программного средства будут манипулирование и обработка графической информации. У системы четко определены входные и выходные данные - изображения, причем первые почти однозначно определяют последние. Эта связь дает возможность сделать систему в высокой степени замкнутой, минимизировать ее внешние зависимости, приблизив к концепции «черного ящика». Данный подход тем более целесообразен, если процесс работы ПС имеет комплексный характер. Так, генерация панорамы состоит из ряда этапов и подэтапов обработки данных различного характера.

Однако, полноценный «черный ящик» в данном случае неприменим, так как пользователь должен иметь возможность управлять и вносить изменения в процесс работы ПС. Необходимо предоставлять ему информацию о процессе наиболее оптимальным образом – графическим (ввиду характера работы системы). В современных операционных системах наиболее распространенным средством ввода и вывода информации являются окна – модульные элементы интерфейса, объединенные общей системой управления. Для разработки приложений, использующих собственные окна, существует множество эффективных программных решений. Это говорит о целесообразности разработки программного средства в виде оконного приложения.

Среди требований присутствуют работы с файлами изображений, что означает взаимодействие с файловой системой действующего компьютера. К счастью, все современные платформы для разработки имеют средства абстрагирования подобных взаимодействий, а также их оптимизации. К последним относится использование системных диалоговых окон для удобной работы пользователя с файлами. Это еще один аргумент в пользу оконного приложения.

3.1.2 Программная архитектура

Процесс генерации панорамы состоит из ряда сложных процедур анализа и преобразования данных различного вида. Это означает наличие высокого риска создания недостаточно гибкой, хрупкой архитектуры, при которой внесение любого изменения в общую структуру может приводить к непредсказуемым последствиям. Поэтому ключевой характеристикой архитектуры должна быть простота, выражаемая в легкости, с которой разработчик сможет ориентироваться в существующей системе и абстрагировать отдельные ее части.

Элементарным приемом для упрощения сложных программных решений является принцип модульности – разделения кода на цельные компоненты. Такой прием позволяет, по возможности, изолировать несвязные фрагменты программы друг от друга, что дает множество преимуществ, среди которых:

- разграничение ответственности;
- возможность параллельной работы нескольких команд разработчиков;
- возможность переиспользования кода;
- простота тестирования.

Разграничение ответственности требует особого способа разбиения кода на модули. Если за некоторый набор функций отвечает четко определенный модуль, работающая над этими функциями команда разработчиков не будет конфликтовать с другими, отвечающими за другие функции. Также ограниченность набора функций модуля означает ограниченный набор способов его использования, и простоту тестирования. Наконец, модульность подразумевает внутреннее разбиение крупных компонент на более мелкие, с отношениями включения. Если схожий код требуется во многих местах, он может быть вынесен в единый модуль, с общей реализацией для всех вариантов использования. Это сокращает общее количество программного кода и избавляет от ошибок при его модификациях.

Модульность упрощает программирование в пределах малого модуля, однако с продвижением вверх по иерархии существенно возрастает уровень ответственности. Современным решением для эффективного абстрагирования и делегирования ответственностей является объектно-ориентированное программирование. Оно основано на представлении программы в виде набора объектов – структур, содержащих информацию и определяющих некоторые операции (методы). Объекты абстрагируются в виде классов, интерфейсов и модулей. Они ссылаются друг на друга и могут составлять иерархии неограниченной сложности. Отношение между двумя классами, когда изменение одного приводит к неизбежному изменению другого, называется зависимостью. Одним из главных достоинств объектно-ориентированного дизайна является то, что он выявляет подобные зависимости и позволяет управлять ими – сокращать или оборачивать вспять. Этой цели служит принцип полиморфизма – использование разных сущностей с помощью одного интерфейса. Использование интерфейсов также означает ограничение взаимодействий с объектом – наличие у него индивидуального пространства

для хранения и использования кода (принцип инкапсуляции). Область влияния этого скрытого кода четко определена, что сильно упрощает отладку программы.

Помимо всего вышеназванного, очевидным достоинством объектно-ориентированной архитектуры является ее способность отражать предметную область. Логично организовывать функции и информацию в классы сообразно сущностям, упомянутым в вариантах использования – будь то изображение, панорама или сам пользователь. Оперирование программными абстракциями, имеющими реальные аналоги, облегчает понимание кода и делает разработку отчасти интуитивной. Не все классы в приложении описывают сущности из предметной области. По мере снижения уровня абстракции появляются классы, имеющие специфические роли уже в терминах конкретной программы, платформы разработки или операционной системы.

