Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Российский государственный университет им.А.Н.Косыгина
(Технологии. Дизайн. Искусство)»

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА МАГИСТРА

на тему: *Приложение для распознавания образов*

Направление подготовки: *09.04.01 Информатика и вычислительная техника*

Программа: *Информационные технологии в логистике*

Выполнил(а)
студент(ка) группы МАГ-В-116 2 курса очной формы обучения

_____ Васильев Константин Евгеньевич
подпись

Научный руководитель _____ к.т.н. Муртазина А.Р.
подпись

Руководитель магистерской программы _____ д.т.н., проф. Севостьянов П.А.
подпись

«Допущен(а) к защите»

Заведующий кафедрой
АСОИиУ _____ к.т.н., доц. Монахов В.И.
подпись

Москва 2018

Содержание

ВВЕДЕНИЕ....3
ОСНОВНАЯ ЧАСТЬ....5
1 Обзор предметной области и актуальность проблемы компьютерного зрения. 5
1.1 Системы компьютерного зрения и их применение....5
1.2 Требования, предъявляемые к формированию признаков изображения 11
2 Методы распознавания изображений....13
2.1 Обзор и анализ существующих методов распознавания объектов на изображениях....13
2.1.1 Метод сравнения с эталоном....13
2.1.2 Статистические методы....14
2.1.3 Структурные и синтаксические методы....14
2.1.4 Точечные методы....16
2.1.5 Нейронные сети....18
2.1.6 Основные виды признаков изображения....20
2.1.7 Методы извлечения точечных признаков....21
2.1.7.1 Метод SIFT....21
2.1.7.2 Метод SURF....23
2.2 Способы представления изображения....34
2.2.1 Общие положения....34
2.2.2 Низкоуровневое представление....35
2.2.3 Контурное представление....38
2.2.4 Структурное представление....39
2.2.5 Признаковое представление....39
3 Проектирование программного обеспечения компьютерного зрения....41
3.1 Выбор программных средств реализации поставленной задачи..41
3.2 Выбор и реализация алгоритмов, позволяющих решить поставленную задачу....48
3.2.1 Общий алгоритм для решения поставленной задачи....48
3.2.2 Выбор конкретных алгоритмов для решения соответствующих этапов основного алгоритма....54
3.3 Проектирование структуры проекта программного обеспечения 56
3.3.1 Разработка базы данных для эталонных изображений....56
3.3.2 Предобработка эталонных изображений....60
3.3.3 Программная архитектура системы компьютерного зрения.…61
3.4 Описание работы программы....66
ЗАКЛЮЧЕНИЕ....71
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ И ЛИТЕРАТУРЫ....75

ПРИЛОЖЕНИЕ ФРАГМЕНТ ЛИСТИНГА ПРОГРАММНЫХ МОДУЛЕЙ.....77

ВВЕДЕНИЕ

Актуальность работы связана с тем, что системы распознавания образов и компьютерного зрения все чаще и чаще используются при автоматическом подсчете количества объектом и других применениях.

Выбор темы обоснован актуальностью применения систем технического зрения в современной промышленности.

Теоретическая значимость работы состоит в изучении методов распознавания изображений, практическая – в применении разработанного программного обеспечения для подсчета деталей.

Системы технического зрения (СТЗ) сейчас становятся одним из основных средств развития АСУ движением в условиях, когда объём априорных данных не достаточен и для реализации задач управления нужен анализ внешней обстановки в онлайн режиме. СТЗ используются в новейших космических, наземных, подводных и надводных мобильных объектах. Хорошая дальность действия и широкий спектр пространственного и цветового разрешения сегодняшних линейных и матричных приемников оптического излучения СТЗ служат отличными источниками данных при автоматическом выполнении задач навигации, наведения или распознавания.

Минимизация времени конвертации зрительных сигналов при большой скорости движения мобильных объектов очень сложна в реализации, и является единственным препятствием для повсеместного применения СТЗ. Сложности также связаны с синтезом алгоритмов необходимых преобразований. Нет универсального алгоритма решения задач зрительного слежения, распознавания, навигации и наведения при стандартном движении аппарата или объекта зрительного слежения. Для любой задачи СТЗ и в конкретной ситуации фотометрических и траекторных условий необходим лишь определенный алгоритм,

причём даже незначительное изменение указанной зрительной сцены часто требует смены применяемого алгоритма исследования фотометрического сигнала. Это ведет к необходимости поддержания алгоритмической полноты и структурной устойчивости проведения задач технического зрения на различных фотометрических ситуациях и траекториях передвижения.

Сложная ситуация алгоритмического обеспечения СТЗ складывается и с конструктивными просчетами их технической реализации — нехватке динамических диапазонов свето- и цветопередачи, дискретностью фотоприёмников, астигматизмом канала наблюдения, погрешностями калибровочных характеристик и привязок к времени потока видеоданных. Поэтому зачастую полезность алгоритма гибнет под ударами искажений и шумов. Борьба с помехами в видеоданных становится основной алгоритмической задачей в процессе применения зрительной обратной связи в центр управления. Такой подход требует серьезного развития математических методов и алгоритмов зрительных преобразований в рамках реализации конкретных задач управления передвижными объектами.

Объектом исследования выпускной квалификационной работы является методы и способы распознавания изображений.

Предметом исследования является программная реализация методов распознавания изображений.

Целью работы является повышение эффективности работы за счет применения проекта программного обеспечения для подсчета количества деталей в системах логистики.

Основными задачами работы являются:

1. Анализ структуры изображения в цифровом виде;
2. Анализ методов и алгоритмов распознавания объектов;
3. Разработка алгоритмов работы программы;
4. Разработка структуры программного обеспечения;
5. Тестирование программного обеспечения.

ОСНОВНАЯ ЧАСТЬ

1 Обзор предметной области и актуальность проблемы компьютерного зрения

1.1 Системы компьютерного зрения и их применение

В настоящее время системы технического зрения (СТЗ) являются одним из основных средств с помощью которых осуществляется развитие автоматических систем для управления движением в условиях, когда объём априорной информации может быть слишком мал и для решения задач управления необходимо произвести анализ внешней обстановки в режиме реального времени. СТЗ могут применяться в современных космических, наземных, авиационных, подводных и надводных мобильных объектах. Благодаря достаточно высокому дальнодействию и уровням пространственного и цветового разрешения современных матричных и линейных приемников для оптического излучения СТЗ служат незаменимыми источниками информации при автоматизации решения задач по распознавания образов, навигации или наведения[18].

Сегодня применение цифровых видео датчиков с повышенным разрешением помогло значительно приблизить информативность созданных зрительных устройств к отдельным характеристикам человеческого глаза. А сами аппаратные возможности, доступные благодаря развитию электроники и вычислительной техники, так шагнули вперед, что вычислительные мощности персональных ЭВМ уже почти приблизились к параметрам «вычислительных мощностей», применяемых для обработки изображений в мозгу.

Одним из ведущих направлений в ИТ сегодня является создание, анализ, а также применение методов для решения задач таких как: анализ, нахождение и оценивание изображений. Результаты различных фундаментальных и прикладных исследований часто могут использоваться в технической диагностике, неразрушающем контроле, удаленном зондировании, отслеживании

экологического состояния, диагностике и прогнозировании в медицине, геологическом поиске, планировании, прогнозировании в химии, автоматизации научной деятельности. Данная тематика привлекает большое количество исследователей, но большинство работ в этой области, в основном, имеют эмпирический характер.

Укажем, что эта проблематика несет немаловажное значение для исследования информационной природы изображений. Цели и итоги работы очень важны для создания и анализа АИС обработки и анализа изображений обширного назначения, не ограниченных узкой предметной областью, работающих с изображениями, полученными в обширных диапазонах условий наблюдения. Разработка новейшего поколения подобных систем часто связана с желанием и необходимостью автоматизации выбора варианта изменения изображения в рамках его семантики и информационной природы. Одним из центральных направлений по созданию подобных систем становится автоматизация реализации их отдельных функций, примером может быть - определение методов описания и преобразования изображений в рамках их информационной природы [18].

Исследования, проводимые для изучения алгоритмов компьютерного зрения, начали появляться в середине 60-70-х годов XX века. В этой «науке» человек заставляет машину научиться находить некоторые особенности мира без участия человека, то есть, чтобы машина сама понимала то, что она видит.

Когда в мире стали устанавливать камеры (на дорогах, улицах, домах и т.д.), стали создавать ПО и устройства, позволяющие на фото выделять разные объекты: лица, номера и другие данные, которые необходимы человеку. Это позволило совершенствовать технологии, которые определяют объекты на фотографиях, проводят сравнение с шаблонами и ориентируются на местности относительно указанных параметров.

При создании бортовых систем видео слежения создается целый класс задач, связанных с сложностями обнаружения, указания и ведения объектов,

находящихся в поле зрения датчика изображений. Примерами таких объектов становятся различные летательные аппараты, люди, суда, транспорт и т.п. При этом радиус охвата видеодатчика подвижен, а априорные данные о характеристиках объектов зачастую скудны или включают в себя лишь примерные размеры объектов и характер их следования.

Между тем видеодатчик, установленный на корпусе мобильного носителя, следует в пространстве вместе с носителем, а также, его ориентацию в пространстве можно поменять при помощи устройства позиционирования. Часто датчик изображения испытывает на себе воздействие вибраций различной природы. Под влиянием перечисленных факторов, составляемые им изображения подвержены искажениям. Они затрудняют, а чаще всего делают невозможным вообще нахождение решения многих задач: нахождение подвижных объектов, градация и параметризация сопровождаемых объектов, находящихся близко друг к другу, анализ траекторий сопутствующих объектов. При этом искажения, связанные с вибрациями корпуса носителя, затрудняют работу оператора системы, поэтому иногда возникает задача стабилизации отслеживаемого датчика.

Природные явления: дождь, ветер, снег, туман могут вносить ощущаемый процент зашумления и колебания в изначально статичную сцену, создавать сильные помехи, перекрывая частично поля зрения камеры. Указанные нюансы часто делают алгоритмы, замечательно работающие в одних условиях, отнюдь ненужными для других.

В итоге следует отметить, что, хоть сейчас есть большое количество разработанных алгоритмов, сама область еще плохо исследована. В процессе создания системы учитывается большинство доступных методов, необходимых для нахождения движения и отметки объектов.

Самой важной особенностью задачи обработки данных, получаемых с датчиков для определения объектов, находящихся на изображении, становится

отсутствие для изучаемых реальных ситуаций или объектов каких-то адекватных математических моделей, благодаря которым можно было бы провести расчеты и составлять количественные или качественные выводы.

Проблема распознавания данных многогранна и проблематична, поскольку изображения имеют высокую информационную емкость и изначальную неопределенность (содержат любые объекты и составляются с помощью различных типов сенсоров), а также высокую изменчивость (корректировка ракурса или освещения приводит к сдвигам значений сразу во всех элементах изображения). Также объекты, присутствующие на изображениях, состоят обычно в сложных иерархических отношениях типа «частное-целое» и «частное-общее». Именно проблематичность решения совокупной задачи распознавания изображений заставляет создавать множество частных методов.

Из-за огромного количества уже имеющихся методов возникает вопрос об их похожести, различиях и результативности применения в процессе применения к той или иной задаче. Основным моментом в процессе подбора метода распознавания становится представление начального изображения. Под представлением зачастую определяют формальную систему, включающую алгоритмы для преобразования в явном виде описаний объектов рассматриваемого класса.

Главными типами представлений становятся:

- Признаковые;
- Низкоуровневые;
- Структурные;
- Контурные.

Вид промежуточного представления данных не станет полноценной характеристикой метода распознавания данных. Зачастую важен также подход к выявлению критерия качества и алгоритм оптимизации данного критерия.

Представления на уровне пикселей становятся опорными для любых приложений по описанию изображений в современных ИС. И поскольку это представление становится общим для разных задач компьютерной обработки изображений, можно сказать, что "изображения - это группы пикселей". Но нужно уметь разделять содержание (само изображение) от его формы (представления).

Представления на уровне пикселей включают в себя все доступные данные о наблюдаемой сцене, но в форме, совершенно не подходящей для автоматического анализа. Зачастую подобные представления почти не используются для распознавания или сравнения изображений. Это и несет необходимость использования других представлений изображений для поиска в них релевантных данных.

Главной проблемой, возникающей при обработке изображений, становится повышение качества изображения, к примеру, увеличение различимости отдельных фрагментов. К факторам, уменьшающим качество изображений, относят:

- Присутствующие яркостные помехи в виде шума;
- Повышенная или пониженная освещенность объектов съемки;
- Недостаточная резкость при создании изображения;
- Небольшие размеры деталей, которые нужно распознать.

Огромное количество методов повышения качества изображений разделяется на две большие категории: методы обработки в пространстве и методы обработки по частоте. Понятие «пространственная область» принадлежит плоскости растрового изображения, формирующегося из отдельных пикселей, и включает в себя методы, базирующиеся на прямом изменении пикселей изображения. Методы обработки по частоте базируются на корректировке сигнала, создаваемого путем использования в изображении преобразования Фурье или подобных преобразований. Часть методов удобно располагается в

пространственной области, а для расположения остальных больше подходит частотная область. Далее будут рассмотрены только пространственные методы увеличения качества изображений, что зачастую связано с проведением большого объема вычислений при реализации преобразований Фурье для растровых изображений размером 800x600 пикселей и выше.

Яркость на фото может значительно снижать различимость отдельных фрагментов изображения, но важно понимать, что для нынешних систем создания изображения в цифровом виде характерно почти полное отсутствие каких-либо существенных яркостных помех, но полученные с помощью их изображения все-таки иногда имеют недостаточную различимость части фрагментов, что может объясняться:

- Малой контрастностью изображения, поскольку известно, что человеческий глаз не может различить границу между фрагментами, яркость которых отлична лишь на единицу градации;
- Малой резкостью изображения, что ведет к замыливанию границ разных фрагментов.

Главными источники шума и повышенной яркости на цифровом изображении становятся: методика его получения, оцифровка, а также его передача. Действие сенсоров, реагирующих на электромагнитное излучение в определенном спектральном диапазоне, может зависеть как от внешних условий в момент получения изображения, так и от чувствительности самих сенсоров. К примеру, в момент получения изображения фотокамерой с ПЗС матрицей, главными факторами, определяющими величину шума, становятся температура самих сенсоров и уровень освещенности.

При передаче изображения подвержены помехам, возникающим в каналах связи. К примеру, при передаче изображения с применением беспроводной сети, оно может подвергнуться изменению в момент разряда молнии или других возмущений в атмосфере.

1.2 Требования, предъявляемые к формированию признаков изображения

Наиболее современные производственные, технологические а так же офисные системы во время прохождения процесса его функционирования используют информацию о маркировке объектов. Возможно использование информации о маркировке различных грузов, таких как: вагоны, контейнеры, а также номерные знаки автомобилей и не только дает возможность наиболее рациональным образом организовывать процесс технологической обработки информации или проводить учет и контролирование изделий, материалов, транспортных средств. Основным в процессе использования маркировок (нанесения текстовоцифровых меток) лежит технология автоматизированного распознавания структурированных символов. Потребность в данной технологии была вызвана необходимостью создания методов, моделей и систем распознавания структурированных символов.

В настоящее время подобные технологии обычно реализуются с помощью трех методов: структурный, признаковый и шаблонный. Каждый из приведенных методов предназначен для использования в определенных условиях и именно в них данные методы наиболее эффективны.

Однако каждый из рассмотренных методов имеет свои недостатки. Самым существенным из них является низкая устойчивость к таким явлениям как: перемены масштаба, смещения, повороты, а также смена ракурса и другим возможным искажениям.

Эти недостатки могут наиболее ярко проявляться при масштабной эксплуатации программнотехнологических систем, использующих в своей основе данные методы. У большинства подобных систем распознавания структурированных символов характеристики точности осуществляют наиболее резкое падение и уже недотягивают до приемлемого уровня если их подвергают искажениям аффинными и проекционными преобразованиями. Вместе с тем

технологические условия получения информации о маркировке не позволяют полностью устранить эти искажения [1].

Требование эффективности процесса распознавания образов является основным требованием к используемой системе признаков данного изображения. Данное требование имеет ряд аспектов:

- к признакам предъявляются требования вычислительного характера, которые заключаются в том, чтобы существует алгоритм расчёта признаков, и данный алгоритм является вычислительно эффективным;
- из требований эффективности выводится необходимость так же удовлетворить требования по устойчивости или инвариантности образа по отношению к ряду искажений объекта.

Так же для ряда практических задач требование инвариантности оказывается достаточно общим и подразумевает все или некоторые из следующих требований:

- инвариантность к шумовым и динамическим искажениям;
- инвариантность к яркостным искажениям (изменению яркости и контраста);
- инвариантность к изменению местоположения объекта;
- инвариантность к изменению масштаба объекта;
- инвариантность к изменению ориентации объекта (к повороту объекта в плоскости изображения);
- инвариантность к произвольным аффинным преобразованиям.

Из-за проведения предварительного преобразования исходного изображения достигается инвариантность к данным искажениям. Первая группа искажений это: шумовые а так же динамические изменения изображения. Такие искажения устраняется путем использования методов и алгоритмов восстановления. Ко второй группе относятся яркостные искажения. Они могут быть эффективно устраняться путем приведения изображения к «нормализованному» виду.

