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

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

Кафедра «Прикладная математика и информатика»
(наименование)

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

Мобильные и сетевые технологии
(направленность (профиль) / специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему «Реализация алгоритма распознавания Бумажного документа в Шаблон электронного документа (в формате Jasper или другой библиотеки для создания отчётов)»

Студент А.И. Захаров
(И.О. Фамилия) (личная подпись)

Руководитель канд. физ.-мат. наук, доц., О.В. Лелонд
(ученая степень, звание, И.О. Фамилия)

Консультант М.В. Дайнеко
(И.О. Фамилия)

Тольятти 2021

Аннотация

Темой выпускной квалификационной работы (ВКР) является «Реализация алгоритма распознавания Бумажного документа в Шаблон электронного документа (в формате Jasper или другой библиотеки для создания отчётов)».

ВКР состоит из введения, трех глав, заключения и списка литературы.

Во введении описывается актуальность данной работы.

Первая глава посвящена описанию понятия и процесса оптического распознавания символов, а также обзору методов и анализу реализаций алгоритмов распознавания.

Во второй главе описывается теоретическая схема работы алгоритма, выбор средств его программной реализации и сама программная реализация системы.

В третьей главе проводится тестирование разработанной системы при различных входных данных. Также приводятся таблицы и графики о работе разработанной системы.

В заключение вынесены выводы о проделанной работе.

Был сделан вывод, что для более точной работы алгоритма расстояние между символами должно быть таким, чтобы их контуры не соприкасались. Также на процесс распознавания в достаточной мере влияет фон документа (его цвет относительно цвета текста, различные помехи и так далее). Данный недочёт возможно в большей мере исправить, используя предварительную обработку и преобразование исходного изображения в монохромное (чёрно-белое). Для увеличения скорости распознавания символов алгоритм возможно усовершенствовать, используя параллельное распознавание.

Данная выпускная квалификационная работа содержит в себе пояснительную записку, состоящую из 42 страниц, 22 рисунков, 3 таблиц и списка литературы из 20 источников.

Abstract

The title of the graduation work is *Implementing a paper record recognition algorithm in an electronic document template.*

This graduation work is devoted to developing an application that can convert text written in an image into printed text with the ability to save it in a document format.

The research dwells on details of the existing software solutions for optical character recognition.

The aim of the research is to develop a programme with a high-precision algorithm for optical character recognition from document images obtained as a result of scanning and other "text" images on Windows to facilitate transforming a paper record into a text format.

The object of the research is an application of optical character recognition from images.

The subject of the research is the character recognition algorithms when working with images containing text.

The key issue of the graduation work is to check the effectiveness of the proposed text recognition algorithm.

The graduation work may be divided into several logically connected parts that describe various existing software tools for optical character recognition. The research also focuses on designing an application, as well as its implementation and testing.

The following implementation tools are selected: Visual Studio, C# programming language, Tesseract implementation technology and EmguCV library.

High accuracy of text recognition is attained by means of image preprocessing.

Overall, the results suggest that the developed algorithm is able to recognize a text in an image with sufficient accuracy.

Оглавление

Введение

В то время как компьютерное зрение – это та область искусственного интеллекта, о которой все любят говорить, типы аннотаций данного семейства, которые работают с текстом, обходят стороной. Однако на самом деле технологии распознавания текста приносят намного больше пользы для современного бизнеса. Это особенно верно для средних и крупных компаний, которые обычно имеют дело с огромными объемами данных. Следует только подумать обо всех отчетах и протоколах, презентациях, корпоративных письмах и юридических формах, счетах, квитанциях и так далее.

Средний документооборот современной компании действительно огромен. На самом деле, исследования показывают, что сотрудники компаний обычно тратят треть своего рабочего времени на поиск необходимой информации. Естественно, документ с текстом, доступным для поиска, лучше, чем фотография или скан документа.

Статистические данные, полученные Бобылевой М.П. [1], подтверждают, что бумажные документы обходятся бизнесу дороже, чем процесс создания цифровых копий и управления ими. Не говоря уже о том, что зависимость от бумажных документов является одной из самых больших уязвимостей информационной безопасности. Эти факты говорят сами за себя: современный бизнес сэкономил бы довольно приличную сумму, просто перейдя с аналоговых на цифровые рабочие процессы. Тем не менее, хотя статистика выступает против использования бумаги, исследования показывают, что она не исчезнет в ближайшем будущем. И именно здесь появляются алгоритмы оптического распознавания символов, чтобы спасти положение. Использование соответствующих программ распознавания значительно упрощает процесс ввода текстовых данных в компьютер, тем самым освобождая значительное время. Это особенно актуально, например, при переводе больших потоков архивных документов в электронный вид для их компактного хранения на современных носителях. Следующим этапом

развития приложений этого класса будет разработка и совершенствование алгоритмов распознавания рукописного текста, что особенно актуально для ввода в компьютер информации с заполненных таможенных деклараций, квитанций, бланков и других отчетных документов.

В любом случае, потребность в распознавании текстов и электронном хранении документов существует практически в каждом офисе, что также подтверждают исследования, производимые Kissell J. [11], потому и столь актуальна данная тема.

Объектом исследования являются приложения оптического распознавания символов с изображений.

Предметом исследования являются алгоритмы распознавания символов при работе с изображениями с текстом.

Цель исследования – разработка программы с алгоритмом высокой точности оптического распознавания символов с изображений документов, полученных в результате сканирования, и иных «текстовых» изображений на ОС Windows для облегчения перевода бумажного документа в текстовый формат.

Задачи исследования:

- Изучить и проанализировать существующие методы и программные решения по распознаванию печатных символов;
- Описать работу алгоритма распознавания;
- Сравнить существующие методы распознавания символов;
- Рассмотреть существующий программный инструментарий;
- Реализовать рабочий прототип приложения, способный распознавать текстовые данные с отсканированного документа и сохранять эти данные в шаблон электронного документа;
- Протестировать приложение с использованием различных изображений.

Глава 1 Анализ существующих технологий оптического распознавания символов

1.1 Понятие и процесс оптического распознавания символов

Оптическим распознаванием символов или же OCR (англ. Optical character recognition), согласно «Википедии» [2], называют преобразование изображений печатного или рукописного текста в текстовый формат. OCR применяется для оцифровки книг и документов, для автоматизации систем учёта или публикаций текста на сайтах, форумах и веб-страницах.

Хотя это обычно остается незамеченным, OCR является незаменимым помощником, когда говорят об автоматизации. Это исключает поток ненужных бумажных документов, позволяет классифицировать, организовывать, хранить, управлять и обмениваться информацией, избегая при этом рисков безопасности, связанных с физическим характером бумажных документов.

Сфера использования OCR становится еще больше. Его можно увидеть в сканерах для билетов в кино или в аэропортах и на вокзалах. Он используется для извлечения данных и наблюдения за безопасностью (учитывайте номерные знаки автомобилей или уличные знаки). Электронные подписи являются еще одной формой OCR. Но, возможно, наиболее распространенным использованием для OCR является превращение изображения бизнес-документов в цифровой текст, который можно искать, редактировать и управлять.

Кроме того, автоматизация на основе OCR — это не только обмен информацией в цифровой форме. Если у предприятия много документов, машины могут использовать их в качестве записей данных для поиска закономерностей. Визуализация также становится проще: если нужен график, схема или электронная таблица, гораздо быстрее использовать цифровые документы, чем составлять визуально приятный отчет вручную. OCR

позволяет тратить гораздо меньше времени на обработку каждого нового документа, экономить затраты на человеческие ресурсы.

Текстовый документ имеет ряд преимуществ над его визуальным представлением. Основными из них являются:

– Возможность редактирования и копирования;
– Облегчённый поиск слов и фраз;
– Хранение в более компактной форме;
– Печать материала без потери качества.

Хоть первые OCR-системы и нуждались в калибровке для работы с каждым определённым шрифтом, сейчас «интеллектуальные» системы способны с высокой точностью распознать большинство известных шрифтов.

OCR – довольно прогрессирующая и активно исследуемая проблема, также неразрывно связанная с компьютерным зрением, распознаванием речи и образов.

Один из самых простых и быстрых способов конвертировать документы в электронный формат – сканировать бумажные копии с помощью сканеров. По мере обработки документа сканером создается графическое изображение (графическое представление документа). Но графическое изображение – это еще не текстовый документ. Люди отлично распознают текстовые символы, даже когда они написаны от руки. Иными словами, чтобы понять, что написано на листе бумаги, достаточно взглянуть на него. Однако для машин это трудная задача. С точки зрения персонального компьютера отсканированный документ – это не текстовый файл, а всего лишь набор цветных точек. Машины требуют алгоритма машинного обучения, чтобы научиться читать, как люди читают, потому что так машины видят текст: как часть изображения. Для этого алгоритм OCR должен пройти через множество этапов подготовки, чтобы иметь возможность обрабатывать изображение текста.

Задача распознавания данных с изображения условно делится на две подзадачи:

– Предварительная обработка;

– Фактическое распознавание.

Предварительная обработка необходима для улучшения качества изображения (удаление шумов, повышение контрастности и т.д.) и выделения текстовой составляющей.

На следующем этапе при помощи программного инструментария выделяются символы (буквы, цифры, знаки препинания и т.д.). От качества изображения прямо зависит точность распознавания. Благодаря обработке изображения при фактическом распознавании текста значительно уменьшается количество ошибок вплоть до их отсутствия.

Обобщённая модель OCR-систем показана на рисунке 1.