Существуют устоявшиеся модели распределения и взаимодействия компонентов в приложении. Использовать их целесообразно при разработке больших программ, при проектировке которых велик риск создания неэффективной архитектуры. Одна из таких моделей – EBC (Entity-Boundary-Control, [X]). Она определяет три категории элементов:

- сущности (entity);
- граничные элементы (boundary);
- управляющие элементы (control).

Сущности – элементы, предназначенные для хранения информации, чаще всего относящиеся к предметной области. Граничные элементы являются интерфейсом всего приложения, они взаимодействуют с пользователем (или другим актером). Элементы управления определяют внутреннюю работу системы по генерации выходных данных на основании входных, полученных из граничных классов. Фактически, элементы управления являются непосредственными реализаторами вариантом использования ПС.

Для проектируемого приложения в разделе 2.3 разработана модель предметной области. Она будет использована для конкретизации категорий элементов. Так, ключевое понятие – «Панорама» - представляет собой наиболее очевидный кандидат на роль сущности «Сегменты», из которых она состоит, являются контейнерами графической информации – это также сущность в приложении. «Генератор» - активный функциональный компонент, следовательно, это должен быть элемент управления. «Настройки», которыми он пользуется – еще один контейнер. «Проекция панорамы» - панорама в виде простого изображения. Платформа разработки предоставит стандартный класс, который заменит эту сущность в приложении. Последнее понятие из модели - «Пользователь». Функции не предполагают хранения какой-либо информации о пользователе. Это понятие отображает инициативную сторону – агента, внешнего по отношению к программному средству. В подобном элементе нет необходимости.

Кандидаты на роль граничных классов были описаны в разделе 3.1.1 – это окно и диалог для управления файлами.

Функции по загрузке/сохранению файлов, запуску генератора и редактирования изображений, будут реализованы элементами управления. Функция панорамы «представляется в виде проекции» реализуется элементом управления «ПредставительПанорамы».

Все разработанные элементы, их роли и зависимости представлены на диаграмме на рисунке 3.1.

3.2 Проектирование пользовательского интерфейса

Пользовательский интерфейс (UI, User Interface) – это способ взаимодействия человека и машины, включающий совокупность действий, совершаемых пользователем, и результатов этих действий. Критериями качества интерфейса является его эффективность и удобство пользования. Структурно интерфейс является совокупностью средств и методов. Средства в данном контексте – элементы, предназначенные для ввода информации в устройство или ее вывода для отображения пользователю. Методы – динамическая составляющая интерфейса, набор правил по использованию имеющихся средств.

Интерфейс разрабатываемого программного средства должен быть двунаправленным. Программе для выполнения своей задачи необходимы данные, определяемые пользователем – сегменты панорамы. Пользователь для возможности предоставления этих данных должен получить и понять запрос от программного средства. Когда программа завершила процесс генерации панорамы, тот предоставляется пользователю, вместе с возможностью редактирования и дальнейшего использования в каких-либо целях.

Ввиду сильной зависимости выходных данных (панорамы) от входных (сегментов), возможность вмешательства в работу может быть излишней в тех случаях, когда система самостоятельно вырабатывает оптимальный результат. Поэтому следует минимизировать необходимость любого вмешательства и предоставить пользователю предварительный результат. В случае неудовлетворительной оценки он должен иметь возможность изменить результат непосредственно или определить этап, на котором дефект может быть устранен, внести изменения в ход этого этапа и оценить их влияние. В случае, если система может самостоятельно оценить качество генерируемой панорамы как неприемлемое, от пользователя сразу потребуется выполнение некоторых корректирующих действий.

Требования к пользователю должны быть ему понятны и очевидны. Следовательно, необходим механизм доставки сообщений, который будет помогать ему в навигации по интерфейсу и при этом не являться помехой для эффективного использования этого интерфейса. К примеру, на окне приложения может быть выделена постоянная область, предназначенная для отображения сообщений пользователю. Так как содержание области будет

