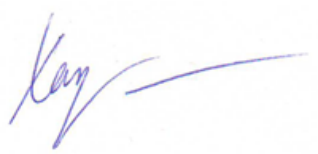
Федеральное государственное автономное образовательное учреждение
высшего образования
«Национальный исследовательский Томский политехнический университет»

На правах рукописи



Хаустов Павел Александрович

АЛГОРИТМЫ РАСПОЗНАВАНИЯ РУКОПИСНЫХ СИМВОЛОВ В УСЛОВИЯХ МАЛОЙ ОБУЧАЮЩЕЙ ВЫБОРКИ

05.13.11 – Математическое и программное обеспечение
вычислительных машин, комплексов и компьютерных сетей

Диссертация
на соискание ученой степени
кандидата технических наук

Научный руководитель
доктор технических наук, профессор
Спицын Владимир Григорьевич

Томск – 2017

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность работы. На сегодняшний день алгоритмы распознавания символов обеспечивают решение ряда научных и прикладных задач, возникающих в процессе извлечения текстовой информации из печатных и рукописных документов [1, 2]. Разработано множество методов распознавания символов [3, 4, 5], среди которых: методы на основе признаковых классификаторов (искусственные нейронные сети, метод опорных векторов, самоорганизующиеся карты Кохонена и др.); статистические методы (подходы с построением гистограмм, метод пересечений, методы на основе зонного описания и др.); методы на основе выделения структурных составляющих, которые основаны на выделении определённых геометрических свойств начертания символа и последующем построении модели, описывающей символ с применением выделенных геометрических свойств [6, 7].

Для создания универсального классификатора начертаний символов наиболее эффективным на сегодня является применение свёрточных нейронных сетей (впервые описаны в работах Y. LeCun [8]), которые с использованием достаточно объёмных обучающих выборок позволяют решить и задачу распознавания, и задачу выработки абстрактных признаков, по которым будет выполняться классификация.

Большой вклад в развитие теории и практики распознавания символов внесли зарубежные учёные, среди которых стоит отметить: S.W. Lee [9], A. Krzyzak и C.Y. Suen [10], Y. LeCun [11], Z. Ping [12], X.R. Chen [13], S. Knerr [14], Y.Lee [15], G.L. Martin и J.A. Pitman [16], L. Seong-Wang и J.K. Young [17], C. Kan [18], A. Krizhevsky [19], C. Cortes [20], S.O. Haykin [21] и другие. Существенный вклад в эту научную область внесли Российские учёные: В.Н. Вапник и А.Я. Червоненкис [22, 23], Ю.П. Пытьев [24, 25], Н.Г. Загоруйко [26, 27], Ю.И. Журавлев [28, 29], А.П. Коробейников [30], В.В. Круглов [21], В.В. Щепин [31],

Ю.Р. Цой [32, 33], Л.М. Местецкий [34], С.С. Садыков [35], В.Л. Арлазоров [36, 37], М.Н. Фаворская [38, 39] и другие.

Все существующие универсальные походы к решению задачи распознавания символов ориентированы на применение опорной базы эталонных изображений, которая для высокой точности распознавания должна быть достаточно велика [40]. В то же время существует ряд задач, в которых количество изначально известных начертаний символов крайне невелико. Примерами таких задач являются: распознавание заполненных нетипичным почерком бланков аттестации, выделение текстовой информации на имеющихся в единственном экземпляре исторических документах, идентификация подписей в банковских документах, идентификации пользователя по рукописной подписи [41, 42].

В многочисленных научных трудах (K.-F. Chan, E. Kavallieratou, М.Н. Фаворская и др.) для распознавания символов применяются методы на основе выделения структурных составляющих. Такие методы опираются на выделение определённых геометрических характеристик изображений символов и последующее построение структурных моделей символов на их основе [43]. В настоящее время для оценки степени схожести структурных моделей используется ряд критериев схожести, например, критерии на основе внутренних расстояний (H. Ling), дистанционного преобразования (S. Hezel), контекстов форм (S. Belongie). На практике использование таких критериев приводит к ошибочным результатам из-за разрывов, лишних слияний и ложных циклов. Поэтому при применении структурных моделей достаточно актуальной является задача выбора критерия схожести, обеспечивающего наибольшую точность распознавания.

Для построения структурной модели символа требуется выполнить предварительную скелетизацию его начертания. Точные методы скелетизации достаточно требовательны к вычислительным ресурсам. Как следствие, обработка изображения символа такими алгоритмами может существенно замедлить процесс распознавания. В то же время

существующие производительные однопроходные алгоритмы для дискретных изображений символов, наиболее известные из которых описаны в работах T.Y. Zhang и C.Y. Suen, A. Rosenfeld, S.N. Srihari, не всегда обеспечивают конфигурацию скелета, пригодную для корректного выделения особых точек. Например, такие алгоритмы допускают наличие на результирующем скелете помех и граничных точек, не являющихся особыми, чего в большинстве случаев не допускают многопроходные алгоритмы скелетизации. Вследствие этого возникает необходимость в разработке алгоритма скелетизации, обладающего высоким быстродействием и не имеющего перечисленных ранее недостатков однопроходных алгоритмов.

Особую сложность в условиях малого количества эталонных изображений представляет задача сегментации рукописного текста. В настоящее время для её решения применяются признаковые классификаторы, но в условиях малой эталонной выборки выполнить качественное обучение таких классификаторов не представляется возможным из-за того, что вместо необходимых для обучения лигатур (пар смежных символов) среди эталонных изображений имеются только разрозненные символы. Общеизвестные методы без применения признаковых классификаторов опираются на использование вертикальных линий для разделения сегментов в слове, тем самым ограничивая возможность корректной сегментации текста, написанного почерком с большим наклоном. Поэтому возникает необходимость в разработке алгоритма сегментации рукописного текста в условиях малой обучающей выборки.

В этой связи проблема разработки алгоритмов распознавания символов, основанных на построении структурных моделей и способных функционировать в условиях малой выборки эталонных изображений, является актуальной.

Объектом исследования являются алгоритмы распознавания рукописных символов в условиях малой обучающей выборки.

Предмет исследования: недостаточно высокая точность распознавания рукописных символов в условиях малой обучающей выборки.

Целью диссертационной работы является разработка алгоритмов распознавания рукописных символов в условиях малой обучающей выборки.

Для достижения поставленной цели необходимо последовательное решение **следующих задач**:

1. Исследовать существующие методы распознавания печатных и рукописных символов.
2. Разработать структурную модель символа для применения в решении задачи распознавания рукописных символов в условиях малой обучающей выборки.
3. Разработать алгоритм построения предложенной структурной модели символа по растровому представлению его начертания.
4. Выбрать критерии схожести структурных моделей символов.
5. Реализовать алгоритмы распознавания рукописных символов в условиях малой обучающей выборки на основе применения предложенной структурной модели символа и выбранных критериев схожести.
6. Реализовать комплекс программ для исследования и сравнительного анализа разработанных и существующих алгоритмов распознавания символов в условиях малой обучающей выборки, и провести вычислительные эксперименты с целью оценки качества и эффективности разработанных алгоритмов.

Апробация результатов работы. Результаты работы были представлены на следующих конференциях и семинарах: The Fifth International Workshop on Mathematical Models and their Applications (Красноярск, 2016), The 11th International Forum on Strategic Technology IFOST (Новосибирск, 2016); II, III, IV Международная научная конференция «Информационные технологии в науке, управлении, социальной сфере и медицине» (Томск, 2014, 2015, 2016); XIII, XIV, XV Международная научно-практическая конференция студентов, аспирантов и молодых ученых

«Молодежь и современные информационные технологии» (Томск, 2013, 2014, 2015); XIX, XXI Международная научная конференция студентов и молодых учёных «Современные техника и технологии» (Томск, 2013, 2015).

Публикации. По теме диссертационной работы опубликовано 16 работ; из них 5 статей в периодических изданиях из перечня ВАК [129-133]; 3 публикации в научных изданиях, индексируемых в базе данных Scopus [129, 135, 136]; 1 статья в рецензируемом журнале [137]; 9 докладов на всероссийских и международных конференциях [135, 136, 138-144]; одно свидетельство РФ о государственной регистрации программ для электронных вычислительных машин [134].

Основное содержание работы

В **первой главе** представлен аналитический обзор ранее предложенных методов и алгоритмов, применяемых для распознавания рукописных и печатных символов. Проанализированы их достоинства и недостатки, на основании чего сделан выбор в пользу группы методов на основе структурных составляющих. Были также рассмотрены алгоритмы предварительной обработки изображений символов.

Во **второй главе** приводится подробное описание всех стадий предложенного алгоритма построения структурной модели: предварительной обработки входного изображения, выделения структурных составляющих, составления структурной модели символа на основе выявленных составляющих. Приводятся три критерия схожести двух ранее построенных структурных моделей символов. Предложен алгоритм на основе комбинации двух существующих алгоритмов скелетизации с использованием собственной операции устранения плоских окончаний. Предложен собственный алгоритм выделения структурных составляющих и последующего построения модели на их основе. Предложен оригинальный алгоритм сегментации рукописного текста на основе разработанных алгоритмов распознавания рукописных символов с применением метода динамического программирования.

В **третьей главе** приведено описание программной реализации описанных алгоритмов. Представлен обзор библиотек для обработки изображений. Приведено обоснование выбора языка программирования и средств для программной реализации алгоритмов. Представлено описание разработанного программного обеспечения.

В **четвертой главе** представлены результаты тестирования разработанных алгоритмов для решения задачи распознавания рукописных символов. Представлены результаты тестирования на общеизвестном наборе рукописных цифр MNIST и наборе рукописных символов из бланков тестовых заданий, аналогичных заданиям единого государственного экзамена для школьников. Проведен сравнительный анализ результатов работы предложенных алгоритмов с существующими алгоритмами, решающими схожие задачи, в условиях малого количества эталонных изображений.

Научной новизной обладают следующие результаты:

1. Предложен алгоритм скелетизации бинарных изображений символов на основе комбинированного подхода с применением операции предварительного устранения плоских окончаний начертания символа и алгоритмов скелетизации Зонга-Суня и Ву-Цая, обладающий высоким быстродействием и позволяющий получить скелетизированное представление начертания символа без удаления таких его важных элементов, как скошенные угловые элементы, закругления, засечки, декоративные элементы начертания.

2. Предложена новая структурная модель символа, отличающаяся от известных графовых моделей принципом разделения ключевых точек и изгибов, группировкой соединяющих рёбер в композитные, дополнительными метками точек и рёбер, позволяющая описать топологию и геометрическую форму его начертания за счёт обобщения схожих по форме представления рёбер.

3. Предложен алгоритм построения структурной модели символа, позволяющий выделить структурные составляющие его начертания (ключевые точки, изгибы, соединяющие и композитные рёбра), отличающийся от известных отсутствием необходимости применения дополнительных итераций алгоритма Ли для определения геометрических характеристик выделенных структурных составляющих.

4. Предложен оригинальный критерий схожести структурных моделей символов, отличающийся от аналогов применением перехода от геометрического представления моделей к задаче нахождения максимального паросочетания наименьшего веса, позволяющий существенно повысить точность распознавания символов в условиях малой обучающей выборки.

5. Предложен оригинальный алгоритм сегментации рукописного текста, позволяющий решать задачу сегментации текста в условиях малой обучающей выборки, отличающийся от аналогов высокой устойчивостью по отношению к наклону символов и отсутствием необходимости использования изображений лигатур для настройки.

Научную ценность работы представляет вклад в развитие алгоритмических средств распознавания рукописных символов в условиях малой обучающей выборки, заключающийся в предложенном многоэтапном алгоритме распознавания рукописных символов, включающем стадии предварительной обработки изображения, выделения структурных составляющих четырех типов, построения структурной модели символа на основе этих составляющих, а также критерии схожести структурных моделей символов. Предложенные методы обладают высоким быстродействием и обеспечивают возможность распознавания рукописных символов с различными особенностями начертания. Главной особенностью предложенных методов является возможность качественной классификации при малом количестве эталонных изображений символов, которого недостаточно для полноценного обучения признаковых классификаторов.

Теоретическая значимость. Разработанные алгоритмы имеют самостоятельное значение и, помимо задачи распознавания символов, могут применяться для решения задач классификации отпечатков пальцев, идентификации почерка, проверки подписей на подлинность и других задач, связанных с анализом бинарных изображений.

Предложенная структурная модель символа может применяться в уже существующих алгоритмах распознавания символов. Аналогично, предложенный критерий схожести может быть использован для других структурных моделей символов. Кроме того, предложенную структурную модель символа можно дополнить информацией о толщине линий и порядке выполнения графических элементов, в результате чего она может быть успешно применена для идентификации пользователя по рукописной подписи.

Практическая значимость. Предложенные алгоритмы и программное обеспечение позволяют эффективно решать задачу распознавания символов в условиях малого количества эталонных изображений, при которых применение универсальных подходов на основе признаковых классификаторов существенно осложняется. Такие условия могут возникнуть при обработке бланков аттестации из-за необходимости учета индивидуальных особенностей почерка или при извлечении текстовой информации из отсканированного изображения, где используется авторский шрифт, существенно отличающийся от общеизвестных.

Апробация реализованных алгоритмических средств была осуществлена на данных из общеизвестной базы изображений рукописных символов.

Разработанные алгоритмы и программные средства могут быть использованы в системах автоматической обработки бланков, проверки тестовых заданий, анализа почерка и могут применяться в социальных организациях и образовательных учреждениях для автоматической обработки большого количества бланков, заполненных вручную.

Методы исследования. Для решения поставленных задач использованы методы искусственного интеллекта, теории графов, вычислительной геометрии, компьютерной графики, алгоритмы цифровой обработки изображений, технология разработки программного обеспечения, а также методы теории вероятностей и математической статистики для обработки результатов численных экспериментов.

Личный вклад. Представленные в диссертационной работе теоретические и практические результаты получены лично автором. В работах, опубликованных в соавторстве с сотрудниками научной группы, диссертант принимал непосредственное участие в разработке и реализации алгоритмов, а также в экспериментальных исследованиях. Постановка задачи диссертационного исследования осуществлялась автором совместно с научным руководителем, д.т.н., профессором В.Г. Спицыным.

Основные положения, выносимые на защиту:

1. Предложен алгоритм скелетизации бинарных изображений символов на основе комбинированного подхода с применением операции предварительного устранения плоских окончаний начертания символа и алгоритмов скелетизации Зонга-Суня и Ву-Цая, сохраняющий информацию о форме начертания символа.
2. Предложена новая структурная модель символа, позволяющая описать топологию и геометрическую форму его начертания за счёт обобщения схожих по форме представления рёбер.
3. Предложен алгоритм построения структурной модели символа, позволяющий выделить структурные составляющие его начертания: ключевые точки, изгибы, соединяющие и композитные рёбра.
4. Предложен оригинальный критерий схожести структурных моделей символов, позволяющий существенно повысить точность распознавания символов в условиях малой обучающей выборки.

5. Предложен оригинальный алгоритм сегментации рукописного текста, позволяющий решать задачу сегментации текста в условиях малой обучающей выборки.

6. Разработан комплекс программ, позволяющий выполнять исследование и сравнительный анализ разработанных и существующих алгоритмов распознавания символов в условиях малой обучающей выборки.

Автор выражает большую благодарность своему научному руководителю профессору, доктору технических наук В.Г. Спицыну за оказание помощи в написании диссертационной работы, ценные замечания и советы, а также за конструктивную критику. Автор благодарит профессора, доктора технических наук Н.Г. Маркова за ценные замечания и обсуждения работы. Выражается благодарность за ценные замечания и всестороннюю помощь кандидатам технических наук: Ю.Р. Цою, А.А. Белоусову, А.А. Друки, А.В. Кудинову, Ю.Я. Кацману, кандидатам физико-математических наук Ю.Б. Буркатовской и Е.С. Сергеевой, а также всем членам научной группы профессора В.Г. Спицына.

Степень достоверности результатов проведенных экспериментов подтверждается результатами численных экспериментов на тестовых задачах различного вида и согласованностью результатов диссертационной работы с результатами других авторов.

Внедрение работы. Результаты диссертационной работы внедрены в Национальном исследовательском Томском политехническом университете на кафедре Информационных систем и технологий при подготовке курса «Методы распознавания образов» для обучения специалистов по магистерской программе «Компьютерный анализ и интерпретация данных»; в ООО «Рубиус Групп» для реализации технологических задач в системе автоматической обработки бланков и анкет.

Реализация результатов работы. Методы, алгоритмы и программные средства, разработанные в диссертационной работе, использовались при выполнении проекта «Создание комплексных технологий распознавания

объектов на изображениях на основе применения моделей зрительного восприятия и методов вычислительного интеллекта», поддержанного грантом РФФИ № 12-08-00296 (2012–2014 гг.).

Структура и объем работы. Диссертация включает в себя введение, четыре главы, заключение, список использованных источников и литературы, содержащий 144 наименования. Общий объём диссертационной работы составляет 181 страницу машинописного текста, 80 рисунков и 30 таблиц.

ГЛАВА 1. АНАЛИТИЧЕСКИЙ ОБЗОР МЕТОДОВ РАСПОЗНАВАНИЯ СИМВОЛОВ

В данной главе представлен аналитический обзор ранее предложенных методов и алгоритмов, применяемых для распознавания символов. Проанализированы их достоинства и недостатки, на основании чего сделан выбор в пользу группы методов на основе структурных составляющих. Были рассмотрены алгоритмы предварительной скелетизации бинарных изображений символов, которые широко применяются для решения задачи распознавания рукописных символов.

1.1. Основные особенности задачи распознавания символов

Задача распознавания символов несколько проще задачи классификации произвольных образов на изображениях. Как правило, решение задачи распознавания символов не подразумевает необходимость отделения границ графического представления символа от фона. Если считать, что каждый из пикселов анализируемого изображения принадлежит либо графическому представлению символа, либо простому фону, то количество возможных представлений символа существенно меньше, чем количество возможных представлений реального объекта, например, на фотоснимках. Однако из-за того, что задача распознавания имеет ряд существенных упрощений по отношению к задачам распознавания более сложных объектов, требования к алгоритмам и методам решения этой задачи существенно выше.

1.2. Различия задач распознавания рукописных и печатных символов

В отличие от печатных символов, рукописные символы имеют намного большее количество графических представлений. Один и тот же символ может быть начертан разными людьми, каждый из которых обладает

собственным почерком. Кроме того, могут варьироваться материал, на котором выполняется начертание, и инструмент, которым оно выполняется.

Человеческий мозг способен отличать рукописные символы по некоторым особенностям их графического представления и априорной информации о возможных способах начертания определенных видов символов. Такая информация накапливается в памяти человека на протяжении всей его жизни, в то время как автоматизированная система гораздо более ограничена в объемах используемой памяти и времени, которое требуется ей для настройки или обучения.

1.3. Существующие методы распознавания символов

В данном разделе проводится обзор и анализ существующих подходов к распознаванию, как рукописных, так и печатных символов.

Существующие подходы можно разделить на следующие категории:

- методы с использованием признаковых классификаторов;
- статистические методы;
- методы с использованием структурных составляющих.

Первая группа методов является одной из наиболее широко распространенных в научной среде. При разработке таких методов можно абстрагироваться от деталей процесса сравнения образов, возложив эту необходимость на один из математических инструментов для классификации. Достаточно широкое распространение получили в задачах распознавания не только символов, но и образов в целом искусственные нейронные сети. В последнее время для подобных задач используются машины опорных векторов. На сегодня во всемирной сети Интернет в свободном доступе существует огромное количество библиотек, реализующих те или иные признаковые классификаторы. С учетом этого, процесс реализации становится сравнительно нетрудоемким, а решение задачи распознавания сводится к правильному выбору пространства

признаков и репрезентативной обучающей и тестовой выборки образов. Огромным недостатком этой группы методов является отсутствие возможности предоставить удобный для восприятия человеком аргументированный отчет о том, почему образ отнесен конкретно к тому или иному классу.

Вторая группа методов так же, как и описанная ранее, достаточно проста в плане программной реализации. Исходное графическое представление образа анализируется с целью выявления различных статистических величин, строятся гистограммы, считаются средние значения и среднеквадратичные отклонения в определенных областях. После чего полученные данные используются для выполнения классификации. Для классификации может использоваться, как простой признаковый классификатор, так и эвристический подход с использованием обнаруженных закономерностей. Гистограммы гораздо более наглядны для человеческого восприятия и могут быть полезны в процессе отладки или улучшения алгоритма.

Третья группа методов отличается тем, что поиск закономерностей и зависимостей графических представлений символов возлагается на разработчика метода. На графическом представлении символа выделяются ключевые элементы – структурные составляющие. Они могут быть представлены, например, графическими примитивами или последовательностью действий оператора при начертании символа. Извлечь из графического начертания подобные составляющие – достаточно трудоемкая задача, но еще более трудной является задача их использования для сравнения двух символов между собой. Как разработка, так и программная реализация таких методов занимает, как правило, существенное количество времени. Однако, подобные алгоритмы, имеют более высокую точность распознавания в задачах, где имеется одна или несколько эталонных форм графического представления символа. При разработке алгоритмов этой группы приходится решать задачу нахождения «золотой

середины» между достаточно точным представлением разбиения на структурные составляющие и не слишком сложным представлением этого разбиения. Ведь в таком случае процесс сравнения может оказаться чересчур требовательным к вычислительным мощностям.

1.3.1. Методы распознавания символов с использованием признаковых классификаторов

Искусственные нейронные сети (ИНС) нашли широкое применение в задачах распознавания символов [44]. ИНС – математическая модель, представляющая из себя множество искусственных нейронов, соединенных между собой с помощью синаптических связей. Нейроны сгруппированы по слоям. Как правило, выделяют входной, выходной и скрытые слои. На вход ИНС поступает вектор признаков, описывающий определенный образ. Размерность пространства признаков равна количеству нейронов во входном слое. Выходной слой содержит количество нейронов, необходимое для хранения информации о результате классификации [45]. Зачастую для каждого возможного класса в выходном слое выделяется отдельный нейрон. Обучением ИНС называется этап настройки весов ее синаптических связей. Стадия обучения, фактически, является стадией построения признакового классификатора на основе ИНС.

Входной вектор признаков для классификатора на основе ИНС зачастую получается с использованием вейвлет-преобразования [46, 47, 48], которое является сверткой вейвлет-функции с исходным сигналом [49].

В статье научной группы из Индии в составе *V.N. Manjunath Aradhya*, *S.K. Niranjan* и *G. Hemantha Kumar* в 2010 году предложено использование вероятностной нейронной сети для распознавания рукописных символов [50]. Вероятностная нейронная сеть или, сокращенно, PNN-сеть – одна из разновидностей радиально-базисных сетей [51].

Вероятностная нейронная сеть имеет три слоя: входной, радиальный и выходной. Входной слой имеет произвольную размерность. Каждый нейрон радиального слоя соответствует одному элементу обучающей выборки. Количество нейронов выходного слоя равняется количеству классов. Каждый нейрон выходного слоя соединен с элементами радиального слоя, принадлежащими соответствующему ему классу (рисунок 1.1).

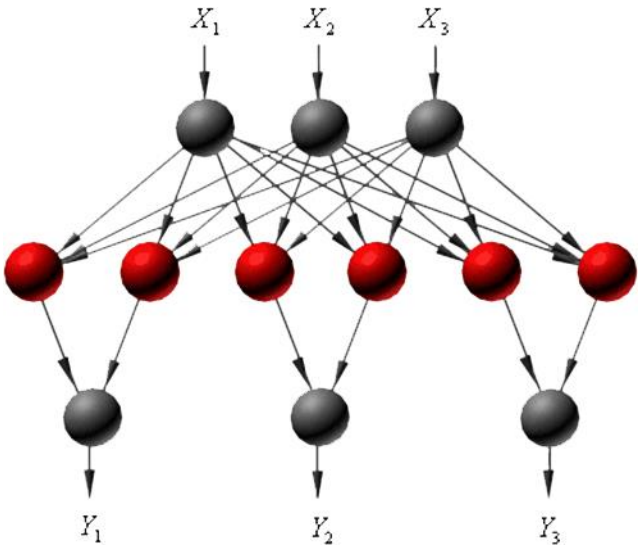


Рисунок 1.1 – Структура вероятностной нейронной сети

По своей сути принцип работы PNN-сети основывается на так называемых ядерных оценках [52, 53]. Для оценки плотности вероятности принадлежности вектора признаков некоторому классу используется информация о наличии в непосредственной близости к этому вектору признаков векторов, соответствующих этому классу в радиальном слое вероятностной нейронной сети.

Для получения вектора признаков графических изображений символов используется дискретное преобразование Фурье. После этого полученный вектор подается на входной слой вероятностной нейронной сети. Нейрон выходного слоя с наибольшим значением радиальной функций соответствует классу, к которому следует отнести изображение.

К существенным преимуществам такого алгоритма можно отнести вероятностный смысл выходных значений искусственной нейронной сети [54]. Такие значения впоследствии могут быть использованы для

улучшения качества классификации с учетом определенной априорной информации, например, словаря [133, 130, 55, 141].

Результаты ученых из Индии для букв английского алфавита сильно зависят от количества изображений в обучающей выборке. Если использовать 175 примеров изображений каждого класса, то процент правильно распознанных символов находится в интервале от 90% до 95%.

Такую сильную зависимость результатов распознавания от количества элементов в обучающей выборке можно объяснить наиболее высоким качеством аппроксимации плотности вероятности при более высоком числе элементов в обучающей выборке. Стоит также отметить, что быстродействие PNN-сети линейно зависит от размера радиального слоя, и, соответственно, от размера обучающего набора изображений. Таким образом, за более высокое качество распознавания приходится «расплачиваться» менее высоким быстродействием.

Такой высокий процент распознавания рукописных символов можно связать и с тем, что большая часть элементов обучающей и тестовой выборки была выполнена одним почерком без излишних деформаций начертания.

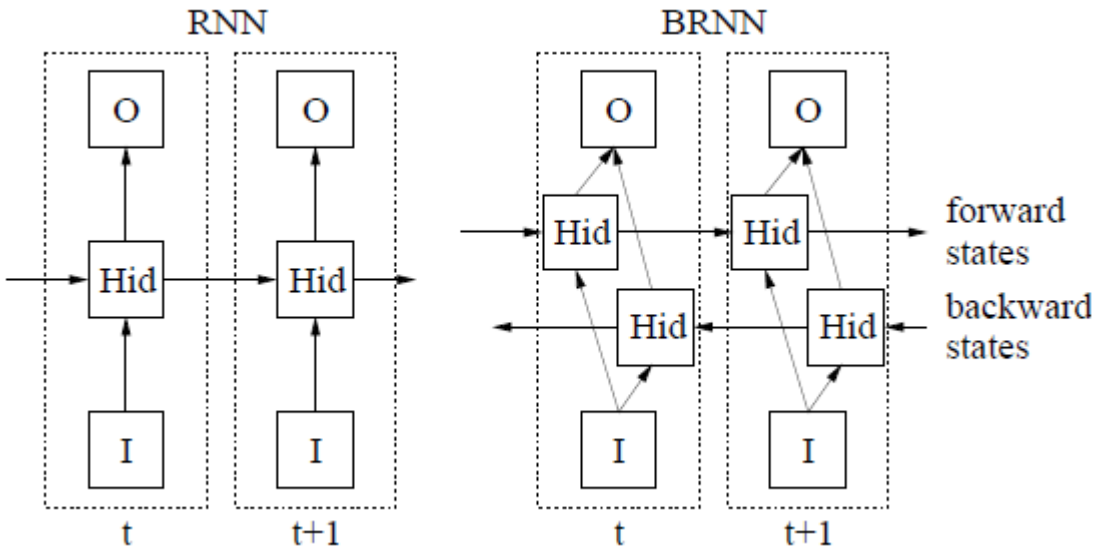


Рисунок 1.2 – Рекуррентная ИНС на основе архитектуры Long Short-Term Memory и ее аналог, развернутый во времени

Группой ученых из Берна в составе *Marcus Liwicki*, *Alex Graves*, *Horst Bunke* и *Jurgen Schmidhuber* в 2010 году был предложен подход для осуществления сегментации рукописного текста с использованием рекуррентной искусственной нейронной сети на основе архитектуры Long Short-Term Memory [56]. Особенность этой модели заключается в том, что некоторые из узлов искусственной нейронной сети способны сохранять информацию о последних значениях функции активации, которые были получены для предыдущих моментов времени.

Огромным преимуществом этой модели является возможность online-сегментации вводимого текста.

Для обучения такого рода ИНС используются изображения заранее сегментированного текста. По откликам рекуррентной нейронной сети можно определить, является ли определенная позиция изображения соединением двух рукописных символов (рисунок 1.3).

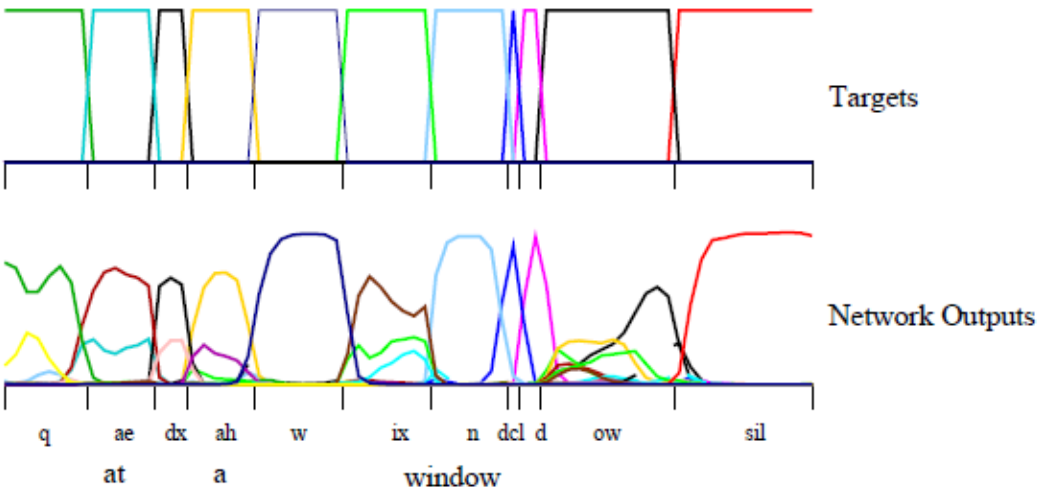


Рисунок 1.3 – Пример визуализации использования откликов ИНС

До того, как предложить такой подход, авторы использовали скрытую марковскую модель, что позволяло правильно сегментировать рукописный текст в 65,4% случаев. Использование же нового подхода приводит к улучшению данного показателя до 74%. В будущем авторы планируют использовать статистические свойства языка для улучшения качества сегментации, и, в последствие, распознавания рукописных символов.

Еще одна биологически-подобная сеть, основанная на двух принципах работы мозга: иерархического представления объектов и использования временной составляющей в процессе зрения – иерархическая временная сеть, разработанная в ТПУ *Ю.А. Болотовой*, используется для распознавания рукописных и печатных символов, в том числе и для распознавания автомобильных номеров [57, 58].

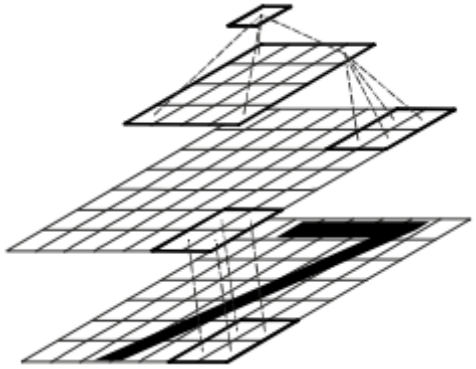


Рисунок 1.4 – структура искусственной временной сети

Такая сеть представлена несколькими уровнями. Каждый из уровней является двумерной решеткой узлов. Узлы верхних слоев связаны с узлами нижних своими рецептивными полями, которые охватывают несколько соседних узлов нижнего слоя.

Также группа ученых из Берна предложила использовать комитетный метод классификации на основе трех различных скрытых марковских моделей для решения задачи сегментации рукописного текста [59]. Марковская модель – модель, которая моделирует работу процесса, схожего с марковским процессом с неизвестными параметрами. Найденные параметры могут использоваться для дальнейшего анализа, например, для решения задачи классификации изображений.

Российские ученые *В. Вапник* и *А. Червоненкис* предложили метод опорных векторов – набор алгоритмов для решения задач классификации и регрессионного анализа [60, 61].

Метод опорных векторов нередко используется для распознавания печатных и рукописных символов. Применение метода в таком случае

сводится к получению некоторого многомерного вектора признаков графического представления символа. Вектора признаков формируют в пространстве признаков группы, которые требуется разделить гиперплоскостью так, чтобы расстояние до ближайшей из плоскостей было максимальным. С увеличением расстояния от гиперплоскости до ближайшего класса будет увеличиваться и точность классификации. Вектора, расположенные ближе всего к искомой гиперплоскости называются опорными векторами.

В случае если выбранное пространство признаков линейно неразделимо, требуется отобразить это пространство в пространство большей размерности так, чтобы условие линейной разделимости выполнялось. Однако стоит отметить, что с ростом размерности пространства увеличивается и сложность процесса нахождения оптимальной гиперплоскости.

Метод нашел широкое применение в задачах распознавания символов, обладая особенным преимуществом в задачах, где обучающая выборка имеет сравнительно небольшие размеры. Существенным же недостатком данного метода является тот факт, что наибольшее влияние на процесс классификации оказывают вектора, которые находятся на границах множеств.

Еще в 1998 году *Y. LeCun* совместно с *L. Bottou*, *Y. Bengio* и *P. Haffner* разработал сверточную нейронную сеть LeNet-5, предназначенную для распознавания рукописных цифр [62]. Архитектура представленной сети приведена на рисунке 1.5.

Тестирование предложенного классификатора выполнялось на бенчмарковом наборе рукописных цифр MNIST, применение которого подразумевает обучение с использованием 60000 изображений, а тестирование – на 10000 изображениях аналогичного типа. Ошибка обученной сверточной нейронной сети составила лишь 0,95% на этом наборе.

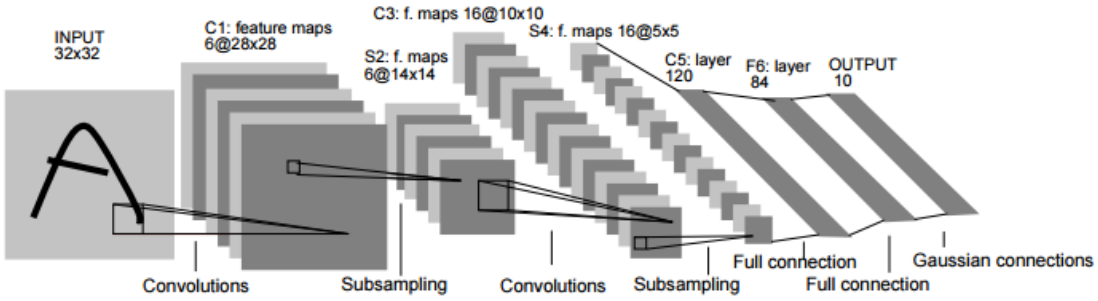


Рисунок 1.5 – Архитектура LeNet-5 для распознавания рукописных цифр

В 2006 году *D. Bouchain* исследовал применение сети LeNet-5, а также предшествующих ее версий, в составе ансамблей классификаторов все на том же общеизвестном наборе рукописных цифр MNIST [63]. В результате применения ансамбля из различных версий сетей архитектуры LeNet была достигнута ошибка, равная 0,7%.

Сегодня сверточные нейронные сети применяются для комплексного решения задачи обнаружения и распознавания текстов на сложном фоне [64, 65].

1.3.2. Статистические методы распознавания символов

Ученые *E. Kavallieratou, N. Fakotakis* и *G. Kokkinakis* из греческого города Патры в 2008 году предложили для распознавания отдельных рукописных символов использовать подход, основанный на анализе различных видов гистограмм [66]. Помимо достаточно распространенных вертикальной и горизонтальной гистограмм предложено использовать радиальную гистограмму и две характеристики, получившие названия «in-out профиль» и «out-in профиль».

Суть радиальной гистограммы заключается в получении данных не для строк или столбцов, а для прямых, проведенных через центр под различными углами. Похожий смысл и у двух профилей. In-out профиль строится путем нахождения наиболее близких к центру пикселей, принадлежащих символу, для каждой из аналогичных прямых. Точно так же строится и out-in профиль,

если вместо наиболее близкого к центру пиксела находить наиболее удаленный.

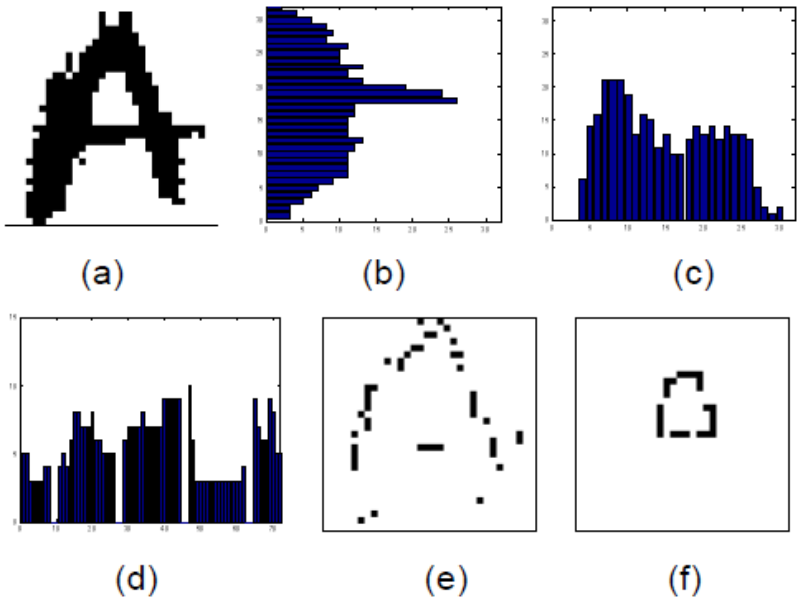


Рисунок 1.6 – (a) изображение символа, (b) горизонтальная гистограмма, (c) вертикальная гистограмма, (d) радиальная гистограмма, (e) out-in профиль, (f) in-out профиль

Несмотря на достаточно простой способ получения вектора признаков, авторы метода опубликовали результаты оценки качества распознавания, которые показывают, что подобный метод очень надежен и с высокой степенью достоверности может распознавать рукописные цифры и буквы латинского алфавита.

Для оценки качества распознавания использовалось два известных набора данных: NIST – для английского языка, и GRUHD – для греческого языка. В каждом из этих наборов было сделано три выборки для оценки среди цифр, строчных букв, заглавных букв и общего набора.

Данные о результатах оценки качества распознавания для набора NIST приведены в таблице 1.1. Стоит отметить, что на одной из выборок цифр этого набора предложенный алгоритм достиг стопроцентного качества распознавания.