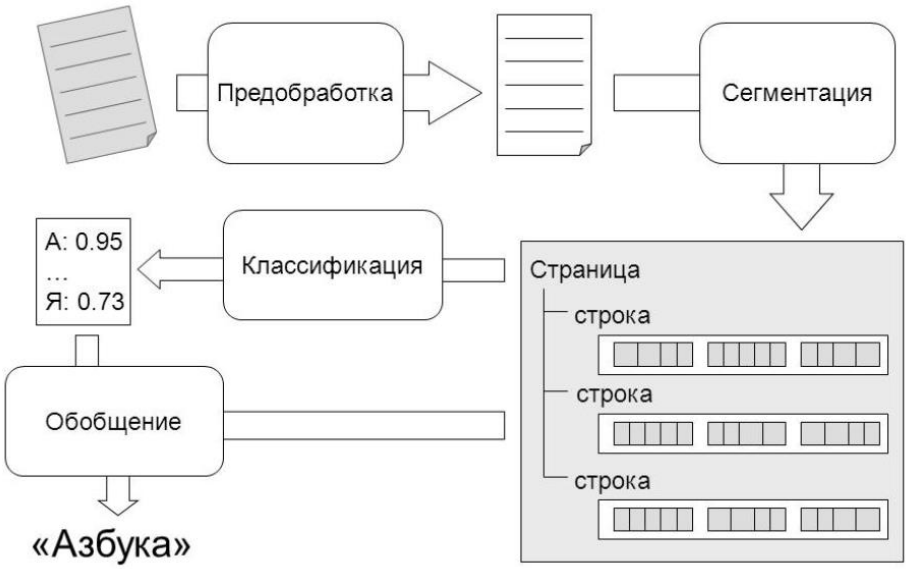

Рисунок 1 – Процесс обработки поступающего документа

Входными данными является отсканированный графический файл, отражающий страницу документа. Для приемлемой работы алгоритмов распознавания входное изображение было максимально возможного качества. Собственно, перед распознаванием изображения алгоритмы проводят преобразование входящего документа (фильтрация, преобразование

в оттенки серого). Модуль сегментации принимает данное готовое изображение, а также находит и определяет строки и слова – основные структурные единицы текста.

По итогам работы модуля сегментации создаётся структура данных, содержащая и отражающая положение текста на странице, – дерево сегментации. Верхний уровень – исходный объект, который содержит массив строк. В то же время массив строк содержит набор объектов слова. Слова – это конечные данные (листья дерева). Также все объекты хранят информацию о площади, покрытой им на изображении.

Несмотря на то что создание программного обеспечения для оцифровки документов задача не их лёгких, текстовый файл, в отличии от графического изображения, является более предпочтительным представлением информации. Использование текстовой информации позволяет упростить, ускорить и реализовать все возможные варианты дальнейшего использования и анализа электронных документов. Поэтому с точки зрения дальнейшего применения преобразование бумажных носителей информации в электронный текстовый документ представляет наибольший интерес.

1.2 Обзор существующих приложений по оптическому распознаванию символов

В настоящее время существует множество программных решений задач OCR. Самыми примечательными из них являются FineReader, CuneiForm, OCRopus и Tesseract, поэтому их и рассмотрим.

FineReader, согласно электронному ресурсу «PC, Mobile Utilites» [5], – это коммерческая система оптического распознавания символов (OCR), разработанная российской компанией ABBYY, работающая на базе технологии ABBYY OCR. Она используется для преобразования отсканированных документов, PDF-документов и графических документов

(включая цифровые фотографии) в редактируемые / доступные для поиска документы. ABBYY FineReader 12 может автоматически распознавать и обрабатывать документы с любой комбинацией 190 языков и обеспечивает полную поддержку словарей для 48 языков. Тонкости работы данной системы описывает в своей книге Huang D. [9]. Программное обеспечение доступно для Linux, MAC OS и Windows. ABBYY FineReader может анализировать уже существующие изображения или файлы PDF, а также документы, находящиеся в процессе сканирования в программу

CuneiForm, как описывает «PC, Mobile Utilites» [14], – OCR-система, разработанная российской компанией CognitiveTechnologies на базе библиотеки распознавания Cognitive, изначально позиционирующаяся как коммерческий продукт. Данный продукт также поставлялся в комплекте с некоторыми популярными моделями сканеров (Epson, Xerox, Canon и др.). Спустя несколько лет после релиза приложения произошло официальное открытие исходных текстов программы. Сейчас CuneiForm – бесплатная (но имеющая минимальные ограничения), шрифтонезависимая система оптического распознавания символов. Программа доступна на Linux, MAC OS и UNIX-подобных системах.

OCRopus, согласно «Википедии» [15], – бесплатная система OCR, работающая на базе технологии Tesseract и направленная на перевод в текстовый формат большого объёма документов. Программа использует код языка моделирования из проекта компании, который поддерживается Google – OpenFST. Она способна распознавать как печатные, так и рукописные материалы. Доступна на Linux и MAC OS.

Tesseract, согласно ресурсу «ВикиПрограммы» [19], – бесплатная система распознавания символов, имеющая одно имя со своим ядром и изначально разрабатывающаяся компанией Hewlett-Packard (HP). Спустя годы после приостановки проекта компания Google выкупила его и открыла исходные коды. Программа работает с UTF-8 и работает более чем с 100 языков, поддержка которых происходит при помощи настройки и установки

дополнительных модулей, и доступна на Windows, MAC OS, Linux и других UNIX-подобных системах.

К сожалению, ни одна из программ не показывает 100% результат при распознавании текста. После рассмотрения четырёх программных решений OCR была составлена таблица А.1, наглядно показывающая разницу между ними.

Поскольку такие программы как OCRopus и CuneiForm недоступны для операционной системы Windows и имеют некоторые ограничения, а ABBYY FineReader вовсе является коммерческим продуктом, было принято решение о рассмотрении системы Tesseract, которая принадлежит компании Google, что даёт уверенность в будущем проекта.

1.3 Механизм распознавания Tesseract

Tesseract – это механизм оптического распознавания символов с открытым исходным кодом, первоначально запущенный как проект компании HP. Позже он был изменен, улучшен и взят на вооружение Google, а ещё позже выпущен в качестве открытого источника в 2005 году. Он очень удобен по сравнению с другими и поддерживает различные платформы. Его фокус больше направлен на обеспечение меньшего отклонения и повышение точности. В настоящее время доступна только версия командной базы, но есть много проектов, в которых поверх нее встроен интерфейс, который может быть разветвлен. На данный момент выпущена и доступна для использования версия Tesseract 3.02. Сейчас Tesseract обеспечивает поддержку около 139 языков.

Tesseract – это система, основанная на примерах. Это делает его эффективным и гибким. Под системами, основанными на примерах, подразумевается, что механизм работает по набору правил, определенных в системе, и результаты зависят от этих данных. Причина гибкости Tesseract заключается в том, что всегда можно изменить или модифицировать правила

в зависимости от требований.

Tesseract OCR работает поэтапно. Первым шагом в цикле является определение интенсивности цвета изображения, называемое адаптивной пороговой обработкой, и преобразование изображения в двоичные изображения. Второй шаг – выполнить анализ связанных компонентов изображения, который выполняет задачу выделения контуров. Этот шаг является основным процессом этого цикла, как и OCR изображения с белым текстом и черным цветом остальной части изображения. Этапы работы алгоритма показаны на рисунке 2.

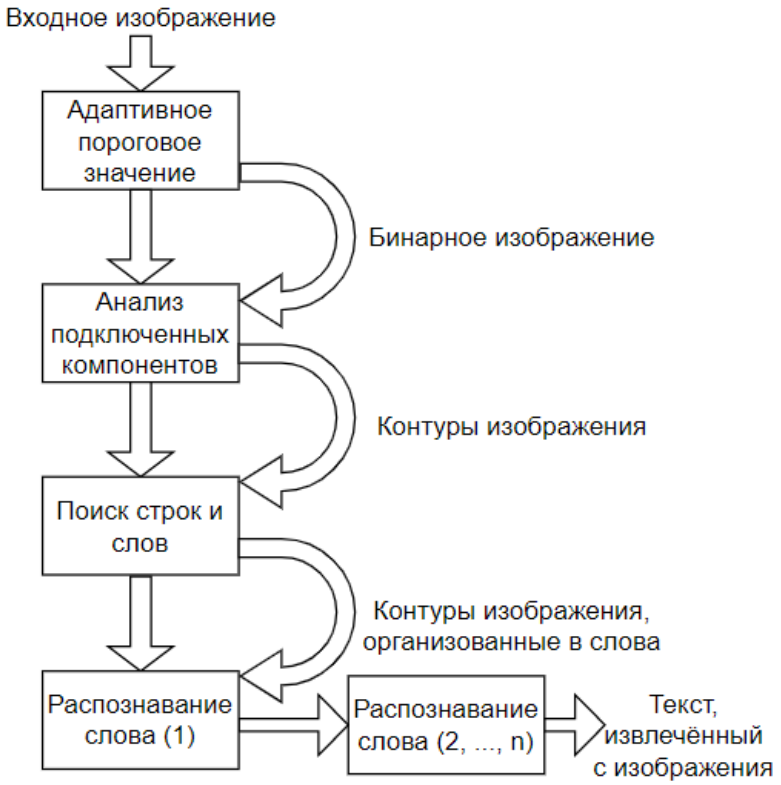

Рисунок 2 – Алгоритм Tesseract

Tesseract был, вероятно, первым, кто использовал эти циклы для обработки входного изображения. После этого наброски, извлеченные из изображения, конвертируются в Blob (двоичные длинные объекты). Затем они организуются в виде линий и областей, и производится дальнейший

анализ проводится для некоторой фиксированной области. После извлечения компоненты разбиваются на слова и разделяются пробелами. Затем начинается распознавание текста, что является двухпроходным процессом. Первая часть – это попытка распознать каждое слово. Каждое удовлетворительное слово принимается, и начинается второй проход, чтобы собрать оставшиеся слова. Это приводит к роли адаптивного классификатора. После этого адаптивный классификатор классифицирует текст более точно. Чтобы адаптивный классификатор работал правильно, его необходимо заранее обучить. Когда классификатор получает какие-то данные, он должен решить проблемы и назначить правильное место в тексте.