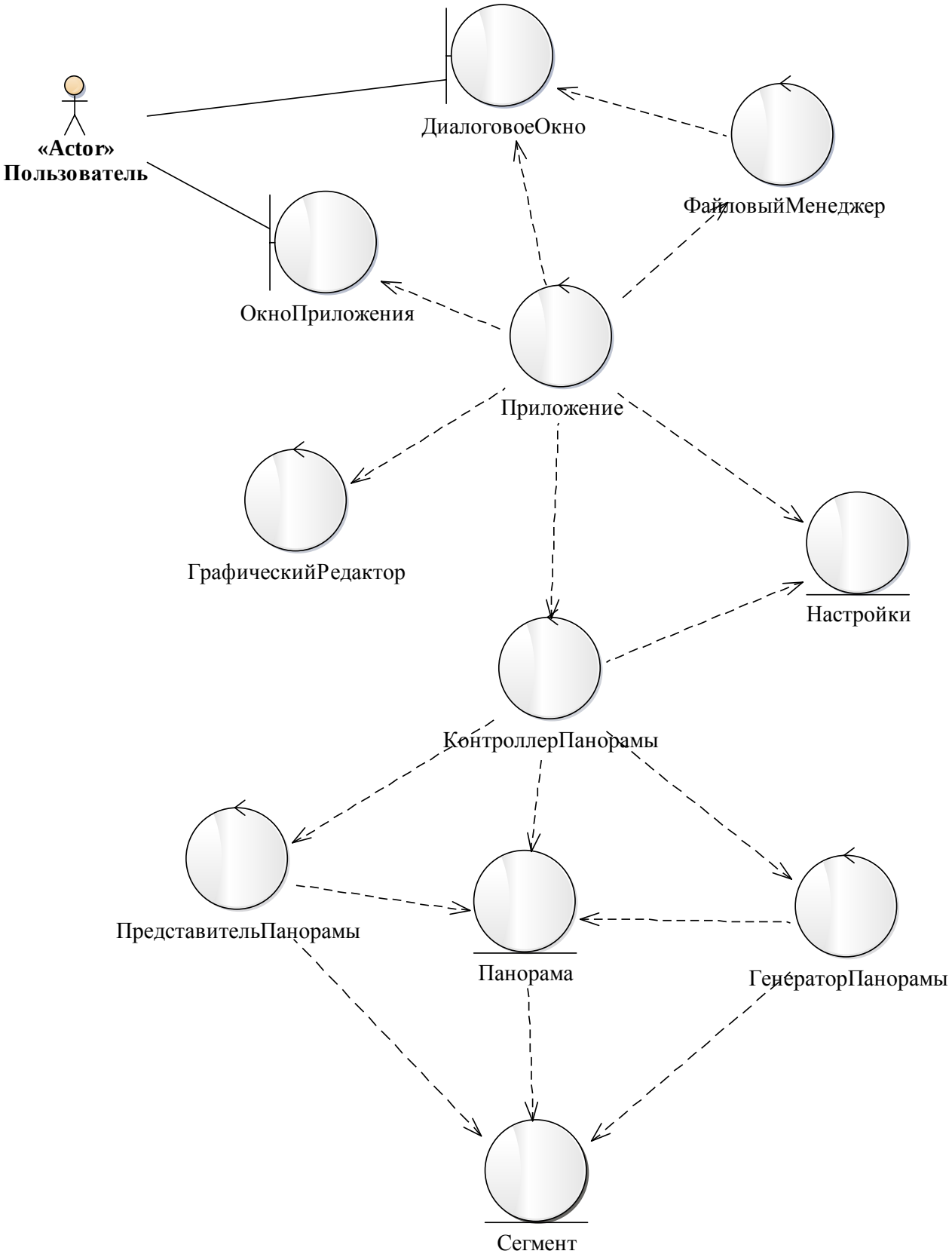

Рисунок 3.1 – Аналитическая модель программного средства

меняться по ходу работы приложения, и поэтому не всегда отвечать на для пользователя актуальные вопросы, следует дать ему возможность задания вопроса. Частой реализацией этого является кнопка «Справка» или ссылки на документацию в контекстном меню окна.

Предоставляемые пользователю инструменты должны быть целесообразны ожидаемым от него действиям и удобны в использовании. Имеет смысл взять за основу средства, получившие широкое распространение среди других популярных приложений, которые будут наиболее близки и понятны пользователю.

3.3 Разработка основных алгоритмов программы

3.3.1 Алгоритм работы программы

Программа начинает свое выполнение после запуска пользователем. Она загружает текущие настройки, которые могут быть заданы пользователем, и использует их при последующей за этим инициализацией компонентов системы. На данном этапе необходимым компонентом является окно приложения, которое предоставит пользователю возможность работы с программой. Для функционирования по назначению программе требуются входные данные, поэтому после запуска окно отобразит интерфейс для передачи этих данных (изображений). Интерфейс состоит из стандартных диалоговых окон для выбора файлов на компьютере.

Получив входные данные, программа способна самостоятельно выработать выходные. Для сопоставления друг с другом изображения проходят через стадию анализа, на которой извлекаются их статистические атрибуты. Пройдя через несколько стадий обработки, эти атрибуты далее служат основой для синтеза выходного результата в ходе процедуры генерации панорамы. Результат генерации – первоначальный вариант панорамы - подается на главное окно и предоставляется пользователю для оценки.