2 Методы распознавания изображений

2.1 Обзор и анализ существующих методов распознавания объектов на изображениях

2.1.1 Метод сравнения с эталоном

Под распознаванием обычно понимают отнесение объекта к конкретному классу. Примерами характерных объектов служат вертолёты, самолёты, автотранспорт, ракеты, корабли. Повышенный интерес представляет проблема разработки систем распознавания объектов, функционирующих в онлайн режиме. Проектирование таких систем может быть полезно для выполнения таких задач, как картография, поисковые работы и спасение в чрезвычайных ситуациях, отслеживание наземного и воздушного движения, минимизация аварийных ситуаций на транспорте [9].

Нынешняя теория распознавания хорошо развита и сложена как область знания, оперирует различными методологиями [20]. Исследование различных источников дает возможность выделить 4 главных подхода к распознаванию.

Детерминированный подход говорит о том, что в любой точке пространства признаков, применяемых для распознавания, с какой-то априорной вероятностью появляется реализации только одного класса объектов. Тогда достаточно найти уравнения границ, делящие в пространстве образов области, относящиеся к тому или иному классу. В случае, если найти эти границы невозможно, разделение проводится путем подсчета расстояния в пространстве признаков между классами изображений и распознаваемым образом. Важно отметить, что распознавание с применением такого подхода имеет наглядное геометрическое представление и для его реализации не нужно дополнительных данных, кроме начального изображения каждого класса. Вычислительные затраты для указанного метода относительно небольшие, что делает его использование оправданным в бортовых системах анализа данных.

2.1.2 Статистические методы

Отличие статистического подхода от детерминированного заключается в том, что в одной точке пространства образов с ненулевой априорной вероятностью могут происходить реализации, относящиеся к разным классам[10]. Это обычно связано с неизбежностью возникновения случайных ошибок, присутствующих в детерминированных признаках, или использованием признаков, которые принципиально являются случайными величинами. Байесовский подход выражается из статистической природы наблюдений. Начальными данными выступает предположение о наличии вероятностной меры на пространстве образов, которая может быть или известна, или оценена. Цель заключается в создании такого классификатора, который позволит точно найти самый вероятный класс для пробного образа. В таком случае задача заключается в определении "наиболее вероятного" класса.

Итак, Байесовский подход к статистическим задачам берет за основу предложение о существовании отдельного распределения вероятностей для конкретного параметра. Минусом такого метода становится необходимость постулирования как наличия априорного распределения для скрытого параметра, так и знание его формы [13].

2.1.3 Структурные и синтаксические методы

Структурно-лингвистический (синтаксический) подход базируется на описании геометрической структуры класса при помощи конкретного формального языка, алфавит которого включает в себя производственные элементы и структуры, а также композиции и операции над ними, определяемые грамматикой языка. Подобный подход предполагает, что характер и структура связей элементов всей совокупности объектов подчиняется конкретным, заранее известным закономерностям. Понятно, что такая методика практически не

подойдет для работы в бортовых ИС реального времени, поскольку ведет к сложной и запутанной программной реализации [14].

Теория распознавания образов изучает множество задач, которые применяются для адаптации вероятностных моделей, а метод минимизации поиска, похож на схему интерпретационного дерева. Главные составляющие подобного метода это допущения такие как:

- Последовательность случайных переменных, условно независимая от всех других, кроме предшествующей переменной;
- Любая случайная переменная определяется измерениями, вероятность которых зависит от состояния [15].

Подобным элементам сопоставима наиболее результативная формальная модель, которую называют скрытой марковской моделью (СММ). Данная модель с успехом применяется в системах созданных для распознавания речи. В которых скрытые состояния отслеживают речевую систему, а под наблюдениями подразумеваются различные акустические измерения. Зачастую скрытые марковские модели соотносятся с каждым словом. После чего модель начинает складываться воедино с помощью модели языка, которая задает вероятность появления следующего слова в рамках распознанного текущего слова. В итоге реализуется другая (зачастую большая) СММ. Предложение, интерпретированное набором акустических измерений, выстраивается с помощью методики логического вывода, применяемой к модели языка.

Деятельность алгоритма по методу динамического программирования всегда начинается с последнего столбца решетки. Заранее известен логарифм правдоподобия пути с одним состоянием, завершающимся в каждой вершине, т.к. данное значение становится весовым коэффициентом этой вершины. Далее рассматривается путь с двумя состояниями, начинающийся в предпоследнем столбце решетки. Затем получается наилучший путь, выходящий из вершины в этом столбце. Переходим к вершине. Мы знаем весовой коэффициент любого

ребра, покидающего вершину, и весовой коэффициент вершины в окончании ребра. Поэтому можно определить сегмент пути с самым большим значением суммы, и такое ребро станет лучшим ребром, выходящим из указанной вершины. Для любой вершины ее весовой коэффициент суммируется со значением лучшего сегмента пути, покидающего вершину весовой коэффициент ребра плюс коэффициент итогового узла. Подобная будет оптимальным значением, которое получается при достижении этой вершины (пусть это будет значение вершины).

Т.к. мы знаем наилучшее значение, получаемое при достижении каждой вершины во втором от конца столбце, можно вычислить наилучшее значение, реализуемое при достижении любой вершины в третьем от конца столбце. Далее в любой вершине третьего от конца столбца анализируются ребра, входящие в вершину, с известным значением. Возьмем ребро с максимальным значением величины (вес ребра + значение вершины), суммируем это значение с весовым коэффициентом начальной вершины в третьем от конца столбце и в итоге выведем значение начальной вершины. Такой процесс можно проводить до получения значения всех вершин первого столбца, при этом максимальное значение будет максимально правдоподобным. В итоге получается путь с самым наибольшим значением функции правдоподобия. При подсчете значения вершины убираются все ребра, покидающие вершину, кроме наилучшего ребра. По факту достижения первого столбца далее нужно идти по пути от узла с наилучшим значением.

2.1.4 Точечные методы

Гистограмма (в фотографии) — это график распределения полутонов изображения, в котором по горизонтальной оси представлен уровень тона канала или яркость, а по вертикали — число пикселей с данным значением. Гистограммы обычно являются весьма важными данными при обработке изображений. Существуют различные методы для построения и сравнения цветовых

гистограмм, которые отличаются между собой изначальной цветовой схемой (RGB, CMY, HSV, grayscale и т. д.), размерностью гистограммы и определением расстояния между гистограммами [5].

Достоинством метода является простота и производительность. Для того что бы вычислить гистограмму потребуется всего лишь один проход по пикселям изображения без каких-либо дополнительных вычислений.

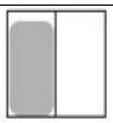
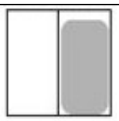
Гистограмма – это массив целых чисел, размер которого равен количеству возможных значений цветового канала, а каждая ячейка массива содержит количество точек изображения с данным значением канала. Если цветовой катал не один, как например каналы палитры RGB, то массивов будет три[10].

Для цветовой схемы RGB уровень каждого канала может изменяться в пределах от 0 до 255. Тогда как для построения гистограммы, для каждого из каналов, самым удобным вариантом будет применение матрицы размером 3x255. В некоторых отличных случаях наиболее удобно преобразовывать исходное изображение в оттенки серого (GrayScale) и использовать только один массив размером 256.

В таблице 1 рассмотрен простейший случай. Рассматриваются одномерные гистограммы только с двумя столбцами. Модель гистограммы имеет значение 1 в левом столбце и 0 в правом. Последние три строки показывают сравнение гистограмм и значения сгенерированные различными метриками.

Таблица 1 - Поведение метрик

Мо-	Метрика

дель				
	Корреля-ция	Хи-квадрат	Пересе-чение	Расстоя-ние Бхатачария
	1	0	1	0
	0,7	0,67	0,5	0,55
	-1	2	0	1

Результаты экспериментов показывают, что при использовании метода гистограмм необходимо определить диапазон значений, в котором будет находиться пороговое значение сходства. Диапазон должен быть определён для каждого метода сравнения свой – отличный от других. Это обусловлено особенностями методов сравнения.

Если гистограммы сравниваются с помощью расстояния Бхатачария, то значение порога должно находиться в пределах от 80 до 100 процентов[11].

Ограничения, накладываемые на порог сходства, необходимы для того, чтобы отсеять некорректные результаты. В случае, когда пороговое значение задаётся ниже, чем минимальный допустимый порог, то результаты сравнения заполняются неточными данными. Это обусловлено особенностью метода и способами сравнения.

2.1.5 Нейронные сети

Искусственная нейронная сеть является некоторой моделью естественной нейронной сети. Каждый элемент искусственной сети (нейрон) является прототипом, имитирующим свойства и работу биологического нейрона.

Качество работы искусственной нейронной сети во многом зависит от весов синаптических связей и от типа применяемых активационных функций.

У метода обратного распространения имеется ряд недостатков:

- паралич сети - это явление происходящее, когда в процессе обучения сети значения весов становятся большими величинами, т.е. у нейронов увеличиваются выходные значения, но из за того что производная активационной функции в случаях с большими значениями мала, то производная в пунктах (7), (10), (13) принимает малое значение. В результате чего процесс обучения сети впадает в состояние «ступора». Что бы решить данную проблему предотвращения паралича следует уменьшить размер для шага η, что в свою очередь, влияет на время обучения ,в следствии чего, оно увеличивается. Так же можно попытаться решить проблему с помощью ввода эвристических правил для обнаружения паралича;
- локальные минимумы – это явление когда в процессе обучения найдены неудовлетворительные значения весов, что происходит из-за того, что обратное распространение является разновидностью градиентного спуска, что означает, осуществление им спуска вниз по поверхности ошибки, непрерывно подстраивая веса по направлению в сторону минимума. И если в случае когда поверхность ошибки примет сложную форму, тогда обучение застревает в локальном экстремуме. Во избежание подобной ситуации могут использоваться, достаточно медленные, статистические методы обучения;
- размер шага – в [6] упоминается коррекции весов, которая должна осуществляться бесконечно малым шагом, однако на практике, обычно требуется, что бы шаг должен быть конечным и достаточно большим. Но в тоже самое время размер шага не должен быть большим, что приведет к параличу сети, а также к постоянной неустойчивости;
- временная неустойчивость – это когда в памяти нейронной сети не остается предыдущих шагов обучения, при попытках ее дальнейшего обучения. Из чего следует, что процесс обучения следует обязательно организовывать так,

чтобы обучение сети осуществлялось на всем множестве для обучения и не было пропусков уже изученного. Так же в рекомендациях в случае потребности в коррекции весов, перед этим на вход сети подавать все обучающее множество. Но может так случиться, входные образы не всегда могут быть повторены и тогда данный подход будет невозможно применить, из-за постоянных осцилляций.

2.1.6 Основные виды признаков изображения

Результат работы детектора это множество из особых точек, для которых требуется построить математическое описание. В настоящем разделе делается обзор некоторых существующих дескрипторов [13].

Дескриптор принимает на вход в качестве данных изображение а так же набор из особых точек, которые должны быть выделены на заданном входном изображении. В качестве выхода для дескриптора становится множество из векторов признаков для исходного набора из особых точек. Так же следует отметить, что некоторые дескрипторы осуществляют решение одновременно двух задач таких как – поиск особых точек, а также построение описателей для найденных особых точек [19].

Построение признаков (описателей) будет осуществляться на основании полученной информации об интенсивности, о цвете, а также о текстуре особой точки. Так же особые точки могут представляться как углами, ребрами или даже контуром объекта, поэтому, обычно, вычисления выполняются только для некой окрестности. В идеальном варианте хорошие признаки должны обладать такими свойствами как [16]:

Повторяемость. Если данные изображениях с одного и того же объекта или сцены, которые, в свою очередь, могут быть сделаны с различных точек зрения и при разной освещенности, наибольшее количество признаков должно быть продетектировано.

Локальность. Для снижения вероятности перекрытий все признаки должны быть максимально локальными.

Репрезентативность. Численность признаков должна быть достаточна, для того чтобы наиболее разумное число признаков детектировалось даже в случае небольшого изображения объекта.

Точность. Признаки должны быть наиболее точно продетектированными по отношению к масштабу и форме объекта.

Эффективность. Для приложений реального времени критично, если процедуры вычисления признаков будут требовать значительных вычислительных затрат. Поэтому требуется оптимизировать сеть в пользу их уменьшения.

2.1.7 Методы извлечения точечных признаков

2.1.7.1 Метод SIFT

Для формирования дескриптора SIFT (Scale Invariant Feature Transform) [4, 16] изначально вычисляют значения магнитуды и ориентации градиента в каждом пикселе, который принадлежит окрестности особой точки размером $16x16$ пикселей. При этом магнитуды градиентов учитываются с весами, пропорционально значению функции плотности нормального распределения с математическим ожиданием в рассматриваемой особой точке и стандартным отклонением, равным половине ширины окрестности (веса Гауссова распределения используются для того, чтобы уменьшить влияние на итоговый дескриптор градиентов, вычисленных в пикселях, находящихся дальше от особой точки).

В каждом квадрате размером $4x4$ пикселя вычисляется гистограмма ориентированных градиентов путем добавления взвешенного значения магнитуды градиента к одному из 8 бинов гистограммы. Чтобы уменьшить различные "граничные" эффекты, связанные с отнесением похожих градиентов к разным квадратам (что может возникнуть вследствие небольшого сдвига расположения особой точки) используют билинейную интерполяцию: значение магнитуды каждого градиента добавляется не только в гистограмму, соответствующую квадрату, к которому данный пиксель относится, но и к гистограммам,

соответствующим соседним квадратам. При этом значение магнитуды добавляется с весом, пропорциональным расстоянию от пикселя, в котором вычислен данный градиент, до центра соответствующего квадрата. Все вычисленные гистограммы объединяются в один вектор, размером, равным 64(число бинов) 128(число квадратов)[19].

Полученный дескриптор следует преобразовать, что уменьшит возможные эффекты при изменении освещенности. Изменение контраста изображения (значение интенсивности каждого пикселя умножается на некоторую константу) приводит к такому же изменению в значениях магнитуд градиентов. Поэтому очевидно, что данный эффект может быть нивелирован путем нормализации дескриптора таким образом, чтобы его длина стала равна единице. Изменения яркости изображения (к значению интенсивности каждого пикселя прибавляется некоторая константа) на значения магнитуд градиентов не влияют. Таким образом, SIFT-дескриптор инвариантен по отношению к аффинным изменениям освещенности. Тем не менее возможно возникновение нелинейных изменений в освещенности вследствие, например, различной ориентации источника света по отношению к поверхностям трехмерного объекта. Данные эффекты могут вызвать большое изменение в отношении магнитуд некоторых градиентов (при этом оказывают незначительное влияние на ориентацию вектора градиента). Для того что бы этого избежать, используют отсечение по некоторому порогу (результаты экспериментов показывают, что наиболее оптимальным является значение 0.2), которое применяется к компонентам нормализованного дескриптора. После применения порога дескриптор вновь нормализуется. Таким образом, значение больших магнитуд градиентов уменьшается и значение распределения ориентаций данных градиентов в окрестности особой точки увеличивается.

2.1.7.2 Метод SURF

Отличие вышеописанных методов от метода Surf заключается в следующем: метод использует точечные особенности изображения, а не все имеющиеся точки (метод гистограмм и усреднённых цветов).

Алгоритм SURF решает такие задачи как: поиск особых точек изображения и создание их дескрипторов, инвариантных к масштабу и вращению. Это означает, что описание ключевой точки не поменяется, даже при изменении размера образца или его повороте. Кроме того, сам поиск ключевых точек так же должен обладать инвариантностью. Так, чтобы повернутый объект сцены имел тот же набор ключевых точек, что и образец.

Метод осуществляет поиск особых точек с помощью матрицы Гессе. Детерминант матрицы Гессе (т.н. гессиан) достигает экстремума в точках максимального изменения градиента яркости. Он хорошо детектирует пятна, углы и края линий.

Гессиан инвариантен относительно вращения. Но так он не инвариантен к масштабу. Из-за чего SURF для нахождения гессианов использует разномасштабные фильтры.

Для каждой ключевой точки подсчитывается направление максимального изменения яркости (градиент) и масштаб, взятый из масштабного коэффициента матрицы Гессе.

Градиент в точке вычисляют при помощи фильтров Хаара.

После нахождения ключевых точек, SURF формирует их дескрипторы. Дескриптор - это набор из 64 (или 128) чисел для каждой ключевой точки. Эти числа отображают флуктуации градиента вокруг ключевой точки (определение флуктуации будет приведено далее). Поскольку ключевая точка представляет собой максимум гессиана, то это гарантирует, что в окрестности точки должны быть участки с разными градиентами. Таким образом, обеспечивается дисперсия (различие) дескрипторов для разных ключевых точек.

Флуктуации градиента окрестностей ключевой точки считаются относительно направления градиента вокруг точки в целом (по всей окрестности ключевой точки). Таким образом, достигается инвариантность дескриптора относительно вращения. Размер же области, на которой считается дескриптор, определяется масштабом матрицы Гессе, что обеспечивает инвариантность относительно масштаба.

Флуктуации градиента можно рассчитать при помощи фильтра Хаара.

Во многих задачах обработки изображений требуется рассчитать яркость прямоугольного участка изображения. Как пример, фильтры Хаара, SURF и многие другие алгоритмы. Суммарную яркость произвольного прямоугольника позволяет быстро рассчитывать интегральное представление, причем время расчета не зависит от площади прямоугольника.

Интегральное представление изображения - это матрица, размерность которой совпадает с размерностью исходного изображения. Элементы матрицы рассчитываются по следующей формуле:

$$II(x,y)=\sum_{i=0}^{i\le x}\sum_{j=0}^{j\le y} I(i$$

где $I(i,j)$ — яркость пикселя исходного изображения.

Каждый элемент матрицы $II(i,j)$ представляет собой сумму пикселей в прямоугольнике от $(0,0)$ до (x,y). Расчёт матрицы занимает линейное время, пропорциональное числу пикселей в изображении.