Tesseract принимает только два аргумента: первый – это входное изображение, которое содержит текст, а второй аргумент – это выходной текстовый файл. Tesseract по умолчанию выбирает расширение выходного файла как .txt. Также он поддерживает различные языки. Каждый язык поставляется с файлом данных для обученного языка. Языковой файл должен храниться в месте, известном Tesseract. При использовании в проекте рекомендуется хранить его в папке проекта.

Эксперименты показывают, что Tesseract способен достигать высокой точности, такой как 95%, но в случае некоторых сложных изображений с многослойным фоном или необычным текстом могут возникнуть проблемы Tesseract обеспечивает лучшую точность результатов, если изображения находятся чёрно-белом режиме, а не в цвете.

Выводы и результаты по главе 1:

- В данной главе были рассмотрены основные понятия в сфере оптического распознавания символов, существующие программные средства для OCR, а также общая модель OCR-систем;
- В ходе рассмотрения предметной области, была выявлена потребность во внедрении OCR-систем на предприятия для улучшения работы с большим документооборотом, присущем большим компаниям;

– Также было сконцентрировано внимание на ядре Tesseract OCR и его особенностях, описана работа алгоритма распознавания Tesseract.

Глава 2 Проектирование алгоритма оптического распознавания символов и разработка приложения

2.1 Выбор средств реализации

Для начала работы над программой распознавания необходимо выбрать среду разработки, а также язык программирования и технологию на котором будет реализовываться OCR-алгоритм.

Начнём с технологии. Tesseract в отличие от своих аналогов имеет ряд преимуществ, с чем и связан его выбор в качестве OCR-модуля в рамках данной работы:

- Обладает открытым исходным кодом;
- Показывает прекрасные результаты при работе с чёрно-белым текстом;
- Позволяет в короткие сроки на своей базе реализовать модуль способный распознавать текст;
- Обладает обширной документацией.

Tesseract использует достаточно новый алгоритм для поиска строк текста на странице. Алгоритм хорошо работает даже при наличии сломанных и соединенных символов, шума и перекоса страниц.

Этот алгоритм работает следующим образом:

- Выполняется анализ подключенных компонентов;
- Медианная высота используется для определения размера текста, а компоненты, которые меньше некоторой доли высоты текста, в основном считаются пунктуацией, диакритическими знаками или шумом.
- Компоненты сортируются (в порядке возрастания) с использованием координаты x (левого края) в качестве ключа сортировки. Такая сортировка позволяет отслеживать перекос по всей странице.
- После того, как текстовые строки были найдены, базовые линии

подгоняются более точно, используя квадратичный сплайн по методу наименьших квадратов.

Tesseract обнаруживает слова путем измерения зазоров в ограниченном вертикальном диапазоне между базовой линией и средней линией. Пространства, близкие к пороговому значению на этом этапе, становятся нечеткими, так что окончательное решение может быть принято после распознавания слов.

Также Tesseract способен к тесной интеграции с библиотеками компьютерного зрения.

Сегодня существует множество приложений в области компьютерного зрения: Computer Vision System Toolbox для MatLab [7], IMAQ Vision [10], LabView [12] и другие. После изучения мануалов данных приложений было принято решение остановиться на наиболее популярной библиотеке – OpenCV и её документации [16].

OpenCV (Open Source Computer Vision) – это библиотека компьютерного зрения с открытым исходным кодом функций и алгоритмов программирования. Их основная цель – повышение вычислительной эффективности, в основном предназначенное для компьютерного зрения в реальном времени. Основным преимуществом OpenCV является простота интерфейса, что позволяет быстро осваивать технологии компьютерного зрения и создавать на их основе сложные приложения.

С момента своего выпуска в январе 1999 года OpenCV использовался во многих приложениях и исследовательских проектах, таких как:

- Снижение шума на медицинских изображениях;
- Системы управления, калибровка камеры;
- Пошив снимков спутниковой карты;
- Выравнивание отсканированных изображений;
- Автоматический мониторинг;
- Анализ объектов, системы обнаружения вторжений;

Структура данной библиотеки компьютерного зрения показана на рисунке 3.

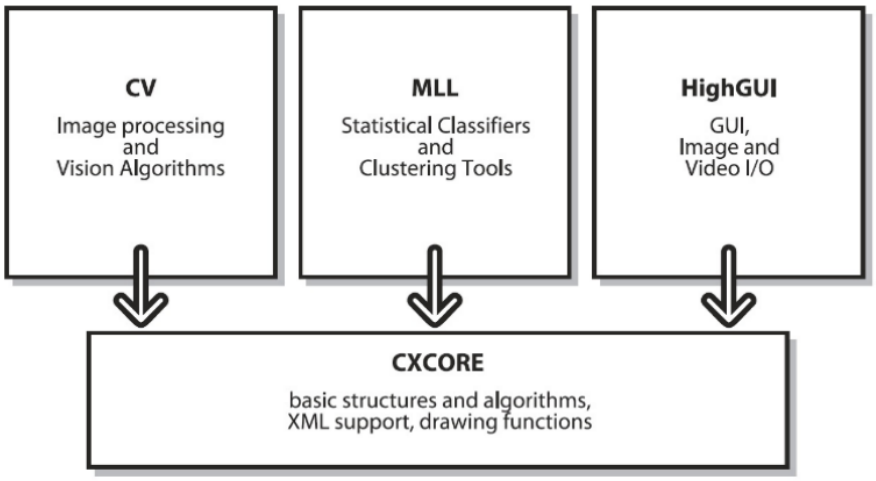

Рисунок 3 – Базовая структура OpenCV

Для OpenCV существует кроссплатформенная обёртка, называемая EmguCV, сохранившая большую часть функций OpenCV для совместимости с языками .NET.

EmguCV, согласно документации [8], имеет свои преимущества, такие как:

- Работа на любой платформе .Net;
- Поддержка как Windows, Linux, Mac OS, так и iOS и Android;
- Чистая реализация C#, обеспечивающая кроссплатформенность библиотеки;
- Возможность перекрёстного языка – может быть использован с нескольких различных языков (в том числе C#, VB.NET, C++ и IronPython);
- Бесплатна для использования в целях разработки.

EmguCV без излишних действий и неудобств во внедрении может использоваться только на Visual studio и Unity. Выбор выпадает на Visual studio, работу с которой в своей книге «Visual Studio 2015 Cookbook» описал

Martin J. [13], с возможностью создания шаблона Windows Forms (окна приложения), с которым в дальнейшем и будет работать пользователь.

Также проще и удобнее всего реализовать приложение OCR будет возможно с родного языка EmguCV – C#, который как раз рассчитан на подобные проекты.

Причина, по которой для разработки будет использоваться C#, заключается в его мощности и возможном расширении для решения большого количества задач, связанных с распознаванием, в будущем.

Таким образом, в качестве средств реализации были выбраны:

– Среда разработки – Visual Studio;
– Язык программирования – C#;
– Технология реализации – Tesseract;
– Библиотека – EmguCV.

Первым шагом к разработке приложения является изучение использования нового языка программирования C# и поиск инструментов, необходимых для разработки этого приложения. Благодаря книге «Microsoft Visual C#. Step by Step» от Sharp J. [18] – автора языка C# достаточно быстрое изучение данного языка становится более чем возможным.

2.2 Реализация программы

Для реализации программы была использована среда разработки – Microsoft Visual Studio 2015. Также в качестве шаблона было выбрано приложение Windows Forms для Visual C#.

Название приложения: «Text-Recognizer».

Tesseract OCR в своих открытых источниках на GitHub [20] имеет уже обученные для распознавания символов того или иного языка модели данных (это является очередным его достоинством). Для прототипа приложения достаточно загрузить лишь два языка – русский и английский. Поместить их

следует в папку проекта для возможности дальнейшего использования приложения на других устройствах.

Sells C. описал в своей книге «Windows Forms Programming in C#» [17] основные тонкости работы языка C# на шаблоне приложения Windows Forms. Учитывая их, можно начать работу над созданием оконного приложения.

Первое что будет находиться на Windows форме это menuStrip, содержащий всего одну вкладку – «Файл» с кнопкой «Выбрать». Её функционал заключается в выборе объекта формата .jpeg, .jpg или .png (т.е. выбор входящего изображения) с помощью проводника. Данные форматы являются стандартными и часто используемыми, согласно форуму «FOTODIZART. Дизайн и разработка сайтов» [4], в котором описываются известные электронные форматы изображений.

Следующий элемент – tableLayoutPanel. Он представляет собой два столбца, развёрнутые на всё приложение (на весь родительский контейнер). В левой ячейке находится pictureBox, выводящий графическое представление выбранного изображения. Правая ячейка – richTextBox. Она будет вбирать в себя распознанную с изображения текстовую информацию.

Весь функционал программы находится на toolStrip, расположенном прямо под menuStrip. Основными функциями являются:

– Выбор языка распознавания;
– Настройка контраста преобразования;
– Функция бинаризации изображения;
– Функция распознавания текста;
– Отмена преобразования.

Выбор языка распознавания происходит в элементе comboBox и имеет два изначально заготовленных варианта – русский и английский, которые ссылаются на соответствующе обученные модели данных, находящиеся в директории проекта Visual Studio.

Настройка контраста преобразования – элемент типа numericUpDown, имеющий некоторое базовое значение. Это значение сравнивается со средним значением всех оттенков изображения (красного, зелёного и синего) и обозначается в программе как «Глубина чёрного». Параметром может являться число в диапазоне от 0 до 255. Для чёрного текста значение должно быть приближённым к минимальному (например, 20), для белого текста – к максимальному (например, 240). Также в случае с документами, имеющими шум на фоне (например, при фотографировании), бежевый фон (от освещения) рекомендуется использовать параметр примерно равный середине всего диапазона (в районе от 100 до 200).