Если результат удовлетворителен по мнению пользователя, он может экспортировать его в виде файла (сохранить изображение). В этом случае программа достигла своей цели и может быть завершена (или использована для новой задачи – на усмотрение пользователя).

Если же результат оказался неудовлетворительным, пользователь может приступить к исправлению выделенных им дефектов. Дефекты могут порождаться на различных стадиях работы программы, поэтому применение изменений, для наибольшего удобства использования, должно быть возможно с использованием инструментов различных уровней. Простейшие средства редактирования изображения являются первой из доступных возможностей. Пользователь уведомляется об этом и применяет инструменты. Если они оказываются недостаточны для исправления дефекта, совершается переход к более раннему этапу процесса, в данном случае – к объединению сегментов в

панораму. Графический интерфейс должен удобно отображать принятые программой решения касательно отношения между сегментами и их свойствами и позволять редактировать эти данные. далее пользователь может перезапустить генератор панорамы и оценить измененный результат. Если и эти средства оказываются недостаточными, остается возможность изменения наборы входных данных. Отображение отношений между сегментами может помочь пользователю выявить отдельные изображения, являющиеся причиной дефекта, и исключить их из исходного набора. Другим вариантом является добавление новых сегментов, что является обычным способом повышения качества панорамы. Обо всех этих возможностях пользователь будет уведомлен посредством сообщений.

Алгоритм работы программы приведен на рисунке 3.2.