Для набора символов GRUHD результаты несколько ниже, что вполне согласуется с более высокой сложностью набора и менее распространенным

греческим языком. Тем не менее, результаты работы алгоритма по-прежнему остаются высокими (таблица 1.2):

Таблица 1.1 – Апробация алгоритма на наборе данных NIST

	Выборка №1	Выборка №2	Выборка №3
Цифры	98,80%	99.91%	100%
Строчные буквы	93,85%	96,54%	98,86%
Заглавные буквы	91,40%	94,50%	98,85%
Смешанная выборка	82,79%	89,27%	96,85%

Таблица 1.2 – Апробация алгоритма на наборе данных GRUHD

	Выборка №1	Выборка №2	Выборка №3
Цифры	94,00%	97.42%	99,54%
Строчные буквы	86,03%	96,54%	98,96%
Заглавные буквы	81,00%	90,36%	96,60%
Смешанная выборка	72,80%	80,04%	88,83%

В обоих случаях существенное улучшения качества распознавания на более поздних выборках объясняется тем, что предыдущие выборки включаются в качестве образцов для обучения при анализе очередной выборки.

К статистическим подходам можно отнести и группу методов на основе метода пересечений [67]. Суть метода пересечений заключается в использовании некоторого набора прямых, каждая из которых накладывается на графическое представление символа. Для каждой прямой подсчитывается количество отрезков, которые она пересекает. Из определенных количеств пересечений и формируется вектор признаков, который используется для дальнейшего выполнения классификации.

Преимуществом этой группы методов является их инвариантность к дисторсии и небольшим стилистическим вариациям символов, а также достаточно высокое быстродействие и низкое потребление оперативной

памяти. Существуют реализации алгоритмов на основе метода пересечений без использования операций над числами с плавающей точкой [131, 137].

Для выбора множества точек, используемого в таких методах, используют, как различные эвристики и наблюдения, так и эволюционные алгоритмы [68]. Как правило, выбор оптимального множества для метода пересечений зависит от типа символов, с которыми будет иметь дело разрабатываемый алгоритм.

1.3.3. Методы распознавания символов на основе выделения структурных составляющих

Ученые университета *The Hong Kong University of Science and Technology Kam-Fai Chan* и *Dit-Yan Yeung* в 1999 году разработали более понятный для человеческого восприятия подход. Ими было предложено использовать анализ на основе гибкого сравнения структуры [69]. Основной отличительной способностью является способ получения дескрипторов. Дескрипторы в этом методе не будут представлять собой какие-то абстрактные числовые значения, которые являются некоторыми частотными характеристиками или усредненными значениями уровней яркости каких-либо частей изображения. Авторы предлагают анализировать графическое представление символа. Для начала рассматривается язык представления изображений на основе PDL-выражений с использованием четырех бинарных операций [70]:

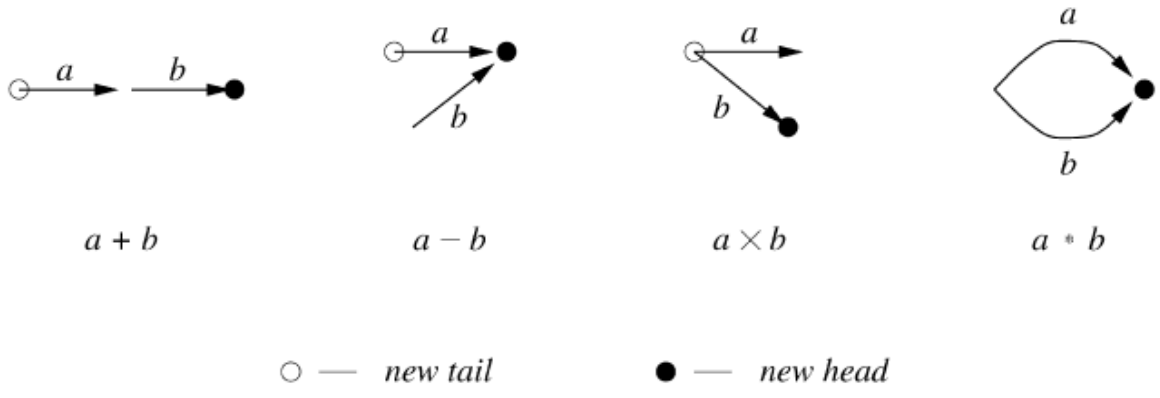


Рисунок 1.7 – Типы бинарных операций для задания PDL-выражений

Если добавить к этим четырем операциям еще одну унарную операцию отрицания, то можно построить некоторый набор начертаний:

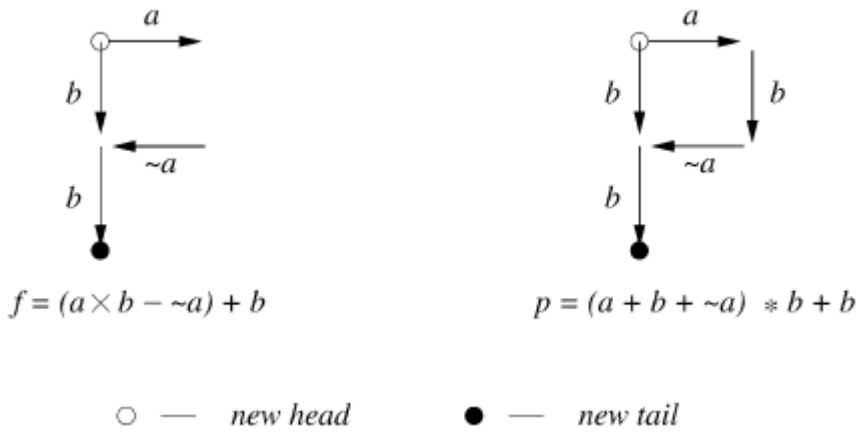


Рисунок 1.8 – Примеры визуализации PDL-выражений

Как можно заметить, такие выражения позволяют задать лишь небольшую часть геометрических представлений некоторых символов. Многие символы и вовсе могут быть заданы множеством различных выражений.

Авторы также рассматривают и PLEX-грамматику, которая основывается на использовании n-связных узлов (nape). В этой грамматике фигурирует три вида графических примитива:

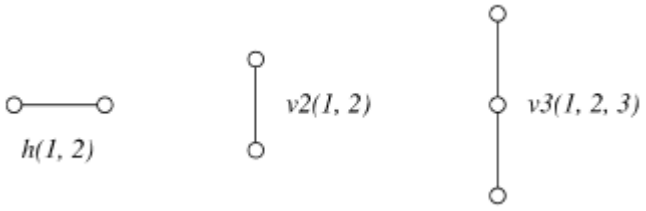


Рисунок 1.9 – Графические примитивы PLEX-модели

Можно образовывать различные графические примитивы и задавать их множеством графических примитивов, множеством точек их соединения и информацией об этих соединениях:

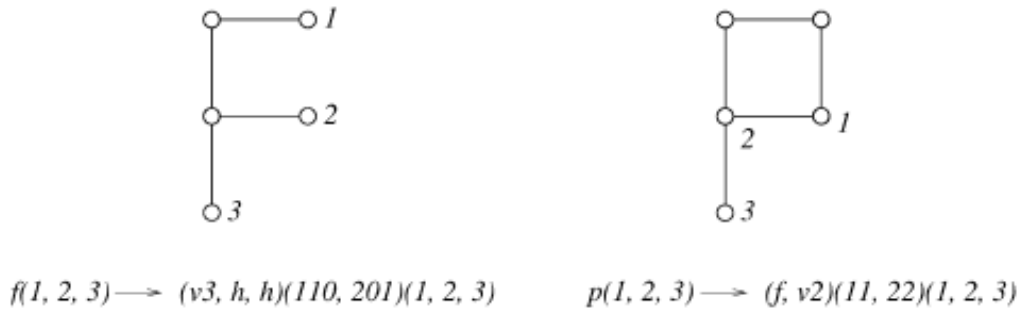


Рисунок 1.10 – Примеры объектов в PLEX-модели

Такая модель тоже имеет ряд существенных недостатков. В том числе и возможность задать один и тот же символ двумя или более способами в PLEX-модели.

Другой способ описания структуры символов, рассмотренный авторами – схема Бертода и Мароя (Berthod and Maroy's encoding scheme [71]). Эта схема использует для описания геометрического представления пять примитивов:

- Прямая линия – T;
- Дуга по часовой стрелке – P;
- Дуга против часовой стрелки – M;
- Перенос пера – L;
- Выступ – R.

Комбинируя такие примитивы можно кодировать каждое начертание символа некоторой строкой:

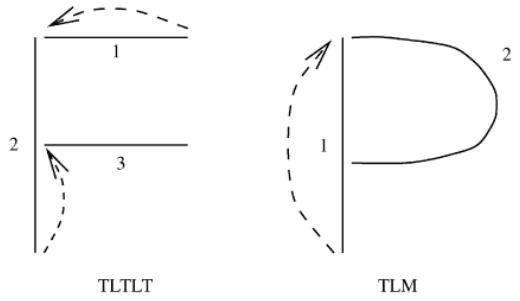


Рисунок 1.11 – Примеры объектов в схеме Бертода и Мароя

Как и у всех предыдущих моделей представления, у этой модели остается возможность задания одного и того же символа двумя или более способами. Однако, как утверждают авторы, количество способов задать один и тот же символ намного меньше, чем у всех ранее рассмотренных способов.

Еще одной схемой, которую рассматривают авторы, является цепной код Фримана. Код использует восемь значений от 0 до 7, которые задают, в каком направлении соединяются два узла:

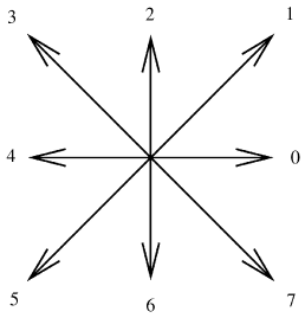


Рисунок 1.12 – Направления в коде Фримана

Так же в коде Фримана используются аналогичные направления начертаний при поднятом пере. Они обозначаются теми же числовыми значениями с горизонтальной чертой над цифрой. Примеры начертаний символов с использованием кода Фримана можно увидеть на изображении:

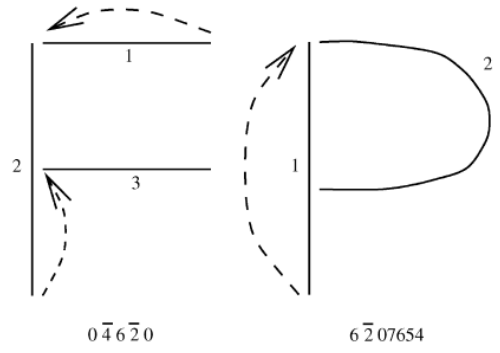


Рисунок 1.13 – Примеры символов в коде Фримана

Существует модификация кода Фримана, в которой горизонтальная черта над цифрами не используется. То есть не учитывается поднято или опущено перо. Эксперименты показывают, что, не смотря на существенное

сокращение количества способов задать один и тот же символ, подобная модификация не сильно ухудшает качество распознавания [72].

Опираясь на код Фримана, авторы разрабатывают собственную грамматику, которая оперирует со следующими примитивами:

- Отрезок – некоторая последовательность пикселей, принадлежащих графическому представлению символа, которая визуально представляет собой прямую линию;
- Дуга по часовой стрелке – некоторая последовательность пикселей, принадлежащих графическому представлению символа, которая визуально расположена существенно выше прямой линии, соединяющей два конца этой последовательности;
- Дуга против часовой стрелки – некоторая последовательность пикселей, принадлежащих графическому представлению символа, которая визуально расположена существенно ниже прямой линии, соединяющей два конца этой последовательности;
- Петля – некоторая последовательность пикселей, принадлежащих графическому представлению символа, которая начинается и заканчивается в одном и том же пикселе;
- Точка – некоторая последовательность пикселей, принадлежащих графическому представлению символа, состоящая из небольшого количества пикселей. Зачастую подобные области являются пиксельным шумом, но отбрасывать такие области заранее недопустимо, ведь они могут являться составной часть таких символов, как «i» или «j» [73].

Далее для хранения информации о структуре символа используется более информативная грамматика, чем в каждом из ранее описанных способов:

```
Character → {StrokeSet}
StrokeSet → Stroke
StrokeSet → Stroke, StrokeSet
Stroke → PrimitiveSet
PrimitiveSet → Primitive
PrimitiveSet → Primitive, PrimitiveSet
```

```
Primitive → {LineType, Direction}
Primitive → loop
Primitive → dot
LineType → line | up | down
Direction → 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7
```

Такого рода грамматика позволяет описать любое начертание символа с точностью до распознавания графических примитивов [74]. Пример описания изображений символов можно увидеть на изображении:

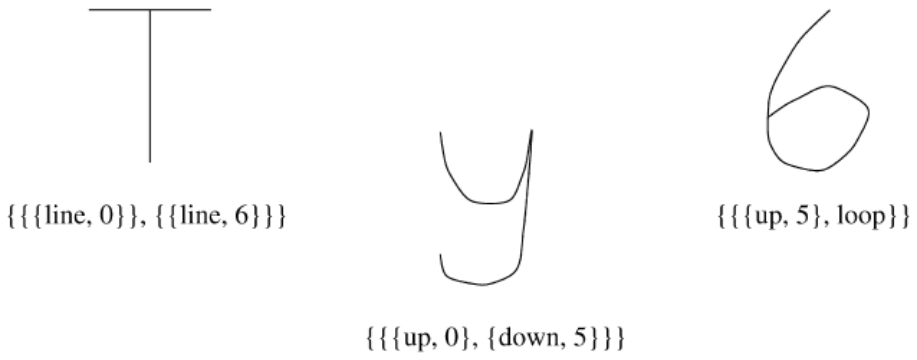


Рисунок 1.14 – Примеры обозначения структуры символа в предложенной авторами грамматике

Для определения типа символа по его структуре используется база образцов. Для каждого из образцов этой базы заранее определяется его структура, а затем очередной символ сравнивается с каждым из символов базы. Очевидно, что некоторые символы имеют несколько видов начертания из-за особенностей почерка или стиля написания. Для учета подобных случаев при сравнении выполняются некоторые преобразования структуры символов из базы, после чего производится тривиальное сравнение.

В качестве базы изображений рукописных символов была использована база «MIT Spoken Language Systems», которая содержит графические представления рукописных начертаний 62 классов символов – цифры (0 – 9), строчные буквы английского алфавита (a – z), прописные буквы английского алфавита (A – Z).

Символ каждого класса в этой базе был начерчен одним из 150 людей собственным почерком. Ниже приведены примеры начертаний символа «3» из этой базы:

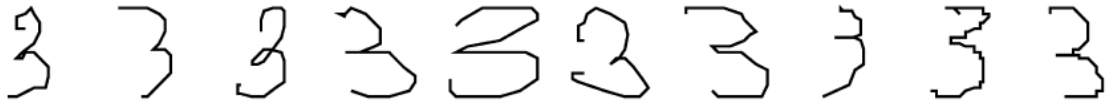


Рисунок 1.15 – Примеры изображений в базе «MIT Spoken Language Systems»

Как можно заметить, некоторые символы были специально введены неаккуратно, некоторые – содержат шумы. В целом же, выборка содержит лишь начертания цифр, которые выполнялись с учетом единого шаблона написания.

Результаты оценки качества распознавания для подготовленной выборки приведены в таблице 1.3.

Таблица 1.3 – Качество распознавания предложенного алгоритма

Цифры	98,60%
Строчные буквы	98,49%
Прописные буквы	97,44%
Общий набор	97,40%

По результатам, полученным авторами *Kam-Fai Chan* и *Dit-Yan Yeung* [69], можно сделать вывод о том, что предложенный метод позволяет с высокой точностью определять класс изображения рукописного символа без использования каких-либо признаковых классификаторов: искусственных нейронный сетей, радиально-базисных сетей, самоорганизующихся карт Кохонена, машин опорных векторов и т.д.

В зарубежной литературе можно встретить огромное количество методов распознавания и сегментации рукописного текста с использованием частотных характеристик [75] и последующего применения инструментов нечеткой логики [76] или искусственных нейронных сетей [77]. Большинство этих методов имеют точность распознавания, превышающую 90% [78], однако точные критерии, по которым тот или иной символ был отнесен к определенному классу, остаются неизвестными в виду отсутствия четко сформулированной логики принятия решений [79, 80].

На общем фоне среди всех рассмотренных методов выделяются методы, которые основываются на анализе общей структуры геометрического представления символа. Такие методы сложнее в

реализации и, возможно, быстродействие таких методов зависит от глубины и сложности анализа графического представления символа. Тем не менее, такие методы обладают гораздо более высокой точностью распознавания, а их логика гораздо понятнее для человеческого восприятия.

Еще одним преимуществом алгоритмов, анализирующих структуру геометрического представления символа, является их независимость от того, рукописные это символы или печатные. Работа алгоритма зависит лишь от набора шаблонов, с использованием которого будет производиться сравнительный анализ текущего изображения.

1.4. Методы скелетизации бинарных растровых изображений

В данном разделе рассмотрены наиболее известные методы скелетизации бинарных растровых изображений, которые обладают достаточно высоким быстродействием, чтобы использоваться для решения задачи распознавания символов [81, 82].

Одним из видов предварительной обработки растровых бинарных изображений является их скелетизация – процесс утоньшения графического представления объекта, в данном случае – символа [83]. Предварительная скелетизация нашла широкое применение в задачах компьютерного зрения. Такой вид предварительной обработки позволяет существенно уменьшить количество возможных способов представить один и тот же символ в виде бинарного изображения [84, 85].

Существенный интерес представляют лишь полиномиальные алгоритмы скелетизации бинарного растрового изображения [86, 87]. Для задачи распознавания символов существенно, чтобы вычислительная сложность алгоритма скелетизации исходного изображения не превышала $O(P)$, где P – количество пикселов исходного изображения. В противном случае запуск алгоритма скелетизации может потребовать больше времени, чем оставшиеся этапы алгоритма распознавания символа.

1.4.1. Алгоритм скелетизации Зонга-Суня

T.Y. Zhang и *C.Y. Suen* в 1984 году предложили собственный алгоритм скелетизации бинарных растровых изображений [88]. При асимптотически оптимальной реализации быстродействие этого алгоритма линейно зависит от количества пикселей исходного изображения.

Идея алгоритма заключается в последовательном выполнении определенного набора действий до тех пор, пока хотя бы один пиксел изображения изменяется после выполнения этих действий.

В ходе описания алгоритма наиболее удобным будет полагать, что пиксели, относящиеся к графическому представлению символа, являются черными, оставшиеся пиксели – белыми. Для каждого пиксела рассматриваются его восемь соседних с ним пикселей бинарного изображения, которые в свою очередь, пронумерованы по схеме, показанной на рисунке 1.16.

P9	P2	P3
P8	**P1**	P4
P7	P6	P5

Рисунок 1.16 – Порядок нумерации соседних пикселей в алгоритме Зонга-Суня

Для пикселей, которые находятся на границе, все несуществующие соседние пиксели полагаются белыми.

Определим A(P1), как количество переходов от белого пиксела к черному при цикличном обходе соседей в порядке P2, P3, P4, P5, P6, P7, P8, P9, P2. Также определим B(P1) равным количеству черных соседей центрального пиксела P1.

На первом шаге алгоритма требуется пометить все пиксели изображения P1, которые одновременно удовлетворяют следующим требованиям:

- Пиксел является черным и имеет восемь соседей;

- Выполняется неравенство $2 \leq B(P1) \leq 6$;
- Выполняется равенство $A(P1) = 1$;
- Как минимум один из пикселов P2, P4 или P6 является белым;
- Как минимум один из пикселов P4, P6 или P8 является белым.

В конце первого шага требуется удалить все пиксели, которые были помечены.

На втором шаге алгоритма требуется пометить все пиксели изображения P1, которые одновременно удовлетворяют несколько иному списку требований:

- Пиксел является черным и имеет восемь соседей;
- Выполняется неравенство $2 \leq B(P1) \leq 6$;
- Выполняется равенство $A(P1) = 1$;
- Как минимум один из пикселов P2, P4 или P8 является белым;
- Как минимум один из пикселов P2, P6 или P8 является белым.

Аналогично первому шагу, после второго шага алгоритма требуется удалить все помеченные во время этого шага пиксели.

Первый и второй шаги алгоритма требуется выполнять до тех пор, пока хотя бы один пиксел помечен в ходе этих действий.

Стоит отметить, что при тривиальном подходе, потребуется многократный проход по исходному изображению, что приведет к вычислительной сложности порядка $O(P^2)$. Однако, если на каждой итерации алгоритма, кроме первой, проверять лишь пиксели, хотя бы один из соседей которых изменился на прошлой итерации, то вычислительная сложность такого алгоритма определяется, как $O(P)$. Реализовать такой подход можно с использованием абстрактного типа данных queue (очередь).

1.4.2. Алгоритм скелетизации Ву-Цая

В 1992 году *Rey-Yao Wu* и *Wen-Hsiang Tsai* предложили собственный метод скелетизации бинарного растрового изображения, обладающий высоким быстродействием [89].

Авторы предложили подход, при котором все пиксели изображения требуется рассмотреть лишь по одному разу. Для выполнения скелетизации предполагается использовать 14 шаблонов, которые приведены на рисунке 1.17.

В приведенных шаблонах символы ‘c’, ‘0’, ‘1’, ‘x’ обозначают, соответственно, рассматриваемый пиксел, белый пиксел, черный пиксел и пиксел произвольного цвета. Символ ‘y’ обладает особым свойством – хотя бы один из таких пикселов, при наложении шаблона, должен быть белым.

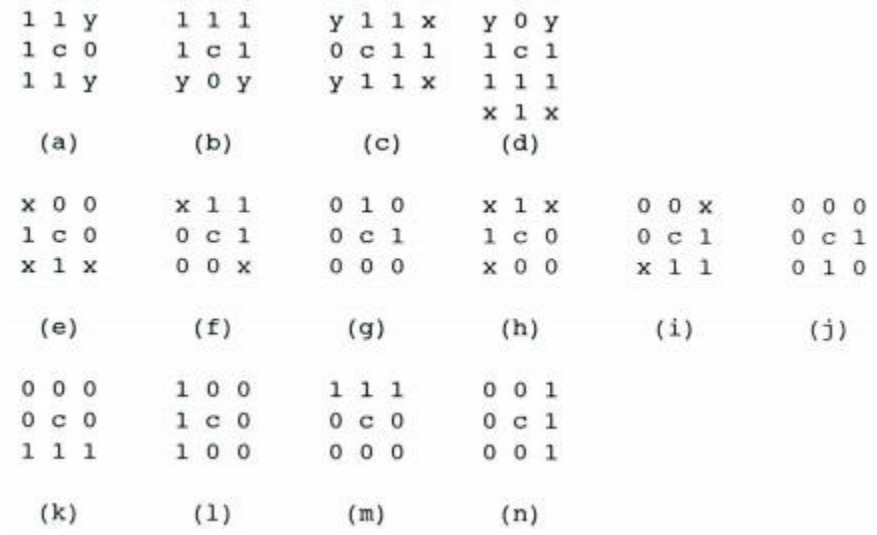


Рисунок 1.17 – Шаблоны, используемые в алгоритме скелетизации Ву-Цая

Каждый из приведенных шаблонов последовательно применяется для каждого из пикселов при линейном проходе по всем пикселам исходного изображения. Пиксел удаляется в случае, если пиксел, с соседними ему пикселами, подходит хотя бы под один из описанных шаблонов. Более того, удаление производится до того, как будет рассматриваться следующий пиксел.

Из-за того, что все пиксели изображения рассматриваются ровно по одному разу, вычислительная сложность алгоритма составляет *O(P)*.

1.4.3. Алгоритм скелетизации Го-Холла

Сотрудники университета Питтсбурга *Zicheng Guo* и *Richard W. Hall* в 1989 году предложили подход [90], чем-то схожий с алгоритмом Зонга-Суня [88]. Основная идея алгоритма, как и в ранее рассмотренных алгоритмах, основывается на оценке восьми соседей каждого из пикселов. Авторы используют обозначения соседей по схеме, приведенной на рисунке 1.18.

p_1 p_2 p_3

p_8 p p_4

p_7 p_6 p_5

Рисунок 1.18 – Схема нумерации соседей пиксела p в алгоритме Го-Холла

Обозначим *B(p)*, как количество соседних пикселей p_i черного цвета. Аналогично, обозначим *C(p)* равным количеству различных компонент восьми-связности, соседствующих с пикселом *p*.

Символами ‘∧’ и ‘∨’ обозначим, соответственно операции конъюнкции и дизъюнкции, в то время, как обычные арифметические операции сложения и умножения стандартно обозначаются знаками ‘+’ и ‘·’ соответственно. Для удобства описания алгоритма введем новое обозначение *N(p)*.

$$N(p) = \min(N_1(p),\ N_2(p)),$$

где $N_1(p) = (p_1 \vee p_2) + (p_3 \vee p_4) + (p_5 \vee p_6) + (p_7 \vee p_8)$,

и $N_2(p) = (p_2 \vee p_3) + (p_4 \vee p_5) + (p_6 \vee p_7) + (p_8 \vee p_1)$.

Фактически, каждое из значений $N_1(p)$ и $N_2(p)$ разбивает всех соседей пиксела p на четыре пары соседних между собой пикселей, считая количество таких пар с одним или двумя черными пикселами.

Используя данные обозначения можно сформулировать алгоритм Го-Холла. Будем выполнять итерации алгоритма до тех пор, пока очередная из них не оставит изображение без изменений. На каждой из итераций удаляются только пиксели, которые соответствуют каждому из следующих условий:

- $C(p) = 1$;
- $2 \leq N(p) \leq 3$;
- Для нечетного номера итерации: $(p_2 \vee p_3 \vee \bar{p}_5) \wedge p_4 = 0$,

а для четного: $(p_6 \vee p_7 \vee \bar{p}_1) \wedge p_8 = 0$.

Аналогично алгоритму Зонга-Суня, алгоритм обладает вычислительной сложностью $O(P^2)$, но при реализации с использованием очереди вычислительная сложность оценивается как $O(P)$.

1.5. Основные результаты и выводы по главе 1

1. В данной главе проведен анализ методов и алгоритмов распознавания символов. Установлено, что большая часть разработанных на сегодня алгоритмов нуждается в большом количестве эталонных изображений для стадии обучения или поиска зависимостей и закономерностей. В процессе исследования было принято решение осуществлять разработку алгоритма на основе выделения структурных составляющих.

2. Проведенный анализ позволил сделать вывод о том, что разработанные ранее методы на основе выделения структурных составляющих создавались с учетом ограниченных вычислительных мощностей. Поэтому предпринимались многочисленные попытки упрощенного представления в виде грамматик или малого количества простейших графических примитивов. В последние же полтора десятилетия наиболее значимые методы и вовсе опираются на использование

признаковых классификаторов, чем осложняют визуализацию процесса классификации для человеческого восприятия.

3. В большинстве рассмотренных алгоритмов распознавания символов используется предварительная обработка исходных графических представлений символов. Например, используются вейвлеты и курвлеты для удаления шумов на изображениях. Для изображений, которые прошли стадию бинаризации, используется скелетизация (утоньшение) или морфологические операции для удаления различных дефектов, допущенных при печати документа, который в последствие был отсканирован. Основным преимуществом такого подхода является возможность существенно уменьшить количество изображений для обучения классификатора или настройки алгоритма.

4. Для решения задачи распознавания образов, в том числе и символов, зачастую используются каскадные классификаторы и прочие методы пост-обработки результатов классификации. Чаще всего подобные средства используются для улучшения результатов искусственных нейронных сетей и других признаковых классификаторов, позволяя существенно улучшить качество распознавания символов. В алгоритмах на основе структурных составляющих такой подход может быть использован на стадии сравнения двух ранее построенных моделей.

ГЛАВА 2. АЛГОРИТМЫ РАСПОЗНАВАНИЯ СИМВОЛОВ В УСЛОВИЯХ МАЛОЙ ОБУЧАЮЩЕЙ ВЫБОРКИ НА ОСНОВЕ ПОСТРОЕНИЯ СТРУКТУРНЫХ МОДЕЛЕЙ

В данной главе приводится подробное описание всех стадий предложенного алгоритма построения структурной модели: предварительной обработки входного изображения, выделения структурных составляющих, составления структурной модели символа на основе выявленных составляющих. Приводятся три критерия схожести двух ранее построенных структурных моделей символов. Предложен алгоритм на основе комбинации двух существующих алгоритмов скелетизации с использованием собственной операции устранения плоских окончаний. Предложен собственный алгоритм выделения структурных составляющих и последующего построения модели на их основе. Предложен оригинальный алгоритм сегментации рукописного текста на основе разработанных алгоритмов распознавания рукописных символов с применением метода динамического программирования.

2.1. Алгоритм предварительной обработки исходного изображения

Предварительную обработку исходного изображения можно разделить на три этапа:

1. Повышение контрастности исходного изображения.
2. Бинаризация полученного на предыдущем этапе изображения.
3. Скелетизация бинарного изображения.

После последовательного выполнения этих трех этапов предварительной обработки, полученное скелетизированное бинарное изображение будет являться исходным для алгоритма распознавания символов.

2.1.1. Применение гистограммной эквализации для увеличения контрастности изображения

При решении задачи распознавания символа, как правило, в качестве входного изображения используется сканированный документ или элемент фотоизображения. В таких случаях процесс распознавания может осложняться различными факторами: неоптимальная настройка устройства, выполняющего съемку или сканирование, плохой уровень освещенности, неоднородность материала, на котором выполнено начертание символа и т.д. В большинстве из таких случаев на исходном изображении символы могут быть плохо отличимыми от фона [91]. Это объясняется низкой контрастностью изображения.

Для повышения контрастности изображения достаточно воспользоваться общеизвестным методом гистограммной эквализации [92], который реализован в большинстве программных библиотек компьютерного зрения [93]. Суть этого метода заключается в том, чтобы интегральная гистограмма изображения была как можно ближе к линейной функции.

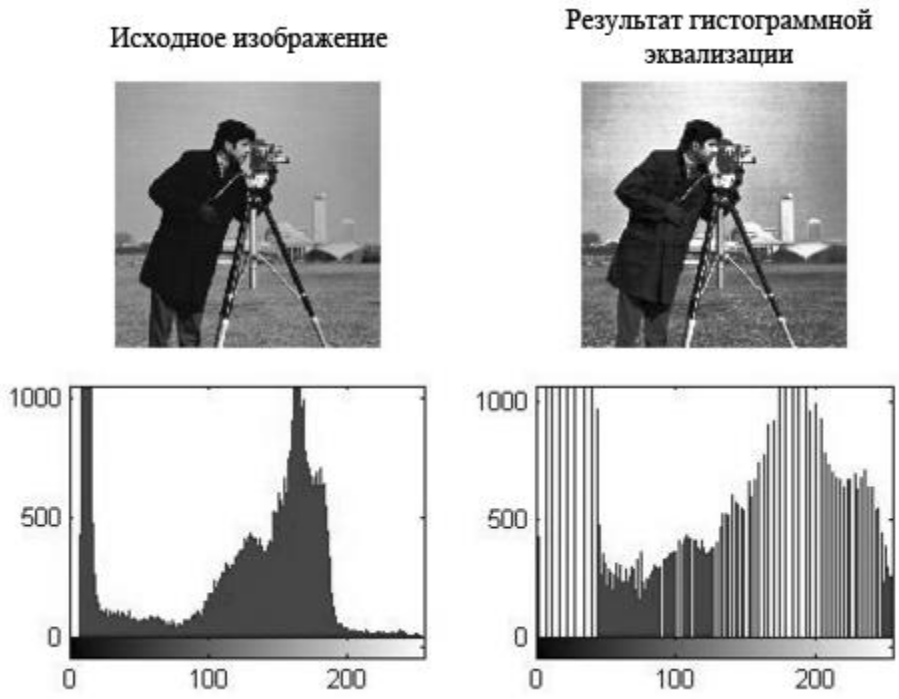


Рисунок 2.1 – Визуализация принципа работы гистограммной эквализации

Построение гистограммы изображения выполняется с использованием однократного прохода по всем пикселам изображения и по всем уровням яркости, представленным на нем.

Преобразовать гистограмму изображения в его интегральную гистограмму можно с помощью одного прохода по всем допустимым значениям яркости для этого изображения:

$$H'_0 = H_0,$$

$$H'_i = H'_{i-1} + H_i \ (i \geq 1),$$

где H_i – значения гистограммы изображения, а H'_i – значения его интегральной гистограммы.

Для увеличения контрастности изображения необходимо повторить линейный проход по всем пикселам исходного изображения, заменив значение яркости каждого пиксела:

$$l_{i,j} = H'_{l_{i,j}},$$

где $l_{i,j}$ – уровень яркости пиксела i-й строки j-го столбца.

Как правило, количество допустимых уровней яркости на изображении намного меньше, чем количество пикселов, следовательно, можно считать, что алгоритм имеет вычислительную сложность O(P), где P – количество пикселей на изображении.

Алгоритм, обладающий таким высоким быстродействием, не окажет существенного влияния на быстродействие процесса распознавания в целом, однако позволит избежать осложнений в ходе распознавания из-за пониженной контрастности исходного изображения.

2.1.2. Адаптивная бинаризация исходного изображения

Для выполнения распознавания символа необходимо предварительно разделить все пиксели этого изображения на пиксели, которые принадлежат графическому представлению символа, и оставшиеся пиксели.

Можно считать, что на изображении в градациях серого пиксели, принадлежащие начертанию символа, существенно отличаются по среднему уровню яркости от пикселей фона. Для определенности будем полагать, что более светлые пиксели принадлежат фону. В таком случае для разделения пикселей на два описанных класса обычно используется некоторое пороговое значение яркости T. Пиксели с уровнем яркости выше T полагаются белыми (принадлежащими фону), оставшиеся пиксели – черными (принадлежащими графическому представлению символа). Процесс выполнения такого разбиения называется бинаризацией, полученное в ходе этого процесса изображение называется бинаризированным, а выбранное значение T – порогом бинаризации. Выбор порогового значения яркости T является нетривиальной задачей, решение которой существенно зависит от конкретного класса изображений.

Существует целый класс методов адаптивной бинаризации, в которой выполняется не только разделение пикселей исходного изображения в зависимости от порогового значения T, но и выполняется автоматический выбор этого значения.

Для определения оптимального порога бинаризации T предложено огромное количество различных алгоритмов. Как правило, такие методы опираются на использование локальной или глобальной гистограммы изображения. Одним из наиболее широко используемых подходов является подход, предложенный Оцу [94, 95].

Метод Оцу основывается на использовании гистограммы изображения. Рассмотрим, для определенности, изображение в оттенках серого. Его гистограмма H содержит 256 столбцов от H_0 до H_{255}. Если рассматривать произвольную подгистограмму этой гистограммы от H_l до H_r $(l \leq r)$, то для нее можно вычислить оценку дисперсии яркости $D(l, r)$. Используя оценку дисперсии, для определенного порога бинаризации T можно определить критерий разделимости $SC(T)$:

$$SC(T) = 1 - \frac{D(0,T)+D(T+1,255)}{D(0,255)}.$$

Критерий $SC(T)$ всегда принимает значение из отрезка [0, 1], причем, чем больше это значение, тем лучше разделимость яркостного распределения на два класса относительно порога бинаризации T.

Алгоритм Оцу предполагает вычисление критерия $SC(T)$ для всех значений T. Оптимальным значением порога бинаризации T_{opt} считается значение T, соответствующее максимальному значению критерия $SC(T)$.

Преимуществом данного алгоритма является понятный статистический смысл и низкая вычислительная сложность.

После того, как алгоритм Оцу был использован для бинаризации исходного изображения, можно считать, что все последующие этапы обработки графического представления символа работают с бинарным изображением, черный пиксел которого принадлежит графическому представлению символа, а белый – фону.

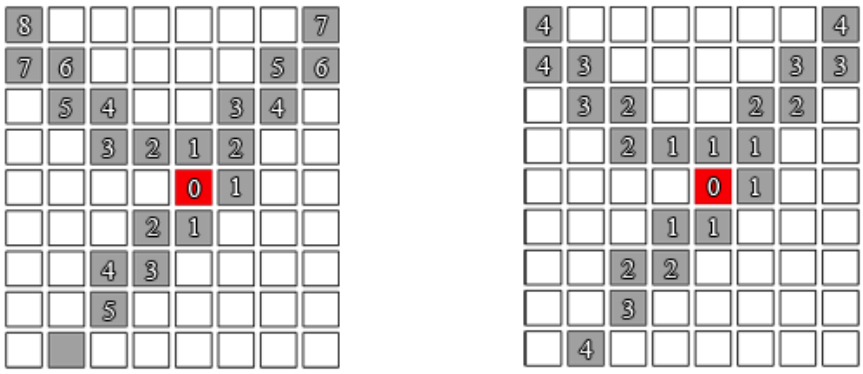
2.1.3. Алгоритм скелетизации бинарного изображения для распознавания символов

Скелетизация – процесс, в ходе которого выполняется максимальное утоньшение всех линий на бинарном изображении, результатом чего является скелет, в котором все линии имеют толщину в один пиксель.

В задаче распознавания символов очень важно, чтобы в процессе скелетизации не происходило потери информации о ключевых элементах графического начертания исходного символа. С другой стороны, для методов, опирающихся на использование скелетизированного изображения, важно, чтобы ни один из пикселов, который можно удалить, не нарушая топологии графического представления, не был пропущен в ходе работы алгоритма.

Алгоритм скелетизации Зонга-Суня [88] находит широкое применение при решении задачи распознавания символов. Основным его достоинством

является низкая вероятность удаления важных элементов графического начертания в ходе процесса скелетизации. В исследованиях *Widiarti A.R.* приведен сравнительный анализ алгоритмов построения скелетов бинаризированных растровых изображений [96]. В приведенном исследовании отмечается, что алгоритм Зонга-Суня показал наиболее высокое быстродействие и низкую вероятность удаления важных элементов графического начертания в ходе процесса скелетизации.



(а) (б)

Рисунок 2.2 – Результат обхода алгоритмом Ли одного и того же бинарного изображения для компонент (а) четырех-связности и (б) восьми-связности

Однако, как показали результаты исследования метода Зонга-Суня, одно из его главных достоинств является и одним из его существенных недостатков. На результирующем изображении достаточно часто остаются пиксели, которые можно удалить без нарушения топологии исходного графического начертания. На рисунке 2.3 показан пример такого результата применения алгоритма.

На первый взгляд, такого рода недостаток не должен оказать существенного влияния на дальнейшее поведение алгоритма распознавания символов. Однако для большинства методов, опирающихся на выделение структурных составляющих, большое количество не до конца скелетизированных участков может оказаться критичным. Например, для анализа структуры графического начертания символа многие методы используют обход пикселей бинарного изображения с помощью алгоритма Ли [97]. Данный алгоритм является частным случаем применения

волнового алгоритма для компонент четырех-связности или восьми-связности в зависимости от решаемой задачи.

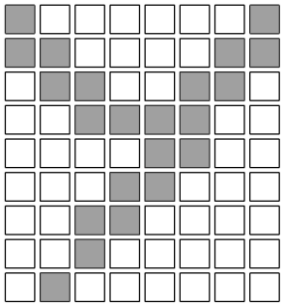


Рисунок 2.3 – Пример не до конца скелетизированного участка изображения символа

На рисунке 2.2 представлена визуализация применения алгоритма Ли к не до конца скелетизированному изображению, как для варианта с компонентами четырех-связности, так и для варианта с компонентами восьми-связности.