Функция распознавания текста – элемент типа Button, обозначающийся в программе как «Распознать текст». Данная функция – основа основ программы. С помощью подключенных библиотек EmguCV (версии 4.1.1.3497) и Tesseract данная функция непосредственно выполняет распознавание символов с изображения, отображающегося в данный момент времени на pictureBox, используя ссылки на обученные модели данных. Листинг данной функции показан на рисунке 4.

```
private void toolStripButton1_Click(object sender, EventArgs e) // Функция распознавания
{
    try // Обработка ошибок
    {
        if (String.IsNullOrEmpty(filePath) || String.IsNullOrWhiteSpace(filePath)) // Если изображение не выбрано
        {
            throw new Exception("Изображение не выбрано!"); // Сообщение об ошибке
        }
        else if (toolStripComboBox1.SelectedItem == null) // Если язык не выбран
        {
            throw new Exception("Язык не выбран!"); // Сообщение об ошибке
        }
        else // Если что нужно - выбрано
        {
            // Установка пути к языковому файлу
            string pathToFile = System.IO.Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "lang_data");
            // Создание тессеракта
            Tesseract tesseract = new Tesseract(pathToFile, lang, OcrEngineMode.TesseractLstmCombined);
            // Установка пути к изображению
            tesseract.SetImage(new Image<Bgr, byte>(filePath));
            // Распознавание букв на изображении
            tesseract.Recognize();
            // Вывод распознанного текста на экран
            richTextBox1.Text = tesseract.GetUTF8Text();
            tesseract.Dispose();
        }
    }
    catch (Exception ex) // Если возникла ошибка
    {
        MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Exclamation); // Окно с ошибкой
    }
}
```

Рисунок 4 – Листинг кода функции распознавания текста программы Text-Recognizer

Функция бинаризации изображения – элемент типа Button, обозначающийся в программе как «Преобразовать». Принцип работы функции преобразования был взят с интернет ресурса «Блог доброго программиста» [6]. Эта функция сравнивает среднее значение RGB каждого пикселя изображения со значением «Глубина чёрного» и, в зависимости от результата (меньше или больше), присваивает пикселям новый цвет (чёрный либо белый). По итогу изображение, находящиеся в pictureBox сменяется своим монохромным (чёрно-белым) аналогом.

«Отмена преобразования» – элемент типа Button и последний из основного функционала. Эта функция необходима для замены полученного при преобразовании монохромного изображения на изначальное. Всё для того чтобы получить возможность изменения значения поля «Глубина чёрного» и нового преобразования.

Также помимо основных функций на toolStrip находятся не менее важные дополнительные функции, которые являются элементами типа Button. В основном они предназначены для экономии времени при работе с распознанным текстом. Данными функциями являются:

- «Копировать» – копирует распознанный текст, находящийся в данный момент в richTextBox в буфер обмена. Позволяет не тратить время на выделение всего текста для копирования;
- «Сохранить» – сохраняет распознанный текст в текстовый шаблон rtf-формата. Данная функция экономит время на сохранение распознанного текста, если необходимо распознать текст с нескольких изображений;
- «Открыть» – открывает папку, в которую сохраняются текстовые документы с распознанным текстом при использовании функции «Сохранить»;

– «Справка» – открывает руководство по использованию программы txt-формата. Руководство представляет из себя текстовый документ, в котором описываются все возможности данного приложения. Руководство представлено на рисунке 5.

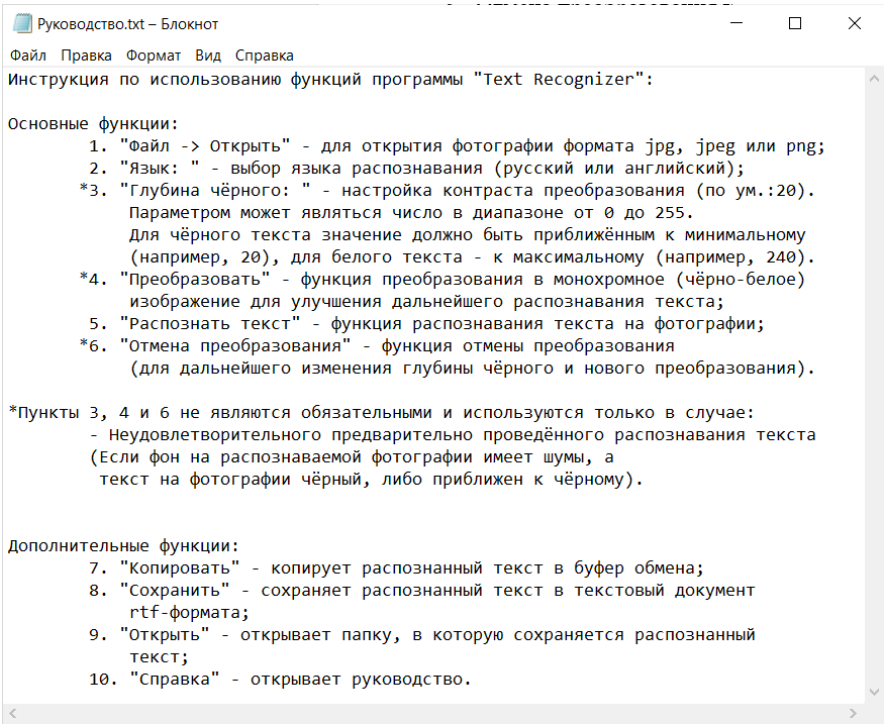

Рисунок 5 – Руководство по использованию программы Text-Recognizer

Руководство помогает новым пользователям сориентироваться в функциях и интерфейсе программы, показанном на рисунке 6.

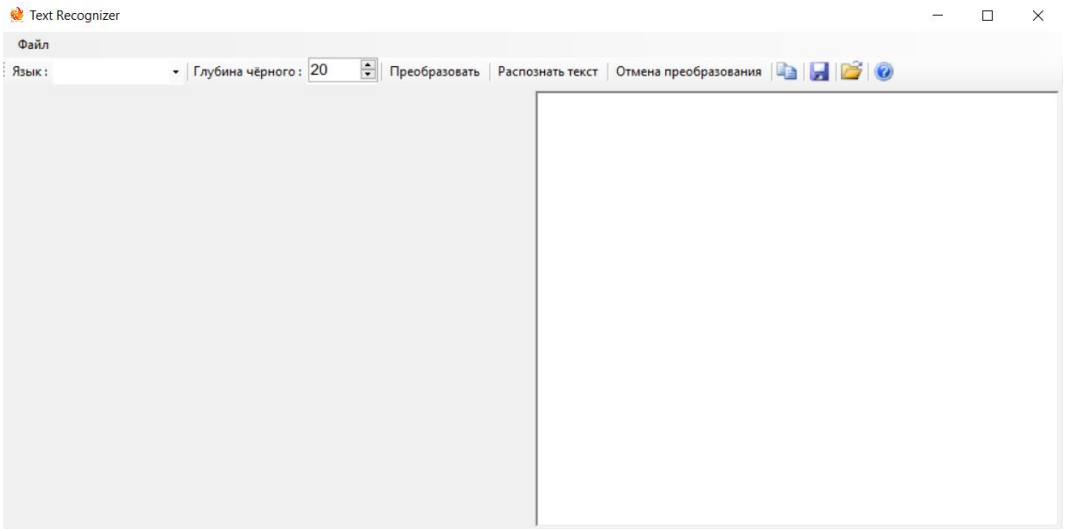

Рисунок 6 – Интерфейс программы Text-Recognizer

Все функции разработанного приложения образуют схему работы, показанную на рисунке 7.

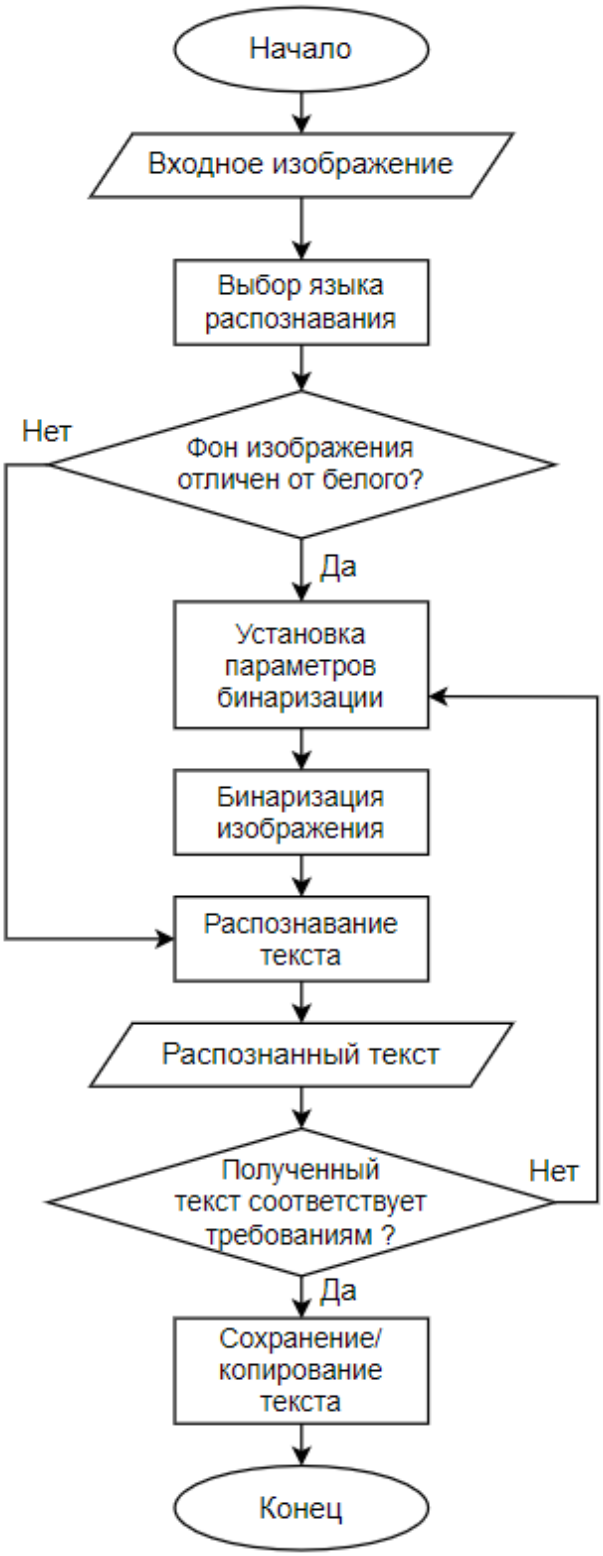

Рисунок 7 – Схема работы приложения Text-Recognizer

Ориентируясь на схему можно с лёгкостью понять основной функционал приложения для работы с текстовым изображением.