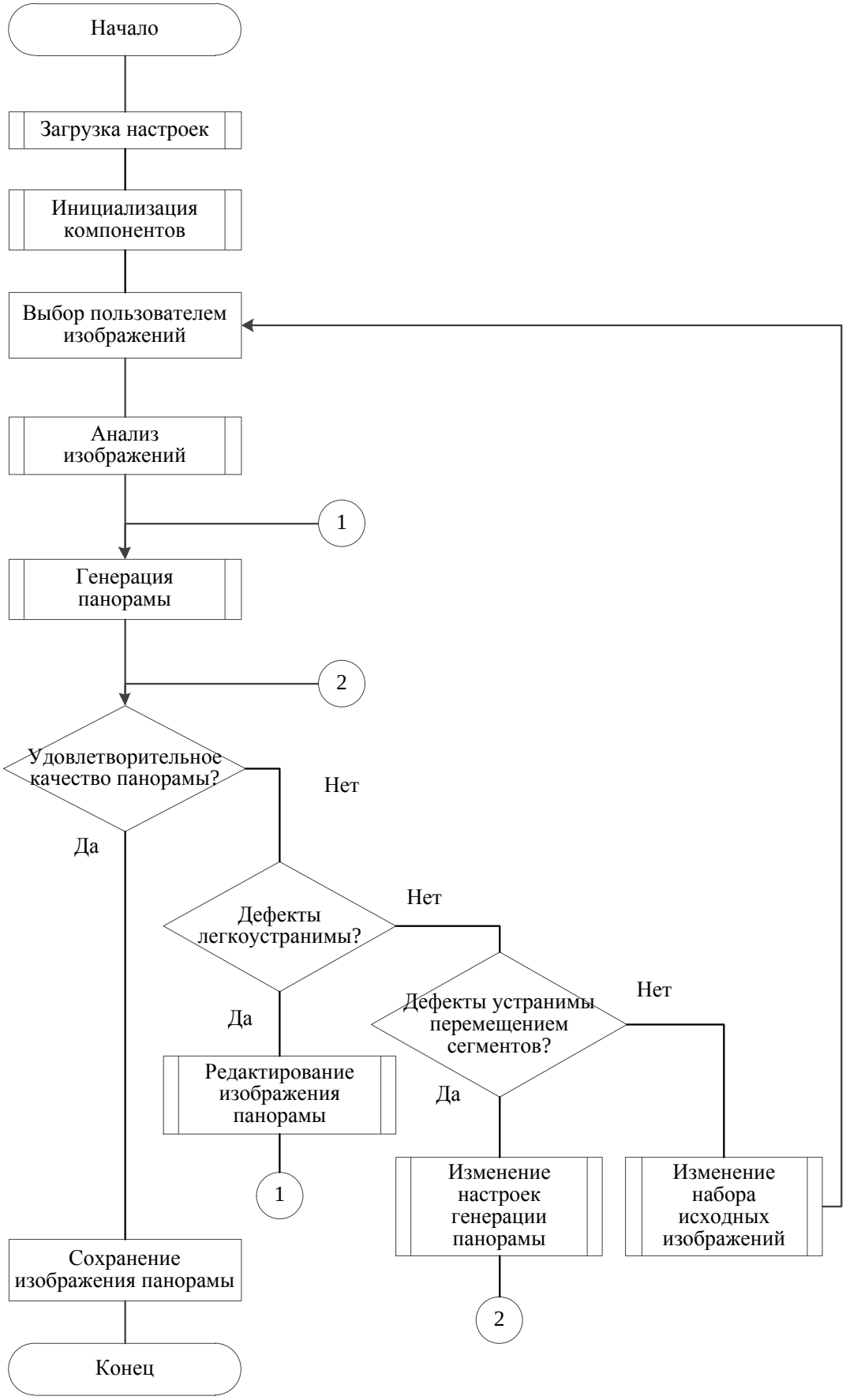

Рисунок 3.2 – Схема работы программы

3.3.2 Алгоритм генерации панорамы

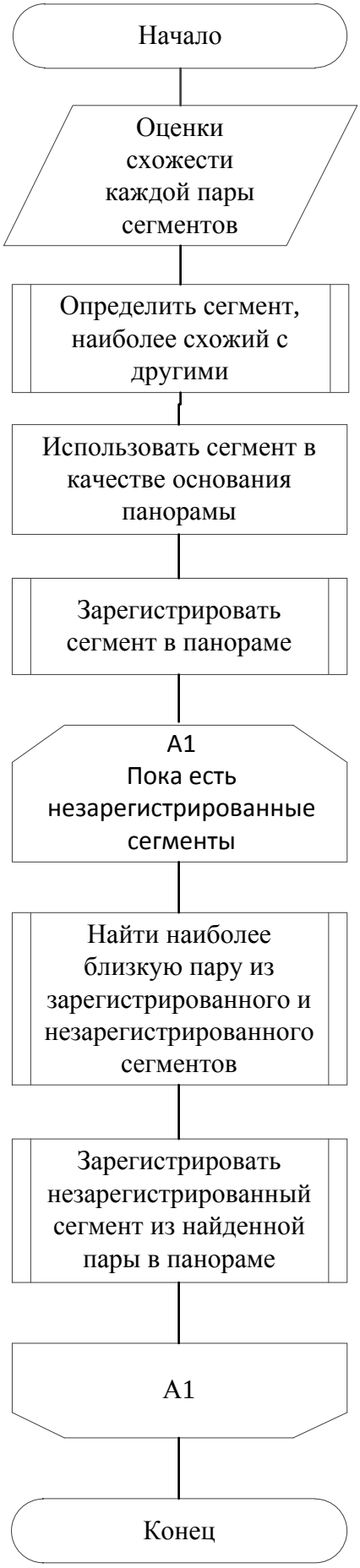
Начало
Оценки схожести каждой пары сегментов
Определить сегмент, наиболее схожий с другими
Использовать сегмент в качестве основания панорамы
Зарегистрировать сегмент в панораме
A1
Пока есть незарегистрированные сегменты
Найти наиболее близкую пару из зарегистрированного и незарегистрированного сегментов
Зарегистрировать незарегистрированный сегмент из найденной пары в панораме
A1
Конец