Как можно заметить, в обоих случаях применение алгоритма Ли к такого рода изображению приводит к некоторым недочетам.

В случае с компонентами четырех-связности два конечных пиксела были посещены с разным номером волны, хотя во втором случае количество шагов, требуемых для достижения этих пикселов из начального пиксела, совпадает. Как следствие, номер волны слабо связан с реальным расстоянием от начальной точки и зависит от того, насколько качественно была выполнена скелетизация области отдельного участка изображения. Но гораздо существеннее тот факт, что один из конечных пикселов и вовсе не посещен в ходе работы алгоритма из-за своей недостижимости при переходе лишь к соседним по стороне пикселам.

В случае с компонентами восьми-связности на изображении имеются соседние по стороне пиксели, имеющие одинаковый номер волны. Это говорит о том, что порядок посещения этих двух пикселов при обходе из произвольной вершины может быть недетерминированным. При оценке топологии объекта критичным может быть наличие более чем одного

простого пути между двумя пикселами, хотя бинарное изображение не содержит ни одного цикла.

Если в первом случае для устранения обеих ошибок необходимо одновременно и удалять определенные пиксели, и добавлять их, то во втором достаточно лишь удаления лишних пикселей. Для устранения областей изображения, где скелетизация была выполнена не полностью, в данной работе впервые предложено дополнительно использовать алгоритм скелетизации Ву-Цая [89]. После применения такого подхода нежелательные участки на результирующем изображении устраняются. Результат обхода алгоритмом Ли после применения алгоритма Ву-Цая показан на рисунке 2.4.

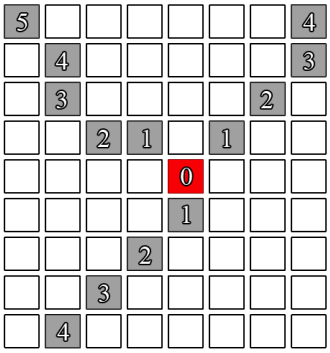


Рисунок 2.4 – Результат применения алгоритма Ли после обработки двумя алгоритмами скелетизации

Стоит отметить, что самостоятельное применение алгоритма Ву-Цая без предварительной обработки алгоритмом Зонга-Суня оказывается нецелесообразным. Алгоритм Ву-Цая достаточно часто нарушает исходную топологию объекта, удаляя небольшие элементы графического представления символа, например, важные с точки зрения топологии выступы на графическом представлении. Можно сделать вывод, что использование лишь одного из двух рассмотренных алгоритмов скелетизации приведет к потере информации о топологии символа или к неоднозначности при попытке извлечения структурных составляющих. В свою очередь, использование сначала алгоритма Зонга-Суня, а затем алгоритма Ву-Цая приводит к устранению обоих типов недостатков [144].

Однако существуют изображения, при обработке которых даже при таком подходе удаляются важные элементы графического представления символа. Пример такого случая изображен на рисунке 2.5.

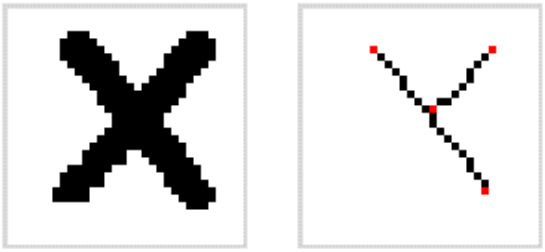


Рисунок 2.5 – Пример некорректной скелетизации последовательным применением алгоритмов Зонга-Суня и Ву-Цая

При подробном анализе такого рода случаев выяснилось, что алгоритм Зонга-Суня иногда тоже допускает удаление важных с точки зрения топологии символа элементов. Следовательно, ошибка допускается уже первым из двух последовательно применяемых алгоритмов.

Как можно заметить, приведенное графическое начертание символа «X» содержит плоское завершение нижнего левого элемента. Алгоритмом оно воспринимается, как локальное утолщение и постепенно удаляется. Такого рода плоских завершений следует избегать. Удаляя такие элементы, можно нарушить исходную топологию объекта. Следовательно, необходимо добавлять некоторые пиксели так, чтобы избегать подобных плоских завершений.

Одним из способов равномерного утолщения бинарного изображения является дилатация – одна из операций морфологической обработки изображений. Однако применение такой операции может привести к соединению двух близко расположенных друг к другу элементов, тем самым нарушить исходную топологию объекта.

В данной работе предложена собственная операция устранения плоских окончаний, имеющая сходство с морфологической операцией дилатации. Для каждого белого пиксела рассмотрим пиксели, граничащие с ним по стороне. Если цвета четырех соседних пикселей не образуют одну из

четырех комбинаций приведенных на рисунке 2.6, то пиксел становится черным.

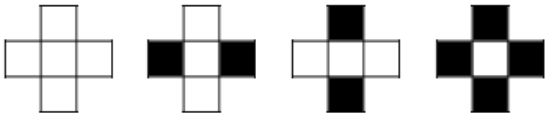


Рисунок 2.6 – Комбинации соседних пикселей, не требующие перекраски центрального пикселя в черный цвет

Предобработка с использованием такой операции позволяет избежать плоских окончаний элементов бинарных изображений, не требующих удалений. Для ранее приведенного примера символа применение такой операции приводит к существенному улучшению результата.

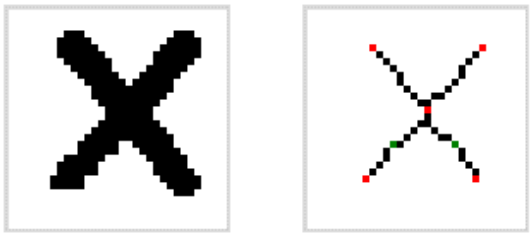


Рисунок 2.7 – Результат выполнения скелетизации после применения предложенной операции

Как можно заметить, применение предложенной операции к бинарному изображению позволяет избежать описанной проблемы. Дальнейшее исследование показало, что для всех рассмотренных в ходе экспериментов случаев предложенный подход к выполнению скелетизации не нарушает исходной топологии символа.

Таким образом, предложенный подход к выполнению скелетизации состоит из трех последовательных этапов:

1. Предварительная обработка бинарного изображения с целью устранения плоских окончаний элементов графического представления.
2. Применение алгоритма скелетизации Зонга-Суня.
3. Применение алгоритма Ву-Цая для устранения участков с незаконченной скелетизацией на полученном изображении.

2.2. Алгоритм выделения структурных составляющих на скелетизированном бинарном изображении

Выполнить выделение структурных составляющих на скелетизированном бинарном изображении существенно проще, чем на произвольном бинарном изображении. Для такого рода изображений гораздо проще сформулировать правила, по которым будут определяться наиболее важные элементы графического представления символа и, как следствие, алгоритмы, которые позволят извлечь эти элементы.

Человеческий мозг способен быстро классифицировать изображенный символ, не рассматривая в течение продолжительного времени каждый пиксел изображения. Достаточно быстро выполнить классификацию позволяет обнаружение наиболее важных точек на графическом начертании символа. На рисунке 2.8 приведены примеры символов, с отмеченными на них наиболее важными для распознавания элементами.

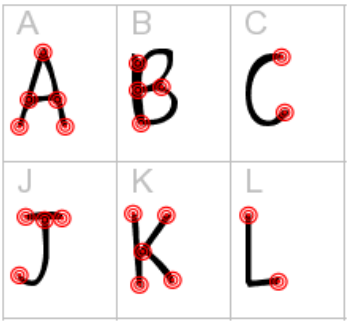


Рисунок 2.8 – Примеры букв латинского алфавита с отмеченными на них ключевыми элементами

На приведенном изображении показаны символы латинского алфавита, на которых отмечены места окончания или стыка простейших графических примитивов: отрезков и дуг. Если несущественно изменять соединяющие их линии, то мы все равно сможем узнать эти символы.

Выделение таких важных точек даже для скелетизированных бинарных изображений остается нетривиальной задачей. Еще более нетривиальной задачей является характеристика соединяющих их элементов. Если научиться извлекать из бинарного изображения информацию о расположении

ключевых точек, а также о характеристиках соединяющих их линий, то на основе этих данных можно построить полноценную структурную модель символа, достаточно информативную для его классификации.

Обозначим термины, которые будут использоваться для описания извлекаемых структурных составляющих представления символа, а также для описания алгоритма их получения.

Ключевые пиксели – черные пиксели скелетизированного бинарного изображения символа, соответствующие ключевым точкам или изгибам его структурной модели.

Соединяющее ребро – структурная составляющая представления символа, описывающая последовательность черных пикселов на скелетизированном бинарном изображении символа, соединяющую два ключевых пикселя.

Ключевая точка – структурная составляющая представления символа, которая является местом стыка одного, трех или более соединяющих ребер. Также ключевой точкой можно назвать место стыка двух соединяющих ребер, направляющие вектора которых, при выборе в качестве начального положения данной ключевой точки, расположены под углом менее 120 градусов.

Направляющий вектор некоторого соединяющего ребра может быть вычислен с использованием следующего принципа. Рассмотрим соединяющее ребро между двумя ключевыми пикселами, как упорядоченную в порядке удаления от начального положения последовательность пикселей. В качестве начального положения следует выбрать один из инцидентных ключевых пикселей. Поочередно проведем вектора из начального ключевого пикселя в каждый из пикселей пути. Последовательно просуммируем в порядке удаления от начального ключевого пиксела все вектора с угасающими коэффициентами: например, первый с множителем 1; второй – с множителем 0,5; третий – с множителем 0,25; и так далее.

Изгиб (точка изгиба) – структурная составляющая представления символа, которая не может быть отнесена к ключевым точкам, но линия, соединяющая два ключевых пиксела и проходящая через этот пиксел, существенно изменяет направление на участке графического представления символа, соответствующем этому пикселу. Фактически, изгиб является стыком двух соединяющих ребер, направляющие вектора которых, при выборе в качестве начального положения данного изгиба, расположены под углом не менее 120 градусов.

Композитное ребро – последовательность соединяющих ребер, начинающаяся и заканчивающаяся в ключевых точках, которая не содержит ни одной ключевой точки, кроме начальной и конечной.

Таким образом, черные пиксели, которые не являются ключевыми пикселями или изгибами, объединяются в соединяющие ребра, которые, в свою очередь, объединяются в композитные ребра.

2.2.2. Алгоритм выделения ключевых пикселей и изгибов на скелетизированном бинарном изображении

Выделение структурных составляющих рационально начать с выделения ключевых пикселей. Как ранее было отмечено, ключевые пиксели находятся на стыке нескольких графических примитивов или являются конечными в графическом преставлении этих примитивов. В обоих случаях обнаружить принадлежность пиксела к классу ключевых можно, рассмотрев восемь его соседних пикселей.

С помощью анализа скелетизированных изображений набора рукописных цифр MNIST был определен набор шаблонов, по которым можно определить пиксели, являющиеся ключевыми или изгибами.

В первую очередь к таким пикселям можно отнести все пиксели, количество соседей у которых не равно двум. Если у пиксела нет соседей, то он является изолированным. Если у пиксела один сосед, то такой пиксел

является точкой окончания какого-либо элемента графического начертания символа. Если же количество соседей у пиксела больше двух, то такой пиксел, скорее всего, является точкой соединения нескольких элементов графического начертания.

Сложнее анализировать пиксел, который имеет два соседа. В таком случае пиксел может являться частью соединяющего ребра или точкой сочленения двух элементов графического начертания. Экспериментально был определен набор шаблонов для определения пикселей, которые являются ключевыми или изгибами.

Можно выделить первую группу шаблонов соседних пикселей, которая всегда соответствует ключевому пикселу или изгибу (см. рисунок 2.9), назовем эту группу шаблонов *α-группа*.

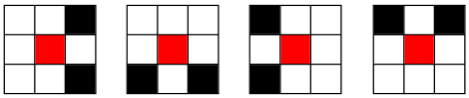


Рисунок 2.9 – α-группа шаблонов соседних пикселей

Для некоторых шаблонов соседних пикселей недостаточно информации лишь об их взаимном расположении. Необходимо также учесть, как расположены соседи непосредственных соседей центрального пиксела. Выделим некоторую группу таких шаблонов соседних пикселей, обозначив ее как *β-группу*:

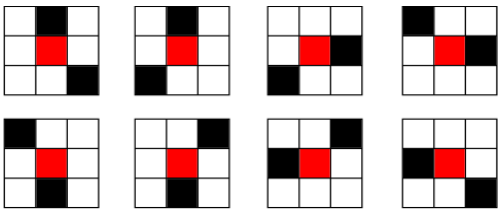


Рисунок 2.10 – β-группа шаблонов соседних пикселей

Как можно заметить, такие участки бинарного изображения могут соответствовать, как участку соединяющего ребра, так и точке стыка нескольких соединительных ребер. Экспериментально была обнаружена закономерность, согласно которой тип центрального пиксела может быть

определен, исходя из положения пикселей, которые являются соседними для соседей данного пикселя. Примеры случаев, когда шаблон β-группы соответствует ключевой точке или точке изгиба показаны на рисунке 2.11:

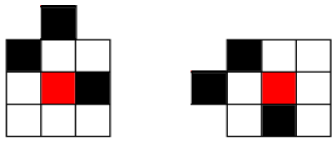


Рисунок 2.11 – Примеры случаев, когда шаблон β-группы не является частью соединяющего ребра

Как можно заметить, один соседний пиксел одного из соседей центрального пиксела является ключевым для определения типа участка бинарного изображения. Однако можно выделить еще одну группу шаблонов соседних пикселей, которая, как и β-группа, характеризуется зависимостью не только от непосредственных соседей центрального пиксела. Назовем такую группу шаблонов *γ–группой*. Формально, такая группа состоит лишь из двух достаточно тривиальных шаблонов (см. рисунок 2.12), которые, на первый взгляд, не могут соответствовать ничему, кроме участка соединяющего ребра.

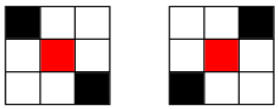


Рисунок 2.12 – γ-группа шаблонов соседних пикселей

Однако многочисленные эксперименты показали, что достаточно часто возникает необходимость в обнаружении случаев, когда подобные шаблоны соответствуют изгибам и даже ключевым пикселам. Определить такие случаи можно, если рассмотреть соседей сразу двух пикселей, соседних с центральным. На рисунке 2.13 показаны примеры подобных случаев.

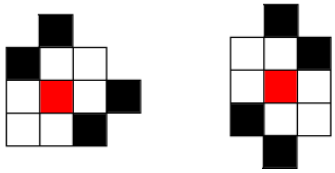


Рисунок 2.13 – Примеры случаев, когда шаблон γ–группы не является частью соединяющего ребра

Таким образом, были выделены три группы шаблонов расположения соседних пикселей: α-группа, β-группа и γ-группа, пиксели принадлежащие которым могут соответствовать, как ключевым пикселам, так и изгибам.

2.2.3. Алгоритм разделения ключевых пикселей и изгибов

Для решения задачи разделения ключевых пикселей и изгибов недостаточно рассматривать ограниченные окрестности этих пикселей, требуется анализ всего бинарного изображения. В частности, необходимо анализировать маршруты между найденными кандидатами в ключевые пиксели по черным пикселам скелетизированного бинарного изображения.

Фактически, скелетизированное бинарное изображение можно представить в виде графа. Черные пиксели этого изображения в таком представлении являются вершинами графа, а ребрами они соединены со всеми черными пикселями, граничащими с ними по стороне или углу. Таким образом, каждая из вершин такого графа может иметь не более восьми инцидентных ребер.

Запуск обхода такого графа в ширину одновременно из всех вершин, соответствующих пикселам α-, β- и γ-групп можно считать частным случаем использования алгоритма Ли для бинарного изображения. В ходе работы алгоритма от каждой из начальных вершин будет расходиться, так называемая, волна. Две встречные волны могут встретиться на середине соединяющего ребра, в таком случае, каждая из волн может носить информацию о вершине, из которой она начата, а также о количестве пикселов на пути от этой вершины до места встречи этих волн.

Однако, не смотря на то, что изображение скелетизировано, на нем все еще могут оставаться участки, где невозможно однозначно определить направление обхода вдоль каждого из ребер. На рисунке 2.14 приведен пример такого случая:

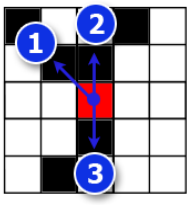


Рисунок 2.14 – Пример неоднозначности направления обхода вдоль ребер

На рисунке показаны три направления запуска волны из центрального пикселя, пронумерованные последовательными целыми числами от одного до трех. Первое и второе направления начинаются с двух смежных пикселей. На следующей итерации алгоритма Ли, при просмотре каждого из этих пикселей, будет рассмотрено ребро в пиксел другого направления. Однако, вопреки общей логике алгоритма, в этом случае не происходит встречи двух волн, соответствующих разным начальным направлениям. Наоборот, волны, в таком случае, еще не успели удалиться друг от друга на достаточное расстояние.

В качестве решения такой проблемы можно отдельно обрабатывать случай, когда рассматриваемый пиксел граничит со стартовой вершиной. В таком случае необходимо игнорировать ребра, ведущие в другие вершины, соседние со стартовой.

При таком подходе возможен случай, когда в определенном месте встретятся две различные волны, стартовавшие из одной вершины. В таком случае можно говорить о наличии петли – ребра направленного из этой вершины в нее же.

Можно заранее определить два правила, по которым однозначно определяются пиксели, являющиеся ключевыми, а не изгибами. Вершина соответствует ключевому пикселю, если:

- имеет инцидентную ей петлю;
- имеет количество исходящих соединяющих ребер отличное от двух.

В оставшихся случаях классификация вершин требует более тщательного анализа. К оставшимся случаям можно отнести лишь вершины,

имеющие ровно два исходящих соединительных ребра. Так как исходящих ребер всего два, угол между направлениями двух этих ребер играет определяющую роль при классификации типа пиксела. Если угол между двумя направляющими векторами исходящих из некоторой вершины соединяющих ребер не менее 120 градусов, то можно считать, что эта вершина соответствует изгибу.

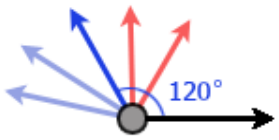


Рисунок 2.15 – Угол между направляющими векторами двух исходящих соединяющих ребер

На рисунке 2.15 черным цветом показан один из направляющих векторов некоторой вершины, синим – направления второго вектора, при которых данная вершина соответствует изгибу, красным – направления второго вектора, при которых данная вершина соответствует ключевому пикселу.

Для того чтобы определить направляющий вектор некоторого соединяющего ребра можно воспользоваться следующим принципом. Рассмотрим соединяющее ребро между двумя ключевыми пикселами как упорядоченную в порядке удаления от начальной точки последовательность пикселей. Поочередно проведем вектора из начального ключевого пикселя в каждый из пикселей пути. Последовательно просуммируем в порядке удаления от начального ключевого пиксела все вектора: первый с множителем равным единице; второй – с множителем 0,5; третий – с множителем 0,25; и так далее. Другими словами, каждый последующий вектор входит в результирующую сумму с множителем, вдвое меньше, чем предыдущий. Такой процесс взвешенного суммирования векторов проиллюстрирован на рисунке 2.16.

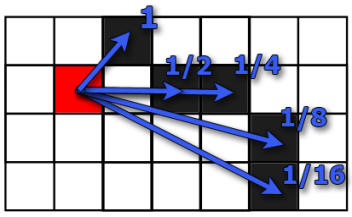


Рисунок 2.16 – Процесс взвешенного суммирования векторов

Таким образом, для каждой пары, состоящей из вершины и инцидентного ей соединяющего ребра, с использованием описанного метода взвешенного суммирования определяется направляющий вектор. Так как множитель уменьшается в геометрической прогрессии, пиксели, достаточно удаленные от начального, не оказывают почти никакого влияния на направление полученного вектора.

Опираясь на рассмотренные утверждения, можно сформулировать алгоритм разделения ключевых точек и изгибов.

Для поиска множества ключевых пикселей и множества пикселей-изгибов необходимо выполнить следующие шаги:

1. *Выделим на бинарном скелетизированном изображении все пиксели соответствующие α-, β- и γ-группам, обозначим все эти пиксели, как ключевые.*
2. *Запустим алгоритм Ли одновременно из всех ключевых пикселей.*
3. *Если в результате работы алгоритма Ли обнаружено, что те из пикселей, которые на этот момент считались ключевыми, на самом деле являются изгибами, обозначим их как изгибы и вернемся к шагу 2.*
4. *Если в результате работы алгоритма Ли не обнаружено новых изгибов, алгоритм завершен.*

Важно заметить, что для каждого последующего запуска алгоритма Ли множество ключевых пикселей, из которых будут одновременно запускаться волны, будет уменьшаться. Это обусловлено тем, что по результатам предыдущего запуска алгоритма, некоторые ключевые пиксели могут стать изгибами, однако изгибы уже не могут снова стать ключевыми пикселями. Следовательно алгоритм имеет ограниченное количество итераций.

Оценить вычислительную сложность такого алгоритма можно как $O(P \cdot K)$, где K – количество пикселей бинарного скелетизированного изображения, соответствующих α-, β- и γ-группам, а P – количество черных пикселей бинарного скелетизированного изображения. Для реальных скелетизированных бинарных изображений величина P намного меньше, чем $H \cdot W$, где W и H – ширина и высота изображения соответственно. Величина K же для реальных изображений намного меньше P и редко превышает 15 [139].

2.2.4. Алгоритм выделения композитных ребер

Ранее были разработаны алгоритмы для выделения ключевых точек и изгибов на скелетизированном бинарном изображении. Для построения полноценной структурной модели также необходимо выделить соединяющие ребра и объединить их в композитные, которые являются последовательностью соединяющих ребер от одного ключевого пиксела до другого.

Для того чтобы сформулировать алгоритм выделения композитных ребер на исходном изображении, необходимо предварительно заметить, что никакой изгиб не принадлежит более чем одному композитному ребру или петле. Это утверждение явно следует из того факта, что на пути между двумя вершинами, соответствующими соседним ключевым пикселям, все вершины соответствуют изгибам и могут иметь лишь два инцидентных соединяющих ребра.

Так как любой пиксел, соответствующий изгибу, принадлежит только одному композитному ребру, посещение в порядке обхода алгоритмом Ли этого пикселя не вызывает неоднозначности при выборе дальнейшего направления обхода. Таким образом, при запуске алгоритма Ли одновременно из всех ключевых пикселей, пиксели, являющиеся изгибами, не могут повлиять на детерминированность обхода. В таком случае две

волны, запущенные из двух ключевых пикселей, связанных композитным ребром, обязательно встретятся где-то между двумя этими пикселами. Если использовать ранее рассмотренный способ определения направляющего вектора волны и подсчета количества посещенных этой волной пикселей, можно сформулировать алгоритм выделения композитных ребер.

Для того чтобы выделить все композитные ребра на скелетизированном бинарном изображении с известным положением ключевых точек и изгибов, необходимо выполнить следующие шаги:

1. *Запустим алгоритм Ли одновременно из всех ключевых пикселей.*
2. *Для каждой комбинации стартовой вершины и начального направления движения заведем стек посещенных изгибов и количества шагов на момент их посещения.*
3. *Если на очередной итерации алгоритма волна посещает вершину, соответствующую изгибу, положим в стек метку этого изгиба.*
4. *Если на очередной итерации алгоритма встретились две волны, объединим два стека этих волн, получив итоговую последовательность изгибов в порядке их посещения и расстояния между двумя соседними изгибами. Для такой последовательности по начальным позициям волн однозначно восстанавливается информация о том, какие два ключевых пиксела соединяются через эту последовательность изгибов.*
5. *По окончании обхода алгоритмом Ли у нас имеется множество последовательностей изгибов, характеризующих пути между соединенными напрямую ключевыми пикселами. Рассмотрим кратчайшие расстояния $d_{u,v}$ между всеми парами соседних изгибов в каждой из последовательностей, а также количество пикселей $p_{u,v}$, посещенных между этими двумя изгибами. Обозначим за коэффициент искривления величину отношения $p_{u,v}$ к $d_{u,v}$. Эта величина в достаточной степени позволяет определить, насколько сильно выгнута линия, соединяющая два изгиба. Посчитаем эту величину для каждого из отрезков между двумя соединенными изгибами.*

6. *Преобразуем каждую из найденных последовательностей в массив соединяющих ребер. Каждое из таких ребер будет однозначно задаваться метками изгибов, которые ими соединяются, направляющим вектором начала и конца этого ребра, а также коэффициентом искривления. Полученный массив и является композитным ребром.*

Вычислительная сложность такого алгоритма составляет $O(P)$, где P – количество черных пикселей на скелетизированном бинарном изображении. Нетрудно убедиться в том, что каждый из P таких пикселей в ходе работы алгоритма будет рассмотрен один раз.

На рисунке 2.17 показаны примеры обычного композитного ребра и композитного ребра, являющегося петлей. Красным цветом показано одно из возможных направлений, в котором могли быть записаны посещенные изгибы для композитного ребра, синим – аналогичное направление для композитного ребра-петли.

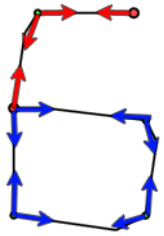


Рисунок 2.17 – Выделение композитных ребер

Композитные ребра и ключевые точки, которые ими соединяются, являются важными структурными составляющими. Именно на их основе можно создать структурную модель символа, которая в дальнейшем будет использоваться в алгоритмах распознавания.

2.3. Построение модели на основе выделенных структурных составляющих

Структурная модель символа, которая будет использоваться при распознавании, должна достаточно точно описывать графическое

представление символа. Если модель будет слишком общей, некоторые символы будут описываться одной и той же моделью. В то же время, если такая модель будет слишком точно описывать графическое представление символа, методы распознавания, в которых она будет использована, не будут инвариантны к несущественным отличиям в образе начертания: измененный наклон, менее острые углы или прерывистость некоторых линий.

Ранее было рассмотрено, как из исходного графического представления рукописного символа извлечь структурные составляющие: положение ключевых пикселей, положение изгибов, информацию о композитных ребрах, включающую коэффициенты искривления и направляющие вектора для каждого из соединяющих ребер.

Если считать все ключевые точки и изгибы вершинами некоторого графа G, а выделенные соединяющие ребра – его ребрами, то полученный граф G будет являться планарным графом, задающим топологическую модель символа. Такая модель не содержит информации о геометрических характеристиках графического представления.

Если дополнить полученную топологическую модель информацией о том, какая вершина соответствует ключевому пикселю, а какая – изгибу, можно существенно увеличить информативность этой модели [138].

Чтобы добавить в модель информацию о геометрических характеристиках графического представления, необходимо добавить положение каждой из вершин такого графа, а для каждого соединяющего ребра – информацию о направляющих векторах с обеих его сторон и коэффициенте искривления.

На рисунке 2.18 показана визуализация примеров таких моделей. Стрелками одного цвета показаны направляющие вектора для внутренних ребер одного композитного ребра. Как можно заметить, каждое соединяющее ребро внутри композитного задается двумя направленными на встречу друг другу векторами.

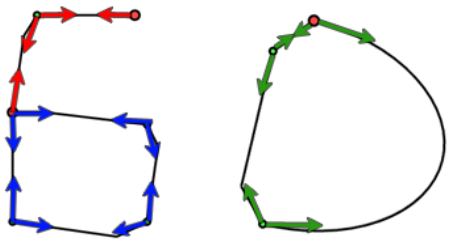


Рисунок 2.18 – Визуализация направляющих векторов внутри композитных ребер

Для дуги, изображенной на рисунке 2.18, коэффициент искривления будет наибольшим из всех соединяющих ребер, представленных на этом изображении. Для отрезков и близких к отрезкам ребер этот коэффициент будет чуть больше единицы.

Таким образом, путем добавления информации о видах вершин и геометрических характеристиках, из топологической модели представления символа была получена гораздо более информативная для распознавания структурная модель символа.

Для того чтобы методы, основанные на использовании такого рода структурных моделей, были инварианты к размеру исходного изображения, координаты вершин структурной модели преобразуются так, чтобы они оказались в интервале от нуля до единицы. Причем преобразование выполняется с одинаковым коэффициентом для обеих осей, чтобы сохранить исходные пропорции элементов структурной модели.

Важным моментом при применении структурной модели в процессе распознавания символов является использование коэффициента искривления. Этот коэффициент показывает, во сколько раз длина кривой, проведенной между двумя ключевыми пикселами больше, чем кратчайшее расстояние между ними. В совокупности с направляющим вектором этот коэффициент можно использовать, чтобы аппроксимировать соединяющее ребро некоторым графическим примитивом.

В случае если коэффициент искривления близок к единице, соединяющее ребро можно представить парой смежных отрезков,

направление первого из которых совпадает с направляющим вектором, второй же отрезок соединяет первый отрезок со вторым концом соединяющего ребра.

Соединяющие ребра с большей степенью искривления удобно представлять в виде дуги окружности. Дуга является графическим примитивом, для которого поиск пересечений с другими дугами и прямыми не является вычислительно затратной задачей. Вид дуги можно однозначно определить, зная коэффициент искривления и направляющий вектор. Умножив кратчайшее расстояние между двумя концами соединяющего отрезка на коэффициент искривления, можно получить длину дуги. Существует лишь два способа провести дугу фиксированной длины через две заданные точки. Дуги, полученные таким способам, будут отличаться лишь направлением обхода: по часовой стрелке или против. Выбрать нужное из этих двух направлений можно с помощью известного направляющего вектора. Необходимо выбрать то направление, при котором угол между направляющим вектором и вектором из начала этого вектора к середине дуги минимален. Однако лишь отрезков и дуг может оказаться недостаточно, чтобы достаточно качественно аппроксимировать соединяющие ребра.

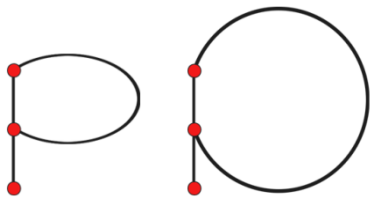


Рисунок 2.19 – Пример структурной модели с эллиптической дугой

На рисунке показан пример структурной модели с эллиптической дугой и аналогичной модели, где предпринята попытка обойтись без нее. Как можно заметить, иногда коэффициент искривления слишком велик, чтобы аппроксимация с использованием дуги окружности достаточно точно описывала исходное графическое представление символа. Следовательно, любое композитное ребро в структурной модели символа может быть

представлено набором, состоящим из графических примитивов одного из трех видов: отрезок, дуга и эллиптическая дуга.

В дальнейшем, для решения задачи распознавания рукописных символов достаточно единожды построенной структурной модели. Само графическое начертание символа после того, как по нему была построена эта модель, может не использоваться.

2.4. Критерии схожести структурных моделей символов

Тривиальным подходом к решению задачи распознавания символов является поочередное сравнение тестируемого графического образа с каждым из эталонных образов, классификация которых известна, и последующий выбор класса наиболее похожего эталонного образа. В общем случае для оценки степени схожести графических представлений двух символов на основе структурных моделей, необходимо построить структурную модель для каждого из этих символов в отдельности, после чего две полученные модели необходимо сравнить для оценки их степени схожести. Однако для всех графических представлений базы эталонных изображений можно заранее построить структурные модели и не затрачивать вычислительные мощности на их построение при каждом запуске процесса распознавания символа.

При таком подходе особенно важен выбор критерия схожести структурных моделей символов. Сравнение структурных моделей является трудно формализуемой задачей. Для того чтобы оценить, какой подход к использованию предложенных структурных моделей является наиболее успешным, было предложено применение трёх критериев схожести структурных моделей символов:

- критерий на основе метода пересечений;
- критерий на основе поиска максимального паросочетания минимального веса;

- критерий на основе вероятностного теста.

Каждый из этих критериев существенно отличается от остальных сложностью реализации, вычислительной сложностью и объемами потребляемой памяти.

2.4.1. Критерий схожести структурных моделей символов на основе метода пересечений

Метод пересечений заключается в анализе пересечений некоторого набора прямых с каждым из сравниваемых символов. Он нашел широкое применение в задачах распознавания символов, однако его применение существенно осложняется тем, что исходное графическое начертание символа задано в растровом представлении. Количество пересечений геометрической прямой на плоскости с растровым изображением не несет никакой информации о реальной пространственной ориентации изображенного объекта.

В общем случае пересечение будет выполняться с каждым из пикселей изображения в определенном порядке. Каждый из пикселей в растровом представлении, при переводе в привычное геометрическое представление, может быть представлен квадратом со сторонами, параллельными координатным осям. В таком случае необходимо оценивать не количество пересечений прямой с растровым изображением, а количество непрерывных участков пересечения с ним. При таком подходе возможны случаи, когда количество таких интервалов будет достаточно большим, но, на самом деле, прямая имела бы лишь одно пересечение с векторным представлением символа [140].

Как показано на рисунке 2.20, прямая многократно пересекает растровое представление символа, хотя векторное представление символа было бы пересечено лишь один раз.

Одним из решений такой проблемы является перевод каждой прямой используемого набора в растровое представление. Такой шаг позволяет избежать необходимости использовать операции с плавающей точкой. Поиск пересечения прямой с растровым представлением теперь может быть представлен в виде подсчета общих пикселей, принадлежащих одновременно и растровому представлению прямой, и исходному изображению [137].

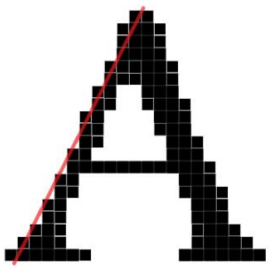


Рисунок 2.20 – Пример некорректной работы алгоритма пересечений для растрового представления символа

Однако стоит заметить, что перевод прямых в растровое представление так же приводит к потере существенной части информации. Кроме того, результат при таком подходе будет зависеть не только от выбранного набора прямых, но и от способа их растеризации.

Если же применить метод пересечений не к исходному графическому представлению символа, а к его структурной модели, то не возникает проблем с растровым представлением. Задача подсчета пересечений в таком случае сводится к тривиальному пересечению определенных графических примитивов. Еще одним преимуществом такого подхода является отсутствие необходимости отслеживать интервалы пересечений – все пересечения являются точками. Исключением являются, наложения прямых и отрезков, которые можно игнорировать, считая отсутствием пересечения.

Выбор набора прямых для пересечения, как и в случае с обработкой растровых представлений, зависит не только от решаемой задачи, но и от типа распознаваемых символов. Если на стадии настройки алгоритма имеется хорошая тестовая выборка изображений, то можно использовать какой-либо эволюционный алгоритм для выбора множества прямых [131].

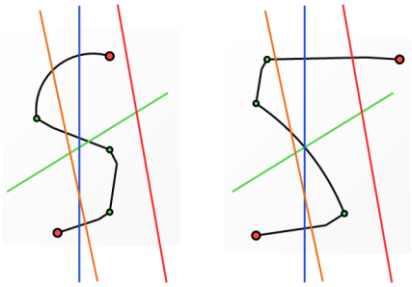


Рисунок 2.21 – Принцип работы метода пересечений на основе структурных моделей

На рисунке 2.21 показаны структурные модели двух символов, которые относятся к одному и тому же классу. Каждая из этих структурных моделей пересекается с одинаковым набором прямых. Как можно заметить, все прямые, кроме красной, имеют одинаковое количество пересечений с первой и второй структурными моделями. Однако красная прямая пересекает вторую модель, но не пересекает первую. Это объясняется тем, что верхний горизонтальный отрезок графического представления правой структурной модели существенно длиннее, чем аналогичный участок левой модели. С точки зрения привычного для нас представления о распознавании символов, такое отличие несущественно, однако, при определенном выборе множества прямых в методе пересечений такое отличие может оказаться определяющим при выполнении классификации. Для того чтобы минимизировать влияние такого недостатка метода пересечений на качество классификации, необходимо использовать достаточно большой набор прямых.

Наиболее простой алгоритм получения вектора признаков для оценки степени схожести структурных моделей на основе метода пересечений может быть сформулирован в виде следующей последовательности шагов:

1. *Предварительно сгенерируем набор прямых для оценки количества пересечений с графическими представлениями символов. Для генерации можно использовать эвристические правила или эволюционные алгоритмы.*

2. *Будем рассматривать все прямые набора в определенном порядке. Выберем первую в порядке рассмотрения прямую* l_i.

3. Изначально полагаем счетчик количества пресечений для прямой l_i равным нулю.

4. Поочередно выполним пересечение прямой l_i с каждым из графических примитивов, представляющих соединяющие ребра структурной модели. Для каждого графического примитива, для которого существует единственная точка пересечения прямой l_i, необходимо выполнить увеличение счетчика количества пересечений для этой прямой на единицу.

5. Выберем следующую в порядке рассмотрения прямую l_i и перейдем к шагу 3. Если все прямые рассмотрены, то перейдем к шагу 6.

6. Объединим счетчики для всех прямых в вектор, который, по своей сути, является вектором признаков для дальнейшей классификации структурных моделей. Алгоритм закончен.

Вычислительная сложность алгоритма на основе метода пересечений для структурных моделей, при таком подходе, оценивается как $O(E \cdot L)$, где E – количество соединяющих ребер в структурной модели, а L – количество прямых, используемых для подсчета количества пересечений. Такая вычислительная сложность характерна для тривиального подхода, при котором каждое соединяющее ребро структурной модели пересекается с каждой из прямых заданного набора.

Если использовать тот факт, что пересечение элементов структурной модели выполняется не со всей прямой, а только с той ее частью, которая умещается внутри квадрата с противоположными сторонами в точках (0; 0) и (1; 1), можно существенно увеличить быстродействие алгоритма. Например, можно использовать охватывающие прямоугольники для всех сравниваемых пар геометрических примитивов. Если охватывающие прямоугольники двух графических примитивов имеют нулевую площадь пересечения, то можно не решать вычислительно затратную геометрическую задачу проверки двух этих примитивов на пересечение.

Для улучшения быстродействия алгоритма на основе метода пересечений для структурных моделей можно также применять k-d деревья.

Такой подход существенно осложнит программную реализацию, однако, позволит существенно ускорить процесс обнаружения пересечений графических примитивов.

2.4.2. Критерий схожести структурных моделей символов на основе нахождения максимального паросочетания минимального веса

Еще одним подходом к оценке степени схожести является подход, основанный на сопоставлении каждому композитному ребру одной структурной модели наиболее близкого по графическому представлению композитного ребра другой структурной модели.

Оценка степени схожести двух композитных ребер является нетривиальной задачей, учитывая, что у них могут отличаться количество составляющих их графических примитивов, типы этих примитивов, их суммарная длина, а также местоположение соединяемых изгибов и ключевых точек.

Однако существует способ, с помощью которого можно определить площадь фигуры, заключенной между двумя композитными ребрами, не смотря на существенные различия в их характеристиках. Эту площадь с противоположным знаком можно использовать как меру схожести между ребрами: чем больше площадь, тем больше отличаются данные ребра.

Помимо площади на оценку степени схожести должно влиять расстояние между соответствующими точками композитных ребер – чем дальше находятся эти точки, тем меньше похожи соответствующие композитные ребра.