Расчёт матрицы можно производить по рекуррентной формуле:

$$II(x,y)=I(x,y)-II(x-1,y-1)+II(x-1,y)+II(x,y-1).$$

У интегрального представления имеется интересная особенность. При помощи интегральной матрицы можно достаточно быстро вычислять сумму яркости пикселей произвольного прямоугольника и произвольной площади.

Пример

Пусть ABCD — интересующий нас прямоугольник:

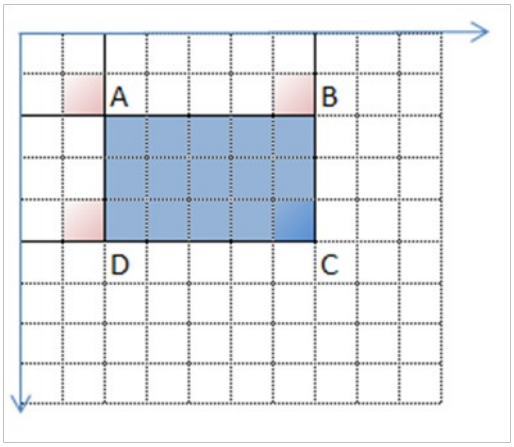


Рисунок 1 Вычисление суммарной яркости прямоугольной области изображения

Исходя из (рисунок 1) можно понять, что сумму внутри прямоугольника возможно выразить через суммы и разности смежных прямоугольников по следующей формуле:

$$II(ABCD) = II(A) + II(C) - II(B) - II(D).$$

Все компоненты нужные для вычисления хранятся в интегральной матрице, вычисление занимает четыре обращения к массиву и три арифметических действия.

Обнаружение особых точек в SURF основывается на вычислении детерминанта матрицы Гессе (гессиана).

Матрица Гессе для двумерной функции и ее детерминант определяется следующим образом:

$$H(f(x,y))=\begin{pmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial y^2} \end{pmatrix},$$

$$\det(H)=\frac{\partial^2 f}{\partial x^2}\frac{\partial^2 f}{\partial y^2}-\left(\frac{\partial^2 f}{\partial x \partial y}\right)^2.$$

Значение гессиана используют при поиске локального минимума или максимума яркости изображения. В данных точках значение гессиана достигает экстремума.

Теоретически, вычисление матрицы Гессе сводится к нахождению Лапласиана Гауссиан. По сути, элементы матрицы Гессе вычисляются как свертка (сумма произведений) пикселей изображения на фильтры (Рисунок 2).

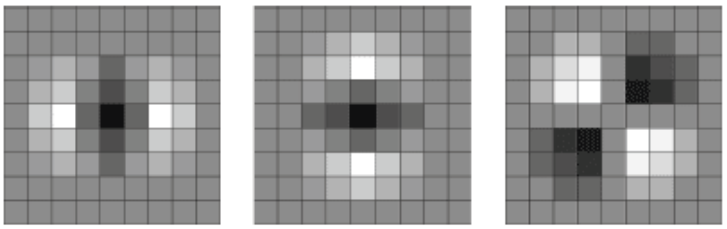


Рисунок 2 Дискретизация матрицы Гессе

На рисунке 3 изображены дискретизированные фильтры которые используются для нахождения четырех элементов матрицы Гессе (поскольку матрица Гессе симметрична то четвертый – совпадает с третьим). Фильтры имеют

пространственный масштаб 9x9 пикселей. Темные участки соответствуют отрицательным значениям фильтра, светлые – положительным.

Однако SURF не использует лапласиан гауссианы в том виде, который изображен на рисунке. Во-первых, по утверждению авторов, дискретизированный лапласиан гауссианы имеет довольно большой разброс значения детерминанта, при вращении образца. А во-вторых, и это главное, фильтр для лапласиана гауссианы имеет непрерывный характер. Почти все пиксели фильтра имеют разные величины яркости. А это не позволяет эффективно использовать такой мощный механизм расчёта, как интегральную матрицу изображения.

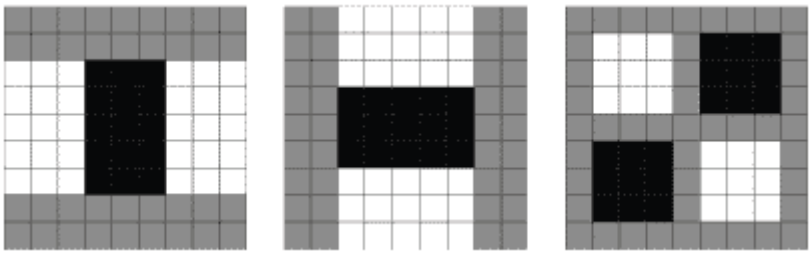


Рисунок 3 Фильтры Fast-Hessian

На рисунке 3 изображены фильтры, используемые для нахождения матрицы Гессе в SURF. Белые области соответствуют значению +1, черные -2 (на третьем фильтре -1), серые – нулевые. Пространственный масштаб – 9x9 пикселов.

Этот фильтр более устойчив к вращению, и его можно эффективно вычислить с помощью интегральной матрицы.

Таким образом, в SURF, гессиан вычисляется так:

$$\det(H) = D_{xx} D_{yy} - (K \cdot D_{xy})^2,$$

где D_{xx}, D_{yy}, D_{xy} – свертки по фильтрам, изображенным на рисунке 3;

K - коэффициент, корректирует приближенный характер вычислений;

$K = 0.9$.

Для нахождения особых точек, SURF пробегается по пикселям изображения и ищет максимум гессиана. Способ нахождения локального максимума гессиана будет рассмотрен позднее. В методе задается пороговое значение гессиана. Если вычисленное значение для пикселя выше порога – пиксель рассматривается как кандидат на ключевую точку.

Исходя из перечисленного, октава содержит не один фильтр, а четыре фильтра, которые хорошо покрывают характерный масштаб октавы (Рисунок 4).

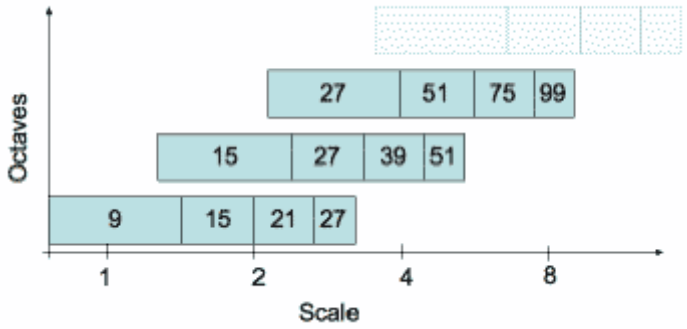


Рисунок 4 Октавы метода SURF

На рисунке 4 показаны первые три октавы метода SURF. Цифры в прямоугольниках показывают размер фильтра Fast-Hessian. Логарифмическая шкала снизу – показывает масштабы, покрываемые октавами.

Октавы значительно перекрываются друг другом. Это увеличивает надежность нахождения локальных максимумов. Почему в октаве именно четыре фильтра станет ясно из следующей главы.

Для нахождения локального максимума гессиана, используется так называемый метод соседних точек 3x3x3. Его смысл понятен из рисунка 5.

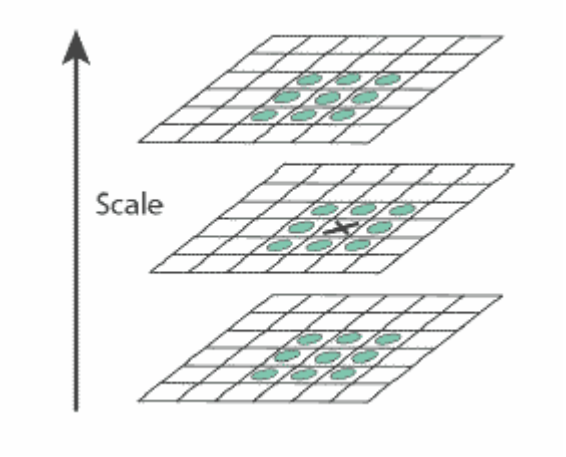


Рисунок 5 Поиск локального максимума

Пиксель, помеченный крестиком, считается локальным максимумом, если его гессиан больше чем у любого его соседа в его масштабе, а также больше любого из соседей масштабом меньше и масштабом больше (всего 26 соседей).

Итак, после нахождения максимального гессиана методом соседних точек 3x3x3, были найдены пиксели, в котором этот максимум достигается. Однако, поскольку, октава перебирает не все точки изображения, то истинный максимум может не совпадать с найденным пикселем, а лежать где-то рядом, в соседних пикселях.

Для вычисления градиента, используется фильтр Хаара. Размер фильтра берется равным $4 \cdot s$, где s – масштаб особой точки.



Рисунок 6 фильтры Хаара

Фильтры Хаара создают точечное значение для перепада яркости по оси x и y соответственно. И поскольку фильтры Хаара обладают прямоугольной формой и их значения легко можно рассчитать при помощи интегральной матрицы. Для того чтобы произвести расчет одного фильтра произвольного размера требуется всего лишь 6 операций.

Исходя из алгоритма, значения фильтров Хаара (dx, dy) для каждой точки перемножаются с весом и затем они сохраняются в массив. Вес определяем как значение гауссианы имеющей центр в особой точке и сигмой равной 2s. Взвешивание на гауссиану необходимо для отсечения случайных помех на далеких от особой точки расстояниях.

Далее, все найденные значения (dx, dy), условно наносятся в виде точек на плоскость (Рисунок 7).

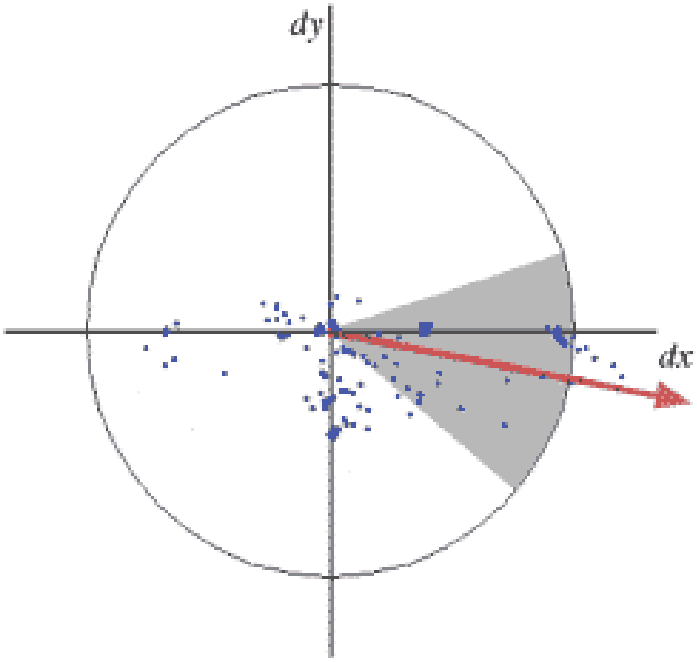


Рисунок 7 Определение ориентации особой точки

После условного нанесения точек на плоскость (Рисунок 9) берется угловое окно (показано серым на рисунке) размером $\pi/3$, и вращается вокруг центра координат. Выбирается такое положение окна, при котором длина суммарного вектора для попавших в окно точек – максимальна. Вычисленный таким образом вектор нормируется и принимается как приоритетное направление в области особой точки.

Манипуляции с окном нужны для уменьшения влияния шумовых точек. На рисунке 8 приведен пример градиента при идеальном крае, и при крае с шумом.

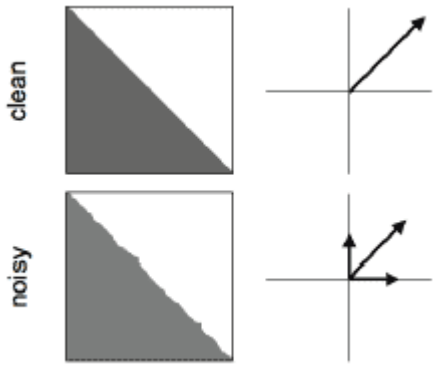


Рисунок 8 Изменение вектора ориентации при наличии помех

Из рисунка 8 следует, что в направлениях, не совпадающих с направлением основного градиента шум дает дополнительные градиенты. Для отсечения шумовых точек следует использовать окно и более точно вычислить истинный градиент.

Дескриптор считается как описание градиента для 16 квадрантов вокруг особой точки.

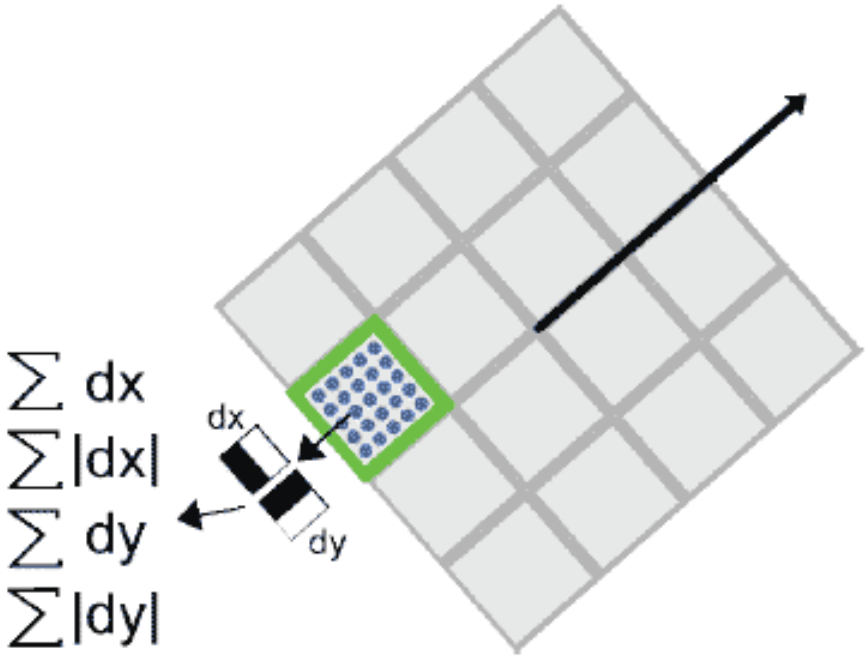


Рисунок 9 Окрестность особой точки

Далее, квадрат разбивается на 16 более маленьких квадрантов, как видно на рисунке. А в каждом квадранте берется регулярная сетка 5x5 и для точки сетки,с помощью фильтра Хаара, ищется градиент. Размер фильтра Хаара принимается равным $2 \cdot s$ и для первой октавы составляет 4x4.

После нахождения 25 точечных градиента квадранта, вычисляются четыре величины, которые собственно и являются компонентами дескриптора:

$$\sum dx, \sum |dx|, \sum dy, \sum |dy|.$$

Две из них есть просто суммарный градиент по квадранту, а две других – сумма модулей точечных градиентов.

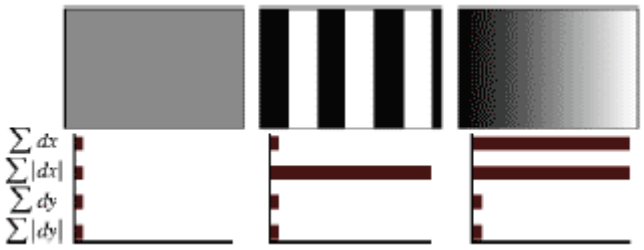


Рисунок 10 Поведение составляющих дескриптора

На Рисунке 12 показано поведение дескриптора на разных изображениях. Где для равномерных областей – все значения близятся к нулю. Для повторяющихся вертикальных полосок – все величины, кроме второй близки к нулю. При увеличении яркости в направлении оси X, две первые компоненты имеют большие значения.

2.2 Способы представления изображения

2.2.1 Общие положения

В настоящее время, несмотря на то, что различного рода изображения все чаще используются для представления и обработки входной, промежуточной и выходной информации во многих прикладных задачах распознавания образов, слабо развиты средства выбора алгоритма обработки изображения в зависимости от его синтаксического и семантического содержания[18]. Наибольшее количество алгоритмов для распознавания изображений, которые обладают вычислительной эффективностью, предназначаются для работы с признаковыми описаниями или моделями изображений. При построении такого описания существенная часть информации, содержащейся в изображении, теряется. Для того, чтобы максимально использовать при анализе изображений содержащуюся в них информацию, необходимо разработать эффективные методы описания изображений и построения моделей изображений. Для решения фундаментальной задачи построения эффективных формальных описаний изображений необходимо провести классификацию и сравнительный анализ средств, позволяющих адекватно описывать изображения и представлять всю информацию, содержащуюся в изображении и необходимую для решения поставленных задач. Решение этой задачи открывает широкие перспективы повышения эффективности выбора алгоритмов анализа изображений и автоматизации (частичной либо полной) обработки изображений.

Моделью изображения называется представление его и/или его характеристик в виде некоторой специальной формальной конструкции, удобной для дальнейшей обработки, распознавания. Для построения модели изображения используются различные математические объекты, в том числе признаки и дескрипторы изображений, логические функции, графы, последовательности преобразований изображений и др. Наиболее часто используются числовые признаки. Числовым признаком изображения называется переменная,

представляющая числовую характеристику свойств изображения, например, геометрических, яркостных, структурных или пространственных. Формальным описанием изображения с помощью числовых признаков называется набор математических объектов (например, чисел, векторов, матриц), содержащих некоторую информацию об изображении, представленную в форме числа и/или набора чисел, и допускаемых алгоритмом распознавания, используемым для решения задачи. Следует отметить, что формальное описание изображения может содержать как полную информацию об изображении (такую, по которой можно восстановить исходное изображение), так и частичную (в случае, когда информации недостаточно для восстановления исходного изображения из его описания).