Рисунок 3.3 – Схема алгоритма генерации панорамы

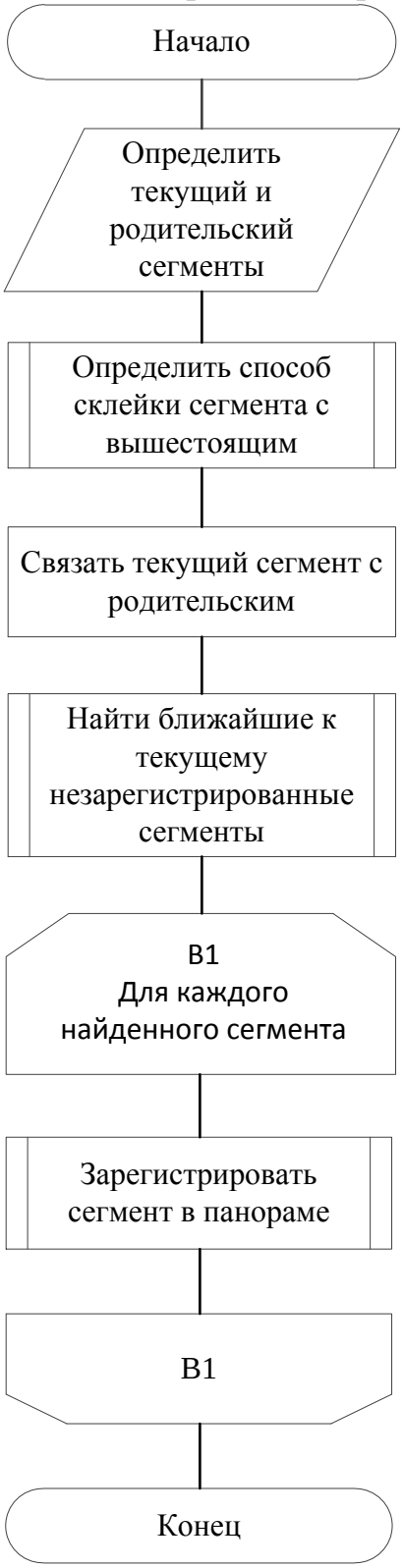

Рисунок 3.4 – Алгоритм регистрации сегмента в панораме

4 КОНСТРУИРОВАНИЕ ПС

5 ТЕСТИРОВАНИЕ ПС

6 ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ ЦЕЛЕСООБРАЗНОСТИ РАЗРАБОТКИ И ИСПОЛЬЗОВАНИЯ ПРОГРАММНОГО СРЕДСТВА ДЛЯ СИНТЕЗА ПАНОРАМНЫХ ИЗОБРАЖЕНИЙ

6.1 Функции, назначение и потенциальные пользователи ПС

Задача программного средства - генерация цифровых панорамных изображений путем автоматизированной обработки набора некоторых исходных графических изображений. ПС предназначено для неограниченного круга пользователей и свободной продажи на рынке информационных технологий. Ранее существующий набор программных решений весьма ограничен, качественно и количественно, особенно с учетом высокой сложности реализуемых процедур обработки. Подобные процедуры лишь в малой мере поддаются ручной реализации (с помощью программ - графических редакторов) и в общем случае требуют значительных трудозатрат. Автоматизация позволяет существенно снизить данный параметр, оставляя за пользователем лишь задание параметров процедуры или простые манипуляции в специальном графическом интерфейсе.

К основным функциональным требованиям относятся следующие:

- загрузка набора исходных изображений;
- генерация панорамы на основе набора исходных изображений;
- графическое представление панорамы с возможностью управления и изменения параметров обзора;
- возможность редактирования панорамы;
- сохранение и загрузка панорамы из файла на диске.

Особенностью программного средства является гибкость настройки и управления процессом генерации панорамы.

Экономическая целесообразность инвестиций в разработку и использование программного средства выявляется на основе расчета и оценки следующих показателей:

- чистый дисконтированный доход;
- рентабельность инвестиций;
- срок окупаемости инвестиций.

6.2 Расчет затрат на разработку и реализацию ПС

Расчет затрат на разработку ПС производится с использованием следующих статей расходов:

- затраты на основную заработную плату разработчиков;
- затраты на дополнительную заработную плату разработчиков;
- отчисления на социальные нужны;
- прочие расходы.

Расчет величины основной заработной платы разработчиков осуществляется по формуле:

$$З_о = \sum_i^n T_{чi} t_i, \quad (6.1)$$

где n – количество исполнителей, занятых в разработке ПС;

$T_{чi}$ – часовая заработная плата i-го исполнителя, р.;

t_i – трудоемкость работ, производимых i-м исполнителем, ч.

Для инженера-программиста квалификационный разряд – 13 (тарифный коэффициент 3.04). При месячной ставке первого разряда 2 млн. р., месячная ставка каждого исполнителя равна:

$$T_{чi} = 2 * 3{,}04 = 6{,}08 \text{ млн. р.}$$

Время на разработку проекта – 4 месяца, количество разработчиков - 2. Таким образом,

$$З_о = 2 * 6{,}08 * 4 = 48{,}64 \text{ млн. р.}$$

Затраты на дополнительную заработную плату включают выплаты, предусмотренные действующим трудовым законодательством, и определяется по формуле:

$$З_д = \frac{З_о Н_д}{100}, \quad (6.2)$$

где $З_о$ – затраты на основную заработную плату с учетом премии, р.;

$Н_д$ – норматив дополнительной заработной платы (20%).

$$З_д = \frac{48{,}64*20}{100} = 9.728 \text{ млн. р.}$$

Отчисления на социальные нужны (фонд социальной защиты и обязательное страхование) определяются в соответствии с действующими законодательными актами по формуле:

$$З_{соц} = \frac{(З_о+З_д)Н_{соц}}{100}, \quad (6.3)$$

где $Н_{соц}$ – норматив отчислений на социальные нужды (34,6%).

$$З_{соц} = \frac{(48{,}64+9{,}728)*34{,}6}{100} = \frac{58{,}368*34{,}6}{100} = 20{,}196 \text{ млн. р.}$$

Расчет прочих затрат осуществляется в виде расчета процентов от затрат на основную заработную плату команды разработчиков с учетом премии по формуле:

$$З_{пз} = \frac{З_о Н_{пз}}{100}, \quad (6.4)$$

где $Н_{пз}$ – норматив прочих затрат (принят равным 110%).

$$З_{пз} = \frac{48{,}64 * 110}{100} = 53.504 \text{ млн. р.}$$

Результаты расчетов приведены в таблице 6.1.