Рассмотрим композитное ребро как маршрут, по которому за одну секунду проходит некоторых объект. Скорость объекта такова, что за эту секунду он пройдет полный путь от одного конца этого композитного ребра до другого. Пусть объекты начали свое движение одновременно вдоль каждого из ребер. В каждый из возможных моментов времени t проведем

отрезок, соединяющий положения объектов. Совокупность таких отрезков и образует искомую площадь, заключенную между двумя композитными ребрами.

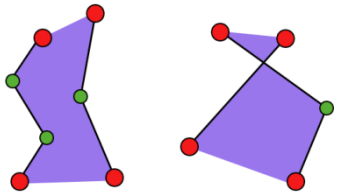


Рисунок 2.22 – Площадь, заключенная между двумя композитными ребрами в качестве схожести этих ребер

Как показано на рисунке 2.22, даже если композитные ребра имеют общие точки, такой способ подсчета площади между ними может быть использован для оценки степени схожести.

Рассмотрим случай, когда описанным способом требуется найти площадь, заключенную между двумя отрезками. Пусть первый отрезок задан двумя своими концами (x_a, y_a) и (x_b, y_b), аналогично второй – двумя своими концами (x_c, y_c) и (x_d, y_d). Тогда в некоторый момент времени t объект на первом отрезке будет находиться в точке (x_1, y_1), на втором – в точке (x_2, y_2). Координаты точки (x_1, y_1) могут быть найдены с помощью формулы (2.1).

$$\begin{cases} x_1 = x_a + (x_b - x_a) \cdot t \\ y_1 = y_a + (y_b - y_a) \cdot t \end{cases} \quad (2.1)$$

Аналогично, используя формулу (2.2), можно определить координаты точки (x_2, y_2):

$$\begin{cases} x_2 = x_c + (x_d - x_c) \cdot t \\ y_2 = y_c + (y_d - y_c) \cdot t \end{cases} \quad (2.2)$$

Квадрат расстояния между ними описывается формулой (2.3):

$$d^2 = (x_1 - x_2)^2 + (y_1 - y_2)^2 \quad (2.3)$$

Значение первого слагаемого формулы (2.3) можно представить с использованием формулы (2.1):

$$\begin{gathered} (x_1 - x_2)^2 = ((x_a - x_c) + (x_b + x_c - x_a - x_d) \cdot t)^2 = \\ (A_x + B_x \cdot t)^2 = {B_x}^2 t^2 + 2 \cdot A_x \cdot B_x \cdot t + {A_x}^2 \end{gathered} \quad (2.4)$$

Аналогичным образом значение второго слагаемого может быть представлено подстановкой формулы (2.2):

$$(y_1 - y_2)^2 = ((y_a - y_c) + (y_b + y_c - y_a - y_d) \cdot t)^2 = \\ = (A_y + B_y \cdot t)^2 = {B_y}^2 t^2 + 2 \cdot A_y \cdot B_y \cdot t + {A_y}^2 \quad (2.5)$$

Используя (2.4) и (2.5), можно получить итоговую формулу для вычисления расстояния между объектами в произвольный момент времени, которая будет иметь вид:

$$d = \sqrt{\left({B_x}^2 + {B_y}^2\right) \cdot t^2 + 2 \cdot \left(A_x B_x + A_y B_y\right) \cdot t + \left({A_x}^2 + {A_y}^2\right)} = \\ = \sqrt{a \cdot t^2 + b \cdot t + c} \quad (2.6)$$

Используя формулу (2.6), можно найти площадь, заключенную между двумя рассматриваемыми отрезками:

$$S = \int_{t_1}^{t_2} \sqrt{a \cdot t^2 + b \cdot t + c}\, dt = \\ = \frac{1}{8 \cdot a^{3/2}} \cdot \left(\left(2 \cdot \sqrt{a} \cdot (2 \cdot a \cdot t_2 + b) \cdot \sqrt{t_2 \cdot (a \cdot t_2 + b) + c}\right) - \right. \\ -(b^2 - 4 \cdot a \cdot c) \cdot \log\left(2 \cdot \sqrt{a} \cdot \sqrt{t_2 \cdot (a \cdot t_2 + b) + c} + 2 \cdot a \cdot t_2 + b\right) - \\ \left. - \left(2 \cdot \sqrt{a} \cdot (2 \cdot a \cdot t_1 + b) \cdot \sqrt{t_1 \cdot (a \cdot t_1 + b) + c}\right) + \right. \\ \left. +(b^2 - 4 \cdot a \cdot c) \cdot \log(2 \cdot \sqrt{a} \cdot \sqrt{t_1 \cdot (a \cdot t_1 + b) + c} + 2 \cdot a \cdot t_1 + b)\right). \quad (2.7)$$

Формула (2.7) может быть использована для участков пары композитных ребер от t_1 до t_2, на которых нет стыков двух соединяющих ребер, при условии, что оба соединяющих ребра на этом отрезке являются отрезками.

Чтобы избежать излишне трудоемких вычислений определенных интегралов для дуг и эллиптических дуг, можно аппроксимировать их последовательностью отрезков [98]. В таком случае композитное ребро будет представлено все так же последовательностью соединяющих ребер, каждое из которых будет являться отрезком, но количество их может быть существенно больше.

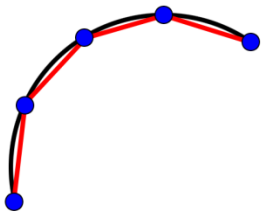


Рисунок 2.23 – Аппроксимация дуги набором отрезков

При таком подходе будут использоваться не точные значения площади, а приближенные. Однако такая погрешность не слишком критична для задачи оценки степени схожести, но разработка и реализация такого подхода существенно проще, чем аналогичная реализация без аппроксимации дуг наборами отрезков.

После того, как для каждой пары композитных ребер посчитана площадь, заключенная между ними, эту площадь можно использовать как меру различия между этими ребрами: чем больше эта площадь, тем больше отличаются эти два ребра. Чтобы получить меру схожести достаточно умножить полученную площадь на минус единицу.

Для каждой пары композитных ребер из разных структурных моделей известна их степень схожести. Можно построить двудольный граф G, в котором вершинами будут являться композитные ребра исходных структурных моделей. К первой доле графа G будут относиться вершины, которые соответствуют композитным ребрам из первой структурной модели, ко второй доле – вершины, которые соответствуют второй модели.

Для того чтобы оценить степень схожести двух структурных моделей, необходимо вершинам первой доли сопоставить вершины второй доли так, чтобы было образовано наибольшее возможное количество пар. Из всех таких распределений по парам необходимо выбрать разбиение, при котором сумма степеней схожести будет максимальной. Так как степень схожести, по сути, является величиной площади с обратным знаком, то такая задача будет равнозначна задаче поиска максимального паросочетания минимального веса в двудольном графе [99], при условии, что ребро такого графа будет иметь вес, равный площади между соответствующими композитными ребрами.

Стоит отметить, что доли построенного графа могут содержать разное количество вершин. В таком случае какие-то вершины не войдут в максимальное паросочетание. Для каждой из таких вершин к общему весу найденного паросочетания необходимо добавить удвоенный минимальный вес инцидентного ему ребра в качестве «штрафа» за несоответствие количества композитных ребер у двух моделей.

Задача нахождения максимального паросочетания минимального веса в двудольном графе сводится к задаче поиска максимального потока наименьшей стоимости в сети [100]. Для этого необходимо построить новый граф G', который будет являться сетью с одним истоком и одним стоком. Чтобы получить такой граф из исходного двудольного графа достаточно добавить две вершины. Первая из этих вершин – исток, ее необходимо соединить ориентированными ребрами со всеми вершинами из первой доли. Все такие ребра должны быть направлены от истока к вершинам первой доли. Вторая добавляемая вершина – сток, ее необходимо соединить ориентированными ребрами со всеми вершинами второй доли. Направление таких ребер будет всегда от вершин второй доли к стоку. Все ребра между долями заменяются аналогичными ориентированными ребрами из первой доли во вторую. Структура полученного графа G' показана на рисунке 2.24:

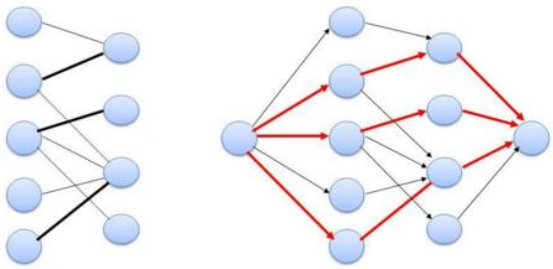


Рисунок 2.24 – Схема сети, полученной из двудольного графа

Так граф G' является сетью, для которой необходимо определить величину максимального потока, для каждого из ее ребер определена величина пропускной способности. В данном случае достаточно каждому из ребер присвоить единичную пропускную способность. Это обусловлено тем,

что если какое-либо ребро уже задействовано в текущем паросочетании, то оно больше не должно участвовать при образовании новых пар.

В полученной сети необходимо найти максимальный поток наименьшей стоимости [101]. Стоимость – величина, характеризующая каждое из ребер этой сети. Стоимость ребра отражает стоимость включения этого ребра в найденный поток. Для построенной сети G' стоимость всех ребер из истока и всех ребер в сток равна нулю. Стоимости ребер, которые проводятся между вершинами долей, соответствуют весам ребер в исходном двудольном графе G.

Одним из наиболее известных методов нахождения максимального потока наименьшей стоимости является алгоритм увеличивающих путей [102]. По своей сути этот алгоритм очень похож на алгоритм Эдмонса-Карпа нахождения максимального потока [103].

Для нахождения максимального потока наименьшей стоимости в сети G' необходимо выполнить следующие шаги:

1. *Предварительно подготовим граф G', добавив для каждого ребра обратное ребро с нулевой пропускной способностью и стоимостью, равной стоимости исходного ребра с противоположным знаком.*

2. *Используя в качестве весов ребер их стоимости, найдем кратчайший путь от истока к стоку.*

3. *Если не существует ни одного пути от истока к стоку, алгоритм завершен, найденный к данной итерации поток является максимальным потоком минимальной стоимости.*

4. *Пустим поток по найденному маршруту. Для этого рассмотрим все его ребра. Определим то, пропускная способность которого минимальна, обозначим ее за f. Уменьшим пропускную способность всех ребер найденного пути на f, одновременно увеличивая пропускную способность обратных им ребер на эту же величину.*

5. *Для обновленной сети снова выполним шаг 2.*

Вычислительная сложность такого алгоритма $O(V^3E)$, где V – количество вершин в сети, E – количество ребер в ней, включая добавленные на первом шаге алгоритма обратные ребра. Такая оценка характерна, при использовании алгоритма Дейкстры для поиска кратчайшего пути. Быстродействие алгоритма можно улучшить, используя для реализации алгоритма Дейкстры бинарную кучу [104].

Так как количество вершин V в полученной сети G', равняется E_1+E_2+2, то при большом количестве композитных ребер в двух структурных моделях нахождение максимального потока наименьшей стоимости может существенно замедлить процесс распознавания символов. На практике количество композитных ребер в каждой из моделей редко превышает 10, поэтому алгоритм несущественно влияет на общее быстродействие процесса распознавания.

Для оценки степени схожести можно использовать и жадные алгоритмы нахождения максимального паросочетания минимального веса. Так как каждая из вершин первой доли связана с каждой из вершин второй доли, такой алгоритм всегда будет находить максимальное паросочетание, однако общий вес такого паросочетания не всегда будет являться минимальным. Тем не менее, паросочетание, найденное таким способом, можно использовать в качестве приближенной оценки степени схожести.

2.4.3. Критерий схожести структурных моделей символов на основе вероятностного теста

Еще одной группой подходов к сравнению структурных моделей символов, которые представляют научный и практический интерес, является группа вероятностных методов. Такие методы, по своей сути, являются тестами двух структурных моделей на схожесть.

Критерий на основе одного из таких методов было предложено применить для оценки степени схожести структурных моделей символов.

Представим, что в некоторой точке структурной модели находится некоторый объект, который способен выполнять следующие операции:

- переместиться вдоль некоторого композитного ребра на расстояние d в направлении единичного вектора v;
- переместиться к ближайшей ключевой точке.

Первая операция должна выполняться только в том случае, если существует композитное ребро, направление которого отличается от заданного не более чем на определенную величину δ. Если таких ребер несколько, то выбирается композитное ребро, направление которого ближе всего к направлению, заданному вектором v. Расстояние d, на которое необходимо переместиться объекту, выбирается таким образом, чтобы оно было меньше самого короткого соединяющего ребра. На практике эта величина составляет порядка 8×10^{-2}.

Один запуск теста заключается в размещении объекта внутри каждой из структурных моделей в точке, наиболее близкой к заданному положению (x_s, y_s), и последующим выполнением набора операций двух описанных ранее типов. После того, как выполнение операций заканчивается, расстояние между объектами на первой и на второй структурных моделях и является мерой различия: чем дальше расположены объекты, тем меньше соответствующие структурные модели похожи между собой.

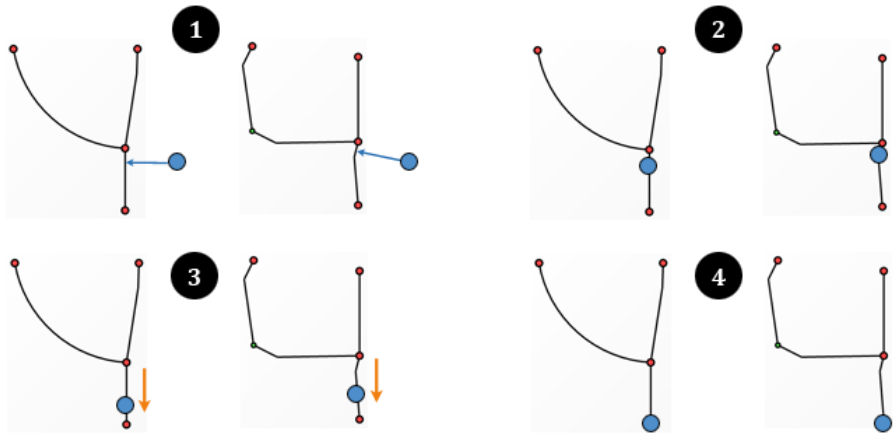


Рисунок 2.25 – Принцип работы теста для оценки степени схожести двух структурных моделей: начальное состояние (1); смещение к наиболее

близкой точке структурной модели (2); перемещение вдоль композитного ребра на величину d (3) и перемещение к ближайшей ключевой точке (4)

На рисунке 2.25 представлена визуализация принципа работы теста на схожесть.

Сам алгоритм классификации структурной модели на основе вероятностного теста может быть сформулирован в виде следующей последовательности шагов:

1. Случайным образом зададим некоторое начальное положение (x_s, y_s).

2. Случайным образом зададим набор команд двух описанных типов.

3. Для структурной модели каждого из эталонных изображений выполним шаги с 4-го по 6-й.

4. Разместим объект в ключевой точке рассматриваемой модели, которая наиболее близка к заданному начальному положению (x_s, y_s).

5. Поочередно выполним все команды заданного набора для рассматриваемой структурной модели.

6. Запишем итоговое положение объекта для рассматриваемой структурной модели.

7. Рассмотрим одну из ранее не рассмотренных структурных моделей соответствующую изображению, которое необходимо классифицировать.

8. Если не рассмотренных структурных моделей не осталось, то алгоритм завершен.

9. Иначе для рассматриваемой модели выполним шаги с 4-го по 5-й и пометим ее, как рассмотренную.

10. Определим структурную, модель соответствующую эталонному изображению, такую, что расстояние между сохраненным для нее итоговым положением и итоговым положением для рассматриваемой модели минимально.

11. Отнесем классифицируемое изображение к классу, соответствующему найденному эталонному изображению.

12. Перейдем к шагу 7.

Выбор количества операций в наборе, как и самих операций, является ключевым при реализации такого метода. Как показывает практика, случайный набор из 1000 операций позволяет адекватно оценить степень схожести двух структурных моделей, однако, при фиксированном количестве команд в наборе и при имеющейся в наличии выборке изображений символов с известными классами, для построения наиболее эффективного набора команд можно использовать генетический алгоритм с целочисленным типом кодирования [105, 106].

Каждый ген при таком типе кодирования будет задавать отдельную операцию и кодироваться лишь одним байтом. Структура такого гена показана на рисунке 2.26:

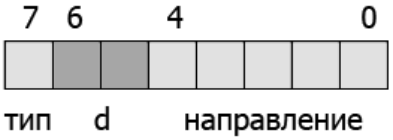


Рисунок 2.26 – Структура гена, задающего операцию

Старший бит такого гена отвечает за тип операции: 0 – первый тип, 1 – второй тип. Два последующих бита образуют число k от 0 до 3, которое задает величину шага d: значение k соответствует величине шага $d = 0{,}04 \cdot (k + 1)$. Таким образом, наибольшее возможное значение d равно 0,16; а наименьшее – 0,04. Оставшиеся пять бит гена, образуя число t от 0 до 31, задают угол, соответствующий направлению v. Число t используется для вычисления величины угла α на тригонометрическом круге

$$\alpha = \frac{2 \cdot \pi \cdot t}{32}. \qquad (2.8)$$

Угол α, вычисленный по формуле (2.8), однозначно задает вектор движения v для первого типа операций.

Значения d и α должны быть проигнорированы при исполнении, если операция имеет второй тип. Однако биты, соответствующие этим значениям по-прежнему могут участвовать в скрещивании и мутации.

Для операции скрещивания можно использовать стандартный для целочисленного кодирования оператор двухточечного кроссинговера [107]. Мутацию при целочисленном кодировании следует выполнять путем инвертирования отдельных битов гена [108].

Используя описанный подход на основе генетического алгоритма, можно получить определенное множество наборов операций, каждый из которых может быть использован в качестве одной итерации теста на схожесть двух структурных моделей. Получив результаты в виде итогового расстояния между объектами для всех итераций теста, можно отбросить четверть наибольших расстояний, а также четверть наименьших расстояний. Сумма оставшихся расстояний и будет являться итоговым результатом теста – оценкой степени различия двух структурных моделей.

Для каждой итерации теста начальное положение (x_s, y_s) выбирается случайным образом, поэтому точность оценки, в общем случае, зависит не только от множества наборов команд, но и от случайно выбранных начальных положений.

Достаточно сильная зависимость результата оценки степени схожести для двух структурных моделей от набора случайных значений является одним из наиболее существенных недостатков такого подхода. Преимуществом такого метода является возможность для определенного множества наборов операций определить заранее итоговое положение объектов на структурных моделях, соответствующих эталонным изображениям символов.

2.5. Алгоритм сегментации рукописного текста на основе построения структурных моделей

Решать задачу распознавания отдельных рукописных символов существенно проще, чем решать аналогичную задачу распознавания для символов, которые являются частью рукописного текста [109, 110]. В таком случае необходимо не только решать задачу распознавания рукописных символов, но и задачу сегментации рукописного текста – выделения из него отдельных символов и соединительных элементов начертания.

2.5.1. Анализ требований к алгоритму сегментации на основе построения структурных моделей

При выполнении начертания рукописного текста почерки различаются не только способом начертания отдельных символов, но и способом выполнения отдельных элементов, соединяющих последовательные буквы текста [111, 112, 113]. Процесс сегментации осложняется характерными для многих почерков искажениями начертаний символов при соединении их графического представления с последующим и предыдущим символами [114].

Для выполнения сегментации рукописного текста необходимо проанализировать исходное начертание символа, определив принадлежность каждой из областей графического представления участка рукописного текста к определенному символу [115]. Рассмотреть все возможные распределения областей графического представления участка рукописного символа не представляется возможным ввиду огромного количества различных распределений даже для растрового представления начертания [116].

Рассмотренный ранее алгоритм построения структурной модели начертания отдельного символа можно применить для построения структурной модели начертания целого слова. Как и в случае с отдельными

символами, в данном случае необходимо предварительно выполнить скелетизацию исходного начертания символа.

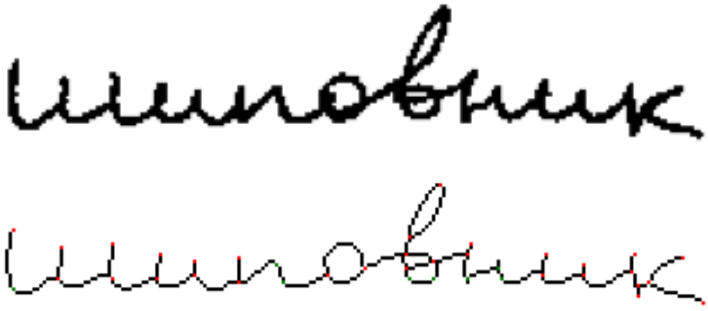


Рисунок 2.27 – Пример изображения отдельного слова рукописного текста и построенного для него скелета

В ходе работы алгоритма будет выполнено выделение структурных составляющих, в том числе ключевых точек и изгибов, от местоположения которых можно отталкиваться при выборе границ, разделяющих области начертания последовательных символов сегментируемого текста.

Если рассмотреть примеры структурных моделей отдельных слов рукописного текста, то можно выделить несколько визуальных особенностей этих моделей.

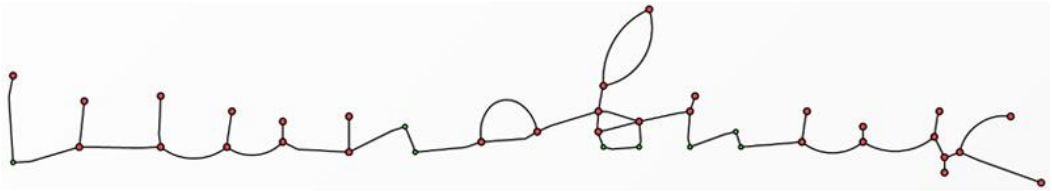


Рисунок 2.28 – Пример структурной модели отдельного слова рукописного текста

В первую очередь стоит отметить, что большинство разделительных линий между соседними буквами слова можно провести через середину соединяющего ребра. Причем любым из концов такого ребра может быть как ключевая точка, так и изгиб. На рисунке 2.29 можно увидеть случаи, когда оба конца являются ключевой точкой, а также случай, когда один из концов является изгибом.

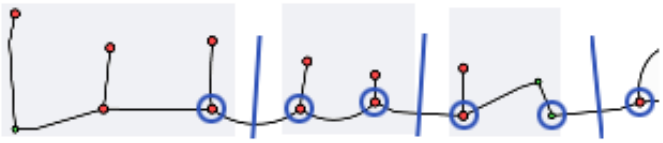


Рисунок 2.29 – Один из вариантов сегментации участка слова

На рисунке 2.29 также показано, что, при таком способе выбора разделительных линий, часть соединительного элемента иногда следует считать частью некоторого символа, а иногда – следует считать частью изображения, которая не относится ни к одному его символу. Например, вторая буква участка слова, показанного на рисунке 2.29, является строчной буквой «и» и, как правило, даже в отдельности от других символов изображается с участком соединительного элемента на конце.

Существуют и изображения, для которых возможен вариант сегментации, при котором разделительная линия может быть проведена через ключевые точки или изгибы. Такое возможно в случаях, когда при начертании отсутствует необходимость в использовании соединительных элементов. Пример такого варианта сегментации показан на рисунке 2.30.

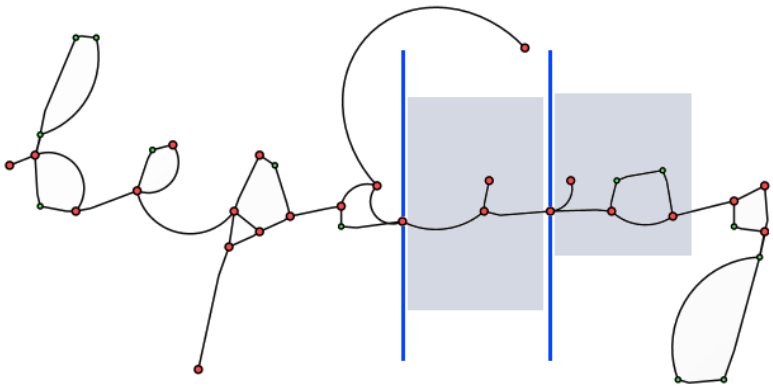


Рисунок 2.30 – Сегментация слова с разделительными линиями, проходящими через ключевые точки

Анализ структурных моделей отдельных слов рукописного текста показал, что точками интереса являются ключевые точки, изгибы, а также середины соединяющих их ребер.

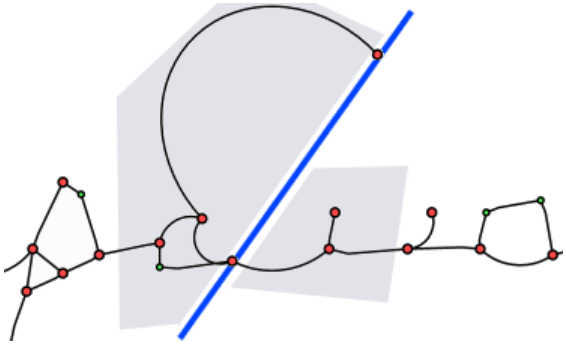


Рисунок 2.31 – Пример необходимости в использовании невертикальной разделительной линии

Как можно заметить, не всегда возможно разделить соседние символы только вертикальными линиями. Иногда графическое представление одного символа может частично находиться над или под графическим представлением соседнего ему символа. Пример такого изображения показан на рисунке 2.31.

Такие случаи возможны не только из-за особенностей графического представления отдельных символов, но и по причине особенностей почерка, связанных с большей степенью наклона букв.

По результатам перечисленных наблюдений можно сформулировать список требований к алгоритму сегментации рукописного текста:

- возможность учитывать присутствие на изображении соединительных элементов между соседними символами;
- возможность отнесения отдельных участков соединительных элементов к начертанию соединяемых символов;
- использование невертикальных прямых в качестве разделяющих соседние символы линий;
- учет возможного различия наклона начертаний различных символов в одном слове.

Кроме того, список требований может быть дополнен типичными требованиями к алгоритмам обработки изображений:

- полиномиальная зависимость времени работы от входных данных;

- полиномиальная зависимость объема потребляемой оперативной памяти от входных данных.

2.5.2. Алгоритм выделения возможных разделяющих линий на структурной модели

Как было ранее отмечено, точками интереса для выбора местоположений границ между соседними символами слова являются ключевые точки, изгибы и центры соединяющих ребер. Однако задать разделяющую прямую, проходящую через некоторую точку интереса, можно, лишь выбрав вторую точку.

Для решения задачи сегментации было сделано предположение о том, что выбор второй точки можно осуществить так же из множества точек интереса.

Процесс построения множества прямых, подмножество которых в итоге станет разделяющими линиями, можно представить в виде перебора всех возможных точек интереса и перебора в пару каждой из них точки для проведения прямой. Для определенности можно полагать, что изначально перебирается нижняя из двух точек, а любая рассмотренная прямая будет впоследствии задаваться вектором из нижней точки в верхнюю. При таком подходе каждая прямая разделяет плоскость структурной модели на две полуплоскости: левую и правую. По большей части элементы структурной модели, принадлежащие правой полуплоскости, относятся к части слова справа от разделительной линии, а оставшиеся, соответственно, – к левой от разделительной линии части. Стоит отметить, что далее будут сделаны уточнения, почему не все точки полуплоскостей могут быть отнесены к соответствующим частям.

Несложно заметить, что в результате работы описанного ранее перебора будет получено $O(K^2)$ прямых, которые могут являться разделяющими линиями при сегментации, где K – количество точек интереса

в структурной модели. Быстродействие любого алгоритма сегментации, опирающегося на данное множество прямых, так или иначе, будет зависеть от размера этого множества. Следовательно, имеет смысл отсеять как можно больше заведомо некорректных прямых, которые не могут являться разделительными линиями.

Анализ имеющихся структурных моделей рукописных слов позволил выявить три типа прямых, которые можно не рассматривать в качестве возможных соединяющих линий:

- горизонтальные или близкие к горизонтальным прямые;
- прямые, которые образованы вектором, пересекающим хотя бы одно соединяющее ребро ровно в одной точке;
- прямые, пересекающие более двух соединяющих ребер.

Далее для выбранной прямой следует сформулировать четкий принцип, по которому следует определять принадлежность каждого из элементов структурной модели к левой или правой от данной прямой части этой модели. Для этого следует опираться не только на геометрические характеристики прямой, но и на местоположение концов вектора, которым задается эта прямая. Как уже ранее было сказано, задающий прямую вектор направлен снизу вверх. В таком случае мы можем четко определить два значения y_1 и y_2 координаты на вертикальной оси Y, между которыми находится вектор. Если значение y-координаты ключевой точки или изгиба находится между значениями y_1 и y_2, то ее принадлежность к левой или правой от разделительной линии части определяется тем, слева или справа данная точка находится от образующего вектора. Для оставшихся точек можно сформулировать следующий алгоритм определения принадлежности к левой или правой части:

1. *Положим некоторый планарный граф G равным графу топологической модели структурной модели.*

2. Если в структурной модели существует ребро, которое пересекается с отрезком текущей разделяющей прямой между y_1 и y_2, то удалим из графа G соответствующее ему ребро.

3. Покрасим в черный цвет вершины графа G, соответствующие ключевым точкам или изгибам, которые находятся слева от образующего текущую разделяющую линию вектора и имеют значение y-координаты между значениями y_1 и y_2.

4. Аналогично, покрасим в белый цвет вершины графа G, соответствующие ключевым точкам или изгибам, которые находятся справа от образующего текущую разделяющую линию вектора и имеют значение y-координаты между значениями y_1 и y_2.

5. Для каждой вершины графа G определим компоненту связности, к которой она относится.

6. Рассмотрим произвольную ранее не рассмотренную компоненту связности графа G.

7. Если в рассматриваемой компоненте связности имеются раскрашенные вершины только одного цвета, то покрасим все вершины этой компоненты связности в этот цвет и перейдем к шагу 9.

8. Если в рассматриваемой компоненте одновременно имеются вершины черного и белого цвета, то пометим текущую разделяющую линию, как некорректную и завершим работу алгоритма.

9. Если в графе G имеются нерассмотренные компоненты связности, перейдем к шагу 6, в противном случае алгоритм закончен.

Описанные критерии позволяют выделить на исходной структурной модели все возможные разделяющие линии, которые, в свою очередь, образуют множество линий-кандидатов *L*.

2.5.3. Алгоритм выбора подмножества разделяющих линий для сегментации структурной модели

Задача сегментации отдельного слова рукописного текста была сведена к задаче сегментации структурной модели, которая, впоследствии, была сведена к задаче выбора подмножества прямых из множества линий-кандидатов L, которыми будет выполнено разделение структурной модели слова на отдельные участки, соответствующие символам.

Для каждой прямой известны: ее уравнение, начало и конец вектора, которым образована эта прямая, а также подмножества ключевых точек и изгибов, по каждую из сторон этой разделяющей прямой.

Обозначим как *G(S)* – значение степени схожести участка структурной модели, содержащего все точки интереса из множества S и соединяющие ребра, оба конца которых находятся в этом множестве, с наиболее похожей эталонной структурной моделью символа.

Введем функцию *F(S)* – максимальное суммарное значение степеней схожести при разделении множества точек интереса S некоторым подмножеством разделяющих линий из множества L.

Для того чтобы определить значение функции *F(S)* для некоторого множества точек интереса S, необходимо рассмотреть все разделяющие прямые множества L, которые разбивают множество S на два непустых подмножества. Пусть прямая l_i разбивает множество S на непустые подмножества $S_1^{(i)}$ и $S_2^{(i)}$. Тогда, рассматривая все допустимые прямые l_i, можно вычислить значение F(S), используя рекуррентную формулу (2.9):

$$F(S) = \max(G(S), \max_i F\left(S_1^{(i)}\right) + F\left(S_2^{(i)}\right) + P_i), \quad (2.9)$$

где P_i – величина штрафа за неиспользование участка структурной модели в процессе классификации.

Суть выражения (2.9) заключается в выборе варианта разбиения множества S таким образом, чтобы максимизировать суммарную величину степени схожести. Рассматриваются все варианты разбиения разделяющими

прямыми множества *L* и вариант, при котором все точки множества *S* относятся к одному символу, и, следовательно, дальнейшее разбиение не производится.

Вычисление функции *F(S)* ввиду рекуррентности ее аналитического представления имеет экспоненциальную вычислительную сложность. Однако использование кеширования уже посчитанных значений функции *F(S)* для определенных множеств *S* существенно улучшает быстродействие алгоритма вычисления этой функции. Высокое быстродействие вычислений с использованием кеширования объясняется малым количеством возможных множеств *S*, для которых необходимо вычислить значение функции *F(S)*. В свою очередь, малое количество возможных множеств можно объяснить спецификой способа их получения. Каждый раз, за редкими исключениями, множество точек интереса, фактически, разбивается на две подмножества: по левую и по правую сторону от разделяющей линии. Таким образом, количество различных множеств на практике имеет квадратичную зависимость от количества точек интереса.

Подход к вычислению функции *F(S)* можно охарактеризовать, как применение метода динамического программирования [117]. При таком подходе состояние будет кодироваться некоторой хэш-функцией от множества *S*, а количество состояний, на практике, квадратично зависит от количества элементов исходного множества, для которого необходимо вычислить значение функции.

В качестве хэш-функции можно использовать легко вычислимый полиномиальный хэш, который позволит представить каждое из множеств в виде соответствующего 64-битного целого числа. Хэширование множества можно выполнить, хэшируя упорядоченный массив номеров точек интереса, которые входят в данное множество.

2.6. Основные результаты и выводы по главе 2

1. Разработан высокопроизводительный алгоритм предварительной обработки входного изображения рукописного символа, включающий стадии гистограммной эквализации, адаптивной бинаризации методом Оцу и скелетизации.

2. Предложен оригинальный подход к скелетизации бинарного изображения на основе алгоритмов скелетизации Зонга-Суня и Ву-Цая, обладающий высоким быстродействием, в том числе предложена собственная операция предварительной обработки бинарного изображения с целью устранения плоских окончаний элементов графического представления.

3. Предложен оригинальный алгоритм выделения структурных составляющих на бинарном скелетизированном изображении на основе многократного обхода скелетизированного бинарного изображения с использованием алгоритма Ли.

4. Предложен оригинальный алгоритм построения структурной модели рукописного символа на основе выделенных структурных составляющих: ключевых точек, изгибов, соединяющих и композитных ребер.

5. Предложены три оригинальных алгоритма оценки степени схожести структурных моделей рукописных символов: на основе метода пересечений, на основе максимального паросочетания минимального веса и вероятностный тест для оценки степени схожести.

6. Предложен алгоритм сегментации рукописного текста на основе построения структурной модели с применением метода динамического программирования.

ГЛАВА 3. ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ДЛЯ РАСПОЗНАВАНИЯ РУКОПИСНЫХ СИМВОЛОВ В УСЛОВИЯХ МАЛОЙ ОБУЧАЮЩЕЙ ВЫБОРКИ

В данной главе приведено описание программной реализации разработанных алгоритмов. Представлен обзор библиотек для обработки изображений. Приведено обоснование выбора языка программирования и средств для программной реализации алгоритмов. Представлено описание разработанного программного обеспечения.

3.1. Общие требования к разрабатываемому программному обеспечению

Современные требования к качеству программного обеспечения подразумевают, что разрабатываемая программная система должна обладать рядом характеристик:

- корректное решение задачи распознавания рукописных символов программной системой;
- корректная реакция программной системы на некорректные значения входных параметров;
- высокое быстродействие программного приложения, отсутствие использования неполиномиальных алгоритмов;
- программная система должна использовать конечное количество оперативной памяти и не должна являться причиной ее утечек;
- программа должна обладать модульной архитектурой, а каждый из модулей должен использоваться для решения отдельной задачи.

Разрабатываемая программная система для распознавания рукописных символов должна соответствовать следующим функциональным требованиям:

1. Возможность выполнения предварительной обработки исходного изображения с целью устранения шума на исходном изображении, а также его бинаризации.

2. Выполнение операции скелетизации бинарного изображения без потери важных структурных элементов графического начертания символа на основе алгоритмов с полиномиальной вычислительной сложностью.

3. Выделение структурных составляющих на скелетизированном бинарном изображении и последующее построение структурной модели символа на их основе.

4. Выполнение распознавания входных изображений с начертаниями рукописных символов с использованием базы структурных моделей эталонных начертаний.

5. Предоставление пользователю данных о результатах решения задачи распознавания входного изображения.

6. Возможность изменения эталонной базы структурных моделей эталонных графических представлений символов.

3.2. Выбор языка программирования для разработки программного обеспечения

При выборе языка программирования для реализации описанного программного обеспечения необходимо руководствоваться рядом требований:

1. Язык программирования должен обладать высокой производительностью – быстродействие алгоритмов теории графов и компьютерной геометрии может существенно варьироваться в зависимости от языка.

2. Язык программирования должен предоставлять разработчику достаточный набор инструментов для реализации хранения необходимых

данных и оперативного доступа к ним: абстрактные типы данных, шаблонизированные контейнеры, функторы и т. д.

3. Для языка программирования должно существовать множество готовых библиотек и решений с реализацией алгоритмов вычислительной математики, компьютерного зрения и компьютерной геометрии.

4. Язык программирования должен предоставлять разработчику возможность самостоятельно выделять и высвобождать память, использовать бинарные операции, самостоятельно оперировать с указателями на необходимые области памяти.

Из-за требований к производительности для реализации алгоритмов построения и сравнения структурных моделей не подходят сценарные языки программирования, обладающие высокими функциональными возможностями, но имеющие невысокую производительность.

Достаточно низкую производительность имеют и языки программирования Java и C#, которые чрезмерно часто оперируют с объектами, что приводит к дополнительным временным затратам на исполнение кода.

Язык программирования Pascal не имеет достаточного набора инструментов для хранения данных с возможностью оперативного доступа к ним, а также не имеет стандартной библиотеки алгоритмов. Кроме того, язык почти не развивается из-за того, что был изначально разработан как язык для обучения основам программирования.

Одним из наиболее популярных языков программирования на сегодняшний день является C++. Этот язык нашел широкое применение для реализации алгоритмов, требующих максимального быстродействия. Существенным преимуществом языка C++ является наличие стандартной библиотеки шаблонов STL, которая предоставляет полезный набор функций и шаблонизированных контейнеров, позволяющих выполнять операции добавления и удаления элементов с логарифмической или константной вычислительной сложностью.

Язык программирования C++ предоставляет разработчику полный контроль над выделением и распределением памяти, обладает высокой степенью гибкости. Поэтому этот язык был выбран в качестве инструмента разработки программного обеспечения для распознавания рукописных символов.

3.3. Выбор средств разработки программного обеспечения

Для языка программирования C++ существует множество библиотек с реализацией методов вычислительной математики, искусственного интеллекта и компьютерного зрения в частности. Наиболее известной из них является библиотека компьютерного зрения OpenCV (Open Source Computer Vision Library) [118]. В ней представлены средства для чтения, записи и интерпретации изображений с различным количеством каналов и глубиной цвета. Библиотека OpenCV содержит большое количество классов и функций для обработки изображений, получения векторов признаков, а также для классификации образов.