Выводы и результаты по главе 2:

– В данной главе был рассмотрен существующий программный инструментарий для выбора среды разработки и языка программирования;
– Также был реализован рабочий прототип приложения, способный распознавать текстовые данные с отсканированного документа и сохранять эти данные в шаблон электронного документа;
– Была создана и встроена в само приложение инструкция по работе с данным приложением. Функционал приложения был описан, а также представлен в виде схемы.

Глава 3 Тестирование разработанного алгоритма

3.1 Тестирование разработанного алгоритма с использованием различных изображений

В ходе выполнения данной работы была спроектирована и разработана OCR-система. В данной главе рассмотрим функции приложения и проведём тестирование данной программы.

Помимо непосредственного распознавания текста в системе распознавания присутствуют следующие функции:

- Выбор языка распознавания (русский, английский);
- Установка параметра бинаризации;
- Преобразование изображения в монохромное;
- Вывод на экран распознанного текста;
- Копирование распознанного текста в буфер обмена;
- Сохранение распознанного текста в шаблон документа rtf-формата;
- Открытие папки вывода шаблонов;
- Открытие руководства по использованию приложения.

Необходимыми действиями, которые должен совершить пользователь, являются:

- Загрузка фотографии в систему;
- Запуск распознавания.

В случае необходимости пользователь может настроить параметр бинаризации и использовать преобразование. Например, если фон изображения не позволяет с достаточной точностью распознать текстовую составляющую исходного изображения.

Остальные действия выполнит система и предоставит пользователю результат работы.

Чтобы проверить работоспособность программы был выбран пример чёрного русского текста на фоне шаблона некоторой презентации, взятой из

открытых источников. Тестируемое изображение показано на рисунке 8.

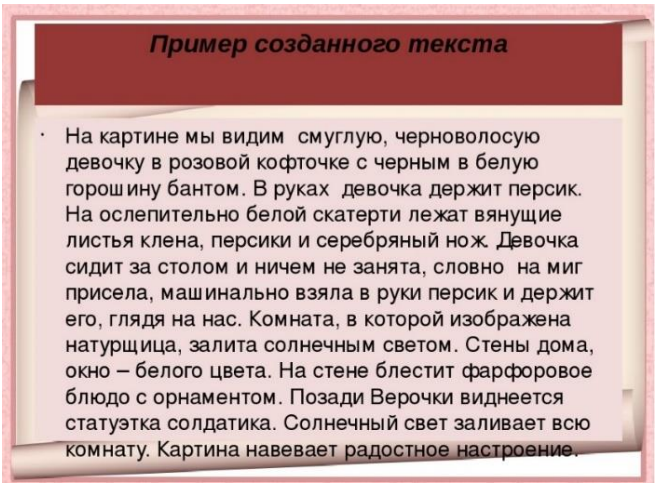

Рисунок 8 – Цветное изображение для тестирования работы алгоритма с чёрным текстом

Для начала используем простое распознавание текста, не прибегая к преобразованию изображения в монохром. Таким образом, получим результат работы программы, представленный на рисунке 9, основанной на технологии Tesseract без предварительной обработки изображения.

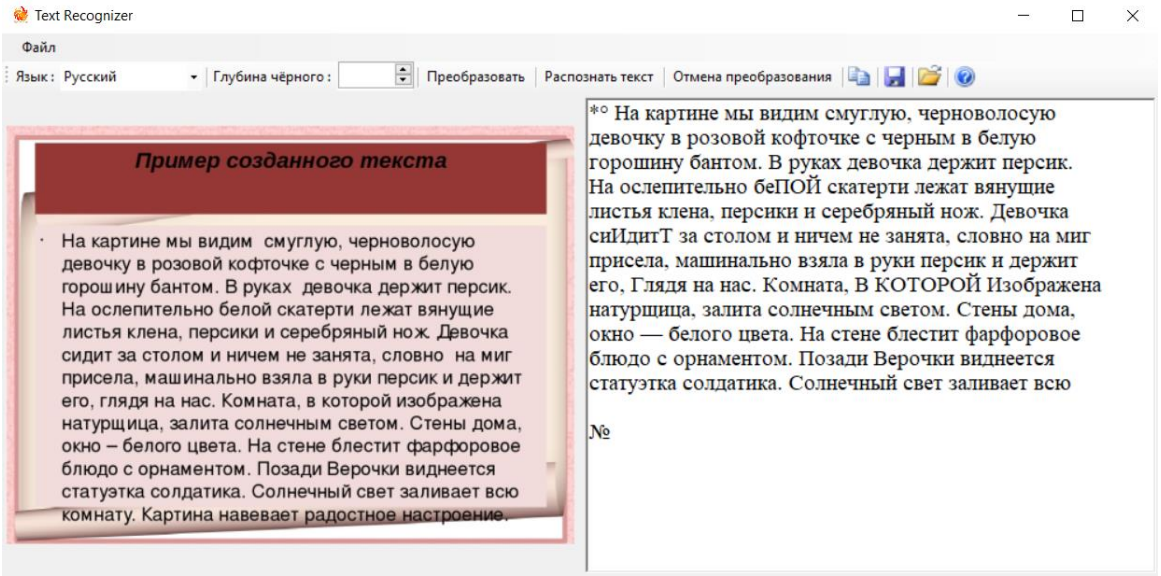

Рисунок 9 – Результат работы алгоритма без обработки изображения с чёрным текстом

После распознавания текста в richTextBox стало видно невооружённым глазом, что текст вывелся в недостаточной точности и с множеством ошибок. Заголовок «Пример созданного текста» не распознался алгоритмом, так как находился на достаточно тёмном фоне. Та же ситуация из-за фона презентации постигла и последнюю строчку данного текста.

Для исправления ситуации и увеличения точности распознавания алгоритмом Tesseract преобразуем изображение в монохромное (чёрно-белое) с помощью встроенных в приложение функций. Результат преобразования показан на рисунке 10.

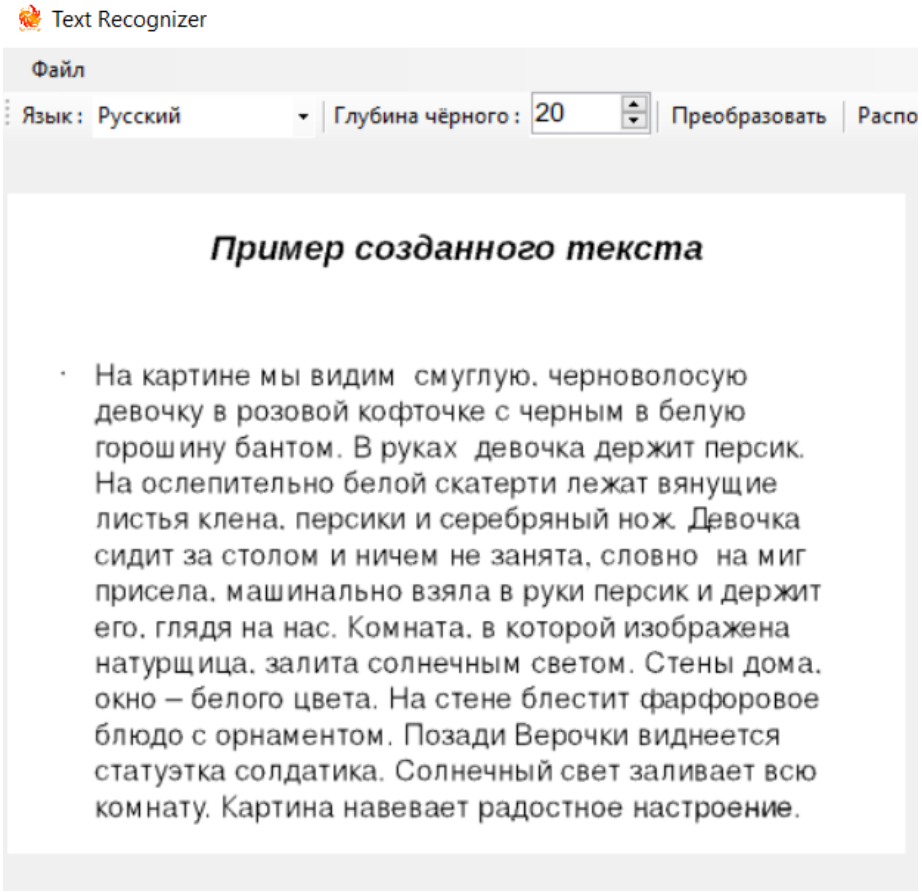

Рисунок 10 – Преобразованное изображение для тестирования работы алгоритма с чёрным текстом

В данном случае изображение имеет чёрные буквы на белом фоне без каких-либо помех. В качестве параметра бинаризации изображения было

выбрано число 20. Фон презентации полностью исчез, так как был значительно светлее чёрных букв.

Буквы на изображении легко различимы даже для приложения, и оно без труда способно распознать информацию с изображения, как показано на рисунке 11.