Свойства признака изображения определяются как свойствами изображения, так и способом извлечения признаков изображения. Признак модели является параметром модели изображения, поэтому тип допустимого пространства признаков, разброс значений признака и методы сравнения значений признаков существенным образом зависят от выбранной модели изображения, в то же время типы алгоритмов распознавания связаны с используемыми типами признаков, поскольку признаки составляют входную информацию алгоритмов распознавания. Отсюда следует, что допустимость применяемых методов распознавания изображений напрямую связана с построенной моделью изображения.

Необходимо отметить, что элементами модели изображения могут быть не только признаки, но и непроизводные элементы, отношения между ними, процедуры и проч.

2.2.2 Низкоуровневое представление

В рамках увеличивающегося объема мультимедиа коллекций особое внимание исследователей направлено на алгоритмы поиска изображений по содержанию (Content Based Image Retrieval, CBIR), которые могут полностью

автоматизировать процесс создания индекса. В рамках отсутствия каких-либо других данных об изображении для создания индекса применяются низкоуровневые характеристики исходного изображения: яркость, цвет, текстура. За последние несколько лет предлагалось огромное количество разнообразных моделей описания подобных характеристик, функций расстояния и определенных с их помощью оценок схожести изображений. Но качество работы систем нахождения изображений по содержанию все же нельзя называть удовлетворительным. Человек, выбирая изображения, сравнивает их семантику, смысловую оценку схожести изображений на базе низкоуровневых характеристик. Главным направлением проводимых в этой области работ становится создание алгоритмов, необходимых для уменьшения «семантического разрыва» между данными анализа изображений системой и восприятием самого пользователя.

Проблемам индексирования и поиска изображений по содержанию отведено большое количество работ, что может говорить о высокой активности исследователей в данной области.

Системы поиска изображений по содержанию применяют разнообразные алгоритмы обработки цифрового сигнала для построения сигнатур - представлений изображения в рамках конкретного набора параметров. Почти во всех предлагаемых подходах одна сигнатура может описать одну из характеристик изображения (цвет, текстура, и т.п.). Т.е. одному и тому же изображению может соответствовать сразу несколько сигнатур, которые описывают каждую из характеристик.

В процессе поиска в рамках запроса зачастую выступает изображение-образец. Итогом такого поиска становится комбинация изображений, близких к изображению-запросу по впечатлениям человека. Различные системы позволяют пользователю создать эскиз того изображения, которое он желает найти. Но оба этих подхода заставляют пользователя как-то конкретизировать свой запрос на языке, понимаемом системой, но не удобном для пользователя. Применение

изображения образца для запроса обязывает пользователя иметь такой образец, что часто просто невозможно. Разработка эскиза нужного изображения заставляет пользователя формулировать запрос на языке форм и цвета. А ведь пользователю всегда удобнее составлять свой запрос на естественном языке, указывая на словах то, что он хочет найти на требуемом изображении. Но сейчас нет системы, позволяющей сгенерировать индекс автоматически в рамках содержания изображения и умеющей понимать запрос на естественном языке.

Большинство исследователей различают сразу несколько уровней содержания изображений. Цвет и яркость принадлежит к низкоуровневому содержанию, находящиеся физические объекты (машина, человек, дом) - к содержанию высокого уровня, а текстура часто принадлежит к среднему уровню. В литературе можно найти огромное количество подходов к моделированию низкоуровневых характеристик изображения и почти все CBIR построены на алгоритмах обработки низкоуровневого содержания. Вот уже несколько лет исследователи понимают нехватку алгоритмов. Важно принять во внимание особенности визуального восприятия человека, его ориентированность на семантику.

Популярным подходом, призванным минимизировать разрыв между представлением изображения системой и человеком становится система обратной связи (relevance feedback). Базирующиеся на итеративном процессе поиска, подобные системы накапливают статистику по реакции пользователя, которая затем может быть учитываться при формировании ответа системы. Примером практической использования подобной системы будет комплекс, созданный в лаборатории CLIPS. Вопросам интерфейса уделяется особое внимание в данной работе, для построения грамотного и полноценного диалога с пользователем.

В работе [1] предоставлен ряд экспериментов, которые показывают возможности использования слов английского языка в качестве основы индекса

изображений. Такой индекс может хорошо согласовываться с семантикой изображения.

2.2.3 Контурное представление

Контурными признаками называют такие признаки, вычисление которых производится на контуре (замкнутом или незамкнутом). Следует отметить, что вычисление контурных признаков происходит в два этапа: 1) выделение пикселов, принадлежащих контуру объекта па изображении; и 2) вычисление собственно признака с использованием выделенных пикселов контура. Как правило, выделяемый контур является бинарным изображением, что существенно уменьшает время работы алгоритмов. Контурные признаки часто используются в задачах распознавания машинописного и рукописного текстов. В [14] для каждого распознаваемого символа выделяются внешний и внутренний контуры, на них вычисляются: признаки, полученные с помощью операции разделения на зоны, коэффициенты сплайна, аппроксимирующего контур, и коэффициенты Фурье для описания эллиптических замкнутых контуров и функции от них[17].

В качестве добавочного признака для распознавания рукописных цифр используется расстояние (по вертикали и по горизонтали) между точками контура цифры и точками выпуклой фигуры, описанной вокруг цифры. В для аппроксимации контуров объектов при распознавании лиц используются коэффициенты, описывающие кривые: коэффициенты прямой - для аппроксимации линии щеки, параболы - для аппроксимации линии подбородка и губ. В [7] контуры объектов используются для вычисления характеристик Эйлера, отражающих топологию описываемого объекта па изображении.

Контурные признаки также часто используются при анализе и классификации изображений отпечатков пальцев [15, 19]. В связи со спецификой таких изображений, на них выделяются контуры яркостных переходов,

характеризующие собственно отпечаток, а затем вычисляются признаки полученных контурных изображений.

2.2.4 Структурное представление

Данный класс содержит модели, представленные символически, или структурные описания. Последние представляют собой структуру данных: набор утверждений, аргументы которых соответствуют частям объекта или сцены (непроизводным элементам) и предикаты которых соответствуют свойствам частей и пространственным взаимоотношениям между ними. Часто структурное описание представляется в виде графа, вершины которого соответствуют частям объекта (или сцепы) и их свойствам, а ребра, связывающие вершины, соответствуют пространственным отношениям.

К сожалению, сегодня науки, связанные с изучением информационных процессов головного мозга человека, восприятия и переработки зрительной информации живыми организмами, не располагают необходимыми данными для создания алгоритма распознавания, успешно моделирующего процессы зрительного восприятия. Поэтому алгоритмы распознавания изображений необходимо основывать на некоторых формальных описаниях.

2.2.5 Признаковое представление

Обозреваемая модель представляет из себя как описание геометрических особенностей изображения с использованием вектора признаков, которые характеризуются особенностями входного изображения. Признаками изображения могут быть как вертикальные, так и горизонтальные линии, кривые или соединения вида «T», «Y» и др. Были разработаны специализированные методы для выделения каждого отдельного признака во входящем изображении вне зависимости от его расположения. После того как все методы будут применены создается вектор признаков, значениями которого являются величины, которые

характеризуют как часто встречается каждый признак на исходном изображении. Дальше в памяти сохраняется список всех возможных признаков, поиск которых ведется на изображении, и вектор признаков, соответствующий исходному изображению. Каждому признаку можно поставить в соответствие некоторый вес с тем, чтобы признаки, инвариантные относительно функций преобразования изображения, оказывали существенное влияние на сопоставление моделей изображений.

При прохождении процесса сопоставления получающийся вектор признаков сравнивается с векторами, что сохраняются в памяти в, с учетом весов, приписанных каждому признаку. Определить меру близости можно с помощью различных способов, например, количество совпадающих признаков, квадратичная разность между элементами векторов и др. Модель, самая ближайшая к исходной модели в смысле заданной меры близости, ставится ей в соответствие.

3 Проектирование программного обеспечения компьютерного зрения

3.1 Выбор программных средств реализации поставленной задачи

Для выбора языка программирования сравним следующие языки – C++, Delphi и C#[1].

Сравнение языков программирования приведено в Таблице 2.

Таблица 2 - Сравнение языков программирования

Параметр	Степень соответствия, %		
	Delphi	C++	C#
Возможность компиляции	8	8	4
Многопоточная компиляция	8	0	8
Интерпретатор командной строки	6	4	0
Многомерные массивы	8	0	8
Динамические массивы	8	8	0
Ассоциативные массивы	4	0	0
Интерфейсы	8	0	4
Мультиметоды	8	0	0
Общая оценка	7,25	2,5	3

При выборе программного обеспечения для разработки приложения будем обращать внимание на такие характеристики как:

- надёжность
- эффективность
- стоимость

- понятность пользователю
- простота настройки
- простота оформления экранных и печатных форм [2]

При выборе IDE для разработки приложения будут рассмотрены:

- Microsoft Visual C# 2008 Express Edition;
- Delphi 7.

Microsoft Visual C# 2008 Express Edition является мощной системой программирования, позволяющей быстро и эффективно создавать приложения для Microsoft Windows.

C# — является типизированным, объектно-ориентированным, простым и в то же время мощным языком программирования, позволяющий разработчикам создавать различные многофункциональные приложения. С использованием возможностей .NET Framework Visual C# 2008 позволяет создавать приложения под платформу Windows, веб-службы, а также инструменты, компоненты и элементы управления базами данных, и многое другое.

Delphi 7 можно назвать знакомой средой для визуальной разработки, основанной на Object Pascal. Этот пакет имеет поддержку новейших перспективных технологий веб-служб. Несет в себе предварительную версию средств для работы с Microsoft .NET Framework. Благодаря Delphi 7 разработчики, применяющие Delphi, могут получить дополнительные навыки программирования для .NET, отлаживать свои приложения для работы под управлением .NET, не теряя при этом навыков и знаний в области проектирования под Windows.

Характерной чертой пакета становится полная поддержка уже созданных и новый Web-сервисов, встроенная модельно-управляемая разработка и возможность предварительного просмотра для Microsoft .NET Framework. Применяя Delphi 7 Studio более 1 миллиона разработчиков, программирующих на Delphi, могут уже сейчас улучшать своих навыки работы с .NET и

реализовывать приложения для .NET с применением уже созданных наработок и навыков взаимодействия на Windows-платформу.

Для разработки информационной системы был выбран язык программирования Delphi.

Основные преимущества и возможности Borland Delphi 7 Studio заключаются в:

- Создание корпоративных приложений с применением MDA — упрощает процесс создания, позволяя разработчикам ускорять процесс от проектирования к установке, минимизируя количество кода и времени, нужного для разработки;
- RAD Visual Web Development — позволяет визуально разрабатывать Web-приложения в среде Delphi 7 Studio и исключает задачи создания общего сервера Application Mode;
- Встроенная межплатформенная поддержка Linux — Delphi 7 Studio будет идти в комплекте с Delphi language version of Borland Kylix 3, первой высокоэффективной, визуальной средой разработки (IDE) для упрощенного создания БД, GUI-, Web-приложений, а также Web-сервисов для ОС Linux;
- Доступность составления отчетов в рамках компании — позволяет разработчикам реализовывать межплатформенные отчеты, определяя эффективность функционирования программ;
- Royalty-Free DataSnap Multi-tier Application Deployment (ранее MIDAS) — обновленное лицензирование гарантирует разработчикам целостных, одиночных, клиент-серверных и многоуровневых приложений реализовать разработку без скрытой платы за все время рабочего использования;
- Создание ПО для Windows XP — Delphi 7 Studio имеет поддержку Windows XP Theme, позволяющую разработчикам разрабатывать ПО, использующее положительные стороны интерфейса Windows XP User.

Для разработки приложения будет использоваться среда разработки RAD Studio Professional.

RAD Studio Professional это мощная IDE включающая в себя:

- интегрированные среды разработки разнообразных приложений под Windows и .NET.
- Интегрированную среду для разработки Delphi
- C++Builder с поддержкой Unicode

Для удобной разработки собственных приложений среда разработки включает в себя сотни различных готовых компонентов и функций, таких как:

- Рефакторинг;
- дополнение кода;
- выделение синтаксиса;
- интерактивные шаблоны;
- полнофункциональная отладка и тестирование модулей

Данная интегрированная среда разработки поддерживает разработку приложений для платформы .NET, включая в себя, поддержку самых последних технологий .NET.

Выбор СУБД представляет собой непростую многопараметрическую задачу и становится одним из самых ответственных этапов при создании приложений БД. Выбранный программный продукт должен отвечать и текущим, и будущим потребностям компании, поэтому важно учитывать денежные затраты на покупку нужного оборудования, самой системы, создание необходимого ПО на ее основе, а также проведение обучения сотрудников. Ещё нужно убедиться, что выбранная СУБД может реально принести компании какие-то выгоды.

При выборе СУБД будет опираться на следующие параметры: стабильность; цена; простота настройки.

При выборе СУБД рассматривались три кандидата:

- Microsoft SQL Server 2012;
- Microsoft Access;
- Oracle;

Microsoft SQL Server 2012 можно назвать совокупным, интегрированным сквозным решением, которое поддерживает организацию безопасной, надежной, и эффективной платформой для реализации обработки промышленных данных и приложений, относящихся к интеллектуальной собственности компании. Microsoft SQL Server 2012 поддерживает мощные, знакомые инструменты как для IT-специалистов, так и для работников IT-сферы, минимизирующие сложность разработки, установки, контроля и применения данных компании и аналитических приложений на платформах от портативных устройств до ИС компании. Благодаря полноценному набору функций, работе с существующими системами и автоматизации типовых задач, SQL Server 2012 позволяет находить решение в области сохранения информации для компаний любого масштаба.

Инструменты разработки: Microsoft SQL Server предлагает встроенные инструменты разработки для ядра БД, извлечения, изменения и загрузки данных, извлечения данных, OLAP и отчётности, которые взаимосвязаны с Microsoft Visual Studio для формирования сквозных возможностей создания ПО. Каждая главная подсистема SQL Server включает в себя собственную объектную модель и набор API для улучшения систем данных в любом направлении, что позволяет оптимально отражать ваш бизнес.

Платформа данных Microsoft SQL Server 2012 дает любой организации такие преимущества как:

- Применение активов данных;
- Увеличение продуктивности;

- Минимизация сложности информационных технологий;
- Уменьшение совокупной стоимость владения.

Microsoft Office Access или просто Microsoft Access представляет собой псевдо-реляционную система управления базами данных разработанная корпорацией Microsoft.

Осуществляет поддержку широкого спектра функций, таких как:

- имеет связанные запросы;
- присутствует связь с внешними таблицами и БД;
- Благодаря наличию языка vba, в самом приложении Access можно реализовывать приложения, работающие с БД;

Oracle Database или Oracle RDBMS является объектно-реляционной системой управления Oracle, занимает верхние строчки на рынке СУБД, а также лидирует на платформах Unix и Windows. В России также просматривается лидерство Oracle, особенно в рамках крупномасштабных ИС государственных структур. Фактически сейчас у нас СУБД Oracle является стандартом для государственных ИС.

Главным преимуществом Oracle перед остальными конкурентами (прежде всего перед SQLServer) является аналогия кода разных версий сервера БД Oracle для всех платформ, что дает идентичность и предсказуемость работы Oracle на любом ПК внутри сети, какие бы конфигурации не входили бы в нее. Все варианты сервера Oracle заключают в себе один и тот же исходный программный код и идентичны функционально, за исключением отдельных опций, которые, добавляются к Oracle Database Enterprise Edition и не включены в Oracle Database Standard Edition. Поэтому для всех платформ создана единая СУБД в различных версиях, которая одинаково работает предоставляет единую функциональность вне зависимости от базовой платформы, на которой она функционирует.

В Таблице 3 приведена таблица сравнения для трех распространенных систем управления базами данных, которые конкурируют на рынке программного обеспечения по основным показателям.

Таблица 3Сравнение СУБД

Показатели	MS Access	баллы	Microsoft SQL Server 2012	баллы	Oracle	баллы
Поддерживаемые операционные системы	Windows Desktop/ Server	3	Windows Desktop/ Server , Linux, Unix, Mac	5	Windows1 Desktop/S22erver, Linux, Unix, 2Mac	5
Поддержка даты и времени	Да	3	Да (без временной зоны)	2	Да	3
Аутентификация	Средствами БД и ActiveDirectory	3	Средствами БД	2	Много разных методов, включающих предыдущие	1
Производительностьпланировщика запросов для сложных запросов	Средняя (умеет параллельные запросы «из коробки»)	1	Очень хорошая	5	Плохая	1
Итого		14		10		10

Таким образом, для разработки системы распознавания образов необходимо и достаточно разработать базу данных в среде СУБД Access 2010 [15].

3.2 Выбор и реализация алгоритмов, позволяющих решить поставленную задачу

3.2.1 Общий алгоритм для решения поставленной задачи

Основным алгоритмом работы системы является распознавание образа на основании метода сопоставления с эталоном. Алгоритм данного процесса приведен на рисунке 11.

Далее работа программы осуществляется в следующем порядке.

Результаты классификации

Результаты классификации представляются классом «TSearchInfo». Данный класс инкапсулирует в себе следующие данные:

1. «ObjectID» - идентификатор образа, с которым производилось сравнение;
2. «ClassID» - идентификатор класса, к которому принадлежит данный образ;
3. «Similarity» - степень сходства входного изображения и образа (количество сходных точек);
4. «Points» - список характеристических точек образа;
5. «Matches» - список сходных точек (точки входного изображения сходные с точками образа);
6. «ComputationTime» - время, затраченное на сравнение точек образа и точек входного изображения.