Библиотека OpenCV реализована на языках C, C++, также разрабатывается для Python, Java, Matlab и других языков и содержит следующие модули:

CXCORE – модуль, который содержит базовые структуры, а также: алгоритмы работы с памятью; алгоритмы преобразования типов данных; алгоритмы работы с матрицами; алгоритмы работы с 2D объектами.

CV – модуль, предназначенный для обработки изображений. Содержит: алгоритмы для обработки и анализа изображений; алгоритмы слежения за объектами; алгоритмы распознания объектов; алгоритмы, предназначенные для калибровки камер.

ML – модуль машинного обучения. Содержит: алгоритмы, предназначенные для классификации образов и анализа данных.

HighGUI – модуль, предназначенный для создания пользовательского интерфейса. Поддерживает: создание окон, вывод изображений, захват видео.

CVAUX – модуль, предназначенный для описания пространственного зрения. Содержит: алгоритмы описания черт лица, описания текстур.

CVCAM – модуль, предназначенный для захвата видео с цифровых камер.

Особый интерес для системы распознавания символов представляют реализации алгоритмов адаптивной бинаризации и гистограммной эквализации, а также общие методы для чтения, конвертирования, хранения и записи изображений различных типов в библиотеке OpenCV.

Еще одной популярной библиотекой для языка программирования C++ является математическая библиотека Eigen – библиотека шаблонов для работы с векторами, матрицами, алгоритмами вычислительной математики, линейной алгебры и аналитической геометрии [119].

К преимуществам этой библиотеки можно отнести возможность работы с векторами и матрицами достаточно больших размеров, в том числе и разреженными. Большинство алгоритмов реализовано с использованием возможностей графических ускорителей.

Стоит так же отметить наличие в библиотеке Eigen большого количества функций, реализующих математические алгоритмы и хороший уровень документации.

Для реализации описанного программного обеспечения особый интерес представляют методы аналитической геометрии и вычислительной математики для оценки степени схожести двух структурных моделей на основе информации об их графических примитивах.

3.4. Разработка программной системы

При разработке программного обеспечения был сделан выбор в пользу модульной архитектуры. Такой подход с использованием нескольких отдельных модулей, каждый из которых решает свою задачу, позволяет упростить процессы разработки и тестирования.

Разработанная программная система получила название «Mediocre OCR Recognition» и имеет модульную архитектуру [143]. Каждый из модулей содержит функции, имеющие схожее функциональное назначение.

3.4.1. Модули разработанной программной системы

Взаимосвязь модулей программной системы может быть представлена с помощью диаграммы компонентов, представленной на рисунке 3.1.

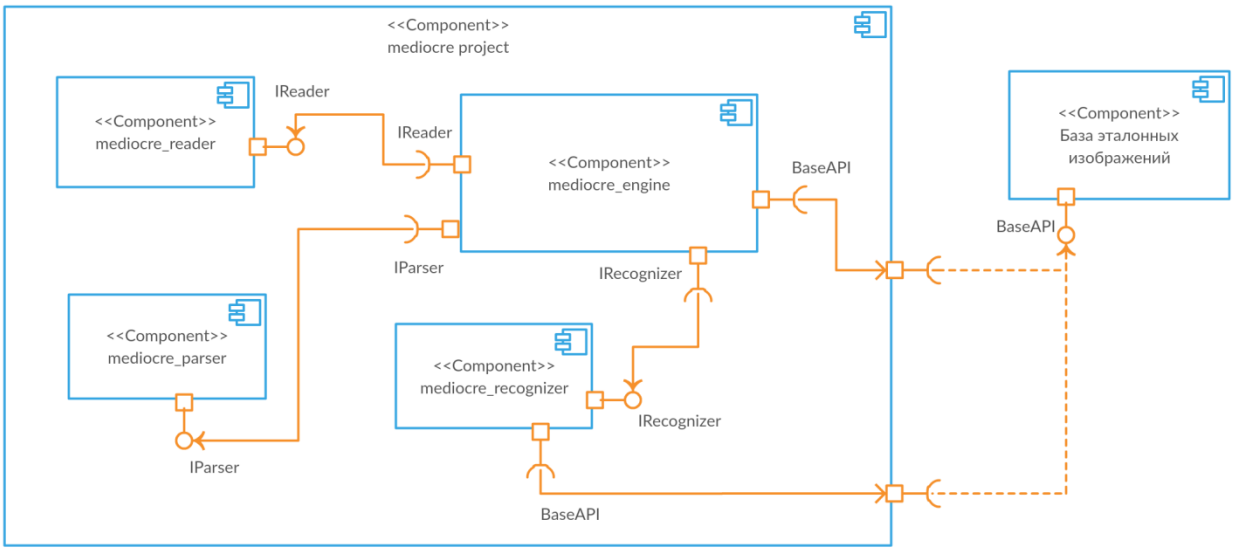


Рисунок 3.1 – Диаграмма компонентов разработанной программной системы

В таблице 3.1 приведено краткое описание каждого из реализованных в программной системе модулей:

Таблица 3.1 – Описание основных модулей программной системы

Название модуля	Описание модуля
`mediocre_engine`	Основной исполняемый модуль. Модуль, принимающий значения входных параметров

	от пользователя и выполняющий все этапы классификации входного изображения рукописного символа. Поочередно обращается к модулям, выполняющим последовательные шаги процесса распознавания рукописных символов.
`mediocre_reader`	Модуль чтения исходного изображения, изменения его размеров, конвертации в простейший растровый формат. Также выполняет стадии первичной обработки исходного изображения: гистограммную эквализацию и адаптивную бинаризацию. Результат работы этого модуля может быть использован в качестве входных данных модулем `mediocre_parser`.
`mediocre_parser`	Модуль извлечения структурных составляющих и последующего построения структурных моделей на их основе. Получает на вход бинарное изображение в растровом формате. Отвечает за предварительную скелетизацию растрового бинарного изображения, выделение структурных составляющих: ключевых точек, изгибов, соединяющих и композитных ребер, а также построение структурных моделей. Модуль также выполняет сохранение полученных моделей на диск в формате XML.
`mediocre_recognizer`	Модуль распознавания рукописных символов. Оперирует набором структурных моделей соответствующих эталонным и входному

	изображениям. Позволяет выполнять классификацию любым из трех описанных ранее способов оценки степени схожести структурных моделей. Результатом является отчет в виде XML-файла, который содержит информацию о результатах распознавания.

Для корректной работы программной системы необходимо, чтобы рабочая станция соответствовала ряду системных требований представленных в таблице 3.2.

Таблица 3.2 – Минимальные системные требования

Название параметра	**Характеристика**
Операционная система	Windows 7 или новее
Центральный процессор	С тактовой частотой не ниже 2.2 ГГц поддерживающий технологию SSE2
Объем ОЗУ	Не менее 1 Гб
Место на жестком диске	Не менее 100 Мб

Стоит отметить, что возможно собрать проект «Mediocre OCR Recognition» в операционной системе Linux, так как используемые библиотеки OpenCV и Eigen являются кроссплатформенными.

3.4.2. Классы для реализации модуля распознавания рукописных символов

В модуле распознавания рукописных символов реализованы все разработанные в данной работе алгоритмы для выделения структурных составляющих, построения структурных моделей, а также оценки степени схожести двух структурных моделей для оценки степени их схожести. Для их реализации используются классы и контейнеры стандартной библиотеки шаблонов STL языка программирования C++, приведенные в таблице 3.3. Кроме того, для программной реализации модуля были реализованы классы

mediocre_image, mediocre_comparator и mediocre_math, отвечающие за реализацию логики процесса распознавания.

Таблица 3.3 – Описание используемых классов и контейнеров стандартной библиотеки шаблонов STL

Название класса	**Описание**
vector	Последовательный контейнер, реализующий динамический массив с поддержкой автоматического изменения размера с выделением и освобождением оперативной памяти. Поддерживает прямой доступ и связанное хранение. В модуле распознавания используется для хранения коллекций объектов, для которых не имеется необходимости в осуществлении эффективного поиска, а также быстрого добавления и удаления элементов с сохранением упорядоченности.
pair	Класс для хранения пар значений произвольных типов. Для объектов этого класса определены операторы сравнения. В модуле распознавания используется для хранения парных геометрических величин, например, координат изгибов и ключевых точек.
set	Контейнер уникальных элементов, реализующий множество. Реализован на основе красно-черного дерева, что позволяет добавлять и удалять элементы этого контейнера за логарифмическое время. В модуле распознавания используется для эффективной реализации алгоритмов выделения ключевых точек и изгибов с целью хранения множеств

	объектов, обладающих определенными свойствами.
`map`	Контейнер, реализующий словарь – аналог ассоциативного массива. Предназначен для хранения соответствий вида «ключ – значение» без повторяющихся ключей. В модуле распознавания используется для биекции между координатами различных пикселей и группами, к которым они принадлежат при реализации алгоритмов выделения структурных составляющих и построения структурных моделей.
`queue`	Контейнер, реализующий очередь по принципу FIFO (First In, First Out). В модуле распознавания используется для эффективной реализации алгоритмов скелетизации Зонга-Суня и Ву-Цая.

Реализованные классы `mediocre_image`, `mediocre_comparator` и `mediocre_math` позволяют полностью инкапсулировать логику процессов построения структурных моделей и их последующего сопоставления.

На рисунке 3.2 приведена диаграмма классов модуля распознавания символов. Данная диаграмма отражает микроархитектуру разработанного модуля, иллюстрирует основные классы для реализации предложенного алгоритма распознавания символов, содержащие поля и методы для выполнения основных операций, лежащих в основе предложенного метода.

Класс mediocre_image

Класс `mediocre_image` предназначен для реализации экземпляра изображения графического представления отдельного символа. Данный

класс содержит поля для описания всех атрибутов исходного графического представления, а также структурной модели изображенного символа.

В классе mediocre_image реализованы вложенные классы `edge`, `compressed_edge`, `vertex`, `compressed_vertex`, `composite_edge`, `compressed_composite_edge`, необходимые для хранения и изменения информации о структурной модели изображенного символа. Описание данных классов приведено в таблице 3.4.

Таблица 3.4 – Описание вложенных классов класса `mediocre_image`

Название класса	**Описание**
`edge`	Класс для хранения информации о соединяющем ребре между двумя изгибами или ключевыми точками: идентификаторы соединяемых ключевых точек или изгибов, длина и коэффициент самого ребра, направляющие вектора.
`compressed_edge`	Класс для хранения информации о соединяющем ребре между двумя изгибами или ключевыми точками, после сжатия символа. Отличается от класса `edge` тем, что использует вещественные типы данных для хранения значений атрибутов. Необходим для сохранении структурной модели символа в формате XML способом, инвариантным к изменению масштаба.
`vertex`	Класс для хранения информации о ключевой точке или изгибе: координаты пикселя на исходном изображении, тип

	вершины (ключевая точка или изгиб), список идентификаторов инцидентных ребер.
`compressed_vertex`	Класс для хранения информации о ключевой точке или изгибе, координаты которой преобразованы после сжатия символа. Отличается от класса `vertex` тем, что использует вещественные типы данных для хранения значений координат. Необходим для сохранении структурной модели символа в формате XML способом, инвариантным к изменению масштаба.
`composite_edge`	Класс для хранения информации о композитном ребре: начальная и конечная ключевые точки, список соединяющих ребер этого композитного ребра, а также общая длина и коэффициент искривления.
`compressed_composite_edge`	Класс для хранения информации о композитном ребре, атрибуты которого преобразованы после сжатия символа. Отличается от класса `composite_edge` тем, что использует вещественные типы данных для хранения значений атрибутов. Необходим для сохранении структурной модели символа в формате XML способом, инвариантным к изменению

	масштаба.

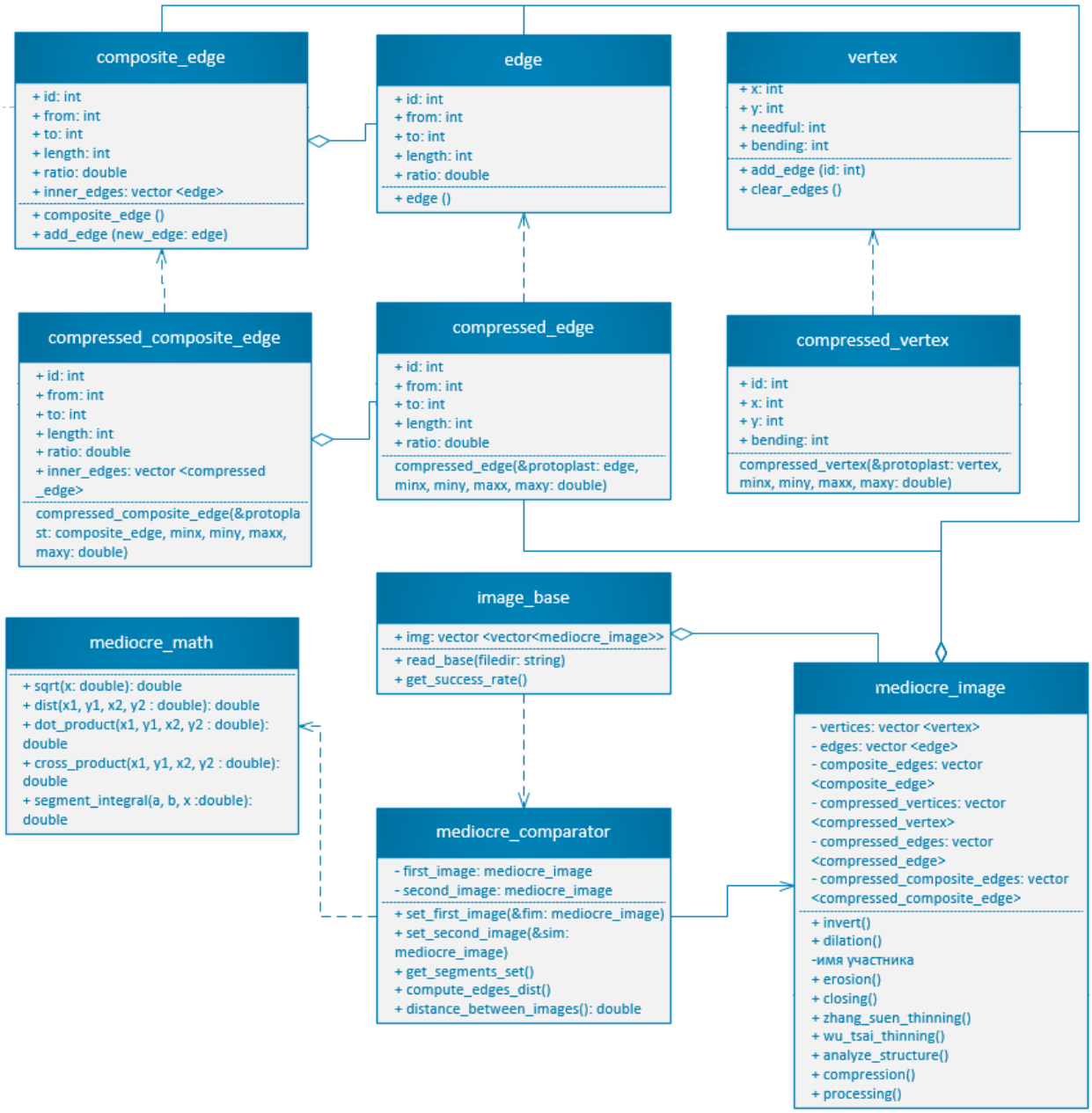


Рисунок 3.2 – Диаграмма классов модуля распознавания символов

Основные атрибуты и переменные класса `mediocre_image` предназначены для хранения информации об исходном изображении, структурных моделях до и после сжатия графического представления символа таким образом, чтобы все значения координат находились в интервале от нуля до единицы. Некоторые поля класса используются для хранения информации о сжатии графического представления символа.

Основные атрибуты и переменные класса `mediocre_image` представлены в таблице 3.5.

Таблица 3.5 – Основные атрибуты и переменные класса mediocre_image

Атрибут/переменная	**Описание**
`vector<vector<int>> original_pixels`	Двумерная бинарная матрица, задающая исходное изображение.
`vector<vector<int>> pixels`	Двумерная матрица, задающая изображение, полученное после скелетизации. Отдельными значениями отмечены ключевые точки и изгибы.
`vector<vertex> vertices`	Вектор всех ключевых точек и изгибов структурной модели.
`vector<edge> edges`	Вектор всех соединяющих ребер структурной модели.
`vector<composite_edge> composite_edges`	Вектор всех композитных ребер структурной модели.
`vector<compressed_vertex> compressed_vertices`	Вектор всех ключевых точек структурной модели после сжатия исходного представления символа.
`vector<compressed_edge> compressed_edges`	Вектор всех соединяющих ребер структурной модели после сжатия исходного представления символа.
`vector<compressed_ composite_edge> compressed_composite_ edges`	Вектор всех композитных ребер структурной модели после сжатия исходного представления символа.
`vector<composite_vertex> composite_bendings`	Вектор всех изгибов структурной модели после сжатия исходного

	представления символа.
`double original_zoom`	Значение коэффициента изменения масштаба при сжатии исходного изображения
`double original_x_shift`	Значение коэффициента изменения масштаба по оси X при сжатии исходного изображения
`double original_y_shift`	Значение коэффициента изменения масштаба по оси Y при сжатии исходного изображения

В классе `mediocre_image` имеются методы для чтения исходного изображения из файла, выполнения над исходным изображением морфологических операций, инверсии и скелетизации, а также методы выделения ключевых точек и изгибов, выделения структурных составляющих, построения структурной модели, а также сохранения и загрузки структурной модели из файла. Описание основных методов класса `mediocre_image` приведено в таблице 3.6.

Таблица 3.6 – Основные методы класса `mediocre_image`

Метод	**Описание**
`invert`	Реализация операции инверсии бинарного изображения.
`dilation`	Реализация морфологической операции дилатации.
`erosion`	Реализация морфологической операции эрозии.
`closing`	Реализация морфологической операции закрытия.
`zhang_suen_thinning`	Реализация однопроходного алгоритма скелетизации Зонга-Суня.

`wu_tsai_thinning`	Реализация однопроходного алгоритма скелетизации Ву-Цая.
`analyze_structure`	Реализация выделения ключевых точек, изгибов, соединяющих ребер, а также их объединения в композитные ребра.
`compression`	Реализация сжатия графического представления символа и последующего построения структурной модели инвариантной к масштабу исходного начертания символа.
`processing`	Метод, объединяющий вызов всех последовательных шагов процесса построения структурной модели по исходному графическому представлению.
`read_from_bmp`	Реализация чтения исходного изображения символа из растрового изображения.
`read_from_xml`	Реализация чтения структурной модели символа из XML-файла.
`print_to_xml`	Реализация записи структурной модели символа в XML-файл.
`cross_dilation`	Реализация операции устранения плоских окончаний графического представления символа на бинарном изображении для дальнейшей скелетизации.

Каждый объект класса `mediocre_image` используется в качестве экземпляра отдельного изображения символа в совокупности с информацией о соответствующей ему структурной модели.

Для создания базы эталонных изображений символов достаточно сохранить их структурные модели в виде XML-файлов. При программной реализации загрузки базы эталонных изображений достаточно объявить контейнер объектов класса `mediocre_image` и для каждого из его элементов вызвать метод `read_from_xml`.

Класс mediocre_comparator

Класс `mediocre_comparator` предназначен для выполнения операции оценки степени схожести для двух объектов класса `mediocre_image`, представляющих структурные модели символов, которые необходимо сравнить.

Основные атрибуты и переменные класса `mediocre_comparator` приведены в таблице 3.7.

Таблица 3.7 – Атрибуты и переменные класса `mediocre_comparator`

Атрибут/переменная	Описание
`mediocre_image first_image`	Объект, описывающий исходное изображение первого сравниваемого символа, а также соответствующую ему структурную модель.
`mediocre_image second_image`	Объект, описывающий исходное изображение второго сравниваемого символа, а также соответствующую ему структурную модель.
`vector<compressed_vertex> first_vertices`	Полный список всех вершин структурной модели первого сравниваемого символа.
`vector<compressed_vertex> second_vertices`	Полный список всех вершин структурной модели второго сравниваемого символа.
`vector`	Полный список всех композитных

<compressed_composite_edge> first_edges	ребер структурной модели первого сравниваемого символа.
vector <compressed_composite_edge> second_edges	Полный список всех композитных ребер структурной модели второго сравниваемого символа.
vector<vector< pair<double, double>>> first_segments	Набор отрезков, аппроксимирующих каждое из композитных ребер структурной модели первого сравниваемого символа.
vector<vector< pair<double, double>>> second_segments	Набор отрезков, аппроксимирующих каждое из композитных ребер структурной модели второго сравниваемого символа.
Vector<vector<double>> edges_dist	Матрица для кеширования попарных расстояний между парами композитных ребер структурных моделей первого и второго сравниваемых символов.

В классе mediocre_comparator реализованы методы для инициализации процесса оценки степени схожести структурных моделей двух сравниваемых символов, а также непосредственно для выполнения самого процесса сравнения. Основные методы класса mediocre_comparator приведены в таблице 3.8.

Таблица 3.8 – Основные методы класса mediocre_comparator

Метод	**Описание**
set_first_image	Метод для задания структурной модели первого сравниваемого символа.
set_second_image	Метод для задания структурной модели второго сравниваемого символа.

`set_images`	Метод для одновременного задания пары структурных моделей сравниваемых символов.
`get_segments_set`	Реализация процесса преобразования композитных ребер любого состава в набор последовательных отрезков, в том числе и аппроксимирующих дуги.
`dist_between`	Реализация вычисления оценки степени схожести двух композитных ребер на основе площади, заключенной между двумя последовательностями отрезков.
`compute_edges_dist`	Реализация и сохранения в матрицу для кеширования расстояний между всеми парами композитных ребер двух сравниваемых структурных моделей.
`distance_between_images`	Метод вычисления расстояния между двумя структурными моделями. Выбор алгоритма сравнения зависит от аргументов, передаваемых методу. По умолчанию используется метод на основе нахождения максимального паросочетания минимального веса.
`print_xml_reports`	Реализация сохранения XML-файлов структурных моделей сравниваемых символов в выбранную директорию под выбранными именами.

Метод `print_xml_reports` может быть использован для изучения причин неправильной классификации, например, если будет вызываться каждый раз, когда результат классификации отличается от эталонного.

Класс mediocre_math

Класс `mediocre_math` предназначен для выполнения необходимых в процессе работы алгоритма распознавания математических операций и реализации геометрической логики.

В данном классе не содержится никаких атрибутов и полей, а содержатся лишь методы, которые используются оставшимися классами для математических вычислений.

Основные методы класса `mediocre_math` приведены в таблице 3.9.

Таблица 3.9 – Основные методы класса `mediocre_math`

Метод	Описание
`sqr`	Метод для вычисления квадрата величины, заданной аргументом.
`dist`	Метод для вычисления величины расстояния между двумя точками на плоскости. Реализована также перегрузка, принимающая в качестве аргументов объекты типа `pair`.
`dot_product`	Метод вычисления скалярного произведения двух двумерных векторов. Как правило, используется для вычисления угла между двумя прямыми графическими примитивами.
`cross_product`	Метод вычисления векторного произведения двух двумерных векторов. Как правило, используется для определения направления ориентированного угла между двумя векторами.
`move_point`	Вспомогательный метод, вычисляющий конечное положение материальной точки

	при выполнении определенной части маршрута. Необходим при вычислении площади, заключенной между двумя последовательностями отрезков.
`segment_integral`	Метод для вычисления интегральной суммы между двумя отрезками сравниваемых последовательностей.

Класс `mediocre_math` позволяет инкапсулировать всю логику математических вычислений и отделить ее от алгоритмов распознавания рукописных символов на основе построения структурных моделей.

3.5. Реализация сохранения структурной модели символа на жесткий диск в виде XML-файла

Во избежание необходимости многократной обработки одних и тех же изображений символов с целью выделения структурных составляющих и построения структурной модели, при разработке была предусмотрена возможность сохранения полученной структурной модели на жесткий диск в виде XML-файла. Также были реализованы средства для загрузки сохраненной модели из XML-файла в оперативную память.

Для реализации работы с XML-файлами была использована библиотека pugixml [120]. Библиотека позиционируется как легковесный простой и производительный парсер XML-файлов для приложений, написанных на языке программирования C++. К ее преимуществам так же можно отнести портируемость на различные операционные системы, включающие Windows, Linux, MacOS и другие.

Библиотека pugixml распространяется под разрешающей лицензией MIT и может быть использована при разработке закрытого программного обеспечения при условии, что текст лицензии будет предоставлен вместе с ним.

3.5.1. Содержимое XML-файла с описанием структурной модели символа

Сам XML-файл содержит теги для описания определенных элементов структурной модели: ключевых точек, изгибов, соединяющих и композитных ребер. Дополнительно в XML-файле содержатся сведения об исходном бинарном изображении и результате его предварительной обработки: скелетизации, сжатии, определении ключевых пикселей.

Теги, используемые для описания структурной модели и исходного графического представления символа, приведены в таблице 3.10.

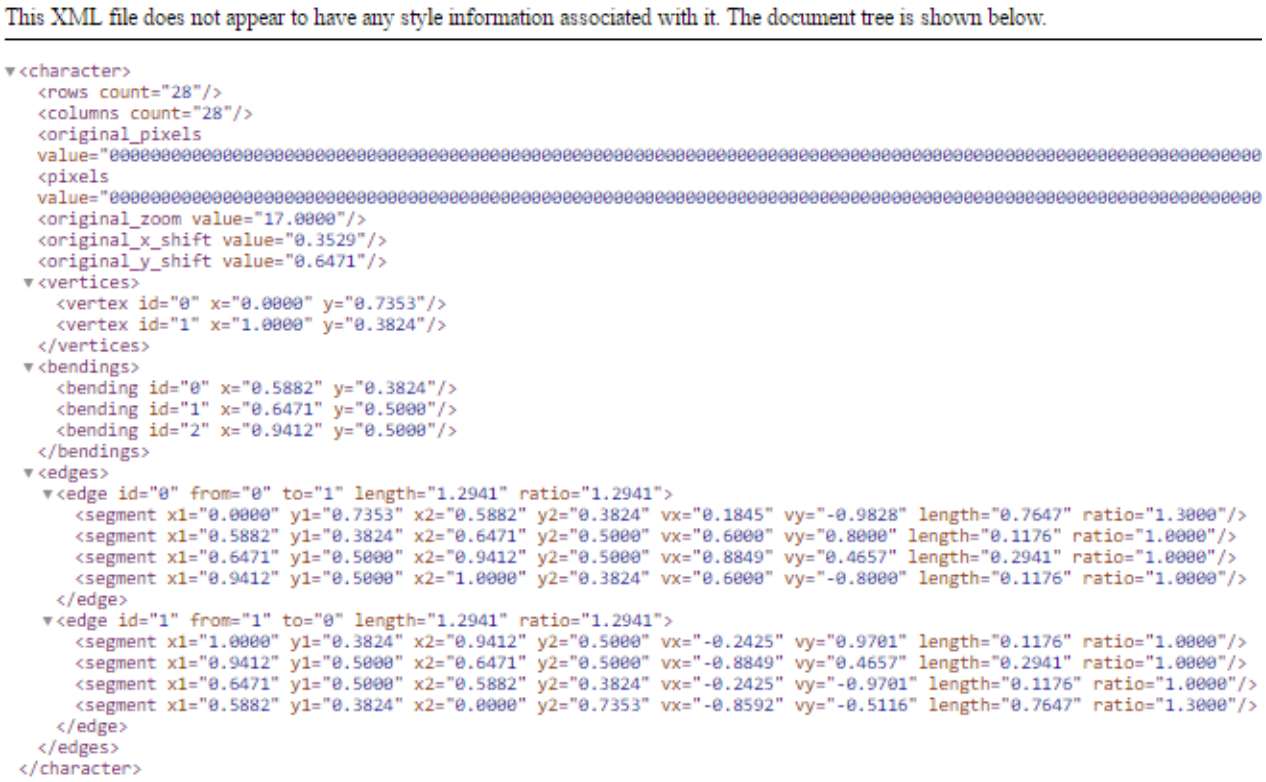
Таблица 3.10 – Теги, используемые для описания структурной модели и исходного графического представления символа

Тег	Описание
`character`	Тег, обозначающий начало описания очередного символа.
`Rows`	Количество рядов пикселей на исходном изображении или высота изображения.
`columns`	Количество столбцов пикселей на исходном изображении или ширина изображения.
`original_pixels`	Последовательность нулей и единиц, задающая исходное бинарное изображение по строкам.
`pixels`	Последовательность цифр, задающая бинарное изображение после выполнения операций скелетизации и выделения пикселей, соответствующих ключевым точкам и изгибам. Используется формат, аналогичный формату элемента `original_pixels`.
`original_zoom`	Вещественное число, показывающее, во сколько раз было уменьшено исходное

	графическое представление символа, чтобы координаты каждой из структурных составляющих оказались в диапазоне от нуля до единицы.
`original_x_shift`	Вещественное число, показывающее, насколько было смещено исходное графическое представление символа по оси X, чтобы после сжатия оно было расположено по центру.
`original_y_shift`	Вещественное число, показывающее, насколько было смещено исходное графическое представление символа по оси Y, чтобы после сжатия оно было расположено по центру.
`vertices`	Тег, открывающий перечень всех ключевых точек структурной модели символа.
`vertex`	Тег, задающий описание очередной ключевой точки структурной модели символа.
`bendings`	Тег, открывающий перечень всех изгибов структурной модели символа.
`bending`	Тег, задающий описание очередного изгиба структурной модели символа.
`edges`	Тег, открывающий перечень всех композитных ребер структурной модели символа.
`edge`	Тег, задающий описание очередного композитного ребра, а также открывающий список всех соединяющих ребер, которые

	входят в его состав.
`segment`	Тег, задающий описание очередного соединяющего ребра в составе композитного.

Язык разметки XML является удобным для восприятия человеком. В большинстве современных браузеров и тестовых редакторов реализованы дополнительные средства для отображения и редактирования XML-файлов. Пример отображения содержимого XML-файла, соответствующего начертанию символа «5» из базы рукописных цифр MNIST в браузере Google Chrome приведен на рисунке 3.3:


This XML file does not appear to have any style information associated with it. The document tree is shown below.
<character>
<rows count="28"/>
<columns count="28"/>
<original_pixels
value="0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000"
<pixels
value="0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000"
<original_zoom value="17.0000"/>
<original_x_shift value="0.3529"/>
<original_y_shift value="0.6471"/>
<vertices>
<vertex id="0" x="0.0000" y="0.7353"/>
<vertex id="1" x="1.0000" y="0.3824"/>
</vertices>
<bendings>
<bending id="0" x="0.5882" y="0.3824"/>
<bending id="1" x="0.6471" y="0.5000"/>
<bending id="2" x="0.9412" y="0.5000"/>
</bendings>
<edges>
<edge id="0" from="0" to="1" length="1.2941" ratio="1.2941">

</edge>
<edge id="1" from="1" to="0" length="1.2941" ratio="1.2941">

</edge>
</edges>
</character>"

Рисунок 3.3 – Пример отображения XML-файла со структурной моделью

Содержимое XML-файла однозначно задает структурную модель символа. Более того, информации в этом файле достаточно, чтобы восстановить исходное изображение и структурную модель без учета сжатия графического представления символа.

3.5.2. Скрипт для визуализации структурных моделей

Для удобства оценки правильности работы алгоритма построения структурной модели, а также методов, отвечающих за ее сохранение и загрузку был реализован скрипт на языке программирования javascript, выполняющий визуализацию структурной модели, записанной в XML-файле. Реализованный скрипт был помещен в HTML-документ, который также содержит область рисования для отображения результата выполнения сценария.

Общий вид HTML-страницы в исходном состоянии приведен на рисунке 3.4.

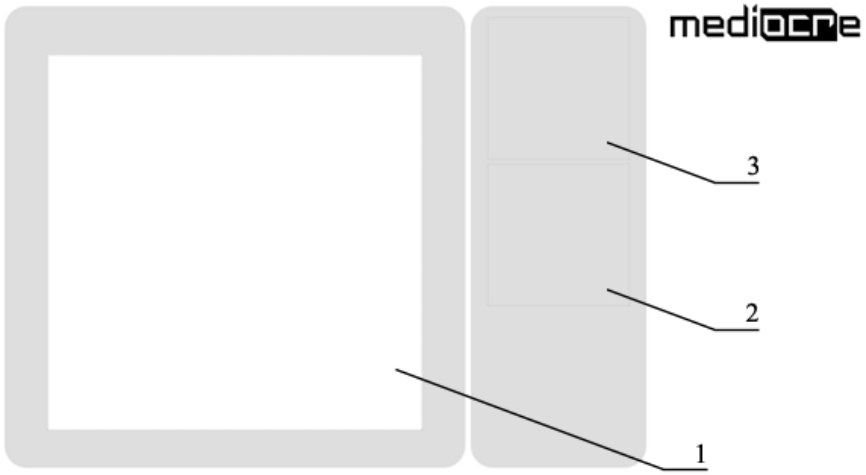


Рисунок 3.4 – Исходное состояние веб-страницы визуализатора структурных моделей: (1) – область визуализации структурной модели; (2) – область визуализации скелетизированного изображения; (3) – область визуализации исходного изображения

При перетягивании XML-файла со структурной моделью символа на открытую веб-страницу визуализатора, происходит отображение на странице этой модели. При перетягивании XML-файла, показанного на рисунке 3.3, HTML-страница визуализатора будет содержать схематичное построение заданной в нем структурной модели. Общий вид такой веб-страницы показан на рисунке 3.5.

На приведенном изображении красным цветом отмечены ключевые точки, зеленым – точки изгиба. Как можно заметить, некоторые соединяющие ребра структурной модели представлены в виде ломаных, некоторые – в виде отрезков, а одно из ребер представлено эллиптической дугой.

В правой части рисунка 3.5 показано исходное бинарное изображение символа. Под ним располагается скелетизированное изображение, на котором отмечены цветом пиксели, соответствующие ключевым точкам и изгибам.

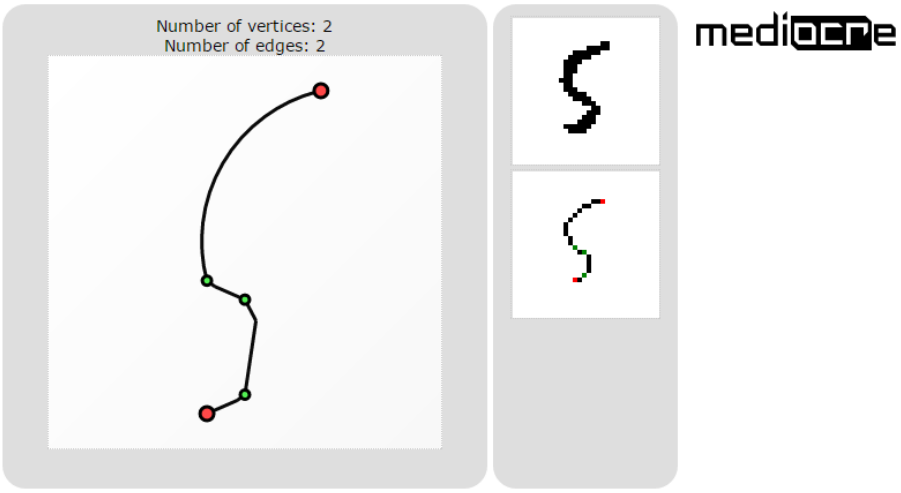


Рисунок 3.5 – Веб-страница с визуализацией структурной модели рукописного символа

Типы графических примитивов, которые будут выбраны для визуализации отдельных соединяющих ребер выбираются по правилами, идентичным тем, что используются в процессе оценки степени схожести двух композитных ребер и в процессе построения аппроксимирующих наборов отрезков.

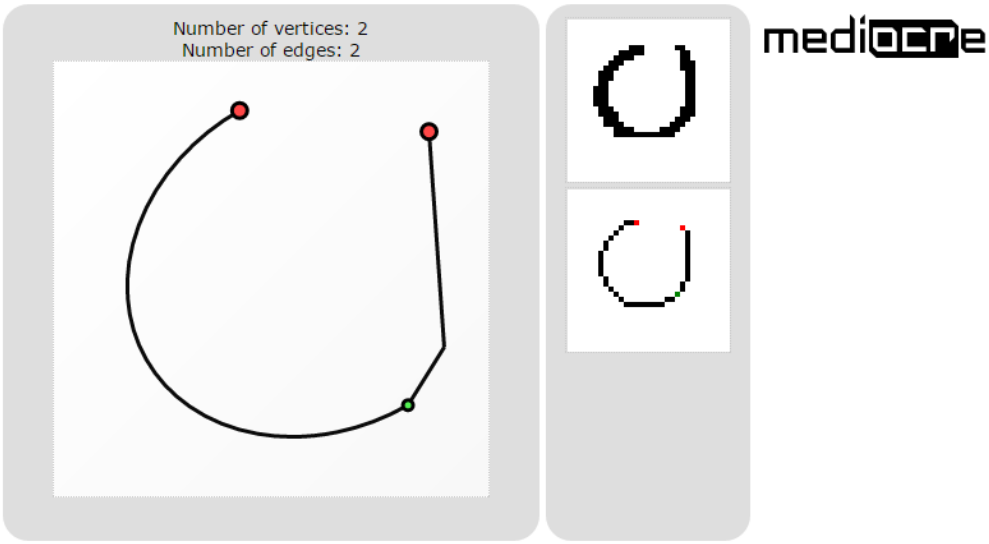


Рисунок 3.6 – Пример визуализации структурной модели символа с использованием дуги окружности

Для соединяющих ребер с большим значением коэффициента искривления в качестве графического примитива выбирается дуга окружности. Пример такого случая показан на рисунке 3.6.

Скрипт для визуализации структурных моделей также был использован для проверки правильности математических расчетов, которые использовались при реализации алгоритмов распознавания символов.

3.6. Интерфейс пользователя программного приложения

Для взаимодействия пользователя с разработанным программным приложением предусмотрен интерфейс командной строки. Пользователь может выбрать режим работы приложения: построение и сохранение структурной модели изображенного символа, сравнение пары структурных моделей или изображений, сравнение изображения или структурной модели с базой эталонных структурных моделей для решения задачи распознавания символов.

В разработанном приложении используется наиболее общий формат команд:

```
[символ_начала_команды]имя_команды [параметр_1 [параметр_2 […]]],
```

где в квадратные скобки помещены части, которые не являются обязательными.

Перечень и описания допустимых параметров разработанного программного приложения приведены в таблице 3.11.

Таблица 3.11 – Допустимые параметры командной строки

Параметр	**Описание**
`Mode`	Символьное обозначение режима, в котором следует запустить программное приложение. Допустимые значения и соответствующие им режимы работы приведены в таблице 3.12.
`input1`	Абсолютный или относительный путь к файлу с информацией о первом анализируемом символе.
`input2`	Абсолютный или относительный путь к файлу с информацией о втором анализируемом символе.
`output1`	Абсолютный или относительный путь к XML-файлу, в который необходимо сохранить структурную модель первого анализируемого символа.
`output2`	Абсолютный или относительный путь к XML-файлу, в который необходимо сохранить структурную модель второго анализируемого символа.
`Basedir`	Абсолютный или относительный путь к директории, в которой находится база структурных моделей эталонных изображений.