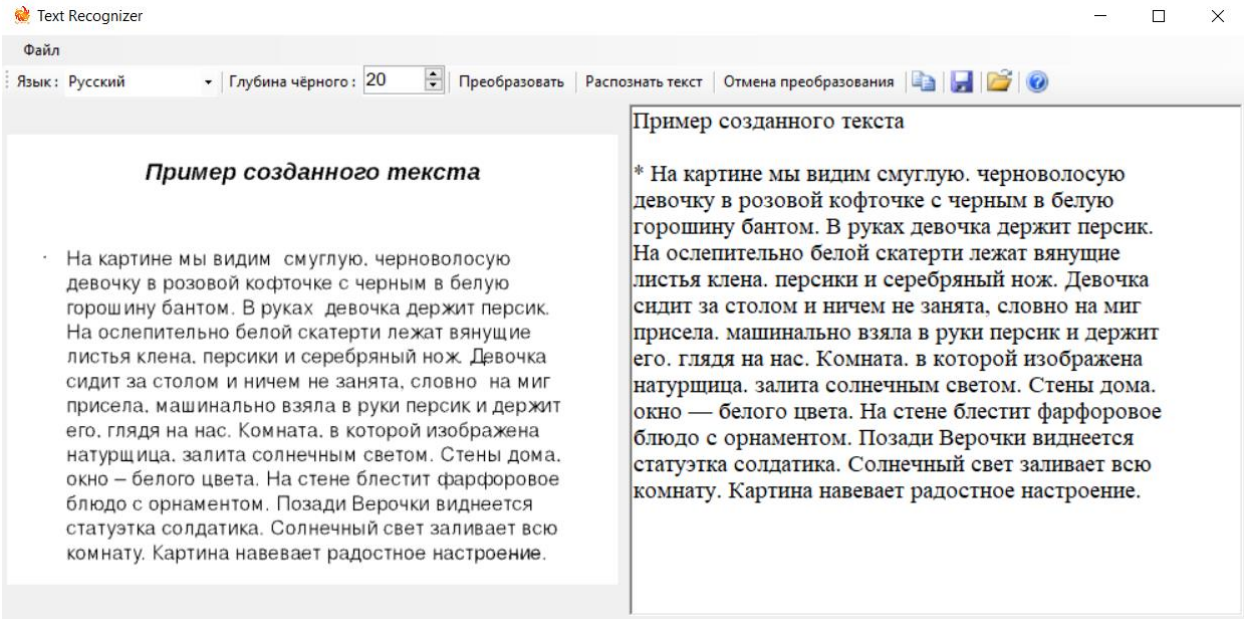

Рисунок 11 – Результат работы алгоритма после обработки изображения с чёрным текстом

В данном случае библиотека EmguCV совместно с технологией Tesseract смогли полностью и безошибочно оцифровать всю информацию с изображения. Был распознан в том числе и заголовок, и последняя строчка текста.

Таким образом, можно сделать вывод, что бинаризация изображения выполняет свою прямую функцию и действительно способствует улучшению качества распознавания текста

Далее рассмотрим работоспособность программы на распознавание текста на тёмном фоне с белым текстом. Для этого возьмём исходное текстовое изображение в инверсированных цветах. Данное изображение

представлено на рисунке 12.

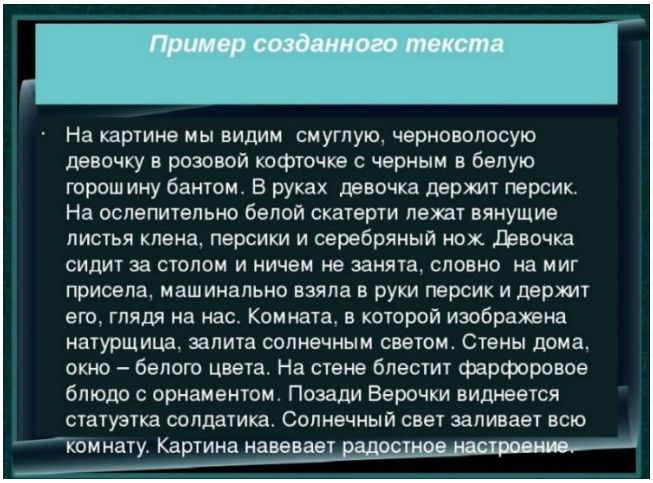

Рисунок 12 – Цветное изображение для тестирования работы алгоритма с белым текстом

Так же в начале используем простое распознавание текста без преобразования изображения в монохром. Данный пример даст информацию о возможности распознавания символов не только чёрного цвета. Результат работы алгоритма представлен на рисунке 13.

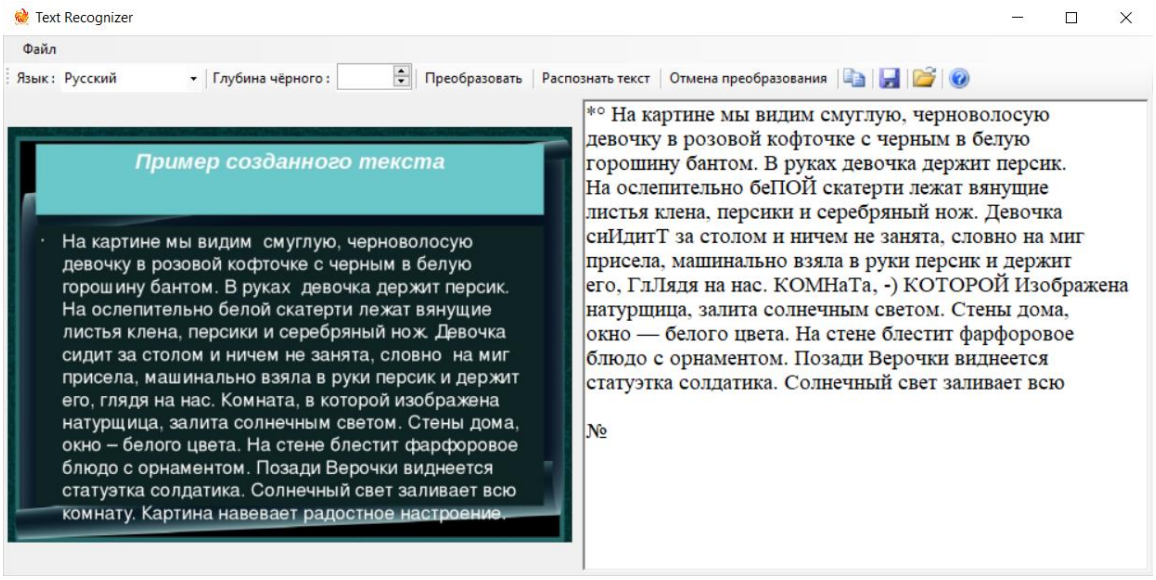

Рисунок 13 – Результат работы алгоритма без обработки изображения с белым текстом

Таким образом, стало известно, что Tesseract и EmguCV дают возможность для OCR белых символов. Функция сработала что является значимым плюсом для программы распознавания несмотря на то, что RichTextBox вывел распознанный текст с некоторыми ошибками и такими же проблемами, что и с чёрным текстом.

Теперь проверим как себя поведёт алгоритм распознавания после преобразования исходного изображения с белым текстом в чёрно-белое. Учитывая то, что в этом случае для правильного преобразования изображения необходимо задать параметр, приближённый максимальному значению, зададим большое значение глубины чёрного, а именно 240 как на рисунке 14.

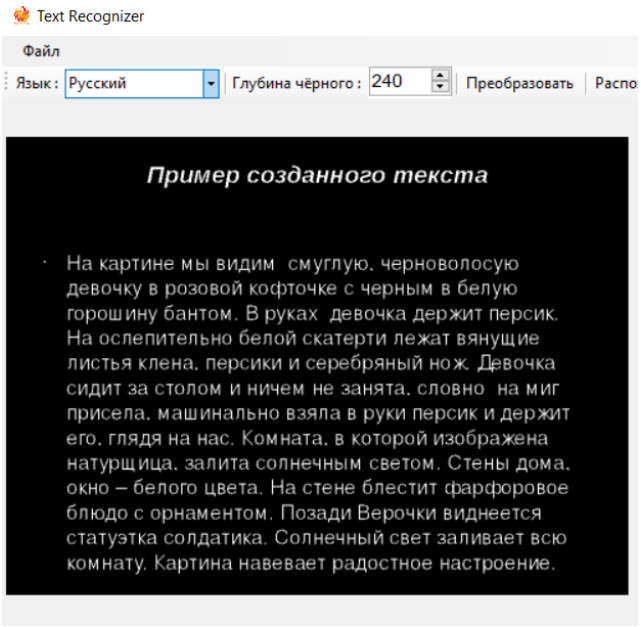

Рисунок 14 – Преобразованное изображение для тестирования работы алгоритма с белым текстом

Благодаря функции преобразования с высоким параметром глубины чёрного удалось выделить даже белый текст на тёмном изображении. В данном случае изображение имеет белые буквы на чёрном фоне без каких-либо помех. Фон презентации полностью исчез, так как был значительно

темнее белых букв. Все буквы на изображении легко различимы. Результат работы алгоритма представлен на рисунке 15.