Класс «TAnalyser» активно использует данную структуру данных для хранения результатов вычислений.

Класс «TAnalyser» ведет историю вычислений (поле «FClassificationHistory : TStringList»). При выполнении классификации, экземпляр класса «TSearchInfo» создается для каждого образа, т.к. сравнение входного изображения производится с каждым образом, который имеется в базе данных.

Также класс «TAnalyser» содержит поле (FClassificationResult: TSearchInfo), которое хранит наилучший результат сравнения (максимальное количество сходных точек). Данное поле и является результатом классификации. Класс входного изображения определяется классом данного образа.

Сравнение точек

Сравнения точек производится по схеме «каждая с каждой». Для примера рассмотрим следующую ситуацию: у нас имеется набор точек входного изображения (279 точек) и набор точек образа (357 точек). В итоге придется выполнить 279*357=99603 сравнений «точка с точкой». А сходство двух точек определяется как среднее геометрическое (расстояние):

$$d=\sqrt{(x_2-x_1)^2+(y_2-y_2)^2}$$

Но т.к. в нашем случае дескриптор точки представляет собой вектор из 64-х составляющих, то формула расчета расстояния между двумя точками будет представлена следующим образом:

$$d=\sqrt{(x_{2,1}-x_{1,1})^2+(x_{2,2}-x_{1,2})^2+(x_{2,3}-x_{1,3})^2+\cdots+(x_{2,64}-x_{1,64})^2}$$

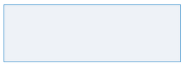


Рисунок 11 - Алгоритм сравнения образа

Алгоритм классификации

Алгоритм классификации входных изображений можно представить следующей формальной блок-схемой (рисунок 11). Работа алгоритма начинается с подачи на его вход изображения (входного изображения). После этого выполняется расчет точек входного изображения (ниже будет описан процесс вычисления точек изображения методом SURF). Следующим действием выполняется подготовка полей класса к сохранению результатов новой классификации. Поле этого выполняется запрос к базе данных, результатом которого является список всех идентификаторов образов, хранящихся в БД. Далее, в цикле выполняется перебор данного списка и последовательная выгрузка из базы данных точек образов. После выгрузки точек образа выполняется их сравнение с точками входного изображения. Полученные результаты сравниваются с предыдущими «лучшими» результатами (переменная выходного результата), и если предыдущие результаты хуже, то они перезаписываются текущими результатами.

После прохождения всего списка образов до конца, алгоритм завершает свою работу, а выходная переменная будет содержать результаты сравнения входного изображения и максимально идентичного образа. Класс входного изображения определяется классом образа. Данная схема реализована в методе «Classification» класса «TAnalyser» (рисунок 12).

Вычисления точек изображения методом SURF

Каждая найденная тока описывается классом «IPoint». Данный класс содержит поля (координаты точки, масштаб, дескриптор точки), которые в полной мере описывают характеристическую точку. Но т.к. на изображении может располагаться множество точек, то для хранения множества экземпляров класса «IPoint» используется список (класс «TStringList»).

Делее будет представлена блок-схема для алгоритма вычисления точек (рисунок 13). Блок-схема соответствует методу «function GetIPoints (B :

TBitmap) : TStringList» класса «TAnalyser». Алгоритм начинается с загрузки входного изображения. Изображение представляется в виде битмапа. Создается переменная (экземпляр класса «TStringList») которая будет содержать результаты работы SURF.



Рисунок 12 - Алгоритм классификации

Для изображения вычисляется его интегральное представление, которое будет использоваться для дальнейших вычислений. Далее производится поиск

особых точек (рисунок 13).



Рисунок 13 - Блок-схема для алгоритма вычисления точек

Завершающим этапом алгоритма является вычисление дескрипторов для найденных особых точек – описание особой точки и её окрестности при помощи метода SURF.

3.2.2 Выбор конкретных алгоритмов для решения соответствующих этапов основного алгоритма

В результате изучения алгоритмов распознавания и сравнение выбраны следующие - для распознавания выбран метод сравнения с эталонами, а для извлечения признаков метод SURF.

Распознаванием образов называют сравнение его с набором векторов-эталонов такой же размерности. Задача по распознаванию и принятию решений о принадлежности данного образа к тому или иному классу, основанная на анализе вычисленных признаков, имеет целый ряд строгих математических решений в рамках детерминистического и вероятностного подходов [12, 14].

В случае, если целью является создания системы для распознавания символьных образов, то для ее реализации наиболее часто используется классификация, основанная на подсчете евклидова расстояния между вектором признаков распознаваемого символа и векторами признаков эталонного описания. Качество распознавания в наибольшей степени зависит от типа и количества признаков. Так же во время анализа заранее подготовленного изображения происходит формирование вектора. Такой процесс называется извлечением признаков. Чтобы получить эталон для каждого класса следует провести аналогичную обработку символов для обучающей выборки [11].

Основными достоинствами признаковых методов являются:

- Простые в реализации;
- хорошая способность к обобщению;
- хорошая устойчивость к изменениям формы символов;
- низкое число отказов от распознавания;
- высокая скорость работы;

Наиболее серьезные недостатки данных методов:

- неустойчивость к различным дефектам изображения;
- на этапе извлечения признаков происходит необратимая потеря части информации о символе;
- Извлечение признаков ведется независимо, поэтому информация о взаимном расположении элементов символа утрачивается [12];

Поскольку четких правил для отбора признаков не может существовать, из-за чего методы разрабатываемые разными разработчиками систем распознавания могут оперировать абсолютно различными наборами признаков[13].

Алгоритм работает по следующему принципу. Для изображения сцены и изображения эталона с помощью метода SURF находятся особые точки и уникальные дескрипторы для них. При сравнении этих наборов дескрипторов, мы выделяем объект-эталон на сцене.

С помощью SURF решается две задачи - поиск особых точек изображения и создание их дескрипторов, которые инвариантны к масштабу и вращению.

Алгоритм SURF использует разномасштабные фильтры для нахождения гессианов [14].

После того как ключевые точки будут найдены, SURF сформирует их дескрипторы. Дескриптор представляет из себя набор из 64(128) чисел для каждой ключевой точки. Данные числа отображают флуктуации градиента вокруг ключевой точки. И поскольку ключевая точка представляет собой максимум гессиана, то это гарантирует, что в окрестности точки должны быть участки с разными градиентами. Таким образом, обеспечивается дисперсия (различие) дескрипторов для разных ключевых точек[15].

Флуктуации градиента окрестностей ключевой точки считаются относительно направления градиента вокруг точки в целом (по всей окрестности ключевой точки). Таким образом, достигается инвариантность дескриптора относительно вращения. Размер же области, на которой считается дескриптор, определяется масштабом матрицы Гессе, что обеспечивает инвариантность относительно масштаба.

Плюс к дескриптору, для описания точки используется знак следа матрицы Гессе, то есть величина sign(Dxx+Dyy). Для темных точек на светлом фоне – положителен, а для светлых точек на темном фоне, след отрицателен. Таким образом, SURF различает светлые и темные пятна.

После применения алгоритма SURF к изображению мы получаем набор дескрипторов который будет уникально идентифицировать эталон на сцене.

Достоинствами метода являются[16]:

1) Инвариантен к поворотам и масштабированию;
2) Инвариантен к разнице общей яркости изображений;
3) Может детектировать более 1 объекта на сцене;

Недостатки метода:

1) Весьма сложный в реализации;
2) Скорость работы алгоритма относительно медленная;

3.3 Проектирование структуры проекта программного обеспечения

3.3.1 Разработка базы данных для эталонных изображений

При разработке базы данных согласно методике предложенной [12] обычно выделяется некоторое количество этапов, с помощью которых осуществляется переход от предметной области к созданию базы данных средствами выбранной СУБД [8].

Перечислим основные этапы [17]:

1. Анализ предметной области.
   - Определение основных объектов (сущностей) предметной области, выявляют их атрибуты и взаимосвязи между собой.
   - Определение примерного состава информационного фонда [3].
2. Информационный анализ класса задач.
   - Анализ внешних моделей данных, которые используются в программе.
   - Анализ источников и возможностей для формирования информационного фонда
3. Построение инфологической модели предметной области.

- Выбор наиболее адекватной модели для описания предметной области.
- Графическое изображение модели.
- Формирование требований к СУБД и ограничений

4. Выбор конкретной СУДБ. Выбор СУБД – претендентов.
   - Сравнение возможностей СУБД-претендентов с учетом требований и ограничений[18].
   - Анализ возможностей отображения инфологической модели в СУБД-ориентированные модели.
   - Окончательный выбор СУБД.
5. Формирование информационного фонда.
   - Определение источников получения информации.
   - Определение характера возможной информации, характеристик информационных массивов (объем, частота, характер использования).
6. Построение концептуальной модели[19].
   - Сопоставление инфологической модели с составом информационного фонда.
   - Отображение инфологической модели в СУБД ориентированную.
   - Описание концептуальной модели (построение концептуальной схемы БД средствами, предлагаемыми СУБД).
7. Построение внешних моделей.
   - Разработка СУБД-ориентированных внешних моделей (в соответствии с результатами этапов 2 и 5).
   - Отображение концептуальной модели во внешние модели.
   - Описание внешних моделей (построение внешних схем БД средствами, предлагаемыми СУБД) [12].

Удобным и наглядным представлением процесса проектирования баз данных могут служить модели данных полученные в результате разработки структуры БД.

Таким образом, можно выделить следующие уровни перехода к конкретной реализации БД [10]:

1. Предметная область
2. Модель предметной области
3. Инфологическая модель
4. Концептуальная модель
5. Внутренняя модель (физический уровень)

ER-диаграмма базы данных разработанной информационной системы приведена на рисунке 14.



Рисунок 14 Диаграмма базы данных на этапе логического проектирования

В таблице Class хранятся данные по описанию классов объектов, в таблице Object_set - данные по наборам элементов, в таблице params – параметры распознавания.

Далее была сформирована физическая модель базы данных в выбранной СУБД (рисунок 15)

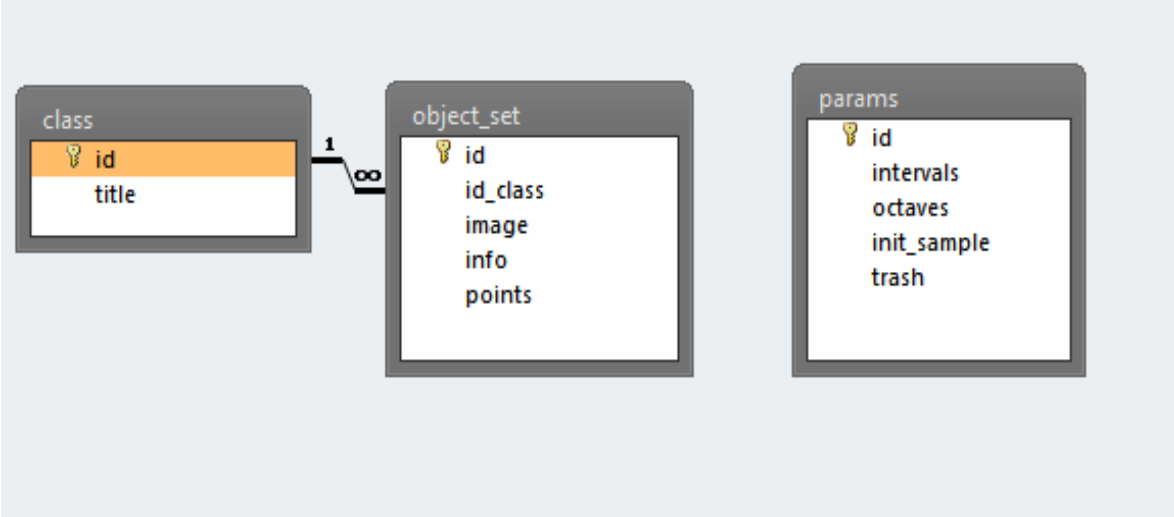


Рисунок 15 - База данных на этапе физического проектирования

Ниже, на рисунках, представлена характеристика таблиц базы данных (рисунки 16-18).

class

Имя поля	Тип данных
id	Счетчик
title	Текстовый

Рисунок 16 - Характеристика таблицы Class

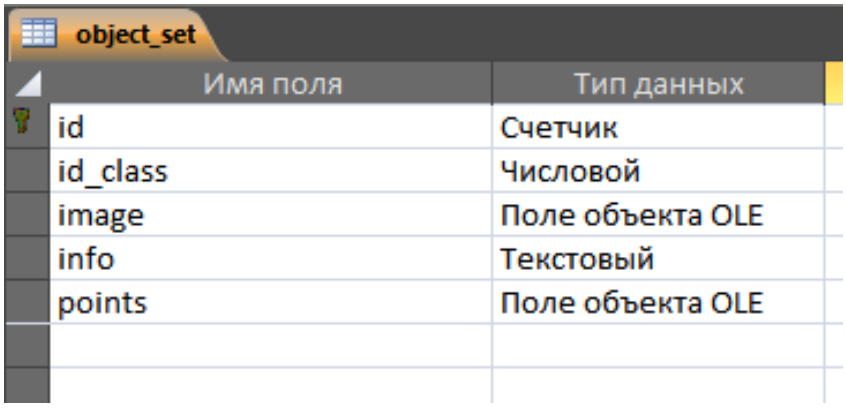


Рисунок 17 - Характеристика таблицы Object_set

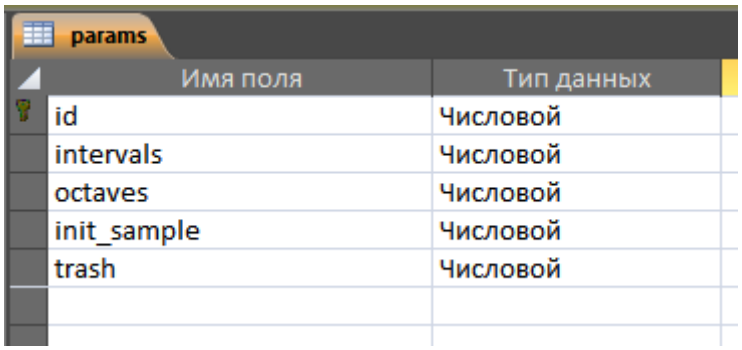


Рисунок 18 - Характеристика таблицы params

3.3.2 Предобработка эталонных изображений

Эталонные изображения в таких системах могут быть получены предварительной съёмкой аналогичных объектов.

Каждый из классов задаётся с помощью прямого указания его членов [3]. Данный подход обычно используют только в том случае, если полная априорная информация о всех возможных объектах распознавания доступна. Осуществляется сравнение попадающих в систему образов с изначально заданными описаниями представителей классов и относящихся к тому же классу, которому принадлежат наиболее схожие с ними образцы [20].

При обосновании допустимых диапазонов необходимо учитывать два фактора, связанные с размером изображения и влияющие на эффективность процедур определения класса подвижных транспортных наземных объектов.

Во-первых, имеет место потеря информации в случаях маленького масштаба искомого объекта из-за дискретного представления цифровых изображений. Поэтому момент времени, после которого возможно начало регистрации, должен назначаться при достаточном приближении ЛА к объекту. С другой стороны, ограничение связано с тем, что регистрация объекта должна быть завершена на некотором расстоянии от него, за время преодоления которого объект должен быть распознан.

3.3.3 Программная архитектура системы компьютерного зрения

Диаграмма классов программы приведена на рисунке 19. Рассмотрим назначение основных классов.

Класс, реализующий алгоритм подсчета количества образов из базы данных на выбранном изображении. Принцип работы данного класса состоит в следующем: задается базовое изображение, которое анализируется при помощи метода SURF, результатом работы которого является множество характеристических точек. После, данное множество точек кластеризуется при помощи алгоритма FOREL.

После, точки каждого кластера сравниваются с точками изображений хранящихся в базе данных, и делается вывод о том, является ли данный кластер схожим с образом БД и производится подсчет результатов.

Класс TResponseLayer выполняет вспомогательную функцию, которая заключается в хранении всех имеющихся фильтров, и результатов их применения. Фильтры имеют свой размер и масштаб. Так как поиск особых точек выполняется в различных масштабах фильтров, то реализация данного класса позволяет упростить способ хранения результатов.

Класс IPoint – осуществляет хранение информации об особых точках. Один экземпляр класса IPoint представляет собой одно описание особой точки. Данный класс описывает такие характеристики особой точки, как координат,

масштаб, дескриптор точки, и т.д. Итогом работы метода SURF является список дескрипторов особых точек. В классе TFastHessian осуществляется заполнение списка особыми точками, а класс TSurf служит для вычисления дескрипторов особых точек.

Класс, реализующий алгоритм идентификации транспортных средств при помощи базы данных образов и алгоритма SURF (таблица 4).

Класс, используемый для хранения результатов классификации (таблица 5).

Класс, выполняющий визуализацию результатов работы алгоритма SURF (таблица 6).