Alg	Символьное обозначение алгоритма, который должен быть использован для оценки степени схожести. Допустимые значения и описание соответствующих им алгоритмов приведены в таблице 3.13.

Параметр basedir игнорируется, если приложение запускается в режиме, не требующем использования базы эталонных изображений. В свою очередь, параметры input2 и output2 игнорируются, если используется режим, не требующий базы эталонных изображений.

Если значения output1 или output2 не указаны, то подразумевается, что сохранение структурной модели в файл не требуется.

В зависимости от расширений файлов input1 и input2 однозначно определяется, в каком формате задаются описания исходных символов: структурными моделями в формате XML или исходными изображениями с графическими представлениями символов.

Для параметра mode допустимыми значениями являются значения, перечисленные в таблице 3.12.

Таблица 3.12 – Режимы работы программного приложения

Режим	**Описание**
Analyze	Режим построения структурной модели заданного изображения и ее последующего сохранения в заданное местоположение.
Compare	Режим сравнения двух символов, заданных структурными моделями или файлами с исходными графическими представлениями.
Classify	Режим определения класса в выбранной базе эталонных изображений, которому наибольшим образом соответствует заданный символ.

Для режимов `compare` и `classify` необходимо указать значение параметра `alg`. Допустимые значения и их описание приведены в таблице 3.13.

Таблица 3.13 – Допустимые значения параметра `alg`

Значение	Описание
`matching`	Алгоритм на основе поиска максимального паросочетания минимального веса.
`intersections`	Алгоритм на основе метода пересечений.
`test`	Алгоритм вероятностного тестирования.

По умолчанию используется алгоритм соответствующий значению «`matching`».

После того, как программное приложение было вызвано с необходимыми значениями параметров, посредством текстового интерфейса пользователю сообщаются результаты работы приложения, а также время исполнения в секундах.

Если при задании значений параметров была допущена ошибка: задано имя несуществующего параметра или задано некорректное значение хотя бы одного из параметров, то пользователю будет выведено сообщение об ошибке с указанием первого имени или значения параметра, где допущена ошибка.

3.7. Основные результаты и выводы по главе 3

1. Сформулированы основные требования к средствам разработки программного приложения: языку программирования и подключаемым библиотекам. На основании данных требований осуществлен выбор средств для программной реализации.

2. Изложено описание основных классов, используемых в модуле распознавания символов. Описаны основные атрибуты, переменные и методы этих классов.

3. Представлен предложенный формат XML-файла для сохранения информации о структурной модели символа на диск. Описан сценарий на языке программирования javascript для визуализации файлов предложенного формата.

4. Разработано программное приложение с использованием выбранных программных средств. Представлен командный пользовательский интерфейс данного приложения. Описаны используемые параметры командной строки и их допустимые значения.

ГЛАВА 4. ТЕСТИРОВАНИЕ РАЗРАБОТАННЫХ АЛГОРИТМОВ И ПРОГРАММНЫХ СРЕДСТВ

В данной главе представлены результаты тестирования разработанных алгоритмов для решения задачи распознавания рукописных символов в условиях малой обучающей выборки. Представлены результаты тестирования на общеизвестном наборе рукописных цифр MNIST и наборе рукописных символов из бланков тестовых заданий, аналогичных заданиям единого государственного экзамена для школьников. Выполнен сравнительный анализ результатов работы предложенных алгоритмов с существующими алгоритмами, способными функционировать в условиях малой обучающей выборки. Приведены результаты исследования предложенного алгоритма сегментации рукописного текста.

4.1. Подготовка базы изображений рукописных символов

Для тестирования алгоритмов распознавания рукописных символов необходимо было подготовить базу изображений с графическими начертаниями рукописных символов.

Чтобы ограничить множество способов графического представления одних и тех же символов необходимо обозначить эталонные формы начертания. В качестве таких образцов можно использовать формы начертания из бланков единого государственного экзамена для школьников. Эталонные формы из данных бланков приведены на рисунке 4.1.

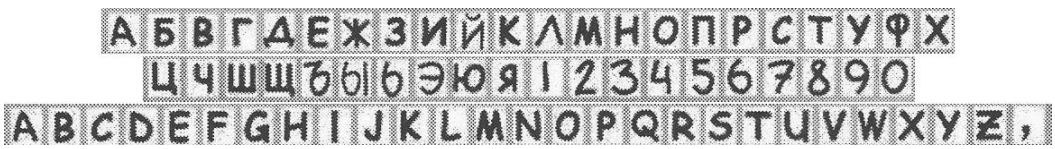


Рисунок 4.1 – Эталонные формы начертания символов из бланков единого государственного экзамена для школьников

Если, при исполнении начертания, оператор старался придерживаться данных образцов, то схожие изображения не могут соответствовать разным

классам. Исключение составляют символы «1» (арабская цифра «1») и «I» (латинская буква «I»), а также схожие по форме начертания латинские и кириллические буквы. Классы этих символов следует считать эквивалентными при решении задачи распознавания таких символов.

Для того чтобы получить большое количество изображений символов каждого класса, группе учащихся было предложено заполнить вручную поля части В бланков единого государственного экзамена. В качестве символов для заполнения были выбраны все десять арабских цифр, а также слова из панграмм на русском и английском языках.

Пример заполнения бланка указанным образом приведен на рисунке 4.2.

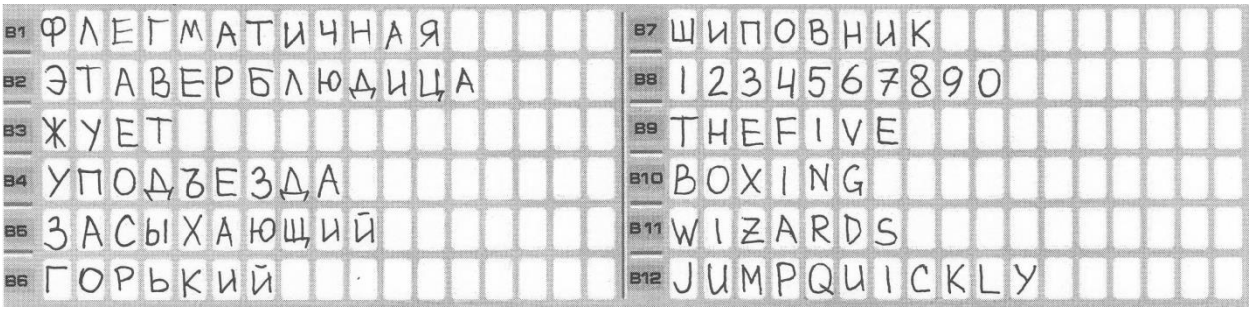
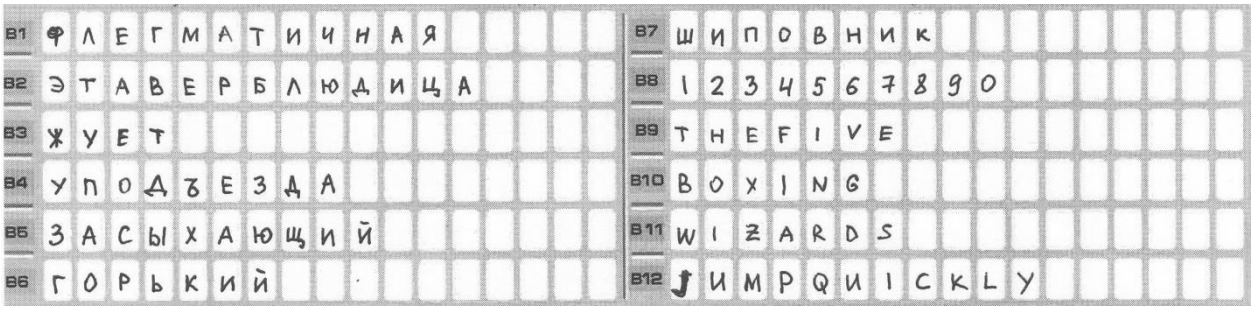
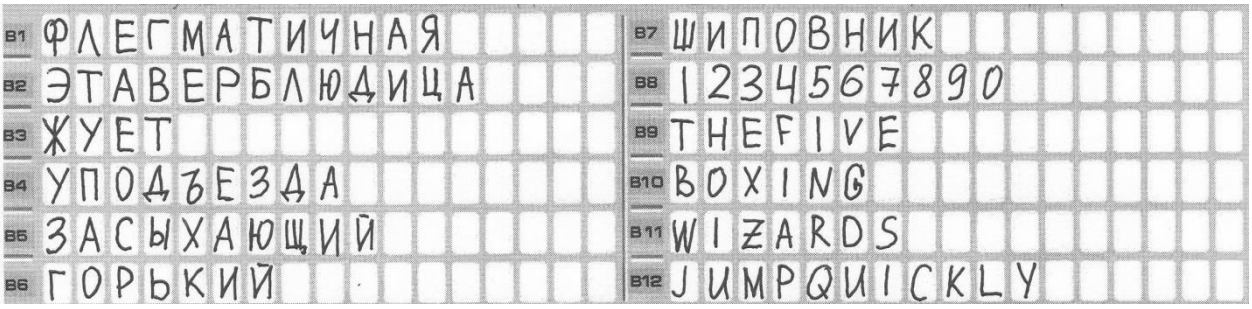


Рисунок 4.2 – Пример заполнения бланка панграммами на русском и английском языках и арабскими цифрами

Стоит отметить, что начертание разных исполнителей отличается не только особенностями выполнения отдельных графических элементов, но и даже размерами изображенных символов.



(а)



(б)

Рисунок 4.3 – Примеры заполнения бланков с использованием (а) чрезмерно мелкого начертания символов и (б) чрезмерно крупного начертания символов.

При чрезмерно мелком начертании задача распознавания символов осложняется неточным выполнением мелких деталей символов. При крупном начертании символов наиболее заметны особенности почерка исполнителя начертания.

После того, как бланки были отсканированы, на них были выделены все области, в которых выполнены начертания символов. Каждая из таких областей была трансформирована так, чтобы ее размер составил 32×32 пикселя. Полученные изображения были сохранены в виде растровых изображений на жесткий диск.

Как можно заметить, приведенный набор символов содержит 68 классов символов: 32 буквы русского алфавита, 26 букв латинского алфавита и 10 арабских цифр.

Среди приведенных 68 классов существует одиннадцать пар и одна тройка идентичных классов. Их описания приведены в таблице 4.1.

Таблица 4.1 – Группы идентичных классов в подготовленной базе изображений рукописных символов

№	Состав группы идентичных классов
1	Латинская буква «A» и кириллическая буква «А»
2	Латинская буква «B» и кириллическая буква «В»
3	Латинская буква «C» и кириллическая буква «С»
4	Латинская буква «E» и кириллическая буква «Е»

5	Латинская буква «H» и кириллическая буква «Н»
6	Латинская буква «K» и кириллическая буква «К»
7	Латинская буква «M» и кириллическая буква «М»
8	Латинская буква «P» и кириллическая буква «Р»
9	Латинская буква «X» и кириллическая буква «Х»
10	Латинская буква «Y» и кириллическая буква «У»
11	Латинская буква «I» и арабская цифра «1»
12	Латинская буква «O», кириллическая буква «О» и арабская цифра «0»

При оценке качества работы алгоритмов распознавания символов результат классификации следует считать правильным, если класс, определенный алгоритмом, совпадает с указанным в базе или находится с ним в одной группе идентичных классов.

Для наполнения базы было использовано 128 заполненных бланков, каждый из которых содержит 104 символа. Следовательно, общее количество изображений с графическими начертаниями символов равно 13312.

4.2. Алгоритмы распознавания символов, используемые для сравнения с предложенными подходами

Для оценки точности предложенных алгоритмов необходимо выполнить сравнение с другими методами распознавания рукописных символов.

Так как предложенные алгоритмы разработаны для корректной работы в условиях ограниченного количества эталонных изображений, сравнение необходимо выполнять с алгоритмами, которые так же способны функционировать при малом количестве эталонных изображений.

К таким подходам нельзя отнести алгоритмы, основанные на использовании признаковых классификаторов, которые требуют настройки большого количества весов или каких-либо других параметров. Как правило,

для подбора значений большого количества параметров, требуется существенное количество эталонных образов.

Тем не менее, существуют классификаторы, которые для корректной работы не требуют большого количества эталонных изображений. К таким классификаторам можно отнести машину опорных векторов (support vector machine, SVM) и вероятностную нейронную сеть на основе RBF-сети (RBF-based probabilistic neural network, PNN-сеть). Точность распознавания данных классификаторов существенно зависит от количества эталонных изображений, но даже при их малом количестве классификация может быть выполнена с достаточно высоким процентом правильно классифицированных образов.

Структура PNN-сети подразумевает необходимость оценки плотности вероятности на основе наблюдений в некоторых точках пространства признаков. В роли таких наблюдений и выступают эталонные образы. Чем больше имеется образов, тем выше качество распознавания.

В машине опорных векторов точность задания границ групп векторов зависит от количества образов, использованных для настройки ее параметров.

Еще одним методом, который может функционировать в условиях ограниченного количества эталонных образов, является метод пересечений. Время работы такого алгоритма, как и разработанных подходов, линейно зависит от количества образцов.

Кроме того, для сравнения были использованы тривиальный подход с использованием вертикальной и горизонтальной гистограмм, а также его модификация с использованием радиальной гистограммы, out-in профиля и in-out профиля [66].

4.2.1. Алгоритм на основе метода опорных векторов

Для реализации алгоритма, использующего машину опорных векторов, был выбран вариант, описанный в одном из руководств авторов библиотеки OpenCV по распознаванию рукописных символов [121]. В качестве вектора признаков при таком подходе используются гистограммы ориентированных градиентов [122]. Метод опорных векторов на основе гистограмм нередко используют для классификации образов [123].

Предварительно исходное изображение подвергается операции deskew – устранению скоса с использованием моментов второго порядка.

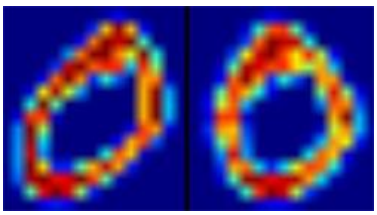


Рисунок 4.4 – Пример результата выполнения операции deskew для изображения из набора рукописных цифр MNIST

Для того чтобы построить гистограмму ориентированных градиентов, для начала необходимо определить производные Собеля в двух направлениях. Затем требуется определить их магнитуду и направление градиента в каждом пикселе. Значение градиента необходимо квантовать на 16 значений. Затем изображение следует разделить на четыре квадратных области, для каждой из которых можно вычислить гистограмму величин магнитуды по 16 направлениям. После объединения четырех векторов, соответствующих каждому из квадратов, будет получен итоговый вектор признаков.

Существенными преимуществами данного подхода являются простота его реализации с использованием библиотеки компьютерного зрения OpenCV и высокое быстродействие.

4.2.2. Алгоритм на основе вероятностной нейронной сети

Подходы на основе вероятностной модификации RBF-сети, которая носит название вероятностной нейронной сети или PNN-сети, могут различаться лишь способом получения вектора признаков. По своей сути искусственная нейронная сеть такого типа решает задачу оценки плотности вероятности по имеющимся данным. Решение такой задачи в данном случае основано на так называемых ядерных оценках [124].

Фактически, делается предположение о том, что существование наблюдения в некоторой точке пространства обеспечивает некоторую плотность вероятности в этой точке. Кластеры из близко лежащих точек указывают на то, что в этом месте плотность вероятности достаточно велика. По расстоянию от ядерных центров оценивается доверие к уровню плотности вероятности. Для оценки общей плотности вероятности используется суммарное значение некоторой функции во всех точках наблюдения. Такую функцию называют ядерной функцией. Зачастую в качестве ядерной функции используют распределение Гаусса. При больших размерах обучающей выборки такой метод дает достаточно точное приближение к истинной плотности вероятности [125].

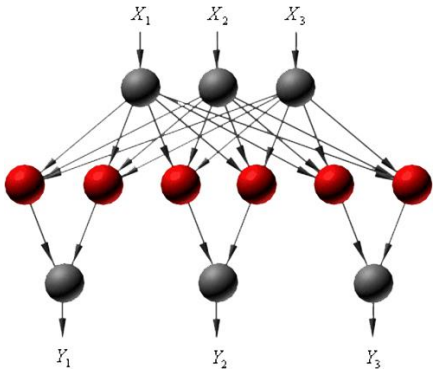


Рисунок 4.5 – Структура вероятностной нейронной сети

Вероятностная нейронная сеть имеет три слоя: входной, радиальный и выходной. Входной слой имеет произвольную размерность. Каждый нейрон радиального слоя соответствует одному элементу обучающей выборки. Количество нейронов выходного слоя равняется количеству классов. Каждый

нейрон выходного слоя соединен с элементами радиального слоя, принадлежащими соответствующему ему классу [142].

Допустим, что все изображения символов, которые требуется классифицировать, являются бинарными изображениями фиксированного размера W×H пикселей. В таком случае можно передавать на входы вероятностной нейронной сети значения яркости каждого из пикселей, которые всегда будут равны нулю или единице. Для таких дескрипторов изображения размерность пространства будет равна количеству бит в изображении – произведению W и H.

Таким образом, для подхода на основе вероятностной нейронной сети роль вектора признаков выполняет вектор значений яркости самого исходного изображения. Это позволяет не тратить вычислительные ресурсы на процесс получения вектора признаков, однако при таком подходе быстродействие алгоритма существенно ухудшается с ростом размеров классифицируемых изображений.

4.2.3. Алгоритмы на основе построения гистограмм

Для реализации методов, основанных на построении гистограмм, были использованы средства библиотеки OpenCV. Сопоставление двух гистограмм выполнялось с использование корреляционного метода сравнения. Функция меры близости при таком подходе задается выражением (4.1):

$$d_{correl}(H_1, H_2) = \frac{\sum_i H_1(i) \cdot H_2(i)}{\sqrt{\sum_i {H_1}^2(i) \cdot {H_2}^2(i)}}. \quad (4.1)$$

В библиотеке OpenCV такая функция реализована в виде вызова функции `compareHist` со значением параметра `method` равным `CV_COMP_CORREL`.

В качестве меры расстояния между in-out и out-in профилями использовалась сумма по всем пикселям расстояний до ближайших черных пикселей сравниваемого профиля.

4.2.4. Алгоритм на основе метода пересечений для растровых изображений

Так как без построения структурной модели или какого-либо другого геометрического представления невозможно реализовать полноценный метод пересечений, в качестве альтернативного решения на основе этого метода была использована модификация для растровых изображений.

Модификация метода пересечений для растровых изображений заключается в использовании метода Брезенхэма для растеризации каждой прямой в наборе для последующего ее наложения на растровое представление начертания символа.

Не смотря на потерю определенной информации из-за растрового представления графических объектов, такой подход имеет существенное преимущество. Он может быть реализован без использования переменных с плавающей точкой [137].

4.3. Оценка качества распознавания предложенных алгоритмов

Как уже отмечалось ранее, предложенный метод подходит для использования в случае, когда имеется лишь малое количество эталонных изображений каждого из классов.

В случае если количество эталонных изображений достаточно велико, предложенные алгоритмы будут обладать низким быстродействием. Это объясняется линейной зависимостью времени их работы от количества эталонных образов.

Оценка качества распознавания предложенных алгоритмов производилась на двух наборах рукописных символов: общеизвестном наборе MNIST (70000 символов, 10 классов) и ранее описанном наборе изображений символов из бланков, аналогичных бланкам единого государственного экзамена (13312 символов, 68 классов).

В оценке участвуют три предложенных алгоритма распознавания символов на основе построения структурных моделей:

- алгоритм на основе поиска максимального паросочетания минимального веса (**SMM**);
- алгоритм на основе метода пересечений для структурных моделей (**SMI**);
- алгоритм на основе вероятностного теста для оценки степени схожести структурных моделей (**SMT**), настройка которого производилась с применением описанного ранее генетического алгоритма в течение 10000 эпох.

Для оценки качества распознавания символов в условиях малой обучающей выборки, предложенные алгоритмы были сравнены с аналогами, способными функционировать в условиях малого количества эталонных изображений:

- метод пересечений для растровых изображений (**IM**);
- алгоритм на основе метода опорных векторов (**SVM**);
- метод на основе PNN-сети (**PNN**);
- гистограммный метод (**HM**);
- гистограммный метод с in-out и out-in профилями (**HMP**).

Для обучения или настройки в каждом из экспериментов использовалось различное количество эталонных изображений E. Эксперименты проводились для нечетных значений *E* от 3 до 15.

Результаты экспериментов для набора MNIST приведены в таблице 4.2.

Таблица 4.2 – Процент верно распознанных символов набора MNIST каждым из сравниваемых алгоритмов

E	SMI	SMM	SMT	IM	SVM	PNN	HM	HMP
3	89,1	**93,2**	78,1	87,4	83,8	70,9	71,5	74,1
5	90,8	**95,1**	79,4	88,7	85,5	73,2	72,2	74,3
7	91,2	**95,1**	80,0	88,9	87,0	74,2	72,6	74,8
9	91,3	**95,2**	80,0	90,4	88,2	74,7	72,8	75,0
11	91,3	**95,2**	80,2	90,4	88,5	75,3	72,9	75,1

13	91,3	**95,3**	80,2	90,6	88,6	75,5	73,0	75,1
15	91,4	**95,3**	80,3	90,7	88,6	75,8	73,1	75,2

Результаты экспериментов также изображены графически на рисунке 4.6.

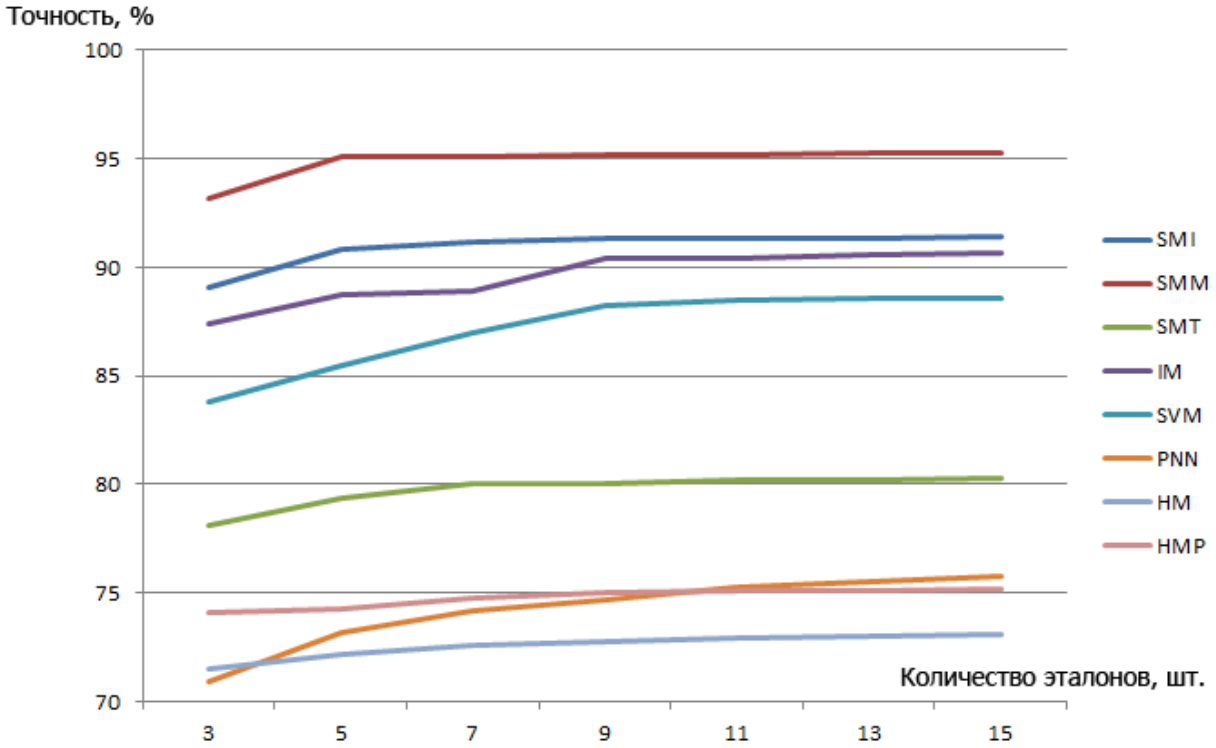


Рисунок 4.6 – Графическое представление результатов экспериментов на наборе рукописных цифр MNIST

Как можно заметить из приведенных результатов экспериментов, алгоритм на основе поиска максимального паросочетания минимального веса чаще верно классифицирует изображения рукописных символов при заданных значениях количества эталонных изображений.

При значении $E = 5$ преимущество такого метода наиболее существенно.

Стоит так же отметить, что алгоритм на основе вероятностного теста для оценки степени схожести оказался недостаточно эффективен и по качеству классификации опередил лишь PNN-сеть и методы на основе сравнения гистограмм.

Важно так же заметить, что при увеличении количества эталонных изображений до нескольких тысяч качество распознавания предложенных

методов улучшается несущественно, в то время как их быстродействие существенно ухудшается.

В то же время при увеличении количества эталонных изображений до нескольких тысяч результаты метода опорных векторов и PNN-сети улучшаются более существенно.

Чтобы оценить, символы каких классов чаще других были неверно распознаны, а также к каким классам они были ошибочно отнесены, можно построить матрицы несоответствий (confusion matrix) для каждого из разработанных подходов сравнения структурных моделей [126, 127].

В первую очередь стоит обратить внимание на матрицу несоответствий для алгоритма на основе нахождения максимального паросочетания минимального веса. В таблице 4.3 приведены данные для эксперимента со значением количества эталонных изображений $E = 15$.

Таблица 4.3 – Матрица несоответствий для результатов алгоритма на основе нахождения максимального паросочетания на наборе MNIST

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	6239	0	0	0	0	0	379	0	0	285
1	0	7775	2	0	2	0	0	98	0	0
2	0	1	6896	2	0	0	0	91	0	0
3	0	0	1	6785	0	0	0	0	210	145
4	0	11	0	0	6779	0	0	12	0	22
5	0	0	0	0	0	6284	25	0	0	4
6	317	0	0	0	0	41	6288	0	230	0
7	0	115	46	0	3	0	0	7129	0	0
8	0	0	0	184	0	0	247	0	6233	161
9	285	0	0	154	7	2	0	0	206	6304

Диагональные элементы приведенной матрицы выражены достаточно явно. Это говорит о том, что для каждого из классов символов предложенный алгоритм, как правило, верно классифицирует относящиеся к нему начертания [128]. Тем не менее, существуют пары классов, структурные модели которых достаточно похожи, что, в совокупности с несовершенством предложенного подхода, приводит к неверной классификации.

Аналогичная матрица несоответствий приведена в таблице 4.4 и для предложенного алгоритма на основе метода пересечений.

Таблица 4.4 – Матрица несоответствий для результатов алгоритма на основе метода пересечений на наборе MNIST

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	5628	0	0	0	0	0	703	0	0	572
1	0	7730	1	0	1	0	0	145	0	0
2	0	0	6912	2	0	0	0	76	0	0
3	0	0	1	6916	0	0	0	0	210	14
4	0	4	0	0	6790	0	0	10	0	20
5	0	0	0	0	0	6242	15	0	0	56
6	732	0	0	0	0	31	5740	0	162	211
7	0	244	37	0	5	0	0	7007	0	0
8	0	0	0	184	0	0	456	0	5462	723
9	611	0	0	23	6	24	330	0	412	5552

Несмотря на то, что количество неверных классификаций увеличилось едва ли не вдвое (6021 по сравнению с 3288), некоторые пары классов стали лучше различимы. Например, классы, соответствующие цифрам «3» и «9» предложенным алгоритмом на основе метода пересечений различаются существенно чаще. Тем не менее, некоторые классы стали существенно менее различимыми. К таким классам можно отнести тройку классов, соответствующих цифрам «0», «6» и «9».

Таблица 4.5 – Матрица несоответствий для результатов алгоритма на основе вероятностного теста на наборе MNIST

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	4769	1	1	2	1	4	913	0	410	802
1	2	7066	124	2	451	2	8	212	3	7
2	0	2	6273	145	1	24	0	413	34	98
3	1	0	214	6013	1	50	28	2	618	214
4	0	4	24	2	6761	1	0	10	2	20
5	2	0	20	33	0	5922	192	4	18	122
6	732	2	1	53	1	152	4282	0	762	891
7	2	244	37	0	5	2	1	6989	0	13
8	354	2	12	461	1	4	712	1	4433	845
9	824	2	56	305	11	24	910	23	1103	3700

При анализе аналогичной матрицы несоответствий для алгоритма на основе вероятностного теста, приведенной в таблице 4.5, можно отметить, что в таблице уже почти не осталось нулей, даже в парах существенно различающихся классов. Это вполне объясняется вероятностным характером предложенного подхода. Существенно увеличилось количество неверных классификаций для тройки классов, соответствующих цифрам «0», «6» и «9»,

а также паре цифр «3» и «8». Наименьшее количество неверных классификаций пришлось на классы символов, обладающих специфичными формами начертания. К таким можно отнести классы рукописных цифр «1», «4» и «7».

Подготовленный набор изображений рукописных символов из бланков, аналогичных бланкам ЕГЭ, содержит начертания, максимально приближенные к определенному эталону, что существенно упрощает задачу распознавания символов. По этой причине результаты исследуемых алгоритмов существенно выше. Результаты экспериментов для данного набора представлены в таблице 4.6.

Таблица 4.6 – Процент верно распознанных символов каждым из сравниваемых алгоритмов для рукописных символов из бланков, аналогичных бланкам ЕГЭ

E	SMI	SMM	SMT	IM	SVM	PNN	HM	HMP
3	98,9	**99,2**	93,2	95,3	88,7	72,1	76,2	78,4
5	99,3	**99,4**	94,4	96,1	89,2	74,8	76,6	78,7
7	99,3	**99,4**	95,1	96,2	90,0	75,7	77,0	78,8
9	99,3	**99,5**	95,3	96,2	90,4	76,8	77,2	79,2
11	99,4	**99,5**	95,4	96,4	90,8	78,1	77,4	79,3
13	99,4	**99,5**	95,4	96,5	90,9	78,5	77,5	79,4
15	99,4	**99,5**	95,5	96,5	90,9	78,7	77,5	79,4

Результаты данного эксперимента представлены на рисунке 4.7.

Из-за малого количества эталонных изображений метод опорных векторов существенно уступает всем трем предложенных алгоритмам. Однако меньшее разнообразие форм начертания дает существенное преимущество методу пересечений, который оказывается достаточно эффективным для данного набора изображений.

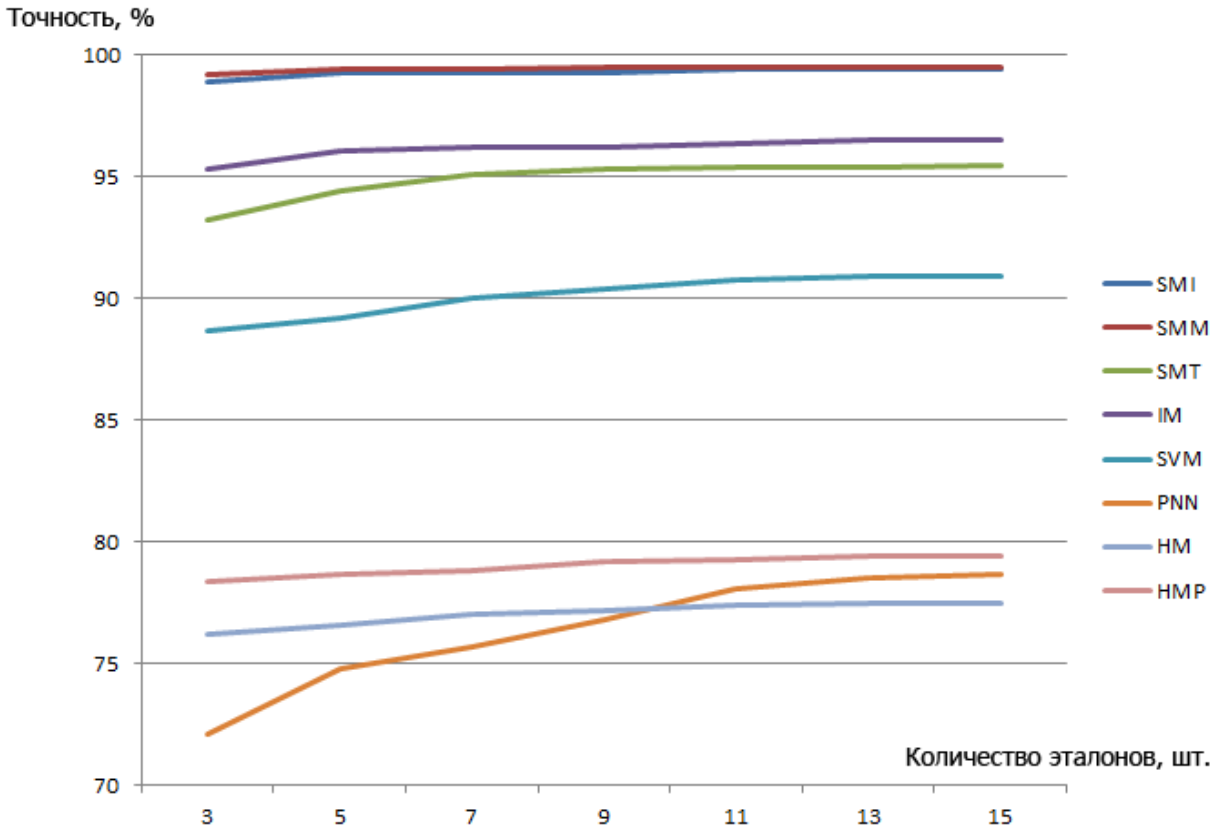


Рисунок 4.7 – Графическое представление результатов экспериментов на наборе рукописных символов из бланков, аналогичных бланкам ЕГЭ

Как и в случае с набором рукописных цифр MNIST, на изображениях из данного набора наилучший процент верно распознанных начертаний был показан предложенным алгоритмом на основе поиска максимального паросочетания минимального веса. Однако существенным отличием результатов для этого набора является большая близость результатов, показанных методом пересечений для структурных моделей, к результатам наиболее эффективного метода.

Из-за большего единообразия форм графического представления символов набора, существенно лучше выглядят результаты предложенного подхода на основе вероятностного теста, опередив для исследуемых значений E в том числе и метод опорных векторов.

Стоит отметить, что данный набор предоставляет идеальные условия для решения задачи распознавания символов разработанными методами: минимальные различия в способах выполнения начертания, почти

одинаковый наклон начертания большинства символов, малая вероятность наличия мелких графических элементов, которые могут осложнить процесс распознавания.

4.4. Оценка быстродействия предложенных алгоритмов

Для оценки быстродействия предложенных алгоритмов каждый из них был запущен на каждом из двух наборов. Как и при оценке качества распознавания, запуск проводился для всех нечетных значений *E* от 3 до 15.

Для каждого из экспериментов программно было измерено время выполнения. Время запуска, при таком подходе, определялось с помощью стандартных средств языка программирования C++.

В оценке участвуют три предложенных алгоритма распознавания символов на основе построения структурных моделей:

- метод на основе поиска максимального паросочетания минимального веса (**SMM**);
- метод пересечений для структурных моделей (**SMI**);
- вероятностный тест для оценки степени схожести структурных моделей (**SMT**).

В таблице для каждого из алгоритмов приведено среднее время обработки отдельного символа (в секундах) при всех исследуемых значениях количества эталонных изображений.

Таблица 4.7 – Среднее время в секундах, затраченное на обработку изображений символов из набора MNIST предложенными алгоритмами

Количество эталонов	**SMM**	**SMI**	**SMT**
E = 3	0,01566	0,00216	0,00144
E = 5	0,02592	0,0036	0,00234
E = 7	0,03645	0,00504	0,00333
E = 9	0,04662	0,00648	0,00423
E = 11	0,05733	0,00801	0,00522
E = 13	0,06741	0,00936	0,00612
E = 15	0,07776	0,0108	0,00702

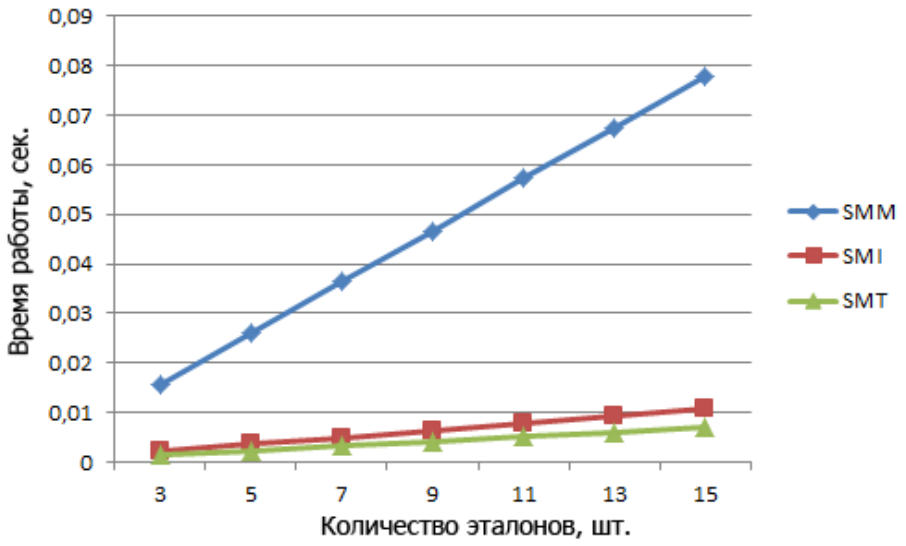


Рисунок 4.8 – Графическая зависимость среднего времени обработки изображения от количества образцов для набора MNIST

Построенная графическая зависимость среднего времени обработки изображения от количества эталонных изображений позволяет убедиться в том, что время работы алгоритмов линейно зависит от количества эталонных изображений.

Таблица 4.8 – Среднее время в секундах, затраченное на обработку изображений символов из бланков, аналогичных бланкам ЕГЭ

Количество эталонов	**SMM**	**SMI**	**SMT**
$E = 3$	0,0162	0,00234	0,00144
$E = 5$	0,02736	0,00396	0,00234
$E = 7$	0,03816	0,00558	0,00333
$E = 9$	0,04833	0,00702	0,00414
$E = 11$	0,0603	0,00882	0,00522
$E = 13$	0,0702	0,01026	0,00603
$E = 15$	0,081	0,01179	0,00702

Приведенный на рисунке 4.9 график зависимости среднего времени обработки изображения от количества образцов, как в случае с набором MNIST, отражает линейный характер зависимости.

Все замеры времени производились на процессоре Intel Core i5-4200 (1.60GHz, 2.30GHz) без использования графических ускорителей.

Стоит отметить, что среднее время обработки символов из набора MNIST несколько ниже для алгоритмов на основе поиска максимального

паросочетания минимального веса и метода пересечений. Это объясняется более простыми формами начертаний символов в наборе MNIST.