Таблица 6.1 – Затраты на разработку ПС

Статья затрат	Сумма, млн. р.
Основная заработная плата разработчиков	48,640
Дополнительная заработная плата разработчиков	9,728
Отчисление на социальные нужды	20,196
Прочие расходы	53,504
Общая сумма затрат на разработку	132,068

Следующие за разработкой процессы реализации и сопровождения также требуют определенных расходов. Затраты на реализацию определяются как:

$$З_{реал} = \frac{З_{разр} Н_{реал}}{100}, \quad (6.5)$$

где $З_{разр}$ – ранее вычисленная сумма затрат на разработку;

$Н_{реал}$ – норматив затрат на реализацию (5%).

$$З_{реал} = \frac{132{,}068 * 5}{100} = 6{,}604 \text{ млн. р.}$$

Затраты на сопровождение ПС определяются по формуле:

$$З_{сопр} = \frac{З_{разр} Н_{сопр}}{100}, \quad (6.6)$$

где $Н_{сопр}$ – норматив затрат на сопровождение (10%).

$$З_{сопр} = \frac{132{,}068 * 10}{100} = 13{,}207 \text{ млн. р.}$$

Итого сумма затрат на разработку, реализацию и сопровождение:

$$З = З_{разр} + З_{реал} + З_{сопр} = 151{,}878 \text{ млн. р.} \quad (6.7)$$

6.3 Оценка эффекта от использования ПС

Экономический эффект для организации-разработчика ПС заключается в получении прибыли от его реализации на рынке информационных технологий. Прибыль, в свою очередь, напрямую зависит от объема продаж, цены реализации и затрат на разработку ПС.

Изучение рынка и статистических данных о продажах программных продуктов схожего функционала позволило рассчитывать на 300-500 покупок лицензий в течение года реализации (при расчетах - 400) при цене не более 1 млн. р. (на основании цен аналогов - PTGui и Photoshop).

Прибыль предприятия от реализации единицы (копии):

$$П_{ед} = Ц - \frac{Ц * НДС}{100\% - НДС} - \frac{З}{N}, \quad (6.8)$$

где Ц – рыночная цена единицы продукта, р.;

НДС – доля налога на добавленную стоимость (20%);

N – количество реализованных копий за год, шт.

$$П_{ед} = 1000 - \frac{200}{0{,}8} - \frac{151878}{400} = 370{,}305 \text{ тыс. р.}$$

Суммарная годовая прибыль:

$$П = П_{ед} * N = 148{,}122 \text{ млн. р.} \quad (6.9)$$

Рентабельность затрат на разработку вычисляется по формуле:

$$Р = \frac{П}{З} * 100\% = \frac{148{,}122}{151{,}878} * 100\% = 97{,}53\%. \quad (6.10)$$

Рассчитанное значение рентабельности существенно превышает средние процентную ставки по банковским депозитным вкладам, что свидетельствует об экономической эффективности проекта.

Чистая прибыль, учитывающая действующий налог на прибыль (18%):

$$П_{ч} = П(1 - 0{,}18) = 121{,}460 \text{ млн. р.} \quad (6.11)$$

Значение чистой прибыли является численным выражением годового экономического эффекта инвестирования в проект.

6.4 Расчет показателей эффективности инвестиций в разработку ПС

Рассчитанный годовой экономический эффект ниже требуемого объема инвестиций, следовательно, они окупятся полностью лишь в течение нескольких лет. Расчет показателей эффективности инвестиций требует учета динамики прибылей и расходов на протяжении этого времени, для чего вводятся коэффициенты дисконтирования:

$$\alpha_t = \frac{1}{(1+E_n)^t}, \quad (6.12)$$

где t – порядковый номер года реализации продукта;

E_n- норма дисконта, не меньшая средней ставки по банковским депозитам на момент осуществления расчетов (40%).

Чистый дисконтированный доход рассчитывается по формуле:

$$\text{ЧДД} = \sum_t^n (\text{П}_t \alpha_t - \text{З}_t \alpha_t), \quad (6.13)$$

где n – порядковый номер года реализации продукта;

П_t – чистая прибыль в t-м году;

З_t – сумма затрат в t-м году.