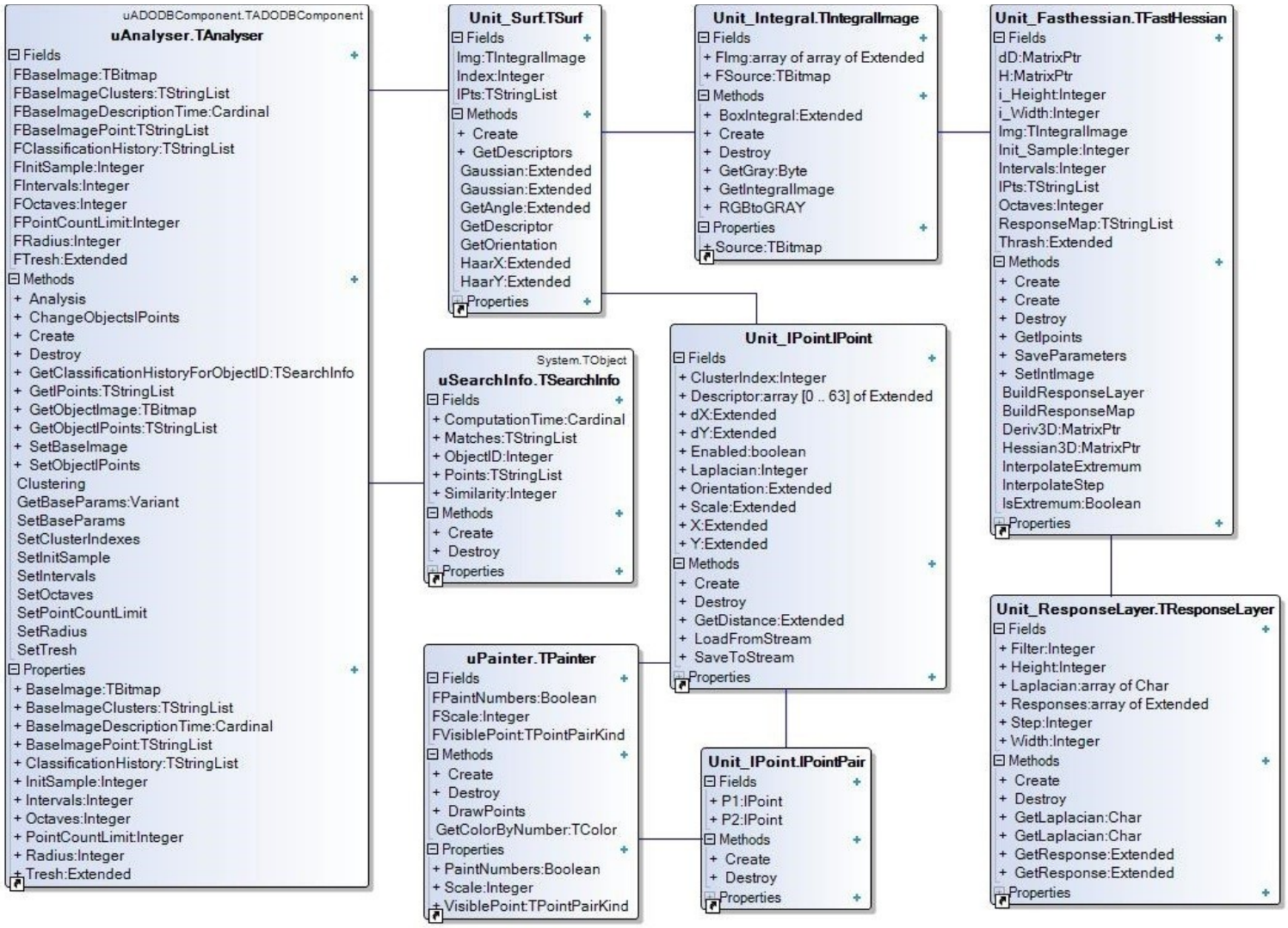


Рисунок 19 - Диаграмма классов проекта

Таблица 4 - Класс «TAnalyser»

Property Summary	
public TBitmap	BaseImage Базовое изображение
public TStringList	BaseImagePoint Характеристические точки исходного изображения
public TStringList	ClassificationHistory История классификации
public uSearchInfo.TSearchInfo	ClassificationResult Результат классификации
public Integer	InitSample Метод установки значения параметра SURF (начальное значение)
public Integer	Intervals Параметр алгоритма SURF (интервалы)
public Integer	Octaves Параметр алгоритма SURF (октавы)
public Extended	Tresh Параметр алгоритма SURF (шум)
Constructor Summary	
Create(Connection: TADO-Connection) Конструкторкласса.	

Таблица 5 - Класс «TSearchInfo»

Field Summary	
public Integer	ClassID Идентификатор класса
public Cardinal	ComputationTime Время вычисления (мс)
public TStringList	Matches Список сходных точек
public Integer	ObjectID Идентификатор объекта
public TStringList	Points Список точек изображения
public Integer	Similarity Степень сходства
Constructor Summary	
Create() Конструктор класса.	
Method Summary	
public Sub	Destroy() Деструктор класса.

Таблица 6 Класс «TPainter»

Property Summary	
public Boolean	PaintNumbers Флаг, который определяет факт отображения чисел около характеристических точек
public Integer	Scale Масштаб
public TPointPairKind	VisiblePoint Флаг определяющий точку, выводимую на изображение (из пары)
Constructor Summary	
Create() Конструктор класса	
Method Summary	
public Sub	Destroy() Деструктор класса
public Sub	DrawPoints(Source: TPicture; Dest: TPicture; Points: TStringList) Методотрисовкиточек

3.4 Описание работы программы

После запуска программы пользователь получает ее главную экранную форму (рисунок 20).

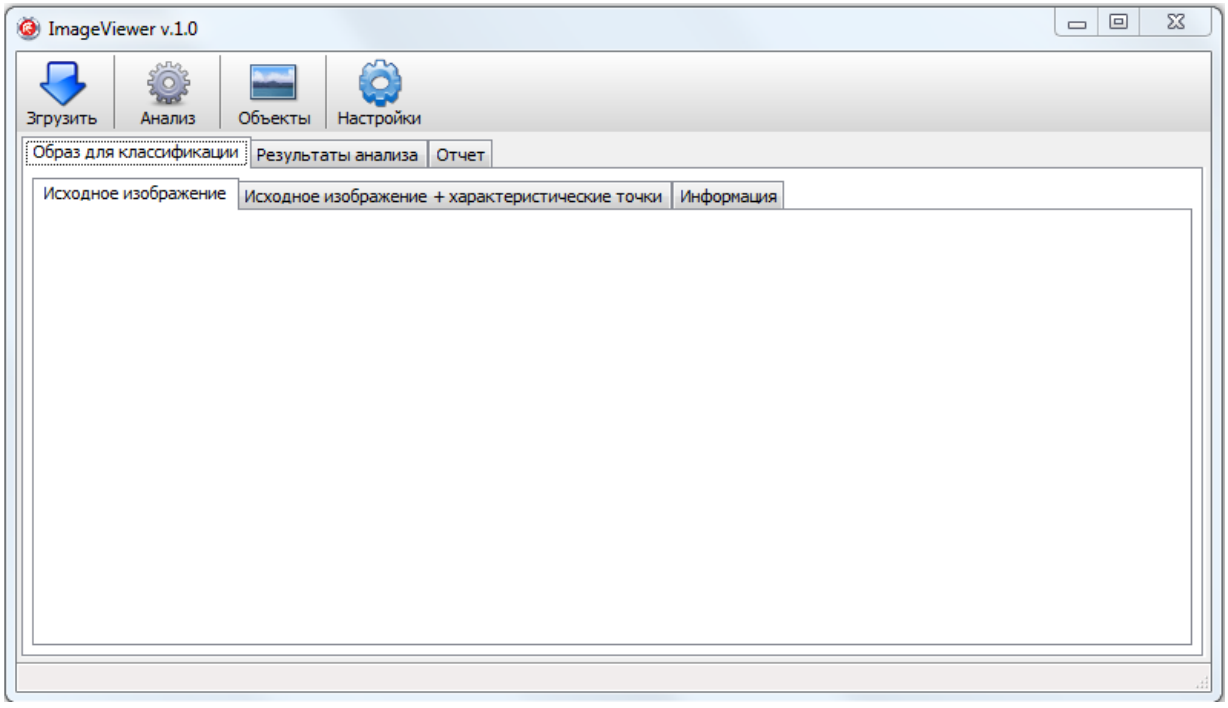


Рисунок 20 - Главная экранная форма

После этого необходимо загрузить наборы эталонных изображений, которые должны быть использованы в ее работе (рисунок 21).

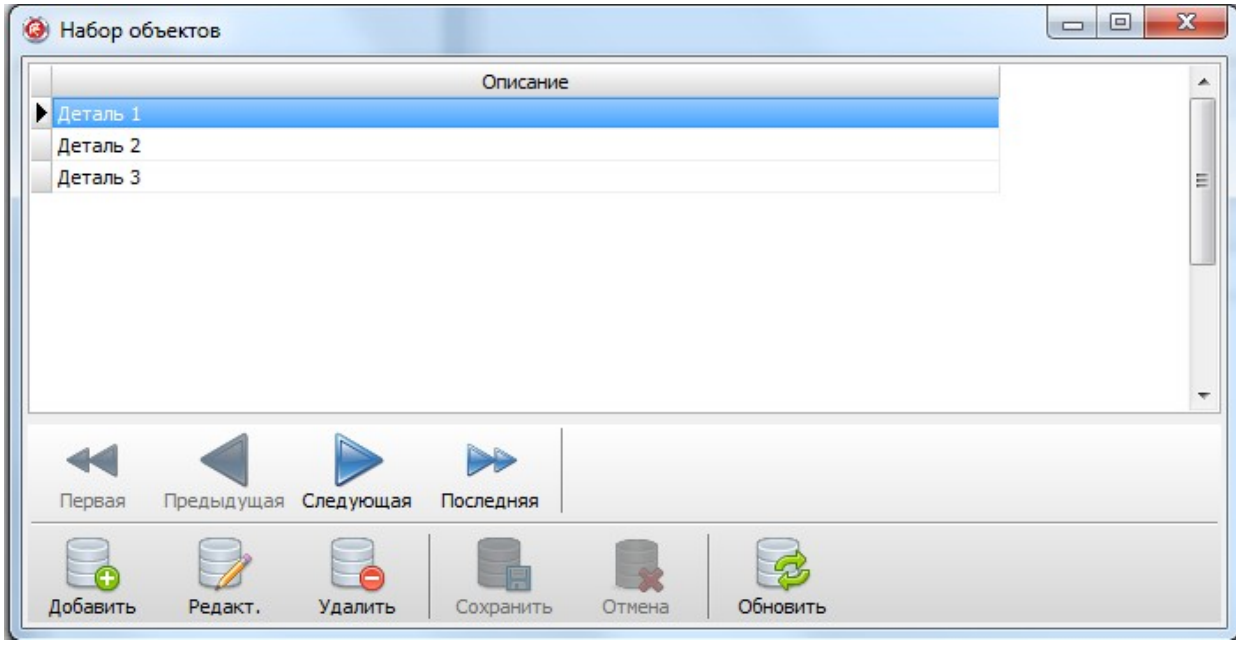


Рисунок 21 - Наборы изображений

Для добавления данных необходимо ввести название и эталонное изображение (рисунок 22).

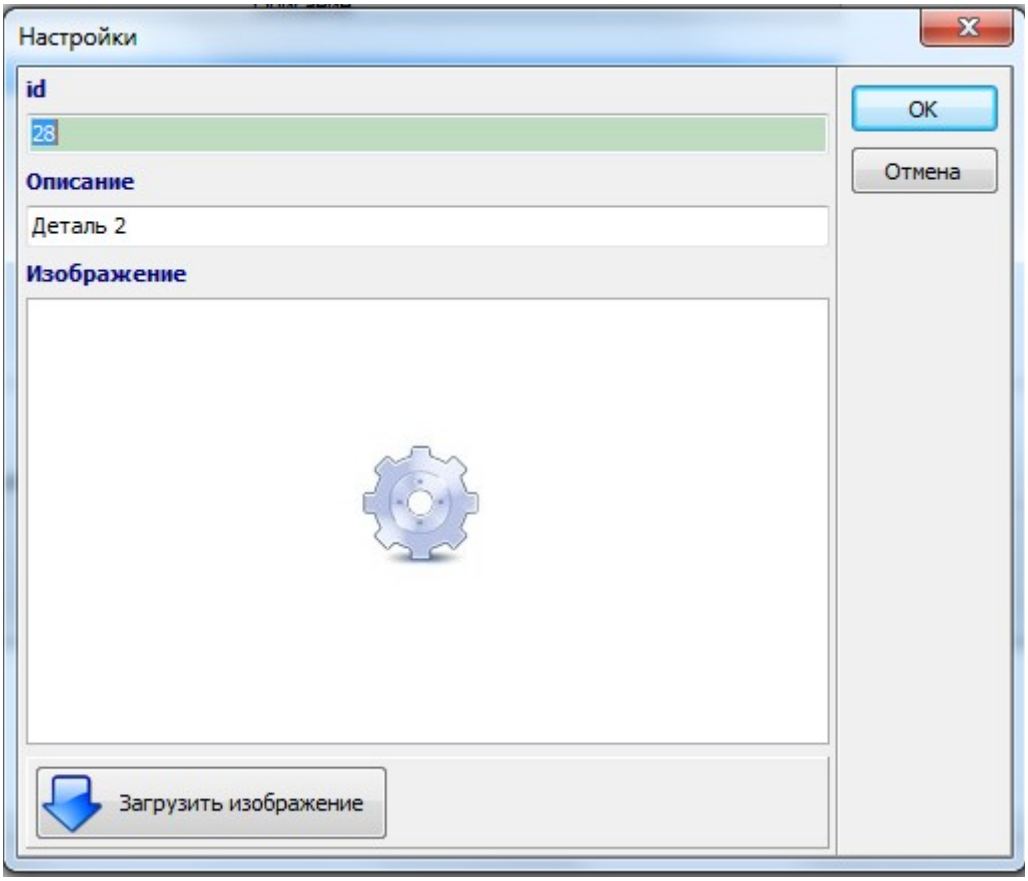


Рисунок 22 - Ввод данных в справочник Классы объектов

После чего осуществляется загрузка файла с изображениями с для их подсчета (рисунок 23).

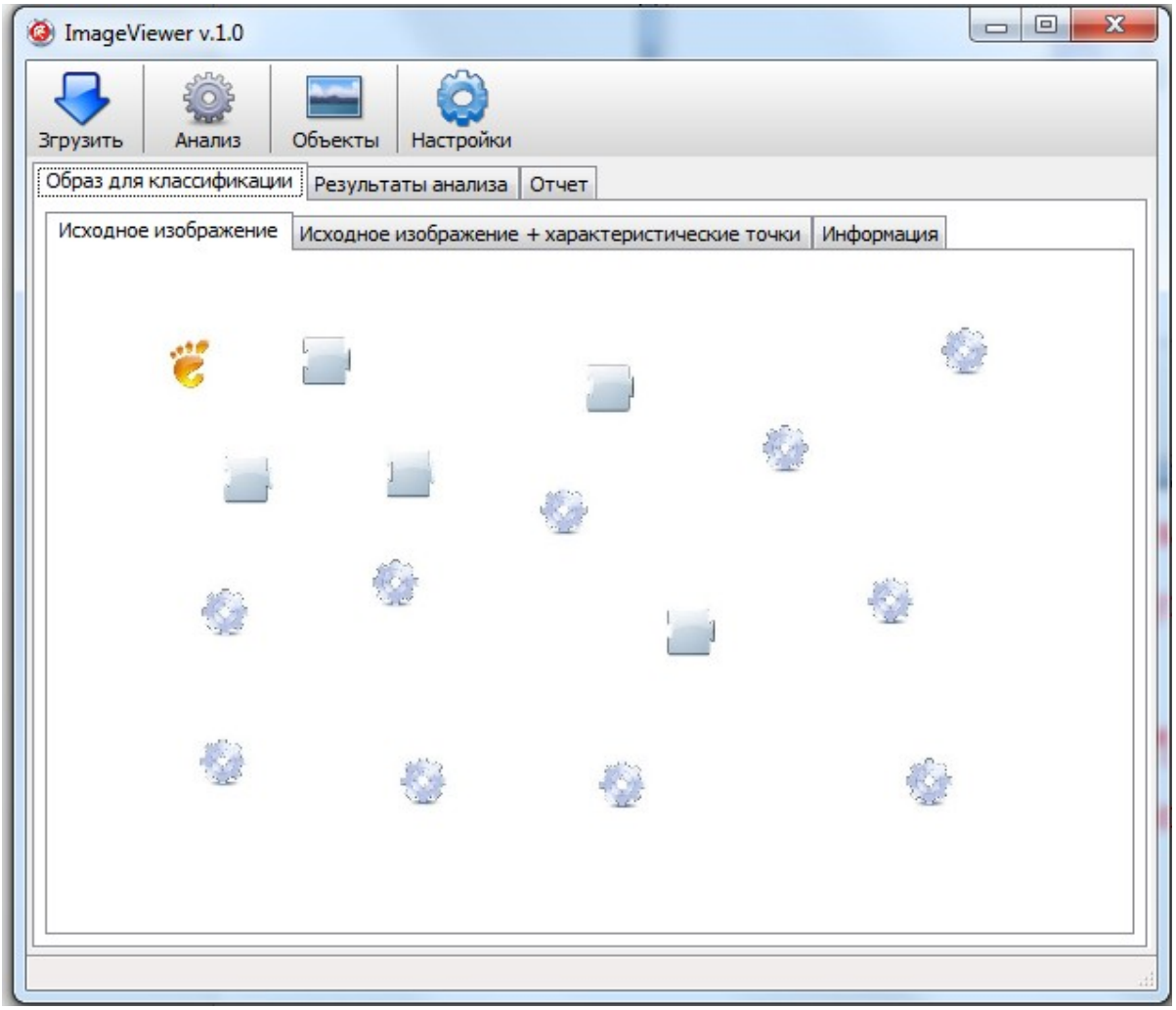


Рисунок 23 - Загруженный файл для подсчета количества деталей

Программа показывает выделение характерных точек (рисунок 24):