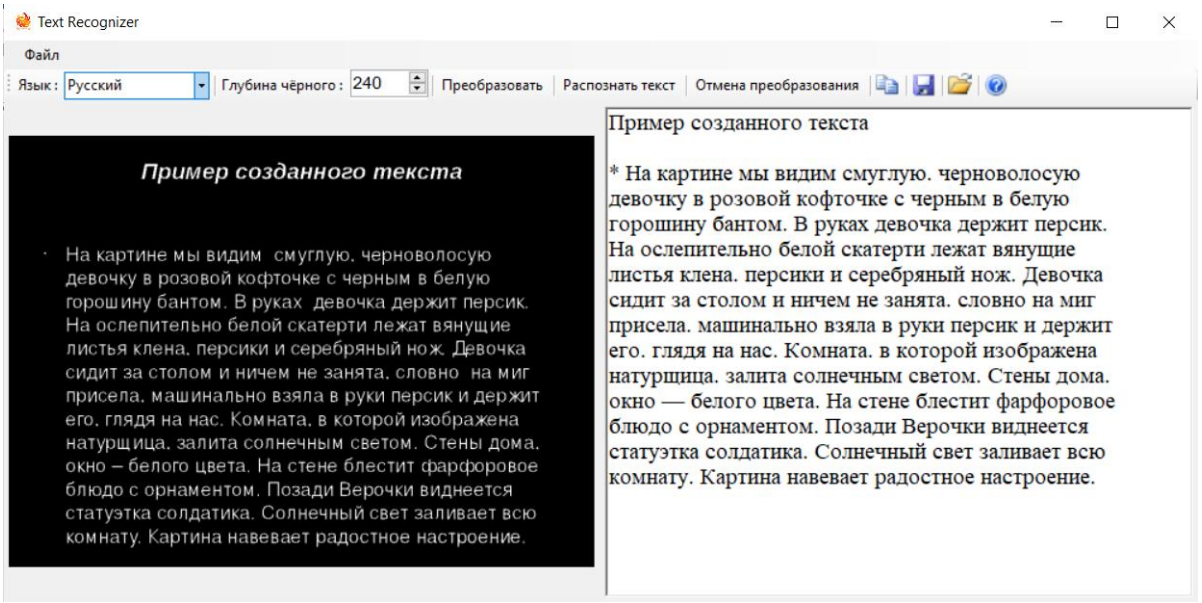

Рисунок 15 – Результат работы алгоритма после обработки изображения с белым текстом

Информация была безошибочно распознана с изображения. Алгоритм OCR показал свою возможность распознавать символы различного цвета.

Однако при всех плюсах Tesseract не может распознавать табличные данные вместе с таблицами, результат показан на рисунке 16.

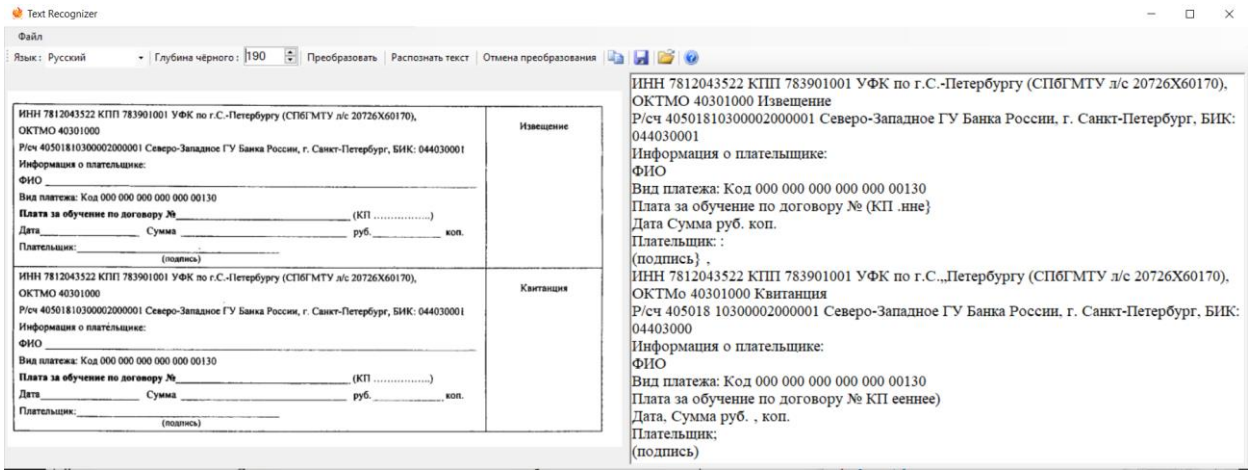

Рисунок 16 – Тестирование алгоритма распознавания над изображением с табличными данными

Однако при данном минусе данные, хранящиеся в ячейках таблиц алгоритм распознаёт с достаточно высокой точностью благодаря функции преобразования исходного изображения.

При использовании функции распознавания с отсутствующим изображением программа вызывает исключение с соответствующим сообщением об ошибке, как на рисунке 17.

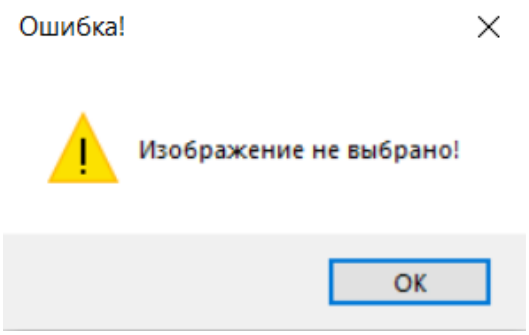

Рисунок 17 – Сообщение об ошибке выбора изображения

Похожее сообщение выводится при отсутствии выбранного языка для распознавания, окно ошибки показано на рисунке 18.

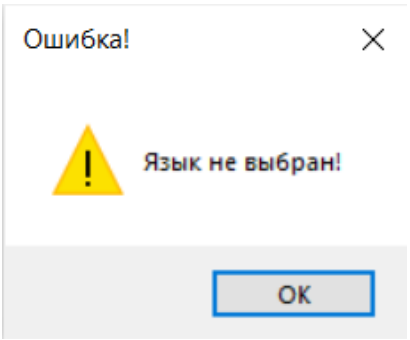

Рисунок 18 – Сообщение об ошибке выбора языка

Из проведенных выше тестов можно сказать, что система работает

хорошо при правильном использовании встроенной функции преобразования изображения в монохромное (если фон изображения мешает алгоритму). В таком случае система с высокой точностью распознаёт необходимый текст и выводит его на экран.

3.2 Подведение итогов тестирования разработанного алгоритма

Для подведения итогов тестирования и объективной оценки разработанного алгоритма проведём сравнительный анализ распознавания.

Основными параметрами тестирования являлись следующие:

– Количество символов в тексте: 545 (без пробелов);
– Количество слов в тексте: 93;
– Текст на изображении со светлым фоном;
– Преобразованное изображение со светлым фоном;
– Текст на изображении с тёмным фоном;
– Преобразованное изображение с тёмным фоном;
– Размер шрифта одинаковый.

Было проведено несколько тестов для каждого изображения. Итоги тестирования программы распознавания символов отражены в таблице 1.

Таблица 1 – Результаты тестирования алгоритма над текстовыми изображениями

Тип изображения	Всего символов	Всего слов	Правильно распознанных		Неверно/не распознанных		Точность OCR
			Символов	Слов	Символов	Слов	
Со светлым фоном	545	93	477	83	68	10	87,5%
С тёмным фоном	545	93	475	81	70	12	87,1%
Монохромное со светлым фоном	545	93	545	93	0	0	100%

Монохромное с тёмным фоном	545	93	545	93	0	0	100%

Результаты тестирования исходя из таблицы показали высокую точность распознавания при работе алгоритма. При этом результаты были стабильными при распознавании одного и того же изображения с одним и тем же параметром бинаризации. Таким образом, алгоритм показал свою стабильную работу, тем самым показав надёжность использования обученных данных из открытого кода Tesseract.

В соответствии с записью на интернет форуме «Студенческая библиотека онлайн» [3], особенностью текста при оптическом распознавании также является размер символа. Для проверки возможностей алгоритма было проведено тестирование распознавания текста разного размера шрифта на белом фоне без помех. Пример одной из восьми разнообразных по величине изображений для данного тестирования показан на рисунке 19.

Рисунок 19 – Слово на белом фоне для проведения тестирования алгоритма на возможность распознавания текста разного размера

Результаты тестов алгоритма с переменной величиной символов на изображении отражены в таблице 2.

Таблица 2 – Результаты тестирования алгоритма над разной величиной шрифта на белом фоне

Размер шрифта	72	48	36	28	24	18	14	10

Точность распознавания	100%	100%	100%	100%	100%	100%	100%	100%

Тестирование показало способность алгоритма к распознаванию как маленьких, так и больших объектов текста на изображении. Все символы были верно распознаны. По полученным данным был построен график, представленный на рисунке 20.

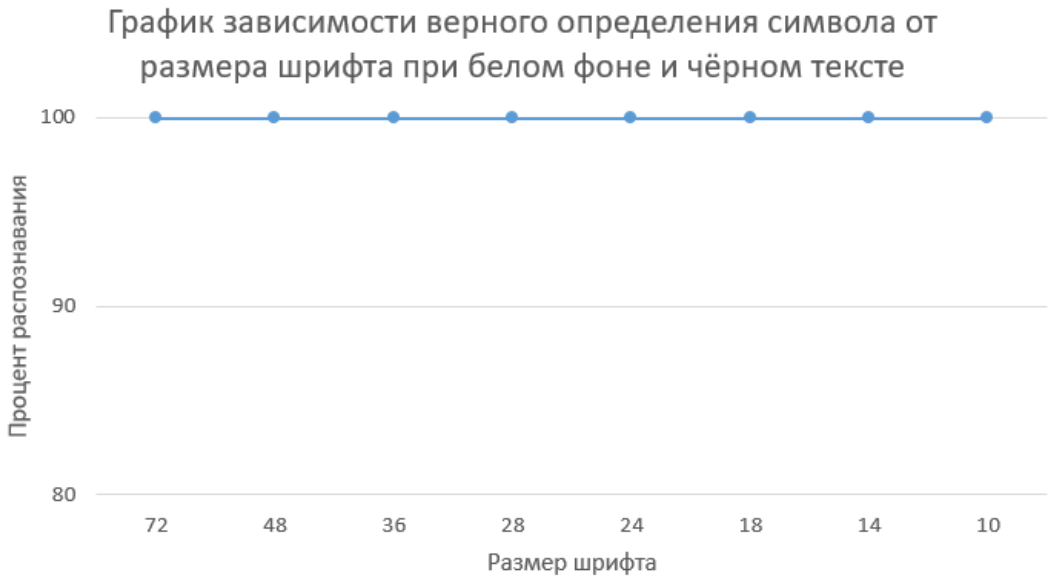

Рисунок 20 – График зависимости верного определения символа от размера шрифта при белом фоне и чёрном тексте.