Результаты расчета дисконтированных значений прибылей и затрат приведены в таблице 6.2.

По результатам расчета, чистый дисконтированный доход за расчетный период составил 45.258 млн. р. Это положительная сумма, что говорит о целесообразности инвестирования.

Таблица 6.2 – Дисконтированные значения прибылей и затрат

Показатель	Разработка	Год реализации			
		1	2	3	4
Чистая прибыль, млн. р.	-	121,460	121,460	121,460	121,460
Дисконтированная прибыль, млн. р.	-	86,757	61,969	44,264	31,617
Затраты, млн. р.	151,878	-	-	-	-
Дисконтированная сумма затрат, млн. р.	151,878	-	-	-	-
Чистый дисконтированный доход, млн. р.	-151,878	-65,121	-3,152	41,112	31,617
Коэффициент дисконтирования	1,000	0,714	0,510	0,364	0,260

Рентабельность инвестиций рассчитывается как

$$P_{\text{и}} = \frac{\sum_t^n \text{П}_t \alpha_t}{\sum_t^n \text{З}_t \alpha_t} * 100\% = \frac{224{,}607}{151{,}878} * 100\% = 147{,}87\%. \quad (6.14)$$

Значение рентабельности превышает 100%, что свидетельствует об эффективности инвестиций.

Срок окупаемости инвестиций – период времени, необходимый для того, чтобы полученная прибыль покрыла всю сумму инвестиций. Иначе говоря, это срок, за который чистый дисконтированный доход принимает положительное значение. Согласно динамике значений дохода (таблица 6.2), этот момент наступает в начале третьего года реализации.

Таким образом, разработка и применение программного средства является эффективным.

7 ЭРГОНОМИЧЕСКИЕ ПОКАЗАТЕЛИ ИСПОЛЬЗОВАНИЯ ПРОГРАММНОГО СРЕДСТВА

ЗАКЛЮЧЕНИЕ

В ходе прохождение преддипломной практики была изучена предметная область темы проекта, существующие проблемы и примеры их решений. Поставлена цель и задача проекта. Разработана базовая функциональная модель программного средства и на ее основе сформулированы требования. Также была определена математическая модель приложения, играющая ключевую роль в его работе.

Таким образом, заложен фундамент для более детальной разработки программного средства, в частности – для проектирования программной архитектуры.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Wikipedia [Электронный ресурс]. – Электронные данные. – Режим доступа: http://en.wikipedia.org/wiki/Computer_vision
[2] Wikipedia [Электронный ресурс]. – Электронные данные. – Режим доступа: http://en.wikipedia.org/wiki/Image_stitching
[3] Szeliski, Richard. Image Alignment and Stitching / R. Szeliski - MSR-TR-2004-92
[4] Cambridge in Colour [Электронный ресурс]. – Электронные данные. – Режим доступа: http://www.cambridgeincolour.com/tutorials/image-projections.htm
[5] Tuytelaars. T. Local Invariant Feature Detectors: A survey / Tinne Tuytelaars, Krystian Mikolajczyk - 2006
[6] Habrahabr [Электронный ресурс]. – Электронные данные. – Режим доступа: http://habrahabr.ru/post/106302/
[7] Habrahabr [Электронный ресурс]. – Электронные данные. – Режим доступа: http://habrahabr.ru/post/244541/
[8] Brown, Mathew. Multi-Image Matching using Multi-Scale oriented Patches / M. Brown, R. Szeliski, S. Winder – Microsoft Research, 2004 MSR-TR-2004-8
[9] Szeliski, Richard. Creating Full View Panoramic Image Mosaics and Environment Maps / Heung-Yeung Shum – MSR-TR-97-8
[10] Zhu, Martin. Efficient Video Panoramic Image Stitching Based on an Improved Selection of Harris Corners and a Multiple-Constraint Corner Matching / M. Zhu, W. Wang, J. Huang – 2013 PLoS ONE 8(12): e81182. doi: 10.1371 / journal.pone.0081182
[11] Belgacem, Mohamed Ben – Panoramic Image Stitching : report – University of Geneva, 2014
[12] Hays, James. Computational Photography : tutorial – Brown University courses, 2010
[13] AI Shack [Электронный ресурс]. – Электронные данные. – Режим доступа: http://www.aishack.in/tutorials/sift-scale-invariant-feature-transform-introduction/
[14] Kwatra. V. Graphcut Textures: Image and Video Synthesis Using Graph Cuts / Vivek Kwatra, Arno Schodl, Irfan Essa, Greg Turk, Aaron Bobick - GVU Center / College of Computing Georgia Institute of Technology