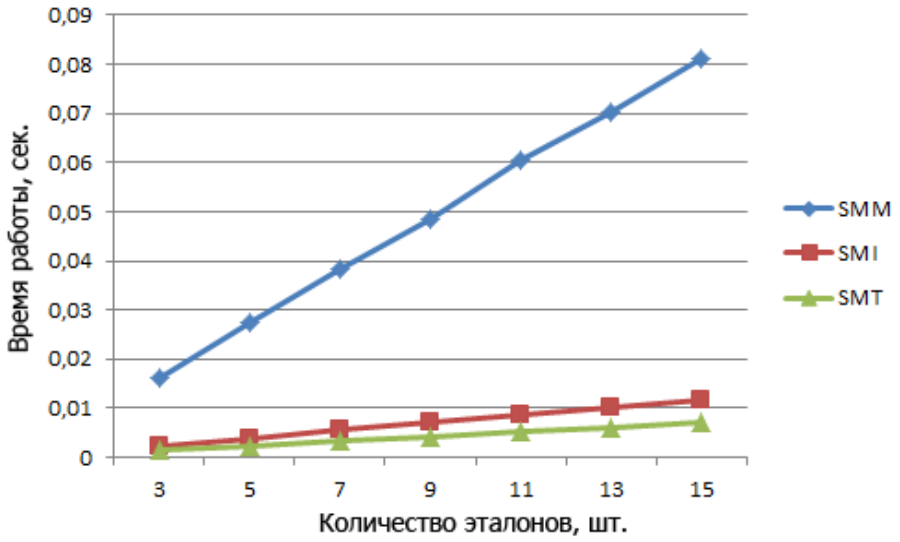


Рисунок 4.9 – Графическая зависимость среднего времени обработки изображения набора от количества образцов для набора символов из бланков, аналогичных бланкам ЕГЭ

Важно, что среднее время, которое необходимо на обработку одного символа, не превышает 100 мс даже для количества эталонных изображений равного 15.

Время обработки одного символа алгоритмом на основе поиска максимального паросочетания минимального веса существенно больше, чем у метода пересечений для структурных моделей. Но для набора символов из бланков, аналогичных бланками ЕГЭ, качество распознавания лишь немногим хуже, что говорит о целесообразности применения метода пересечений в случае, если требуется максимизировать быстродействие при несущественных потерях точности используемых алгоритмов.

4.5. Оценка предложенного подхода к утоньшению бинарных изображений

Как было ранее отмечено, скелетизация бинарного изображения символа выполняется с использованием совместного применения алгоритмов

Зонга-Суня и Ву-Цая, а также операции устранения плоских окончаний графического представления символа.

Предположение о необходимости в использовании дополнительной операции устранения плоских окончаний было сделано на основе анализа результатов тестирования предложенного алгоритма. Однако подробный анализ каждой из построенных моделей используемых наборов данных затруднителен, ведь единственный надежный способ оценки качества построенной структурной модели – это оценка визуального сходства, которая может быть произведена лишь человеком.

Тем не менее, можно использовать оценку точности распознавания в качестве параметра, который косвенно отражает качество построения структурных моделей. Стоит помнить, что точность распознавания также существенно зависит от других параметров, особенностей алгоритмов сравнения структурных моделей и, вероятно, присутствует влияние элемента случайности.

Сравнение точности распознавания производилось для каждого из трех предложенных алгоритмов для количества эталонных изображений E = 15 на наборе рукописных цифр MNIST и подготовленном наборе символов из бланков, аналогичных бланкам ЕГЭ.

Таблица 4.9 – Точность распознавания предложенных алгоритмов с использованием операции устранения плоских окончаний и без нее на символах набора MNIST

	SMI	**SMM**	**SMT**
С устранением	91,4%	95,3%	80,3%
Без устранения	74,3%	82,6%	63,9%

Таблица 4.10 – Точность распознавания предложенных алгоритмов с использованием операции устранения плоских окончаний и без нее на символах из бланков

	SMI	**SMM**	**SMT**
С устранением	99,4%	99,5%	95,5%
Без устранения	82,1%	86,9%	74,7%

Как можно заметить, для каждого из трех алгоритмов точность распознавания существенно улучшается вследствие применения операции устранения плоских окончаний.

Еще одним критерием, по которому можно оценить полезность операции устранения плоских окончаний, является соотношение количества пикселей скелетизированного изображения до и после применения данной операции. Для удобства восприятия такого рода информации человеком изменение количества пикселей следует измерять в процентах.

Было выделено семь наиболее значимых интервалов изменения количества пикселей:

- два интервала уменьшения количества пикселей: -(5%; ∞), -(0%; 5%];
- один интервал неизменности количества пикселей: {0%};
- четыре интервала увеличения количества пикселей: +(0%; 5%], +(5%; 15%], +(15%; 25%] и +(25%; ∞).

Таблица 4.11 – Изменение количества черных пикселей при использовании операции устранения плоских окончаний

	-(5; ∞)	-(0;5]	+{0}	+(0; 5]	+(5; 15]	+(15; 25]	+(25; ∞)
№ п.п.	1	2	3	4	5	6	7
MNIST	0,12%	1,94%	60,63%	23,07%	7,60%	5,42%	1,22%
Бланки	0,27%	2,39%	50,52%	38,12%	4,52%	3,29%	0,89%

Как можно заметить, достаточно редко использование такого рода операции приводит к уменьшению количества пикселей бинарного изображения в результате скелетизации. Как правило, если такое уменьшение происходит, то оно составляет не более 5% от исходного количества пикселей.

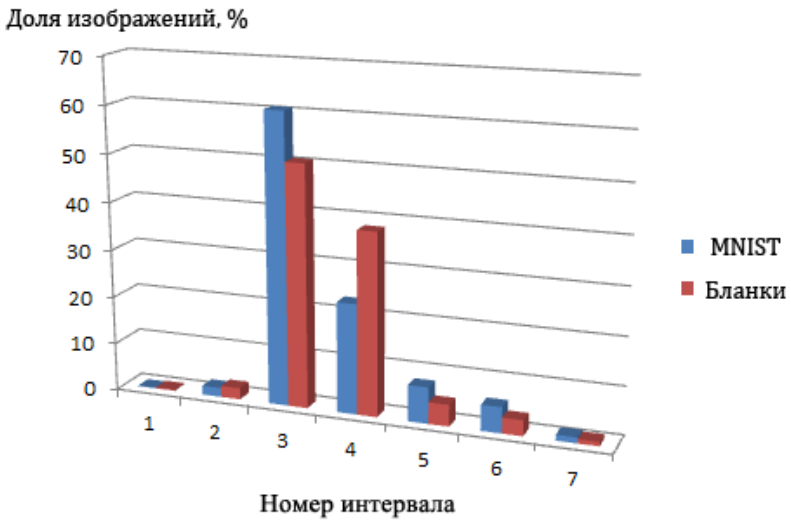


Рисунок 4.10 – Графическое представление распределения исследуемых изображений по интервалам изменения количества черных пикселей после применения операции устранения плоских окончаний

В свою очередь, увеличение количества пикселей, как правило, соответствует устранению эффекта чрезмерной скелетизации – удаления пикселей, которое приводит к нарушению топологии символа. Как видно, примерно в одном проценте случаев сохраняется более 25% пикселей. На практике такое существенное увеличение количества пикселей после скелетизации всегда соответствует сохранению какого-либо структурного элемента топологии символа.

Графическое представление проведенных экспериментов на рисунке 4.10 наглядно демонстрирует низкую вероятность потери структурных элементов изображения символа.

Результаты проведенных экспериментов позволяют сделать выводы о целесообразности применения операции устранения плоских окончаний элементов на бинарном изображении, однако необходимо исследовать и необходимость в использовании алгоритма скелетизации Ву-Цая для устранения недоскелетизированных участков результатов работы алгоритма Зонга-Суня. Оценить такую необходимость в терминах точных значений существенно проще. Количество пикселей, удаленных в ходе работы

алгоритма Ву-Цая, соответствует количеству пикселей, которые должны быть удалены алгоритмом Зонга-Суня, но были пропущены.

Таблица 4.12 – Распределение изображений используемых наборов по количеству пропущенных пикселей

	0	1	2-3	4-7	8-15	16-∞
№ п.п.	1	2	3	4	5	6
MNIST	73,67%	11,92%	9,91%	3,27%	1,17%	0,06%
Бланки	66,78%	16,02%	9,92%	4,58%	2,56%	0,14%

Как видно из таблицы 4.12, дополнительная обработка алгоритмом Ву-Цая необходима примерно четверти изображений набора MNIST и трети изображений набора из бланков, аналогичных бланкам ЕГЭ. На графическом представлении результатов экспериментов заметно, что для большинства изображений количество пропущенных пикселей, которые необходимо удалить, находится в диапазоне от 0 до 7. Крайне редко возникает необходимость в удалении большего числа пикселей.

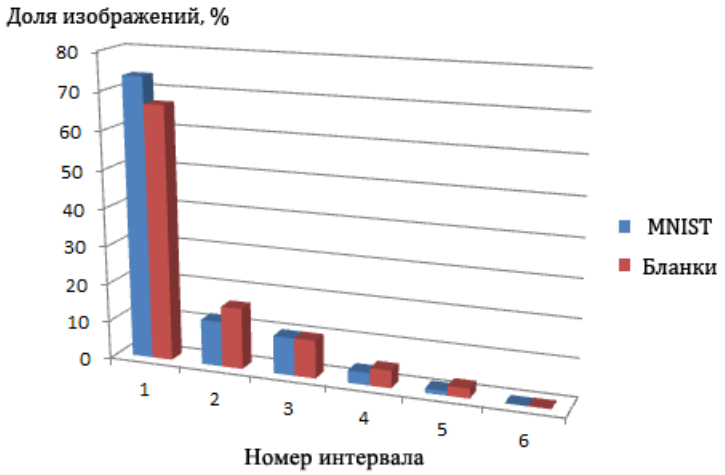


Рисунок 4.11 – Графическое преставление распределения изображений используемых наборов по интервалам количества пропущенных пикселей

Полученные результаты позволяют сделать вывод о необходимости использования для существенной части изображений обработки алгоритмом Ву-Цая с целью устранения пикселей, которые образуют области на бинарном изображении, являющиеся причиной неоднозначности при выборе порядка обхода изображения алгоритмом Ли.

4.6. Выбор оптимального порогового значения угла для разделения изгибов и ключевых точек

В алгоритме выделения структурных составляющих на стадии разделения ключевых точек и изгибов используется пороговое значение угла, равное 120 градусам. Если угол между направляющими векторами в текущей точке превышает данное пороговое значение, то точка считается изгибом, в противном случае – ключевой точкой.

Выбранное пороговое значение было определено по результатам оценки точности распознавания трёх предложенных алгоритмов на наборе рукописных цифр MNIST. Пороговое значение угла последовательно выбиралось из интервала от 15 до 165 с шагом, равным 15 градусам. Исследование производилось для количества эталонных изображений $E = 15$. Данные эксперимента по оценке точности распознавания предложенных алгоритмов приведены в таблице 4.13 и на рисунке 4.12.

Таблица 4.13 – Точность распознавания предложенных алгоритмов классификации для различных пороговых значений угла

Пороговое значение угла	**SMM**	**SMI**	**SMT**	**Среднее**
15	59,9	91,4	63,9	71,7
30	71,2	91,4	65,1	75,9
45	79,9	91,4	66,3	79,2
60	82,4	91,4	72,1	82,0
75	93,0	91,4	77,5	87,3
90	93,1	91,4	77,5	87,3
105	95,3	91,4	80,3	89,0
120	95,3	91,4	80,3	89,0
135	94,0	91,4	80,3	88,6
150	86,2	91,4	74,2	83,9
165	67,1	91,4	60,8	73,1

Полученные результаты демонстрируют, что точность метода пересечений для структурных моделей не зависит от порогового значения. Это объясняется тем, что пороговое значение влияет лишь на распределение соединяющих ребер по композитным, но не влияет на само геометрическое представление символа. Наиболее чувствительным к изменениям порогового значения является алгоритм на основе поиска максимального паросочетания

наименьшей стоимости, который опирается на характеристики композитных ребер структурных моделей.

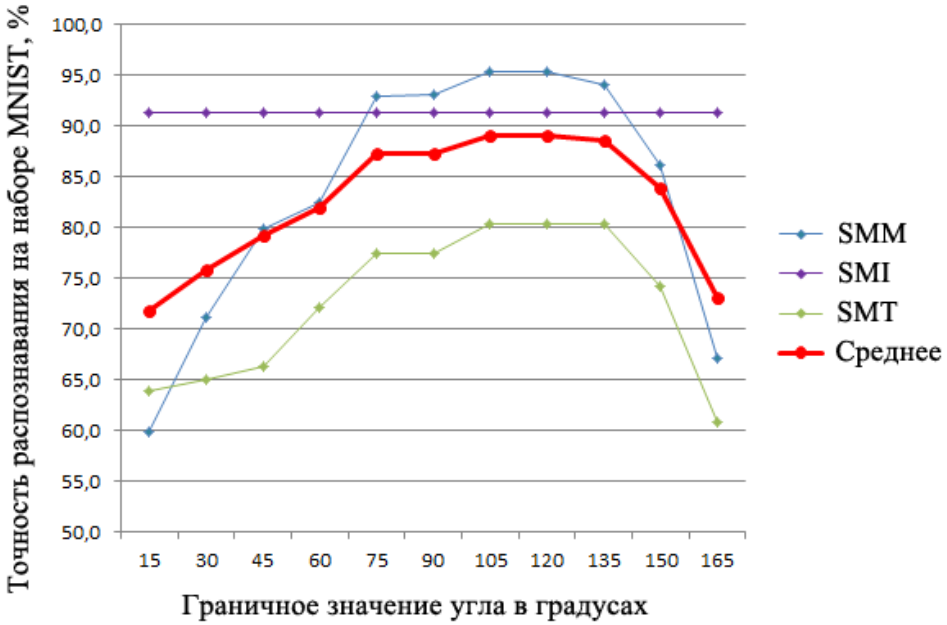


Рисунок 4.12 – Графическое представление данных из таблицы 4.13

Приведенные результаты экспериментов показывают, что оптимальное пороговое значение угла находится в диапазоне от 105 до 120 градусов. Причем значения точности распознавания для пороговых значений 105 и 120 достаточно близки, что говорит о том, что среди изображений базы MNIST очень мало таких, для которых такое изменение порогового значения приводит к изменению результирующей точности классификации.

4.7. Анализ быстродействия алгоритма сегментации на основе построения структурных моделей

Реализованный в программном приложении алгоритм сегментации рукописного текста на основе динамического программирования, как ранее было отмечено, теоретически, обладает экспоненциальной вычислительной сложностью в виду экспоненциальной оценки наибольшего возможного количества состояний. Однако было сформулировано утверждение о том, что на практике количество состояний будет достаточно невелико и, в частности,

его зависимость от количества точек интереса в исходном множестве будет носить квадратичный характер.

Для исследования быстродействия алгоритма сегментации рукописных слов на основе построения структурной модели с применением динамического программирования были выполнены начертания слов различной длины, составляющих панграммы. Для оценки степени схожести отдельных символов тем же почерком были выполнены начертания отдельных рукописных символов. Отдельные символы были начертаны так, чтобы их графические представления были максимально похожи на те, что встречаются в составе рукописных слов. Используемые при оценке степени схожести начертания символов изображены на рисунке 4.13:

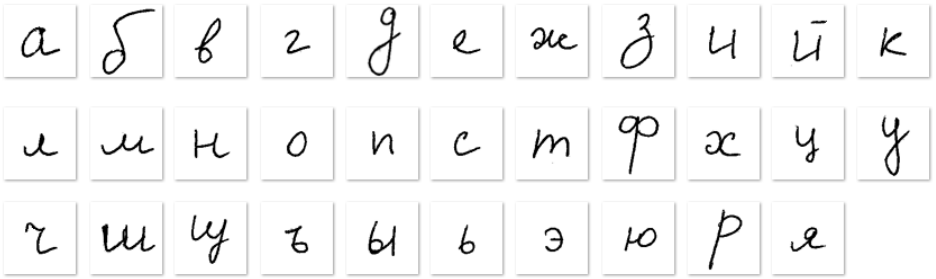


Рисунок 4.13 – Графические начертания отдельных рукописных символов

Примеры начертаний отдельных слов, которые были использованы для исследования быстродействия алгоритма сегментации, приведены на рисунке 4.14:

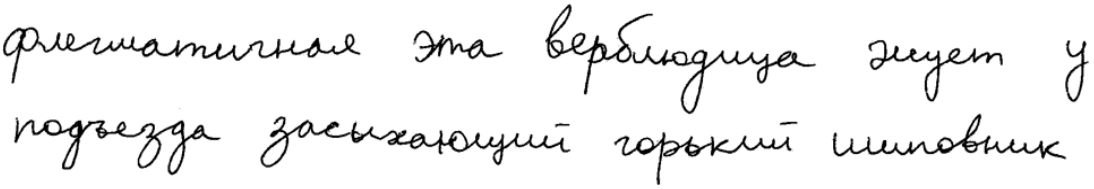


Рисунок 4.14 – Графические начертания рукописных слов

Каждое слово приведенной панграммы было записано 20 раз одним и тем же почерком. Начертания одних и тех же слов при исполнении имеют существенные отличия, поэтому для оценки быстродействия были использованы все полученные начертания слов. Иногда разные начертания

одного слова оказываются преимущественно похожими, но различаются выполнениями отдельных соединительных элементов и масштабом.

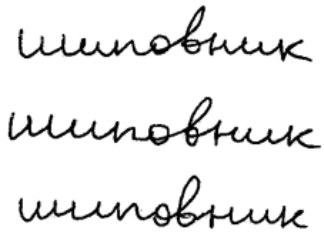


Рисунок 4.15 – Примеры схожих начертаний одного слова

В ходе тестирования алгоритма сегментации выполнялся его запуск для каждого из изображений начертаний. В процессе работы алгоритма определялись следующие параметры: время работы алгоритма, количество линий-кандидатов и количество состояний в методе динамического программирования, которое одновременно является количеством значений в хэш-таблице. Стоит отметить, что время работы включает в себя время работы всех этапов обработки исходного представления рукописного слова: построения структурной модели слова, построения множества линий-кандидатов и непосредственно поиска оптимального множества разделяющих линий.

Результаты тестирования, приведенные в таблице 4.14, демонстрируют, что средние количества точек интереса, линий кандидатов и состояний динамического программирования являются величинами, несущественно влияющими на потребление оперативной памяти. Более того, количество состояний динамического программирования, одновременное являющееся и количеством элементов в хэш-таблице, на практике оказывается настолько небольшим, что общее время работы ни на одном из изображений не превысило 1,7 секунды.

Отдельно можно исследовать сделанное ранее предположение о зависимости количества состояний динамического программирования от количества линий-кандидатов. Предполагалось, что такого рода зависимость будет близка к квадратичной.

Таблица 4.14 – Результаты тестирования быстродействия алгоритма сегментации на основе построения структурных моделей

Слово	Ср. кол-во точек интереса	Ср. кол-во линий-кандидатов	Ср. кол-во состояний	Ср. время работы
Флегматичная	77,10	156,50	12275,90	1,448
Эта	27,45	57,70	1739,55	0,392
Верблюдица	71,95	146,05	10692,95	1,292
Жует	46,40	95,70	4636,60	0,693
У	10,00	20,75	252,55	0,230
Подъезда	58,95	119,80	7204,20	0,946
Засыхающий	82,80	168,40	14208,90	1,669
Горький	43,25	87,00	3855,80	0,599
Шиповник	59,950	122,450	7531,050	0,978

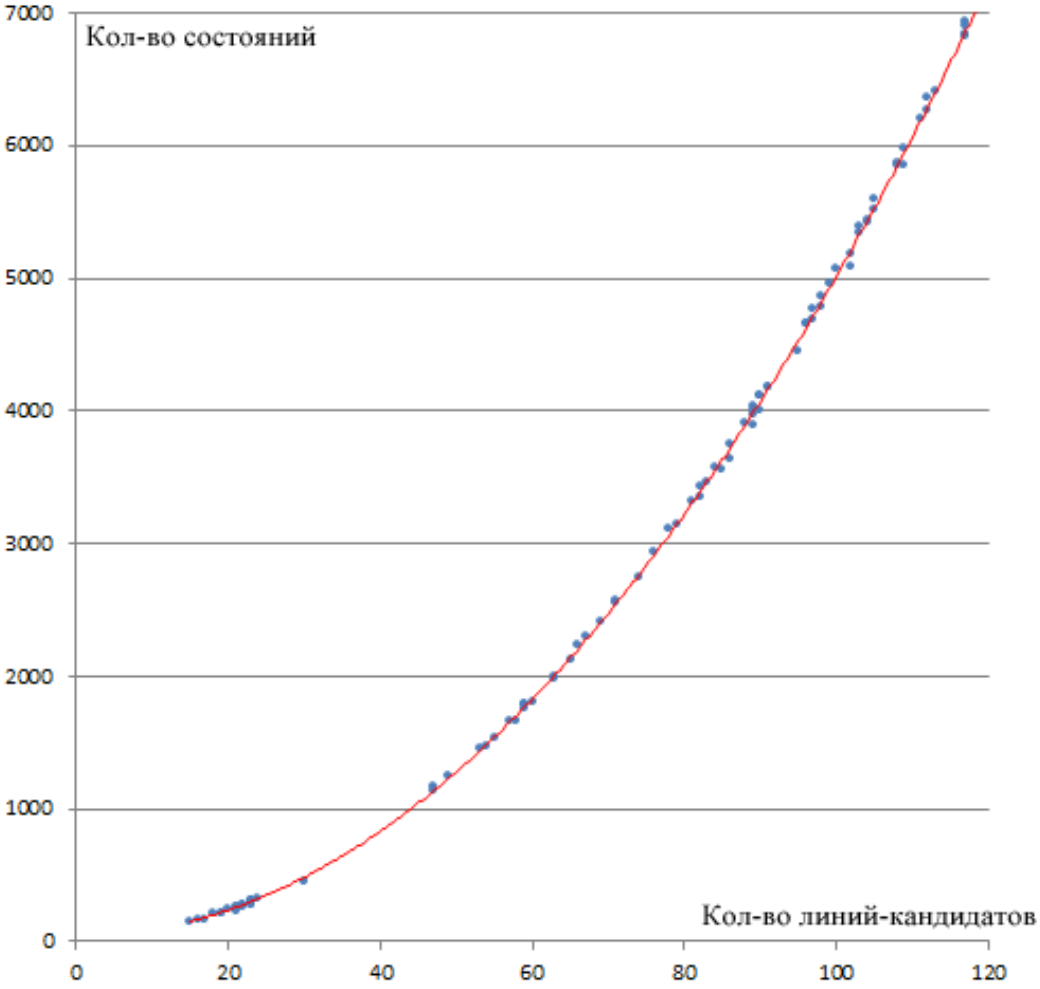


Рисунок 4.16 – Визуализация зависимости количества состояний динамического программирования от количества линий-кандидатов

Для подтверждения данной гипотезы была построена квадратичная функция, аппроксимирующая исследуемую зависимость. В качестве измеряемых данных были использованы пары значений для каждого изображения набора: количество линий-кандидатов и соответствующее ему количество состояний динамического программирования. Измеряемые данные и построенная для них квадратичная аппроксимирующая функция показаны на рисунке 4.16.

Графическое представление результатов аппроксимации наглядно демонстрирует, что зависимость количества состояний динамического программирования от количества линий-кандидатов, как и предполагалось, близка к квадратичной. Отклонения от квадратичной зависимости в меньшую сторону объясняются недостижимостью некоторых состояний из-за пересечений определенных линий-кандидатов.

Стоит также отметить, что величина отклонений значений, полученных опытным путем, увеличивается с ростом количества линий-кандидатов.

4.8. Оценка точности алгоритма сегментации структурной модели слова

Оценить точность алгоритма сегментации независимо от алгоритма классификации отдельных символов практически невозможно. В литературе можно найти множество различных методик оценки качества сегментации, но каждая из них опирается либо на дискретность пространства изображения, что не выполняется для структурных моделей, либо на итоговое качество распознавания, что приводит к косвенной оценке сегментации через результаты последующих этапов анализа изображения.

Однако и косвенная оценка точности сегментации через точность распознавания отдельных символов может быть выполнена огромным количеством способов. Некоторые виды ошибок при сегментации или при распознавании отдельных символов могут быть компенсированы средствами

словарного поиска или с помощью других алгоритмов постобработки результатов [128].

Одним из параметров результата работы алгоритма сегментации является разность между правильным и полученным количествами сегментов. Нулевое значение такой разности не гарантирует верно выполненной сегментации, ведь излишняя сегментация в одной части структурной модели может быть компенсирована недостаточной сегментацией в другой. Однако значения, отличные от нулевых, косвенно отражают качество выполняемой сегментации. На рисунке 4.17 приведено распределение результатов работы алгоритма сегментации по значению разности между правильным и полученным количествами сегментов.

Рисунок 4.17 – Распределение отклонения полученного количества сегментов от правильного

Как можно заметить, в большинстве случаев количество сегментов совпадает с правильным, а, примерно, в 13% случаев отличается от правильного на единицу.

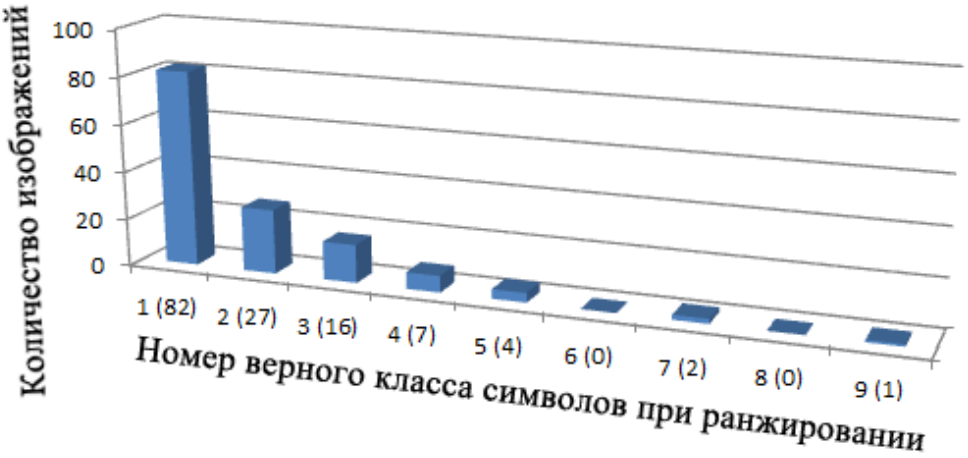


Рисунок 4.18 – Распределение значения самой низкой позиции при ранжировании отдельных символов по степени схожести

В тех случаях, когда количество сегментов совпадает, можно определить, насколько правильно выполнена классификация самих начертаний внутри сегментов. Таким способом можно косвенно оценить качество выполненной сегментации. Результаты большинства алгоритмов сегментации впоследствии нуждаются в постобработке алгоритмами словарного поиска, поэтому важно определить не только наиболее похожий, с точки зрения результата классификации, символ, но и определить, насколько высоко при ранжировании по степени схожести он оказался. С данной целью было построено распределение результатов сегментации по самой низкой позиции символа при ранжировании по степени схожести.

Представленное распределение показывает, что в 90% случаев верный символ всегда оказывался среди первых трех в порядке ранжирования, вариантов. С высокой долей вероятности в таких случаях средства словарного поиска могут быть эффективно использованы для исправления ошибок при ранжировании отдельных символов по степени схожести с эталонами.

4.9. Анализ корректности результатов алгоритма построения структурной модели

Если для решения задачи распознавания рукописных символов используется малое количество эталонных образов, то необходимо обеспечить максимальное разнообразие форм начертания символов среди изображений, которые используются в роли эталонных.

В общем случае, это означает, что при выборе эталонных изображений следует выбирать изображения символов, которым соответствуют наименее похожие структурные модели.

Однако стоит помнить, что структурные модели для некоторых изображений могут быть построены с существенными ошибками. Это может привести к тому, что построенная структурная модель не будет соответствовать геометрическому представлению начертанного на исходном изображении символа.

На рисунке 4.19 приведены достаточно похожие друг на друга начертания символов и визуализации соответствующих им структурных моделей.

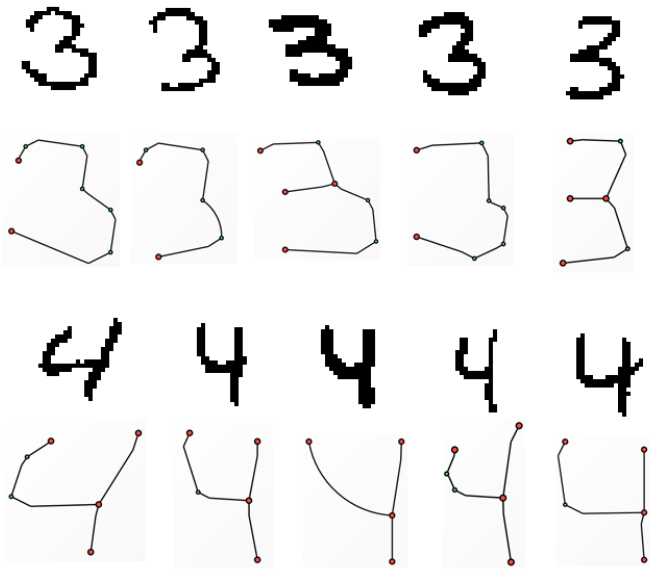


Рисунок 4.19 – Изображения рукописных символов и соответствующих им структурных моделей

Как можно заметить, структурные модели, как правило, различаются, но имеют достаточно много похожих структурных составляющих. В целом структурные модели соответствуют геометрии исходного начертания символа, и взгляда на структурную модель без исходного изображения человеку достаточно, чтобы понять, какому символу она соответствует.

Структурные модели двух не похожих между собой начертаний символов одного класса могут различаться еще более существенно из-за несовершенства алгоритмов. Пример различающихся начертаний одного и того же символа и визуализаций соответствующих им структурных моделей представлены на рисунке 4.20.

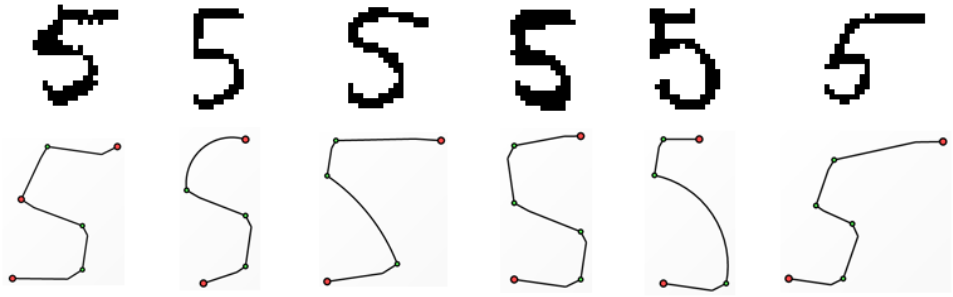


Рисунок 4.20 – Изображения с различными начертаниями рукописных символов одного класса и соответствующих им структурных моделей

В случае с такими изображениями структурные модели различаются более существенно: разные пропорции схожих по положению геометрических элементов и разные типы геометрических примитивов, задающих соединяющие ребра. Как и ранее, структурные модели достаточно близко аппроксимируют исходные представления символов.

При анализе изображений символов, которые были классифицированы неверно, было обнаружено, что для них были построены неверные структурные модели. Примеры изображений таких символов, результатов их скелетизации, а также визуализации соответствующих им структурных моделей показаны на рисунке 4.21.

Как можно заметить, нередко к построению неправильной структурной модели приводит результат работы алгоритма скелетизации. Однако, как,

например, в случае с изображение арабской цифры «4» на приведенном изображении, сказывается и несовершенство алгоритма выделения структурных составляющих. Чаще всего такое происходит в случае, когда некоторые важные точки изображения после скелетизации становятся малозаметными, или несколько ключевых пикселей оказываются слишком близко друг к другу, в результате чего объединяются в одну ключевую точку при построении структурной модели.

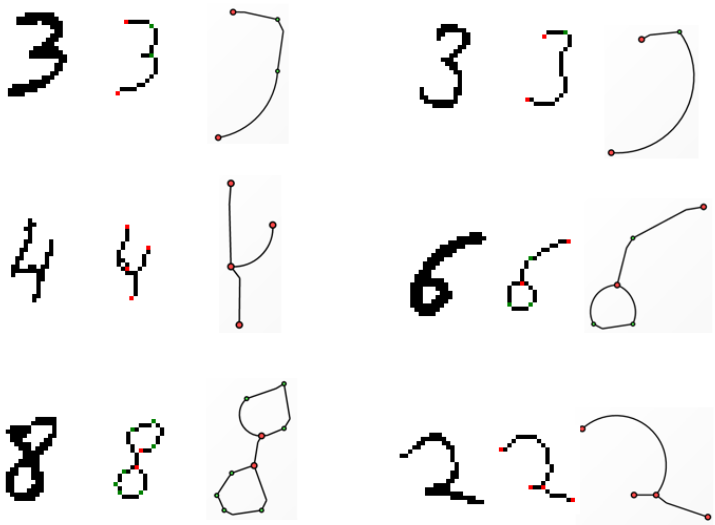


Рисунок 4.21 – Изображения символов до и после скелетизации, а также соответствующих им структурных моделей

В наборе символов из бланков, аналогичных бланкам ЕГЭ, как правило, малое разнообразие форм начертания привело к тому, что структурные модели редко строились некорректно. Причиной некорректной структурной модели могло стать лишь некорректное исполнение начертания самим исполнителем. Примеры таких некорректных исполнений и визуализаций соответствующих им структурных моделей представлены на рисунке 4.22.

Стоит отметить, что из всех приведенных изображений, только одно изображение кириллической буквы «Ж» было некорректно классифицировано из-за неточной структурной модели.

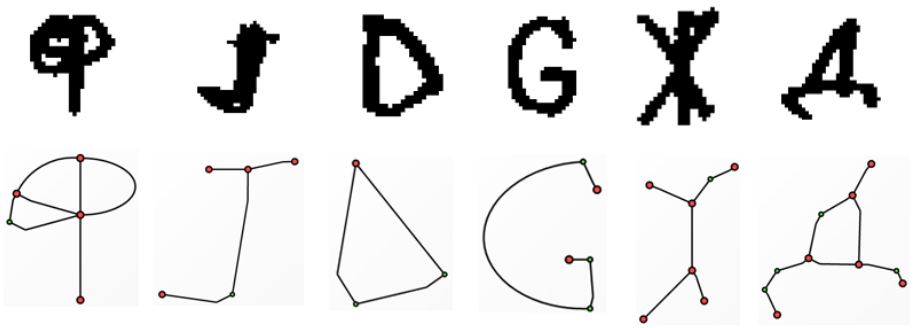


Рисунок 4.22 – Изображения некорректно начертанных символов и соответствующих им структурных моделей

В большинстве случаев достаточно аккуратные начертания данного набора изображений не составляли трудностей для алгоритма построения структурных моделей. Структурные модели, в основном, содержат лишь несущественные отклонения от эталонных представлений.

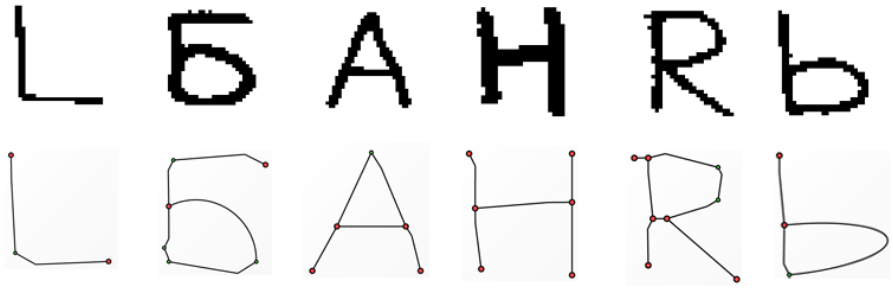


Рисунок 4.23 – Примеры изображений символов и соответствующих им структурных моделей

В результате визуального анализа структурных моделей можно сделать вывод, что первоочередным направлением для улучшения качества построения структурных моделей является улучшение подхода к скелетизации бинарного изображения символа.

4.10. Анализ корректности результатов алгоритма построения структурных моделей слов

Визуальный анализ корректности построенных описанным алгоритмом структурных моделей слов показал, что, в целом, полученные модели достаточно точно описывают исходные графические представления слов.

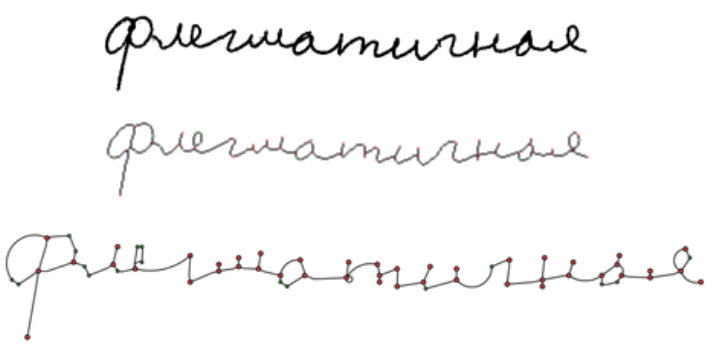


Рисунок 4.24 – Пример начертания слова, результата его скелетизации и структурной модели

Недостатки, характерные полученным структурным моделям, аналогичны недостаткам моделей отдельных символов, так как качество выполнения операции скелетизации оказывает наиболее существенное влияние на топологию и геометрические характеристики итоговой структурной модели.

Рисунок 4.25 – Пример результата некорректной скелетизации начертания слова и соответствующей структурной модели

Пример недостаточно точной структурной модели слова приведен на рисунке 4.25. Особое внимание стоит обратить на пятую букву слова, топология которой, в результате неточной скелетизации, была искажена. Такого рода неточность не повлияла на сегментацию, однако привела к тому, что пятый символ был ошибочно классифицирован, как «И», вместо «К».

Анализ результатов работы алгоритма сегментации показал, что причиной неверной сегментации чаще всего является схожесть отдельных

элементов начертания рукописных символов, а неточности, допущенные при построении структурной модели, как правило, приводят к неверной классификации отдельных символов, но не оказывают существенного влияния на качество сегментации.

4.11. Основные результаты и выводы по главе 4

1. В данной главе представлены результаты тестирования предложенных алгоритмов на двух наборах изображений рукописных символов в условиях малого количества эталонных изображений. При тестировании оценивалось качество распознавания разработанных алгоритмов и среднее время распознавания одного изображения.

2. Представлены результаты сравнения качества распознавания предложенных алгоритмов с аналогами, способными функционировать в условиях малой обучающей выборки. Установлено, что в подобных условиях предложенные алгоритмы гораздо более эффективны для решения задачи распознавания рукописных символов.

3. Исследована эффективность предложенного подхода к скелетизации бинарного изображения. Экспериментально подтверждена необходимость применения алгоритма Ву-Цая и дополнительной операции предварительного устранения плоских окончаний.

4. Исследован алгоритм сегментации рукописного текста с целью оценки его точности и быстродействия. Подтверждена близкая к полиномиальной зависимость времени работы от количества линий-кандидатов.