Однако стоит отметить, что использование алгоритма на изображении чёрного текста на тёмном фоне может привести к ошибкам распознавания. Для этого тестирования были взяты те же изображения, но уже с тёмным фоном. Пример одного из изображений показан на рисунке 21.

Рисунок 21 – Слово на тёмном фоне для проведения тестирования алгоритма на возможность распознавания текста разного размера

Результаты тестирования чёрного текста на тёмном фоне были занесены в таблицу 3.

Таблица 3 – Результаты тестирования алгоритма над разной величиной шрифта на тёмном фоне

Размер шрифта	72	48	36	28	24	18	14	10
Точность распознавания	17%	0%	17%	17%	100%	100%	17%	100%

Из полученных данных можно сделать вывод, что точность определения символа действительно зависит от размера текста и самого символа. По представленной выше таблице был построен график, представленный на рисунке 22.

Рисунок 22 – График зависимости верного определения символа от размера шрифта при схожести цветов фона и текста

По неизвестной причине текст с меньшим размером чаще получал более высокую точность распознавания. Однако благодаря программной функции бинаризации изображения, при которой символы относительно фона будут более явно выражены, подобных ошибок в распознавании символов можно избежать.

Еще одним недостатком алгоритма является то, что при увеличении количества символов на изображении, значительно увеличивается время определения наличия символа на изображении, что в свою очередь увеличивает время работы программы. Это связано с тем, что найденные контуры на изображении обрабатываются последовательно. Хорошей доработкой данного алгоритма является создание его распараллеленной версии, что уменьшит время обработки изображения за счет увеличения количества нагруженных потоков.

Таким образом, в результате тестирования разработанного алгоритма были получены данные о его работе при различных условиях. Входные изображения при тестировании отличались видом шрифта, цветом символов и размером символов на изображении. После проведения всех тестов данные были занесены в таблицы и был построен график зависимости верного определения символа от размера шрифта. Также были выявлены недостатки алгоритма и описаны возможные варианты для его улучшения.

Также для более точной работы алгоритма расстояние между буквами должно быть таким, чтобы контуры букв не соприкасались. В таком случае модуль сегментации сможет более явно и с большей вероятностью определить тот или иной символ.

Еще один недостаток алгоритма заключается в том, что время определения наличия символа на изображении значительно увеличивается с увеличением количества символов на изображении что, в свою очередь, увеличивает также и время выполнения программы. Это связано с тем, что обработка найденных на изображении контуров происходит

последовательно. Хорошим усовершенствованием этого алгоритма является создание его распараллеленной версии, которая сократит время обработки изображения за счет увеличения количества загружаемых потоков, а также возможность распознавания табличных данных с изображения.

Выводы и результаты по главе 3:

- В результате тестирования разработанного алгоритма были получены данные о его работе в различных условиях. Входные изображения во время теста различались цветом шрифта и размером шрифта в изображении;
- После проведения всех тестов данные были вставлены в таблицы. Был построен график зависимости правильного определения шрифта от размера шрифта;
- Также были выявлены недостатки алгоритма и описаны возможные варианты его улучшения.

Заключение

Выпускная квалификационная работы посвящена разработке алгоритма распознавания текста на изображении про помощи метода Tesseract и библиотеки EmguCV, являющейся кроссплатформенной обёрткой библиотеки компьютерного зрения OpenCV. В результате ее выполнения был спроектирован, реализован и протестирован алгоритм, позволяющий распознать на изображении буквы, слова и перевести распознанные символы в печатный текст с возможностями:

- Сохранения его в шаблон документа формата rtf;
- Копирования текста в буфер обмена;
- Редактирования распознанного текста.

Исходный код программы представлен в приложении Б.

Во время выполнения данной работы были успешно завершены следующие задачи:

- Изучены и проанализированы существующие методы и программные решения по распознаванию печатных символов;
- Описана общая схема работы алгоритмов распознавания;
- Произведено сравнение методов распознавания символов для выбора наиболее подходящего к реализации приложения;
- Рассмотрен существующий программный инструментарий для реализации приложения;
- Реализован прототип приложения, способный распознавать текстовые данные с отсканированного документа и сохранять эти данные в шаблон электронного документа;
- Реализованное приложение было протестировано с использованием различных графических изображений текста.

Для тестирования работоспособности алгоритма на вход системы подавались изображения, имеющие различные особенности, такие как:

- Цвет символов;

– Размер символов;
– Цвет фона изображения.

По окончанию тестирования были построены таблицы и графики, показывающие эффективность алгоритма. По результатам тестирования полученной программы были выявлены ее недостатки, а также предложены варианты ее улучшения.

Хотя это и не часто обсуждается за пределами индустрии машинного обучения, оптическое распознавание символов имеет один из самых высоких рейтингов использования в сфере искусственного интеллекта. Предприятия по-прежнему работают на основе огромных объемов бумажной документации, что является устаревшей и почти вредной практикой. OCR может помочь бизнесу справиться с этим путем оцифровки рабочего процесса.

Более того, сфера применения для OCR на этом не заканчивается. Любой текст, будь то аккуратно организованный отчет, случайный знак магазина или рукописная записка, может быть обработан и преобразован OCR в машиночитаемый текст. Это шаг к автоматизации больших данных.

Любопытно, что, несмотря на то, что создание алгоритмов распознавания текста не является новой технологией, это сложно, как никогда. Конечно, общественность имеет доступ к открытым исходным кодам OCR алгоритмов. Но если необходима модель распознавания текста, которая будет служить конкретным целям, то лучше построить ее самостоятельно.

Список используемой литературы и используемых источников

1. Бобылева М.П. Эффективный документооборот: от традиционного к электронному / М.П. Бобылева. – М.: Издательство МЭИ, 2004, – 172 с.
2. Оптическое распознавание символов [Электронный ресурс]: Википедия. Свободная энциклопедия. – URL: https://ru.wikipedia.org/wiki/Оптическое_распознавание_символов
3. Типовые проблемы, связанные с распознаванием символов [Электронный ресурс]: Студенческая библиотека онлайн. – URL: https://studbooks.net/2250220/informatika/tipovye_problemy_svyazannye_raspoznavaniem_simvolov
4. Форматы изображения [Электронный ресурс]: FOTODIZART. Дизайн и разработка сайтов. – URL: https://fotodizart.ru/formaty-izobrazheniya.html
5. ABBYY FineReader 10 Home Edition [Электронный ресурс]: PC, Mobile Utilites. – URL: https://pcutilites.com/43-abbyy-finereader-10-home-edition.html
6. C#: Перевод изображения в монохром [Электронный ресурс]: Блог доброго программиста. – URL: https://blog.foolsoft.ru/c-perevod-izobrazheniya-v-monoxrom/
7. Computer Vision Toolbox. Design and test computer vision, 3D vision, and video processing systems [Электронный ресурс]: MathWorks. – URL: https://www.mathworks.com/products/computer-vision.html
8. EmguCV: API Documentation [Электронный ресурс]: Emgu. – URL: https://www.emgu.com/wiki/index.php/Documentation
9. Huang D. Optical Character Recognition Technology. Application of ABBYY FineReader 11 Kindle Edition / D. Huang. – Kindle Store. – 2013. – Jule. – 253 с.
10. IMAQ. IMAQ Vision for LabVIEW. User Manual [Электронный ресурс]: National Instruments. – URL: https://www.ni.com/pdf/manuals/371007a

11. Kissell J. Take Control of Your Paperless Office, 3rd Edition – Kindle Edition / J. Kissell. – Kindle Store. – 2017 – Mar. – 178 с.

12. LabVIEW – первое знакомство [Электронный ресурс]: habr. – URL: https://habr.com/ru/post/57859/

13. Martin J. Visual Studio 2015 Cookbook – Second Edition / J. Martin. – Kindle Store. – 2016. – Aug. – 368 с.

14. OCR CuneiForm [Электронный ресурс]: PC, Mobile Utilites. – URL: https://pcutilites.com/356-ocr-cuneiform.html

15. OCRopus – OCRopus [Электронный ресурс]: Википедия, бесплатная энциклопедия. – URL: https://ru.xcv.wiki/wiki/OCRopus

16. OpenCV: API Documentation [Электронный ресурс]: OpenCV Documentation. – URL: https://docs.opencv.org/2.4/modules/refman.html

17. Sells C. Windows Forms Programming in C# – Illustrated Edition / C. Sells. – Amazon. – 2003. – Sep. – 682 с.

18. Sharp J. Microsoft Visual C#. Step by Step – Ninth Edition / J. Sharp. – Microsoft Press Store. – 2018. – June. – 832 с.

19. Tesseract-OCR [Электронный ресурс]: ВикиПрограммы. – URL: https://wikiprograms.org/tesseract-ocr/

20. Tesseract-OCR. Open source [Электронный ресурс]: GitHub. – URL: https://github.com/tesseract-ocr/tesseract

Приложение А

Сравнение программных решений

Таблица А.1 – Сравнение популярных программных решений для оптического распознавания символов

Программа	Лицензия	Windows	MAC	Linux	Язык программирования	Языки	Шрифты	Форматы вывода
Tesseract	Бесплатная	+	+	+	C++	100+	Любой печатный	TXT, ALTO, hOCR, PDF, другие с другими пользовательскими интерфейсами или API
Abbyy FineReader	Коммерческая	+	+	+	C/C++	192	Все	DOC, DOCX, XLS, XLSX, PPTX, RTF, PDF, HTML, CSV, TXT, ODT, DjVu, EPUB, FB2
CuneiForm	Бесплатная, с минимальными ограничениями	-	+	+	C/C++	28	Любой печатный	HTML, hOCR, RTF, TeX, TXT
OCRopus	Бесплатная	-	+	+	C++, Python	Латиница	Латиница	TXT, hOCR, PDF

Приложение Б

Код программы

Исходный код программы доступен для скачивания в электронном в виде.

URL: https://disk.yandex.ru/d/PluMJSnp4UcpCQ