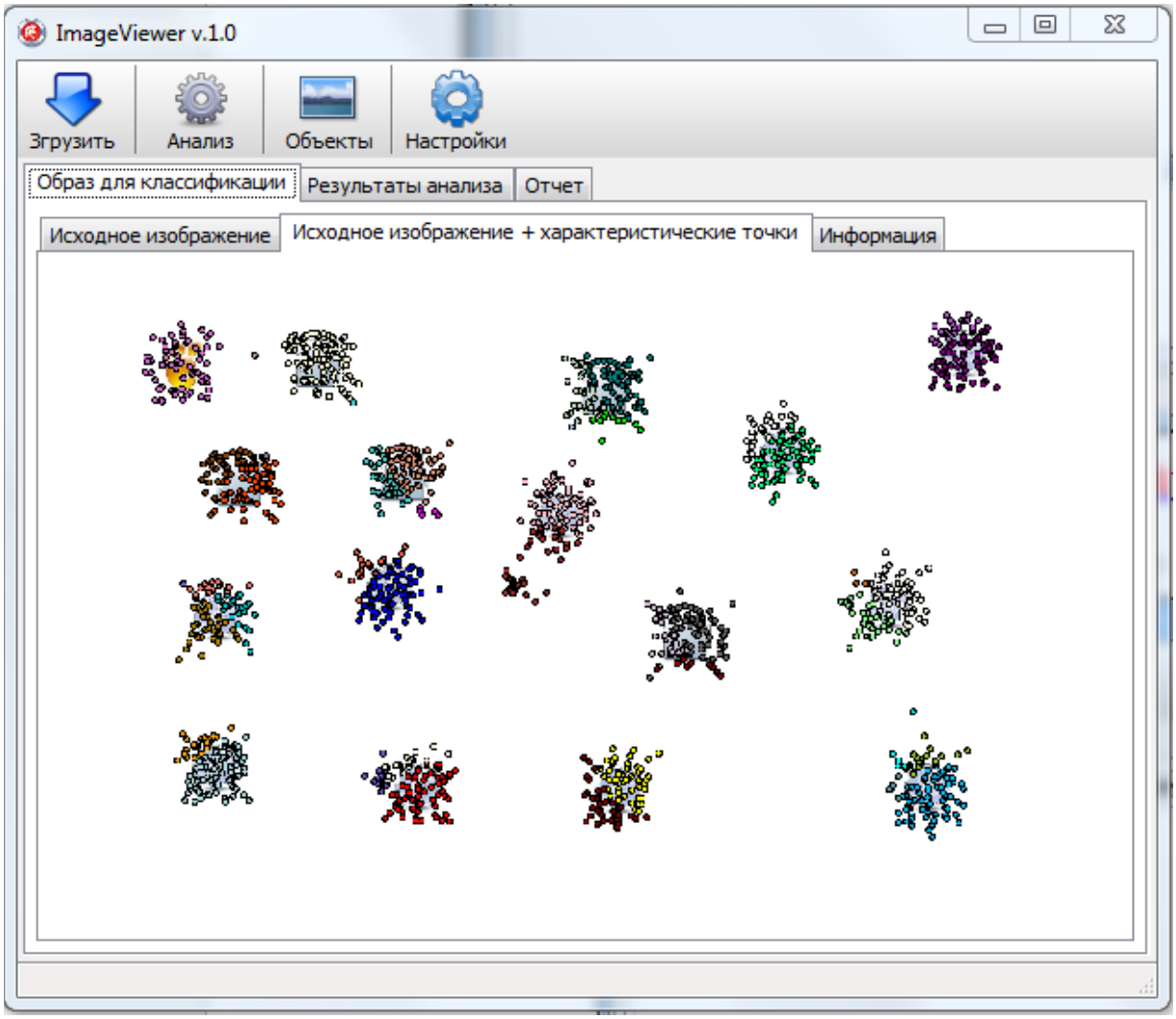


Рисунок 24 - Изображение с характерными точками

И предоставляет отчет о посчитанном количестве (рисунок 25).

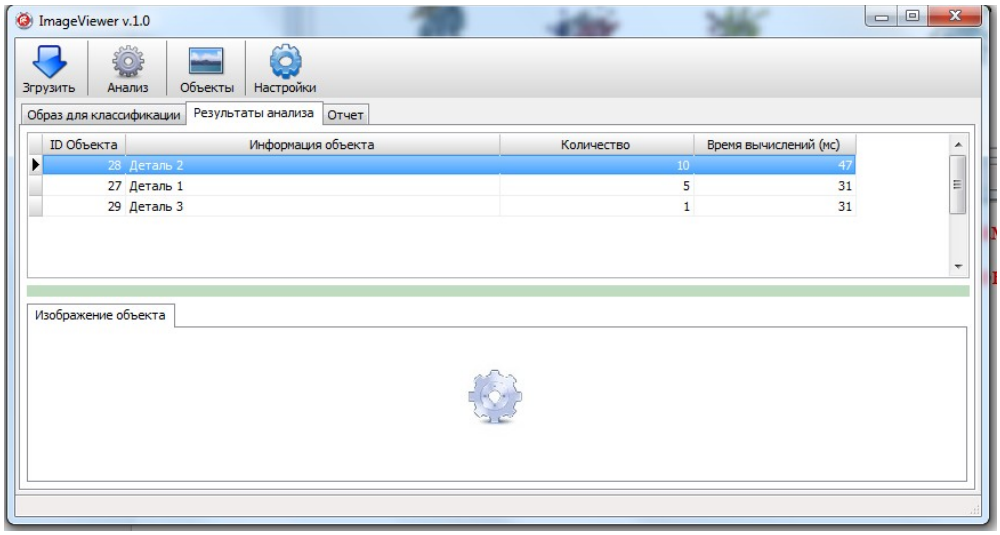


Рисунок 25 - Итоги работы программы

Листинг программных модулей приведен в Приложении.

ЗАКЛЮЧЕНИЕ

Специфика создания конкретных систем технического зрения выражается следующими главными особенностями. Во-первых, при проектировании подсистем информационного обеспечения, требуется решать не общую проблематику автоматического понимания изображения произвольной сцены, а более определённую и узкую задачу проблемно-ориентированного представления изображения. Зачастую эта задача сводится к нахождению и выявлению на изображении отдельного набора объектов известных заранее типов, наличие которых влияет на построение управления. Во-вторых, к алгоритмам анализа изображений в подсистемах информационного обеспечения выдвигаются специальные требования, связанные с определенными характеристиками и назначением создаваемой или уже существующей среды управления.

В рамках специфики конкретной задачи, система автоматического управления запрашивает у подсистемы анализа изображений такие параметры, как локализация, робастность и вычислительная реализуемость в пределах указанной аппаратной архитектуры. При этом одна из главных проблем, отличающих область обработки изображений, к примеру, от теории обработки сигналов, состоит в проектировании методов нахождения объектов, плохо реагирующих на разнообразные виды изменчивости, характерные только для изображений, например: искажения оптических сенсоров, блики, тени, загораживания, искажения формы, расфокусировка, шум.

Важно также понимать, что существенное увеличение возможностей подсистем технического зрения достигается при совместном применении датчиков различной физической природы (ДРФП). Финальная эффективность такого решения задачи управления целиком зависит как от применяемого набора и параметров датчиков, так и от алгоритмов обработки и комплексирования данных.

Алгоритмы технического зрения часто проверяются на математических моделях характерных целей, получаемых искусственным путём с помощью ЭВМ. Этот метод можно назвать самым быстрым, также он дает возможность отрабатывать алгоритмы в отношении объектов, реальные изображения которых недоступны. Но разрабатываемые алгоритмы должны уметь работать не только на модельных, но и на вполне реальных изображениях. При этом каждый, кто создавал алгоритмы обработки реальных изображений, прекрасно знает, какими неопределенными и изменчивыми могут быть факторы, отвечающие за качество реальных изображений от датчиков и соответственно, все это влияет на степень обнаружения и распознавания объектов на данных изображениях.

Укажем данные факторы:

- Шум — имеют несколько видов источников возникновения, к числу которых относят несовершенство сенсоров принимающей аппаратуры и устройства оцифровки изображений, сложные условия съёмки, малая освещенность и ряд других;
- Сложный текстурированный фон, на котором нужно обнаружить объекты, к примеру, нахождение штриховой наклейки на газетной странице и т. п.;
- Загораживание (заслонение) одних объектов другими, обычно, не определённой заранее формы, к примеру — облако на космофотоснимке и т. п., другие загораживающие помехи;
- Изменяющие оптические эффекты в виде расфокусировок, а также ракурсных искажений и др.;
- Резкая смена освещения, тень, блики, особенно в динамичных сценах;

- Изменяемость и разнообразность самих объектов обнаружения — переменная структура (штрих-код), временные перемены формы, дефекты, вегетационные циклы для растений и т. п.;
- Изменение среды между сенсорами и объектами съемки — осадки, пыль, туман, дым, искусственные помехи и т.д.;
- Рассинхрон в записи и обработка данных в динамических задачах обнаружения, связанная с нехваткой средств сохранения и анализа изображений - наиболее критичны для приложений с малым временем реакции системы нахождения объектов; сюда относятся также сбои в компьютерном ПО для обработки.

По итогам выполненной работы делаем несколько выводов. Исходя из проведенного исследования, различные алгоритмы включают свои плюсы и минусы, поэтому все алгоритмы имеют персональную область использования.

В рамках проведенной работы были решены следующие задачи:

- Изучены имеющиеся алгоритмы распознавания изображений. Отражены их положительные и отрицательные стороны;
- Изучены методы эффективного практического внедрения алгоритмов нахождения объектов на изображениях;
- Описано исследование некоторых алгоритмов поиска объектов на 2D-изображениях;
- Применена часть алгоритмов: алгоритм SURF и алгоритм сравнения с эталонной моделью.

Данная выпускная квалификационная работа не может осветить все возможные аспекты, которые могут быть связаны с такой большой областью исследований, как поиск и распознание объектов на изображениях, по причине ее обширности. Не затрагивались такие темы как: повышение устойчивости методов к аффинным преобразованиям, улучшение качества

распознания на зашумленных изображениях, пространственный поиск и распознание объектов и многие другие, данные темы слишком объемны и чтобы полностью отобразить их в одной работе и каждая из них заслуживает отдельной.

Таким образом, цель работы выполнена. Практическая значимость исследования заключается в возможности использования разработанного программного обеспечения на практике.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ И ЛИТЕРАТУРЫ

1. Березин С.А., Березин Б.А., Начальный курс С и С++, М., Диалог-МИФИ, 2017 г., 288 с.
2. Брайан У. Керниган, Деннис М. Ритчи, Язык программирования С, М., Вильямс, 2017 г, 288 с.
3. Брайан У. Керниган, Роб Пайк, Практика программирования, М., Вильямс, 2017 г., 288 с.
4. Бунаков П.А., Лопатин А. В, Практикум по решению задач на ЭВМ в среде Delphi. Учебное пособие, М., Инфра-М, 2018 г, 304 с.
5. Васильев Р.А, Калянов Г.А., Левочкина Г.А.,Стратегическое управление информационными системами, М, Интернет-университет информационных технологий, Бином. Лаборатория знаний, 2017 г, 512 с.
6. Вон Вернон, Реализация методов предметно-ориентированного проектирования, М, Вильямс, 2017 г, 688 с.
7. Восстановление изображений, различие и оценка параметров объекта, М, LAP Lambert Academic Publishing, 2013, 76 с.
8. Гольштейн Е.А., Юдин Д.А., Специальные направления в линейном программировании, М, Красанд, Editorial URSS, 2018 г, 526 с.
9. Гонсалес Р., Вудс Р., Эддинс С. Цифровая обработка изображений
10. Грацианова Т.А., Программирование в примерах и задачах, М., Лаборатория знаний, 2018 г, 368 с.
11. Дунаев А., Использование теории графов для распознавания образов, М, Издательский дом Университета "Синергия", 2017, 145 с.
12. Иванова Г.А., Технология программирования, М, КноРус, 2018 г, 336 с.

13. Исаев Г.А, Теоретико-методологические основы качества информационных систем, М. Инфра-М, 2018 г. 258 с.

14. Л. Рубанов, П. Чочиа, Рафаэль Гонсалес, Ричард Вудс, Цифровая обработка изображений, М, Техносфера, 2012 , 1104 с.

15. Распределенные вычисления и ГРИД-технологии в науке и образовании. Труды 5-й международной конференции Дубна, 16-21 июля 2012 г., Под редакцией: Т. А. Стриж Дубна: Объединенный институт ядерных исследований, 2012.

16. Старовойтов В.В., Голуб Ю.И. Цифровые изображения: от получения до обработки, Минск: ОИПИ НАН Беларуси, 2014. — 202 с.

17. Стенли Б. Липпман, Жози Лажойе, Барбара Э. Му, Язык программирования C++. Базовый курс, М, Вильямс, 2017 г, 1120 с.

18. Техническое зрение в системах управления мобильными объектами, Труды научно-технической конференции-семинара / под ред. Р.Р. Назирова. – М.: Механика, управление и автоматика, 2011. – 328 с.

19. Уильям Гибсон, Распознавание образов, М, Азбука, Азбука-Аттикус, 2017, 384 с.

20. Ягольников С.А., Александр Храмичев, Александр Катулев, Цифровая обработка 2D слабоконтрастных изображений, формируемых оптико-электронным прибором в сложных фоновых условиях. Обнаружение, распознавание, сопровождение динамических объектов, М, Радиотехника, 2018, 4-8 с.