5. Подготовлен и описан оригинальный набор рукописных символов, отличительными особенностями которого являются максимальная приближенность начертаний всех символов к определенному образцу, а также наличие символов 68 классов: кириллического алфавита, латинского

алфавита и арабских цифр. Подготовленный набор был использован для апробации предложенных алгоритмов.

6. С помощью визуализатора структурных моделей обнаружены примеры неправильно построенных структурных моделей. Установлено, что чаще всего неверная структурная модель строится из-за несовершенства используемых алгоритмов скелетизации. Однако встречаются и случаи, когда причиной становятся неточности разработанного алгоритма выделения структурных составляющих.

ЗАКЛЮЧЕНИЕ

Диссертационная работа посвящена разработке алгоритмов распознавания рукописных символов в условиях малой обучающей выборки. По итогам её выполнения получены следующие научные и практические результаты:

1. Исследованы существующие методы распознавания печатных и рукописных символов.
2. Предложена структурная модель символа для применения в решении задачи распознавания рукописных символов в условиях малой обучающей выборки.
3. Разработан алгоритм построения предложенной структурной модели символа по растровому представлению его начертания.
4. Выбраны критерии схожести структурных моделей символов.
5. Реализованы алгоритмы распознавания рукописных символов в условиях малой обучающей выборки на основе применения предложенной структурной модели символа и выбранных критериев схожести структурных моделей символов.
6. Реализован комплекс программ для исследования и сравнительного анализа разработанных и существующих алгоритмов распознавания символов в условиях малой обучающей выборки, и проведены вычислительные эксперименты с целью оценки качества и эффективности разработанных алгоритмов.

Разработанные алгоритмы имеют самостоятельное значение и, помимо задачи распознавания символов, могут применяться для решения задач классификации отпечатков пальцев, идентификации почерка, проверки подписей на подлинность и других задач, связанных с анализом бинарных изображений.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ И ЛИТЕРАТУРЫ

1. Фомин Я. А. Распознавание образов: теория и практика. Издание третье, дополненное / Я. А. Фомин // М.: ФАЗИС. – 2014. – 460 с.

2. Anderson N. Optical Character Recognition – Part 1 / N. Anderson // IMPACT Best Practice Guide. – 2010.

3. ABBYY FineReader [Электронный ресурс]. – Режим доступа: http://www.abbyy.ru/finereader/, свободный. – Загл. с экрана (дата обращения: 01.04.2017).

4. Cuneiform Windows [Электронный ресурс]. – Режим доступа: http://cognitiveforms.com/ru/products_and_services/ cuneiform, свободный. – Загл. с экрана (дата обращения: 01.04.2017).

5. Dynamsoft OCR SDK [Электронный ресурс]. – Режим доступа: http://www.dynamsoft.com, свободный. – Загл. с экрана (дата обращения: 01.04.2017).

6. Lam S. W. An Adaptive Approach to Document Classification and Understanding / S. W. Lam // IAPR Workshop on Document Analysis Systems, Kaiserslautern, Germany. – 1994. – P. 231-251.

7. Schantz H. F. History of OCR, Optical Character Recognition / Schantz H. F. // Recognition Technologies Users Association, 1982. – 114 p.

8. LeCun Y. Handwritten Zipcode Recognition With Multilayer Networks / Y. LeCun, O. Matan, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, L. D. Jackel, H. S. Baird // Proc. of International Conference on Pattern Recognition, Atlantic City. – 1990. – P. 541-551.

9. Lee S.-W. Off-line Recognition of Totally Unconstrained Handwritten Numerals Using Multilayer Cluster Neural Network / S.-W. Lee, Y. J. Kim // Proc. of the 12th IAPR International Conference on Pattern Recognition. Jerusalem, Israel. – 1994. – P. 507-509.

10. Krzyzak A. Unconstrained Handwritten Character Classification Using Modified Backpropagation Model / A. Krzyzak, W. Dai, C. Y. Suen // Proc. 1st Int. Workshop on Frontiers in Handwriting Recognition, Montreal, Canada. – 1990. – P. 155-166.

11. LeCun Y. Efficient BackProp / Y. LeCun, L. Bottou, G. B. Orr, K.-R. Muller // Neural Networks: tricks of the trade. Springer. – 1998. – P. 9-48.

12. Ping Z. Documents filters using morphological and geometrical features of characters / Z. Ping, C. Lihui // Image and Vision Computing. 2001. – vol. 19. – P. 847-855.

13. Chen X. R. Detecting and Reading Text in Natural Scenes / X. R. Chen // Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition – 2004. – P. 366-373

14. Knerr S. Handwritten digit recognition by neural networks with single-layer training / S. Knerr, L. Personnaz, G. Dreyfus // IEEE Transactions on Neural Networks 3 – 1992. – P. 962-968.

15. Lee Y. Handwritten digit recognition using K neares-neighbor, radialbasis function, and back-propagation neural networks / Y. Lee // Neural Computation 3, – 1991. – P. 440-449.

16. Martin G. L. Recognizing hand-printed letters and digits using backpropagation learning. / G. L. Martin, J. A. Pitman // Neural Computation 3. – 1991. – P. 258-267.

17. Seong-Wang L. Off-line Recognition of Totally Unconstrained Handwritten Numerals Using Multilayer Cluster Neural Network / L. Seong-Wang, J. K. Young // Proc. Of the 12th IAPR International Conference on Pattern Recognition. Jerusalem, Israel. – 1994. – P. 507-509.

18. Kan C. Invariant Character Recognition with Zernike and Orthogonal Fourier-Mellin Moments / C. Kan, M. D. Srinath // Pattern Recognition. – 2000. – Vol. 35. – P. 143–154.

19. Krizhevsky A. ImageNet Classification with Deep Convolutional Neural Networks / A. Krizhevsky, I. Sutskever, G. Hinton // Conference on Neural Information Processing Systems (NIPS). – 2012. – 27 p.

20. Vapnik V. N. Support Vector Networks / V. Vapnik, C. Cortes // Machine Learning. – 1995. – № 20(3). – P. 273-297.

21. Круглов В. В., Дли М. И., Голунов Р. Ю. Нечеткая логика и искусственные нейронные сети – М.: Физматлит, 2001. – 224 с.

22. Вапник В.Н., Червоненкис А.Я. Теория распознавания образов. М.: Наука, 1974. – 415 с.

23. Vapnik V. N. A Training Algorithm for Optimal Margin Classifiers / V. N. Vapnik, E. Boser, I. M. Guyon // Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory. – 1992. – № 18. – P. 144-152.

24. Пытьев Ю. П. Морфологический анализ изображений / Ю. П. Пытьев // Докл. АН СССР. – Т. 269. – № 5. – 1983. – С. 1061-1064.

25. Пытьев Ю. П. Задачи морфологического анализа изображений / Ю. П. Пытьев // Математические методы исследования природных ресурсов Земли из космоса. – М: Наука, 1984. – С. 41-83.

26. Загоруйко Н. Г. Методы распознавания и их применение. – М.: Сов.радио, 1972. – 208 с.

27. Загоруйко Н. Г. Прикладные методы анализа данных и знаний. – Новосибирск: ИМ СО РАН, 1999. – 270 с.

28. Журавлев Ю. И. Распознавание. Математические методы. Программная система. Практические применения / Ю.И. Журавлев, В.В. Рязанов, О.В. Сенько. М.: ФАЗИС, 2006. – 176 с.

29. Журавлев Ю. И. Об алгебраическом подходе к решению задач распознавания или классификации / Ю. И. Журавлев // Проблемы кибернетики. – 1978. – Т. 33. – С. 5-68.

30. Коробейников А. П. Методы распознавания образов: Учебное пособие / А. П. Коробейников. – Ростов-на-Дону: Издательский центр ДГГУ, 1999. – 51 с.

31. Щепин Е. В. К топологическому подходу в анализе изображений / Е. В. Щепин, Г. М. Непомнящий // Межвузовский сборник научных трудов «Геометрия, топология и приложения». – Москва, Мин. высшего и средн. спец. образ. РСФСР, Московский институт приборостроения. – 1990. – С. 13-25.

32. Спицын В. Г. Применение искусственных нейронных сетей для обработки информации / В.Г. Спицын, Ю.Р. Цой. – Томск: ТПУ, 2007. – 32 с.

33. Спицын В. Г. Применение генетического алгоритма для решения задач оптимизации / В. Г. Спицын, Ю. Р. Цой. – Томск: ТПУ, 2007. – 27 с.

34. Местецкий Л. М. Математические методы распознавания образов / Л. М. Местецкий. – М.: МГУ, ВМиК, 2002. – 85 с.

35. Садыков С. С. Скелетизация бинарных изображений / С. С. Садыков, И. Р. Самандаров // Зарубежная радиоэлектроника. – 1985. – № 11. – С. 30-37.

36. Арлазаров В.Л., Астахов А.Д., Троянкер В.В., Котович Н.В. Адаптивное распознавание символов // Интеллектуальные технологии ввода и обработки информации. – 1998. – С. 39-56.

37. Арлазаров В.Л., Славин О.А. Алгоритмы распознавания и технологии ввода текстов в ЭВМ. // Информационные технологии и вычислительные системы. – 1996. – № 1. – С. 48-54.

38. Фаворская М. Н. Модель распознавания изображений рукописного текста / М. Н. Фаворская, А. Н. Горошкин // Вестник Сибирского государственного аэрокосмического университета имени академика М. Ф. Решетнева. – 2008. – № 2 (19).– С. 52-58.

39. Фаворская М. Н. Морфологическая обработка контурных изображений в системах распознавания текстовых символов / М. Н. Фаворская, А. С. Зотин, А. Н. Горошкин // Вестник Сибирского государственного аэрокосмического университета имени академика М. Ф. Решетнева. – 2007. – № 1 (14). – С. 70-75.

40. Хайкин С. Нейронные сети. Полный курс 2-е изд. Пер. с англ. – М.: Издательский дом «Вильямс». – 2006. – 1104 с.

41. Богданов В. Системы распознавания текстов в офисе / В. Богданов, К. Ахметов // Компьютер-пресс. – 1999. – № 3.– С. 40-42.

42. Аникеев М. В. Алгоритм распознавания бланков факсимильных сообщений. / М. В. Аникеев, В. М. Федоров, Н. Н. Цопкало // Известия ТРТУ. Специальный выпуск «Материалы XLVII научно-технической конференции» – 2002. – № 1(24). – С. 146-147.

43. Wang P.S.P. An improved structural approach for automated recognition of handprinted characters, in: P.S.P. Wang (Ed.) / P.S.P. Wang, A. Gupta / Character and Handwriting Recognition: Expanding Frontiers, World Scientific, Singapore. – 1991. – P. 97-121.

44. Буй Т. Т. Ч. Алгоритмическое и программное обеспечение для классификации цифровых изображений с помощью вейвлет-пребразования Хаара и нейронных сетей. / Т. Т. Ч. Буй, Н. Х. Фан, В. Г. Спицын // Известия Томского политехнического университета. – 2011. – Т. 319. – № 5. – С. 103-106.

45. Назаров Л. Е. Нейросетевые алгоритмы обнаружения, классификации и распознавания объектов на изображениях / Л. Е. Назаров, Н. С. Томашевич, А. Н. Балухто // Нейрокомпьютеры в прикладных задачах обработки изображений. – 2007. – Кн. 25. – С. 25-54.

46. Рутковская Д. Нейронные сети, генетические алгоритмы и нечёткие системы / Д. Рутковская, М. Пилиньский, Л. Рутковский. – М.: Горячая линия-Телеком, 2004. – 452 с.

47. Fukushima K. Neocognition: a hierarchical neural network capable of visual pattern recognition / K. Fukushima // Neural Networks. – 1988. – Vol. 1(2). – P. 119-130.

48. Фан Н. Х. Распознавание печатных текстов на основе применения вейвлет-преобразования и метода главных компонент. / Н. Х. Фан,

Т. Т. Ч. Буй, В. Г. Спицын // Известия Томского политехнического университета. – 2012. – Т. 321. – № 5. – С. 154-158.

49. Фрейзер М. Введение в вэйвлеты в свете линейной алгебры: Пер. с англ. / М. Фрейзер // М.: БИНОМ. Лаборатория знаний. – 2008. – 487 с.

50. Manjunath Aradhya V. N., Probabilistic Neural Network based Approach for Handwritten Character Recognition. / V. N. Manjunath Aradhya, S. K. Niranjan, G. Hemantha Kumar // Special Issue of IJCCT 2010 for International Conference [ACCTA-2010], 3-5 August 2010. – Vol.1 Issue 2, 3, 4.

51. Pujari A. K. An intelligent character recognizer for Telugu scripts using multiresolution analysis and associative memory / A. K. Pujari, C. D. Naidu, S. Rao, B. C. Jinaga // Image and Vision Computing. – 2004. – Vol. 22, is. 14. – P. 1221–1227.

52. Specht D. F. Probabilistic Neural Networks. / D. F. Specht // Neural Networks. – 1990. – № 3. – P. 109-188

53. Bishop C. M. Neural Networks for Pattern Recognition – Clarendon Press, Oxford. – 1995. – 482 p.

54. Dhandra B. V. Kannada, Telugu and Devanagari Handwritten Numeral Recognition with Probabilistic Neural Network: A Novel Approach / B. V. Dhandra, R. G. Benne, M. Hangarge // IJCA Special Issue on «Recent Trends in Image Processing and Pattern Recognition» RTIPPR. – 2010. – P. 83-88.

55. Трегубов А. А. Контекстное нейросетевое распознавание символов с учетом словаря и переходных вероятностей / А. А. Трегубов, Н. Н. Цопкало // Сборник трудов научно-практической конференции «Информационная безопасность», Россия, Таганрог, 28-31 мая. – 2002. – С. 112-118.

56. Liwicki M. A Novel Approach to On-Line Handwriting Recognition Based on Bidirectional Long Short-Term Memory Networks / M. Liwicki, A. Graves, H. Bunke, J. Schmidhuber // Proc. 9th Int. Conf. on Document Analysis and Recognition. – 2007. – P. 367–371.

57. Болотова Ю. А. Алгоритмы обработки и анализа изображений иерархической временной сетью: диссертация на соискание ученой степени кандидата наук 05.13.01 / Ю. А. Болотова. – 2013. – 162 с.

58. Болотова Ю. А. «Распознавание автомобильных номеров на основе метода связных компонент и иерархической временной сети» / Ю. А. Болотова, В. Г. Спицын, М. Н. Рудометкина // Компьютерная оптика. – 2015. – № 39(2). – С. 275-280.

59. Bertolami R. Combination of Multiple Handwritten Text Line Recognition Systems with a Recursive Approach / R. Bertolami, B. Halter, H. Bunke // Proc. 10th Int. Workshop Frontiers in Handwriting Recognition. – 2006. – P. 61-65.

60. Vapnik V. N. Support Vector Networks / V. Vapnik, C. Cortes // Machine Learning. – 1995. – № 20(3). – P. 273-297.

61. Vapnik V. N. A Training Algorithm for Optimal Margin Classifiers / V. N. Vapnik, E. Boser, I. M. Guyon // Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory. – 1992. – № 18. – P. 144-152.

62. LeCun Y. Gradient-Based Learning Applied to Document Recognition [Электронный ресурс] / Y. LeCun, L. Bottou, Y. Bengio, P. Haffner // Proceedings of the IEEE, Vol. 86, Issue 11, Nov 1998. – 1998. – URL: http://yann.lecun.com/exdb/publis/pdf/lecun-98.pdf (дата обращения: 01.04.2017).

63. Bouchain D. Character Recognition Using Convolutional Neural Networks [Электронный ресурс] / D. Bouchain // Seminar Statistical Learning Theory, Winter 2006/2007. – 2017. – URL: http://bertrand.granado.free.fr/iWeb/archiparalleles/Travaux_Pratiques_files/Bouchain.pdf (дата обращения: 01.04.2017).

64. Друки А. А. Применение сверточных нейронных сетей для выделения и распознавания автомобильных номерных знаков на изображениях / А. А. Друки // Известия Томского политехнического университета. – 2014. – Т. 324. – № 5. – С. 85-91.

65. Wang T. End-to-end text recognition with convolutional neural networks [Электронный ресурс] / T. Wang, D. J. Wu, A. Coates, A. Y. Ng // Pattern Recognition (ICPR), 2012 21st International Conference on. – 2012. – URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.252.8930&rep=rep1&type=pdf (дата обращения: 01.04.2017).

66. Kavallieratou E. Handwritten Character Recognition based on Structural Characteristics / E. Kavallieratou, K. Sgarbas, N. Fakotakis, G. Kokkinakis // Proceeding ICDAR '03 Proceedings of the Seventh International Conference on Document Analysis and Recognition. – 2003. – Vol. 1. – P. 139-142.

67. Андреев С. В., Бондаренко А. В., Горемычкин В. И., Ермаков А. В., Желтов С. Ю. Исследование подходов к построению систем автоматического считывания символьной информации. Препринт ИПМ им. М.В.Келдыша РАН №44. – 2003. – 18 с.

68. Tu M. Genetic Algorithms for Automated Negotiations: A FSM-based Application Approach / M. Tu, E. Wolff, W. Lamersdorf // Proceedings of the 11-th International Conference on Database and Expert Systems. – 2000.

69. Chan K.-F. Recognizing on-line handwritten alphanumeric characters through flexible structural matching / K.-F. Chan, D.-Y. Yeung // Pattern Recognition 32. – 1999. – P. 1099-1114.

70. Lucas S. A comparison of syntactic and statistical techniques for off-line OCR / S. Lucas, E. Vidal, A. Amiri, S. Hanlon, J. C. Amengual // in: R.C. Carrasco, J. Oncina (Eds.), Grammatical Inference and Applications (ICGI-94), Springer, Berlin, September. – 1994. – P. 168-179.

71. Berthod M. Learning in syntactic recognition of symbols drawn on a graphic tablet / M. Berthod, J. P. Maroy // Computer Graphics Image Process 9. – 1979. – P. 166-182.

72. Kassel R. A comparison of approaches to on-line handwritten character recognition / R. Kassel // Ph.D. Thesis, MIT Department of Electrical Engineering and Computer Science. – 1995. – 162 p.

73. Chan K. F. Towards efficient structural analysis of mathematical expressions / K.-F. Chan, D.-Y. Yeung // in: A. Amin, D. Dori, P. Pudil, H. Freeman (Eds.), Advances in Pattern Recognition, Springer, Berlin. – 1998. – P. 437-444.

74. Bunke H. Syntactic and Structural Pattern Recognition – Theory and Applications / H. Bunke, A. Sanfeliu (Eds.) // World Scientific, Singapore. – 1990. – 554 p.

75. Ishidera E. A Study on Top-down Word Image Generation for Handwritten Word Recognition / E. Ishidera, D. Nishiwaki // Proc. of the 7th Int'l Conf. on Document Analysis and Recognition. Edinburgh, Scotland, August 3-6, 2003. – Vol. 2. – 2003. – P. 1173-1177.

76. Kim G. An Architecture for Handwritten Text Recognition Systems / G. Kim, V. Govindaraju, S. N. Srihari // Int'l Journal on Document Analysis and Recognition. – 1999. – № 2(1). – P. 37-44.

77. Kornai A. Recognition of Cursive Writing on Personal Checks / A. Kornai, K. M. Mohiuddin, S. D. Connell // Proc. of the 5th Int'l Workshop on Frontiers in Handwriting Recognition. Colchester, UK, September 2-5. – 1996. – P. 373-378.

78. Madhvanath S. The Role of Holistic Paradigms in Handwritten Word Recognition. / S. Madhvanath, V. Govindaraju // Trans. on Pattern Analysis and Machine Intelligence. – 2001. – № 23(2). – P. 149-164.

79. Marti U.-V. Using a Statistical Language Model to Improve the Performance of an HMM-Based Cursive Handwriting Recognition System / U.-V. Marti, H. Bunke // Int'l Journal of Pattern Recognition and Artifical Intelligence. – 2001. – № 15(1). – P. 65-90.

80. Цопкало Н. Н. Распознавание символов печатного текста по структурным признакам с использованием нейросети. / Н. Н. Цопкало // Сборник трудов второго регионального научно-практического семинара «Информационная безопасность — юг России». – 2000. – С. 130-135.

81. Pavlidis T. A Thinning Algorithm for Discrete Binary Images. / T. Pavlidis // CGIP, 13. – 1980. – P. 142-157.

82. Pavlidis T. Structural Pattern Recognition / T. Pavlidis // Berlin, Heidelberg, New York: Springer Verlag. – 1977. – 302 p.

83. Stefanelli R. Some Parallel Thinning Algorithms for Digital Pictures / R. Stefanelli, A. Rosenfeld // JACM, 18. – 1971. – P. 255-264.

84. Lu H. E. A comment on «A Fast Parallel Algorithm for Thinning Digital Patterns» / H. E. Lu, P. S. P. Wang // Commun. ACM. – 1986. – Vol. 19 – № 3. – P. 239-242.

85. Rosenfeld A. A characterization of parallel thinning algorithms / A. Rosenfeld // Information and control. – 1975. – Vol. 29. – P. 286-291.

86. Deng W. A fast parallel thinning algorithm for the binary image skeletonization / W. Deng, S. Iyengar, N. Brener // The International Journal of High Performance Computing Applications. – 2000. – Vol. 14. – № 1. – P. 65-81.

87. Lam L. An Evaluation of Parallel Thinning Algorithms for Character Recognition / L. Lam, C. Y. Suen // IEEE Transactions on Pattern Analysis and Machine Intelligence. – 1995. – Vol. 17. – № 9.– P. 914-919.

88. Zhang T. Y., A Fast Parallel Algorithm for Thinning Digital Patterns / T. Y. Zhang, C. Y. Suen // Image Processing and Computer Vision. – 1984. – № 3. – P. 236-239.

89. Wu R.-Y. A new one-pass parallel thinning algorithm for binary images / R.-Y. Wu, W.-H. Tsai // Pattern Recognition Letters. – 1992. – № 13. – P. 715-723.

90. Guo Z. Parallel Thinning with Two-Subiteration Algorithms / Z. Guo, R. W. Hall // Image Processing and Computer Vision. 3. – 1989. – P. 359-373.

91. Лагунов Н. А. Влияние предобработки изображений на качество обучения нейронной сети для их распознавания / Н. А. Лагунов, О. С. Мезенцева // Вестник Северо-Кавказского государственного технического университета. – 2015. – № 1(46). – С. 21-25.

92. Raju A. A Comparative Analysis of Histogram Equalization based Techniques for Contrast Enhancement and Brightness Preserving / A. Raju, G. S. Dwarakish, D. Venkat Reddy // International Journal of Signal Processing, Image Processing and Pattern Recognition. – 2013. – Vol. 6 – № 5. – P. 353-366.

93. Соколов А. Е. Исследование библиотек технического зрения для построения системы учёта движения на перекрёстках / А. Е. Соколов // Международная конференция «Современные проблемы математики, информатики и биоинформатики». Собр. тезисов. – С. 49-52.

94. Otsu N. A Threshold Selection Method From Gray-level Histograms / N. Otsu // IEEE Trans. Sys., Man., Cyber. 9. – 1979. – P. 62-66.

95. Jianzhuang L. Automatic Thresholding of Gray-level Pictures Using Two-Dimension Otsu Method / L. Jianzhuang, L. Wenqing, T. Yupeng // Circuits and Systems. Conference Proceedings. – 1991. – P. 325-327.

96. Widiarti A. R. Comparing Hilditch, Rosenfeld, Zhang-Suen, and Nagendraprasad-Wang-Gupta Thinning / A. R. Widiarti // International Journal of Computer, Electrical, Automation, Control and Information Engineering. – 2011. – Vol. 6, Issue 5. – P. 20-24.

97. Lee C. Y. An Algorithm for Path Connections and Its Applications / C. Y. Lee // IRE Transactions on Electronic Computers, EC-10. – 1961. – № 2 – P. 346-365.

98. Ильин В. А. Аналитическая геометрия / В. А. Ильин, Э. Г. Позняк // М.: ФИЗМАТЛИТ, 2002. – 240 с.

99. Буркатовская Ю. Б. Теория графов: учебное пособие / Ю. Б. Буркатовская; Национальный исследовательский Томский политехнический университет (ТПУ). – 2014. – Ч. 1. – 213 с.

100. Берж К. Теория графов и ее применения. – М.:Изд. иностр. лит. – 1962. – 320 с.

101. Харари Ф. Теория графов. – М.: Мир. – 2006. – Изд. 3. – 296 с.

102. Кристофидес Н. Теория графов. Алгоритмический подход. – М.: Мир, 1978. – 432 с.

103. Майника Э. Алгоритмы оптимизации на сетях и графах. – М.: Мир, 1981. – 324 с.

104. Евстигнеев В. А. Применение теории графов в программировании / В. А. Евстигнеев. – М.: Наука, 1985. – 125 с.

105. Гладков Л. А., Курейчик В. В., Курейчик В. М. Генетические алгоритмы. – М.: Физматлит, 2006. – 320 с.

106. Nedjah N. Mealy Finite State Machines: An Evolutionary Approach / N. Nedjah, L. Mourelle // International Journal of Innovative Computing, Information and Control. – 2006. – Vol. 2 – № 4.– P. 58-67.

107. Davis L. Adapting operator probabilities in genetic algorithms / L. Davis // Proceedings of the Third International Conference on Genetic Algorithms. – La Jolla: Morgan Kaufmann Publishers. – 1989. – P. 60-69.

108. Miller B. Genetic algorithms, tournament selection, and the effects of noise / B. Miller, M. Goldberg // Complex Systems. – 1995. – № 3– P. 193-212.

109. Pacquet T. Recognition of Handwritten Sentences using a Restricted Lexicon / T. Pacquet, Y. Lecourtier // Pattern Recognition – 1993. – № 26(3). – P. 391-407.

110. Plamondon R. On-Line and Off-Line Handwriting Recognition: A Comprehensive Survey. / R. Plamondon, S. N. Srihari // IEEE Trans. on Pattern Analysis and Machine Intelligence 22:1 (2000) 63-84.

111. Rabiner L. R. A Tutorial on Hidden Markov Models and Selected Applications / L. R. Rabiner // Speech Recognition. Proc. of the IEEE. – 1989. – № 77(2). – P. 257-286.

112. Rath T. M. Features for Word Spotting in Historical Manuscripts. / T. M. Rath, R. Manmatha // Proc. of the 7th Int'l Conf. on Document Analysis and Recognition (ICDAR), Edinburgh, Scotland, August 3-6, 2003. – 2003. – Vol. 1. – P. 218-222.

113. Vinciarelli A. Offline Recognition of Large Vocabulary Cursive Handwritten Text. / A. Vinciarelli, S. Bengio, H. Bunke // Proc. of the 7th Int'l

Conf. on Document Analysis and Recognition. Edinburgh, Scotland, August 3-6. – 2003. – Vol. 1. – P. 1101-1105.

114. Rath T. M. Word Image Matching Using Dynamic Time Warping. / T. M. Rath, R. Manmatha // Proc. of the Conf. on Computer Vision and Pattern Recognition, Madison, WI, June 18-20. – 2003. – Vol. 2. – P. 521-527.

115. Tomai C. I. Transcript Mapping for Historic Handwritten Document Images. / C. I. Tomai, B. Zhang, V. Govindaraju // Proc. of the 8th Int'l Workshop on Frontiers in Handwriting Recognition 2002, Niagara-on-the-Lake, ON, August 6-8. – 2002. – P. 413-418.

116. Trier O. D. Feature Extraction Methods for Character Recognition – A Survey / O. D. Trier, A. K. Jain, T. Taxt // Pattern Recognition. – 1996. – № 29(4). – P. 641-662.

117. Кормен Т., Лейзерсон Ч., Ривест Р. Алгоритмы: построение и анализ. МЦНМО. – 2001. – 1296 с.

118. OpenCV Library [Электронный ресурс]. – Режим доступа: http://opencv.org/downloads.html. Дата обращения: 01.04.2017.

119. Eigen [Электронный ресурс]. – Режим доступа: http://eigen.tuxfamily.org/index.php?title=Main_Page. Дата обращения: 01.04.2017.

120. pugixml.org - Home[Электронный ресурс]. – Режим доступа: http://pugixml.org. Дата обращения: 01.04.2017.

121. OpenCV: OCR of Hand-written Data using SVM [Электронный ресурс]. – Режим доступа: http://docs.opencv.org/3.1.0/dd/d3b/tutorial_py_svm_opencv.html. Дата обращения: 01.04.2017.

122. Dalal N. Histograms of Oriented Gradients for Human Detection / N. Dalal, B. Triggs // INRIA. – 2005. – P. 120-123.

123. Chapetle O. Support vector machines for histogram-based image classification / O. Chapetle, P. Haffner, V. Vapnik // IEEE Neural Networks. – 1999. – № 10(5). – P. 1055-1064.

124. Probabilistic Neural Networks [Электронный ресурс]. Режим доступа: http://courses.cs.tamu.edu/rgutier/cpsc636_s10/specht1990pnn.pdf. Дата обращения: 01.04.2017.

125. An Introduction to Probabilistic Neural Networks [Электронный ресурс]. Режим доступа: http://www.psi.toronto.edu/~vincent/research/presentations/PNN.pdf. Дата обращения: 01.04.2017.

126. Kohavi R. On Applied Research in Machine Learning / R. Kohavi, F. Provost // Editorial for the Special Issue on Applications of Machine Learning and the Knowledge Discovery Process, Columbia University, New York. – Vol. 30. – 1998. – P. 127-132.

127. Classifier performance evaluation [Электронный ресурс]. – Режим доступа: http://cmp.felk.cvut.cz/~hlavac/TeachPresEn/31PattRecog/13ClassifierPerformance.pdf. Дата обращения: 01.04.2017.

128. Оценка классификатора (точность, полнота, F-мера) [Электронный ресурс]. – Режим доступа: http://bazhenov.me/blog/2012/07/21/classification-performance-evaluation.html. Дата обращения: 01.04.2017.

Публикации автора по теме диссертации:

129. Хаустов П. А. Алгоритмы распознавания рукописных символов на основе построения структурных моделей / П. А. Хаустов // Компьютерная оптика. – 2017. – № 41(1). – С. 67-78. – DOI: 10.18287/2412-6179-2017-41-1-67-78.

130. Хаустов П. А. Разработка системы оптического распознавания символов на основе совместного применения вероятностной нейронной сети и вейвлетт преобразования / П. А. Хаустов, Д. С. Григорьев, В. Г. Спицын // Известия Томского политехнического университета. – 2013. – Т. 323. – № 5. – С. 107-112.

131. Хаустов П. А. Генетический алгоритм поиска множества кривых для оптического распознавания символов с использованием метода пересечений [Электронный ресурс] / П. А. Хаустов, В. Г. Спицын, Е. И. Максимова // Современные проблемы науки и образования. – 2014. – № 6. – URL: http://www.science-education.ru/pdf/2014/6/541.pdf (дата обращения: 01.04.2017).

132. Хаустов П. А. Алгоритм сегментации рукописного текста на основе построения структурных моделей / П. А. Хаустов // Фундаментальные исследования. – 2017. – № 4(1). – С. 88-93.

133. Григорьев Д. С. Улучшение качества метода оптического распознавания текстов с помощью совместного применения вейвлетт-преобразований, курвлетт-преобразований и алгоритмов словарного поиска / Д. С. Григорьев, П. А. Хаустов, В. Г. Спицын // Известия Томского политехнического университета. – 2013. – Т. 323 – № 5. – С. 101-107.

134. Свидетельство о государственной регистрации программы для ЭВМ № 2016619504 «Оптическое распознавание рукописных символов на основе построения структурных моделей» / Хаустов П. А.; правообладатель федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский Томский политехнический университет» (RU). Заявка № 2016614414, заявл. 04.05.2016, дата государственной регистрации в Реестре программ для ЭВМ 22.08.2016.

135. Khaustov P. A. Structural model constructing for optical handwritten character recognition / P. A. Khaustov, V. G. Spitsyn , E. I. Maksimova // IOP Conference Series: Materials Science and Engineering, 2017, vol. 173, № 1. – DOI: 10.1088/1757-899X/173/1/012006.

136. Khaustov P. A. Algorithm for Optical Handwritten Characters Recognition Based on Structural Components Extraction / P. A. Khaustov, V. G. Spitsyn, E. I. Maksimova // IFOST-2016: Information and Communication Technologies: Proceedings. – 2016. – vol. 1. – P. 355-358.

137. Хаустов П. А. Использование растеризации методом Брезенхэма для метода пересечений при оптическом распознавании печатных символов [Электронный ресурс] / П. А. Хаустов, Е. И. Максимова // Вестник науки Сибири. – 2014. – № 4(14). – URL: http://sjs.tpu.ru/journal/article/view/1101 (дата обращения: 01.04.2017).

138. Хаустов П. А. Алгоритм построения топологической модели рукописного символа с целью анализа правильности его написания / П. А. Хаустов // Современные техника и технологии: сборник трудов XXI Международной научной конференции студентов, аспирантов и молодых ученых: в 2 т, Томск, 5-9 октября 2015. – 2015. – Т. 2. – С. 91-94.

139. Хаустов П. А. Алгоритм выделения структурных составляющих для решения задачи оптического распознания рукописных символов / П. А. Хаустов, Е. И. Максимова // Молодежь и современные информационные технологии: сборник трудов XIII Международной научно-практической конференции студентов, аспирантов и молодых ученых, Томск, 9-13 ноября 2015. – 2016. – Т. 1. – С. 132-133.

140. Хаустов П. А. Применение растеризации методом Брезенхэма в методе пересечений при оптическом распознавании печатных символов / П. А. Хаустов, Е. И. Максимова // Молодежь и современные информационные технологии: сборник трудов XII Всероссийской научно-практической конференции студентов, аспирантов и молодых ученых: в 2 т., Томск, 12-14 ноября 2014. – 2014. – Т. 2. – С. 269-270.

141. Хаустов П. А. Алгоритм словарного поиска для системы оптического распознавания текстов на основе динамического программирования [Электронный ресурс] / П. А. Хаустов, В. Г. Спицын // Молодежь и современные информационные технологии: сборник трудов XI Международной научно-практической конференции студентов, аспирантов и молодых ученых, Томск, 13-16 ноября 2013. – 2013. – С. 153-155.

142. Хаустов П. А. Решение задачи классификации образов с использованием вероятностных нейронных сетей / П. А. Хаустов //

Современные техника и технологии: сборник трудов XIX Международной научно-практической конференции студентов, аспирантов и молодых ученых: в 3 т., Томск, 15-19 апреля 2013. – 2013. – Т. 2. – С. 383-384.

143. Хаустов П. А. Разработка автоматической системы для проверки правильности написания символов / П. А. Хаустов // Информационные технологии в науке, управлении, социальной сфере и медицине: сборник научных трудов II Международной конференции, Томск, 19-22 мая 2015. – 2015. – С. 217-219.

144. Хаустов П. А. Алгоритм оптического распознавания рукописных символов на основе построения структурной модели / П. А. Хаустов, В. Г. Спицын // Информационные технологии в науке, управлении, социальной сфере и медицине: сборник научных трудов III Международной конференции, Томск, 23-26 мая 2016. – 2016. – С. 501-503.

ПРИЛОЖЕНИЕ А. АКТЫ ВНЕДРЕНИЯ РЕЗУЛЬТАТОВ РАБОТЫ

Rubius

(3822) 9-7777-2
www.rubius.com
info@rubius.com

Организация:	Общество с ограниченной ответственностью «Рубиус Групп»
Генеральный директор:	Дорофеева Мария Александровна
ИНН / КПП / ОГРН	7017252288 / 701701001 / 1097017021584
Юридический адрес:	634034, г. Томск, ул. Нахимова, 13/1
Банк получателя:	ПАО «Сбербанк России», г. Томск, БИК 046902606, к/с 30101810800000000606, р/с 40702810164000003190

« 20 » июня 2016 г.

АКТ ВНЕДРЕНИЯ

результатов диссертационной работы Хаустова Павла Александровича «Алгоритмы распознавания рукописных символов в условиях малой обучающей выборки»

Настоящим утверждается, что результаты диссертационной работы Хаустова П.А. «Алгоритмы распознавания рукописных символов в условиях малой обучающей выборки», выполненной на соискание учёной степени кандидата технических наук по специальности «05.13.11 – Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей», являются актуальными, обладают практической значимостью и были внедрены в ООО «Рубиус Групп» для реализации технологических задач в системе автоматической обработки бланков и анкет.

Результаты диссертационной работы включают в себя программную реализацию следующих алгоритмов:

1) Алгоритм предварительной обработки исходного представления символа, включающий этапы гистограммной эквализации, адаптивной бинаризации и однопроходной скелетизации.
2) Алгоритм построения и сохранения структурной модели символа по скелетизированному бинарному изображению его начертания.
3) Алгоритм распознавания рукописных символов с применением построенных структурных моделей символов.
4) Алгоритм графического представления ранее построенных структурных моделей символов.

Реализованные алгоритмы являются эффективными при решении соответствующих задач в области обработки изображений, классификации и распознавания образов.

Генеральный директор ООО «Рубиус Групп» / _____ / М.А. Дорофеева

Rubius Group

КОПИЯ ВЕРНА

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

«УТВЕРЖДАЮ»
Проректор по образовательной
деятельности ТПУ,
д.т.н., профессор
_____ Ю. С. Боровиков
«4» 07 2017 г.

АКТ ВНЕДРЕНИЯ

результатов диссертационной работы Хаустова П.А.
«Алгоритмы распознавания рукописных символов в условиях малой обучающей выборки»

Настоящим подтверждается, что результаты диссертационной работы Хаустова П. А. «Алгоритмы распознавания рукописных символов в условиях малой обучающей выборки» на соискание учёной степени кандидата технических наук по специальности 05.13.11 «Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей», используются в учебном процессе на кафедре информационных систем и технологий Института кибернетики Томского политехнического университета при проведении лабораторных занятий по дисциплине «Методы распознавания образов» для магистрантов, обучающихся по направлению 09.04.01 «Информатика и вычислительная техника» магистерской программы «Компьютерный анализ и интерпретация данных».

В рамках данной дисциплины осуществляется реализация разработанных Хаустовым П. А. алгоритмов построения структурных моделей символов для распознавания рукописных символов. Цикл лабораторных работ по этой дисциплине апробирован при обучении магистрантов гр. 8ВМ4А.

Директор ИК ТПУ,
к.т.н.
_____ С. А. Байдали
«04» 07 2017 г.

И.о. зав. кафедрой ИСТ,
к.т.н., доцент
_____ А. В. Погребной
«4» 07 2017 г.

КОПИЯ ВЕРНА

ПРИЛОЖЕНИЕ Б. СВИДЕТЕЛЬСТВО НА ПРОГРАММУ ДЛЯ ЭВМ

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2016619504

Оптическое распознавание рукописных символов на основе построения структурных моделей

Правообладатель: ***федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский Томский политехнический университет» (RU)***

Автор: ***Хаустов Павел Александрович (RU)***

Заявка № **2016614414**

Дата поступления **04 мая 2016 г.**

Дата государственной регистрации в Реестре программ для ЭВМ ***22 августа 2016 г.***

Руководитель Федеральной службы по интеллектуальной собственности

Г.П. Ивлиев

КОПИЯ ВЕРНА