ПРИЛОЖЕНИЕ ФРАГМЕНТ ЛИСТИНГА ПРОГРАММНЫХ МОДУЛЕЙ

```
unit uAnalyser;

interface

uses Windows, Classes, Graphics, Unit_Integral, Unit_Fasthessian, Unit_Surf, Unit_IPoint,
  uADODBComponent, ADODB, SysUtils, DB, uSearchInfo;

type
  /// <summary>
  /// Класс, реализующий алгоритм подсчета
  /// количества деталей при помощи базы данных образов и алгоритма SURF
/// </summary>
  TAnalyser = class(TADODBComponent)
  protected
    /// <summary>
/// Радиус сферы для алгоритма Forel
    /// </summary>
    FRadius: Integer;
    /// <summary>
     /// Минимальное количество сходных точек, при котором образ считается идентифи-
цирован
/// </summary>
    FPointCountLimit: Integer;
    /// <summary>
    /// Историяклассификации
/// </summary>
    FAnalysisHistory: TStringList;
    /// <summary>
    /// Характеристические точки исходного изображения
    /// </summary>
    FBaseImagePoint: TStringList;
    /// <summary>
    /// Кластеры, образованные характеристическими точками исходного изображения
/// </summary>
    FBaseImageClusters: TStringList;
    /// <summary>
```

```
        /// Базовое изображение
    /// </summary>
        FBaseImage: TBitmap;
        /// <summary>
        /// Время вычисления характеристических точек исходного изображения
    /// </summary>
        FBaseImageDescriptionTime: Cardinal;
        /// <summary>
        /// Параметралгоритма SURF (октавы)
        /// </summary>
        FOctaves: Integer;
        /// <summary>
        /// Параметралгоритма SURF (интервалы)
        /// </summary>
        FIntervals: Integer;
        /// <summary>
        /// Параметралгоритма SURF (начальноезначение)
        /// </summary>
        FInitSample: Integer;
        /// <summary>
        /// Параметралгоритма SURF (шум)
    /// </summary>
        FTresh: Extended;
        /// <summary>
         /// Задает параметр "Минимальное количество сходных точек, при котором образ считается идентифицирован"
    /// </summary>
        procedure SetPointCountLimit(const Value: Integer);
    /// <summary>
        /// Задает параметр "Радиус сферы для алгоритма Forel"
    /// </summary>
        procedure SetRadius(const Value: Integer);
    /// <summary>
        /// Метод установки значения параметра SURF (октавы)
    /// </summary>
        procedure SetOctaves(Value: Integer);
```

```
/// <summary>
      /// Метод установки значения параметра SURF (интервалы)
/// </summary>
      procedure SetIntervals(Value: Integer);
/// <summary>
      /// Метод установки значения параметра SURF (начальное значение)
/// </summary>
      procedure SetInitSample(Value: Integer);
/// <summary>
      /// Метод установки значения параметра SURF (шум)
/// </summary>
      procedure SetTresh(Value: Extended);
/// <summary>
      /// Метод сохранения параметра в БД
/// </summary>
      procedure SetBaseParams(FieldName: string; Value: Variant);
/// <summary>
      /// Метод извлечения параметра из БД
/// </summary>
      function GetBaseParams(FieldName: string): Variant;
/// <summary>
      /// Метод кластеризации точек исходного изображения
/// </summary>
      procedure Clustering(SourcePoints: TStringList);
/// <summary>
      /// Устанавливает значение ClusterIndex для каждой точки кластера
/// </summary>
      procedure SetClusterIndexes;

   public
      /// <summary>
      /// Конструкторкласса.
      /// </summary>
      constructor Create(Connection: TADOConnection);
      /// <summary>
      /// Деструкторкласса
```

```
        /// </summary>
        destructor Destroy; override;
/// <summary>
        /// Метод загрузки базового изображения
        /// </summary>
/// <param name="Image">
        /// Входноеизображение
        /// </param>
        procedure SetBaseImage(Image: TBitmap);
/// <summary>
        /// Процедура идентификации.
        /// </summary>
        /// <returns>
        /// Номер класса объекта (код класса из базы данных)
        /// </returns>
        procedure Analysis;
        /// <summary>
        /// Функция получения характеристических точек по битмапу
/// </summary>
        function GetIPoints(B: TBitmap): TStringList;
/// <summary>
        /// Метод выгрузки картинки (образа) из базы данных
/// </summary>
        function GetObjectImage(ObjectKey: Integer): TBitmap;
/// <summary>
        /// Получение элемента истории по идентификатору объекта
/// </summary>
        function GetAnalysisHistoryForObjectID(ObjectKey: Integer): TSearchInfo;
/// <summary>
        /// Метод сохранения характеристических точек объекта
/// </summary>
        procedure SetObjectIPoints(ObjectKey: Integer; ObjectIPoints: TStringList);
/// <summary>
        /// Метод выгрузки характеристических точек объекта
/// </summary>
        function GetObjectIPoints(ObjectKey: Integer): TStringList;
```

```
/// <summary>
        /// Процедура перерасчета характеристических точек, хранящихся в БД
/// </summary>
        procedure ChangeObjectsIPoints;
        /// <summary>
        /// Историяклассификации
        /// </summary>
        property ClassificationHistory: TStringList read FAnalysisHistory;
        /// <summary>
        /// Базовоеизображение
        /// </summary>
        property BaseImage: TBitmap read FBaseImage;
/// <summary>
        /// Характеристические точки исходного изображения
        /// </summary>
property BaseImagePoint: TStringList read FBaseImagePoint;
/// <summary>
        /// Время вычисления характеристических точек исходого изображения
/// </summary>
        property BaseImageDescriptionTime: Cardinal read FBaseImageDescriptionTime;
        /// <summary>
        /// Параметралгоритма SURF (октавы)
        /// </summary>
        property Octaves: Integer read FOctaves write SetOctaves;
/// <summary>
        /// Параметр алгоритма SURF (интервалы)
/// </summary>
        property Intervals: Integer read FIntervals write SetIntervals;
/// <summary>
        /// Метод установки значения параметра SURF (начальное значение)
/// </summary>
        property InitSample: Integer read FInitSample write SetInitSample;
        /// <summary>
        /// Параметралгоритма SURF (шум)
        /// </summary>
        property Tresh: Extended read FTresh write SetTresh;
```

```
        /// <summary>
            /// Радиус сферы для алгоритма Forel
        /// </summary>
            property Radius: Integer read FRadius write SetRadius;
        /// <summary>
             /// Минимальное количество сходных точек, при котором образ считается идентифи-
цирован
        /// </summary>
            property PointCountLimit: Integer read FPointCountLimit write SetPointCountLimit;
        /// <summary>
            /// Кластеры, образованные характеристическими точками исходного изображения
        /// </summary>
            property BaseImageClusters: TStringList read FBaseImageClusters;

        end;

        implementation

        procedure TAnalyser.ChangeObjectsIPoints;
        var
          //
          I: Integer;
          Q: TADOQuery;
          P2: TStringList;
          Image2: TBitmap;
        begin
          //
          Q := GetQuery('select id_class, id from object_set order by id_class, id');
          try
            //
            for I := 1 to Q.RecordCount do
            begin
              //
              Q.RecNo := I;
              Image2 := GetObjectImage(Q.FieldByName('id').AsInteger);
              if Image2 = nil then
```

```
            Continue;
          try
            //
            P2 := GetIPoints(Image2);
          finally
            Image2.Free;
          end;
          SetObjectIPoints(Q.FieldByName('id').AsInteger, P2);
        end;
    finally
      Q.Free;
    end;
end;

procedure TAnalyser.SetBaseParams(FieldName: string; Value: Variant);
var
    Q: TADOQuery;
begin
    //
    Q := GetQuery('select * from params where id = 1');
    try
      //
      Q.Edit;
      Q.FieldByName(FieldName).AsVariant := Value;
      Q.Post;
    finally
      Q.Free;
    end;
end;

function TAnalyser.GetBaseParams(FieldName: string): Variant;
var
    Q: TADOQuery;
begin
    //
    Q := GetQuery('select * from params where id = 1');
```

```
    try
      //
      Result := Q.FieldByName(FieldName).AsVariant;
    finally
      Q.Free;
    end;
end;

procedure TAnalyser.SetObjectIPoints(ObjectKey: Integer; ObjectIPoints: TStringList);
var
    //
    Q: TADOQuery;
    MS: TMemoryStream;
    BF: TBlobField;
    procedure IPLSaveToStream(Points: TStringList; MemoryStream: TMemoryStream);
    var
      //
      I, PointsCount: Integer;
      IP: IPoint;

    begin
      //
      MemoryStream.Position := 0;
      PointsCount := Points.Count;
      MemoryStream.WriteBuffer(PointsCount, sizeof(Integer));
      for I := 0 to Points.Count - 1 do
      begin
        //
        IP := Points.Objects[I] as IPoint;
        IP.SaveToStream(MemoryStream);
      end;
    end;

begin
    //
    Q := Self.GetQuery('select * from object_set where id = ' + IntToStr(ObjectKey));
```

```
    try
      if Q.RecordCount = 0 then
        raise Exception.Create('Объектненайден');
      MS := TMemoryStream.Create;
      try
        //
        IPLSaveToStream(ObjectIPoints, MS);
        MS.Position := 0;
        BF := Q.FieldByName('points') as TBlobField;
        Q.Edit;
        BF.Clear;
        BF.LoadFromStream(MS);
        Q.Post;
      finally
        MS.Free;
      end;
    finally
      Q.Free;
    end;
end;

function TAnalyser.GetObjectIPoints(ObjectKey: Integer): TStringList;
var
    //
    MS: TMemoryStream;
    Q: TADOQuery;
    BF: TBlobField;
    L: TStringList;

    procedure IPLLoadFromStream(Points: TStringList; MemoryStream: TMemoryStream);
    var
      //
      I, PointsCount: Integer;
      IP: IPoint;
    begin
      //
```

```
      MemoryStream.Position := 0;
      MemoryStream.ReadBuffer(PointsCount, sizeof(Integer));
      for I := 0 to PointsCount - 1 do
      begin
        //
        IP := IPoint.Create;
        IP.LoadFromStream(MemoryStream);
        Points.AddObject('', IP);
      end;
    end;

begin
    Result := nil;
    Q := Self.GetQuery('select * from object_set where id = ' + IntToStr(ObjectKey));
    try
      if Q.RecordCount = 0 then
        raise Exception.Create('Объектненайден');
      MS := TMemoryStream.Create;
      try
        //
        BF := Q.FieldByName('points') as TBlobField;
        if BF.IsNull = false then
        begin
          //
          BF.SaveToStream(MS);
          MS.Position := 0;
          Result := TStringList.Create;
          try
            //
            Result := TStringList.Create;
            IPLLoadFromStream(Result, MS);
          except
            Result.Free;
            raise;
          end;
        end;
```

```
            finally
              MS.Free;
            end;
          finally
            Q.Free;
          end;
        end;

        procedure TAnalyser.SetOctaves(Value: Integer);
        begin
          //
          if (Value > 0) and (Value <= 4) then
            FOctaves := Value
          else
            raise Exception.Create('Значениеоктавывыходитзапределыдопустимогодиапазона');
          SetBaseParams('octaves', FOctaves);
        end;

        procedure TAnalyser.SetPointCountLimit(const Value: Integer);
        begin
          if (Value <= 0) then
              raise Exception.Create('Значение количества точек выходит за пределы допустимого
диапазона');
        FPointCountLimit := Value;
          SetBaseParams('point_count_limit', FPointCountLimit);
        end;

        procedure TAnalyser.SetRadius(const Value: Integer);
        begin
          if (Value <= 0) then
            raise Exception.Create('Значениерадиусавыходитзапределыдопустимогодиапазона');
          FRadius := Value;
          SetBaseParams('radius', FRadius);
        end;

        procedure TAnalyser.SetIntervals(Value: Integer);
```

```
begin
  //
  if (Value > 0) and (Value <= 4) then
    FIntervals := Value
  else
    raise Exception.Create('Значениеинтервалавыходитзапределыдопустимогодиапазона');
  SetBaseParams('intervals', FIntervals);
end;

procedure TAnalyser.SetInitSample(Value: Integer);
begin
  //
  if (Value > 0) and (Value <= 6) then
FInitSample := Value
  else
    raise Exception.Create('Начальное значение выходит за пределы допустимого диапазо-
на');
SetBaseParams('init_sample', FInitSample);
end;

procedure TAnalyser.SetTresh(Value: Extended);
begin
  //
  if (Value >= 0) then
    FTresh := Value
else
    raise Exception.Create('Значение уровня шума выходит за пределы допустимого диапа-
зона');
SetBaseParams('trash', FTresh);
end;

constructor TAnalyser.Create(Connection: TADOConnection);
begin
  //
  inherited Create(Connection);
  FAnalysisHistory := TStringList.Create;
```

```
    FAnalysisHistory.OwnsObjects := True;
    FBaseImageClusters := TStringList.Create;
    FBaseImageClusters.OwnsObjects := True;

    FBaseImagePoint := nil;
    FBaseImage := TBitmap.Create;
    FBaseImageDescriptionTime := 0;
    FOctaves := GetBaseParams('octaves');
    FIntervals := GetBaseParams('intervals');
    FInitSample := GetBaseParams('init_sample');
    FTresh := GetBaseParams('trash');
    FRadius := GetBaseParams('radius');
    PointCountLimit := GetBaseParams('point_count_limit');

end;

destructor TAnalyser.Destroy;
begin
    //
    FAnalysisHistory.Free;

    FBaseImage.Free;
    FBaseImageClusters.Free;
    inherited Destroy;
end;

procedure TAnalyser.SetBaseImage(Image: TBitmap);
var
    //
    T: Cardinal;
begin
    //
    FBaseImage.Assign(Image);
    if FBaseImagePoint <> nil then
        FBaseImagePoint.Free;
    T := GetTickCount;
```

```
    FBaseImagePoint := GetIPoints(FBaseImage);
    Clustering(FBaseImagePoint);
    SetClusterIndexes;
    FBaseImageDescriptionTime := GetTickCount - T;
end;

procedure TAnalyser.Analysis;
var
    //
    I, J: Integer;
    Q: TADOQuery;
    P1, P2, Matches: TStringList;
    Image2: TBitmap;
    MatchesCount: Integer;
    SI: TSearchInfo;
    T: Cardinal;

begin
    //

    if FBaseImagePoint = nil then
        raise Exception.Create('Базовоеизображениенеподгружено');

    Image2 := nil;
    P2 := nil;
    Matches := nil;
    FAnalysisHistory.Clear;
    Q := GetQuery('select id_class, id from object_set order by id_class, id');
    try
        //
        for I := 1 to Q.RecordCount do
        begin
            //
            T := GetTickCount;
            Q.RecNo := I;
            P2 := GetObjectIPoints(Q.FieldByName('id').AsInteger);
```

```
      if P2 = nil then
      begin
        Image2 := GetObjectImage(Q.FieldByName('id').AsInteger);
        if Image2 = nil then
          Continue;
        try
          //
          P2 := GetIPoints(Image2);
        finally
          Image2.Free;
        end;
        SetObjectIPoints(Q.FieldByName('id').AsInteger, P2);
      end;

      Matches := TStringList.Create;
      try
        Matches.OwnsObjects := True;
        SI := TSearchInfo.Create;
        SI.ObjectID := Q.FieldByName('id').AsInteger;
        FAnalysisHistory.AddObject('', SI);

        for J := 0 to FBaseImageClusters.Count - 1 do
        begin
          P1 := FBaseImageClusters.Objects[J] as TStringList;
          GetMatches(P1, P2, Matches);
          if Matches.Count >= PointCountLimit then
          begin
            SI.Similarity := SI.Similarity + 1;
          end;
        end;
        SI.ComputationTime := GetTickCount - T;
      finally
        Matches.Free;
      end;
    end;
  finally
```

```
      Q.Free;
    end;

  end;

  procedure TAnalyser.Clustering(SourcePoints: TStringList);
  var
    Radius: Extended;

    Points, NeighbourPoints: TStringList;

    CurrentPoint, CenterPoint: IPoint;
    X, Y: Extended;

    function GetRandomPoint: IPoint;
    begin
      Result := Points.Objects[Random(Points.Count)] as IPoint;
    end;

    function GetNeighbourPoints(Point: IPoint): TStringList;
    var
      I: Integer;
      P: IPoint;
    begin
      Result := TStringList.Create;
      try
        for I := 0 to Points.Count - 1 do
        begin
          P := Points.Objects[I] as IPoint;
          if Sqrt((P.X - Point.X) * (P.X - Point.X) + (P.Y - Point.Y) * (P.Y - Point.Y)) <= Radius
then
          begin
            Result.AddObject(IntToStr(I), P);
          end;
        end;
      except
```

```
      Result.Free;
      raise;
    end;
  end;

  function GetCenterOfPoints(Points: TStringList): IPoint;
  var
    I, J: Integer;
    P1, P2: IPoint;

    DeltaSum, Delta: Extended;
  begin
    Delta := MaxInt;

    for I := 0 to Points.Count - 1 do
    begin
      P1 := Points.Objects[I] as IPoint;
      for J := 0 to Points.Count - 1 do
      begin
        P2 := Points.Objects[J] as IPoint;
        // if I <> J then
        begin
          DeltaSum := Sqrt((P1.X - P2.X) * (P1.X - P2.X) + (P1.Y - P2.Y) * (P1.Y - P2.Y));
        end;
      end;
      if Delta > DeltaSum then
      begin
        Delta := DeltaSum;
        Result := P1;
      end;
    end;
  end;

  procedure DeletePoints(NeighbourPoints: TStringList);
  var
    I: Integer;
```

```
      P: IPoint;
    begin
      for I := NeighbourPoints.Count - 1 downto 0 do
      begin
        Points.Delete(StrToInt(NeighbourPoints.Strings[I]));
      end;
    end;

  begin
    FBaseImageClusters.Clear;
    Radius := Self.Radius;
    Points := TStringList.Create;
    try
      Points.AddStrings(SourcePoints);

      while Points.Count > 0 do
      begin
        CurrentPoint := GetRandomPoint;
        NeighbourPoints := GetNeighbourPoints(CurrentPoint);
        CenterPoint := GetCenterOfPoints(NeighbourPoints);

        while (CurrentPoint <> CenterPoint) do
        begin
          CurrentPoint := CenterPoint;
          NeighbourPoints.Free;
          NeighbourPoints := GetNeighbourPoints(CurrentPoint);
          CenterPoint := GetCenterOfPoints(NeighbourPoints);
        end;
          FBaseImageClusters.AddObject(IntToStr(FBaseImageClusters.Count + 1), Neighbour-
Points);
        DeletePoints(NeighbourPoints);
      end;

    finally
      Points.Free;
    end;
```

```
end;

procedure TAnalyser.SetClusterIndexes;
var
   I, J: Integer;
   Points: TStringList;
   P: IPoint;
begin
   for I := 0 to FBaseImageClusters.Count - 1 do
   begin
      Points := FBaseImageClusters.Objects[I] as TStringList;
      for J := 0 to Points.Count - 1 do
      begin
         P := Points.Objects[J] as IPoint;
         P.ClusterIndex := I + 1;
      end;
   end;
end;

function TAnalyser.GetAnalysisHistoryForObjectID(ObjectKey: Integer): TSearchInfo;
var
   //
   I: Integer;
   SI: TSearchInfo;
begin
   //
   Result := nil;
   for I := 0 to FAnalysisHistory.Count - 1 do
   begin
      //
      SI := FAnalysisHistory.Objects[I] as TSearchInfo;
      if SI.ObjectID = ObjectKey then
      begin
         //
         Result := SI;
```

```
            Exit;
        end;
    end;
end;

function TAnalyser.GetObjectImage(ObjectKey: Integer): TBitmap;
var
    //
    Q: TADOQuery;
    F: TBlobField;
    MS: TMemoryStream;
begin
    //
    Result := nil;
    Q := Self.GetQuery('select * from object_set where id = ' + IntToStr(ObjectKey));
    try
        //
        if Q.RecordCount = 0 then
            raise Exception.Create('Объектненайден');
        F := Q.FieldByName('image') as TBlobField;
        if F.IsNull = false then
        begin
            //
            MS := TMemoryStream.Create;
            try
                //
                F.SaveToStream(MS);
                MS.Position := 0;
                Result := TBitmap.Create;
                Result.LoadFromStream(MS);
            finally
                //
                MS.Free;
            end;
        end;
    finally
```

```
      Q.Free;
    end;
end;

function TAnalyser.GetIPoints(B: TBitmap): TStringList;
var
    FastHessian: TFastHessian;
    IntegralImage: TIntegralImage;
    Surf: TSurf;
begin
    //
    try
        Result := nil;
        FastHessian := nil;
        Surf := nil;

        Result := TStringList.Create;
        Result.OwnsObjects := True;
        IntegralImage := TIntegralImage.Create(B);
        FastHessian := TFastHessian.Create(Result, FOctaves, FIntervals, FInitSample, FTresh);
        FastHessian.SetIntImage(IntegralImage);
        FastHessian.GetIPoints;
        Surf := TSurf.Create(IntegralImage, Result);
        Surf.GetDescriptors;

        if Surf <> nil then
            Surf.Free;
        if IntegralImage <> nil then
            IntegralImage.Free;
        if FastHessian <> nil then
            FastHessian.Free;
    except
        //
        if Surf <> nil then
            Surf.Free;
        if IntegralImage <> nil then
```

```
            IntegralImage.Free;
        if FastHessian <> nil then
            FastHessian.Free;
        if Result <> nil then
            Result.Free;
        raise;
    end;
end;

end.
```