Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Языки программирования (ЯП)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему

Программное средство для распознавания текста

БГУИР КП I–40 01 01 21 ПЗ

Студент: гр. 051004 Павловец С.В.

Руководитель: асс. Шостак Е.В.

Минск 2021

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

УТВЕРЖДАЮ

Заведующий кафедрой ПОИТ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

Лапицкая Н.В. 2021г.

ЗАДАНИЕ

по курсовому проектированию

Студенту Павловцу Сергею Валерьевичу \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Тема работы Программное средство для распознавания текста \_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2. Срок сдачи законченной работы 28.12.2021г.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3. Исходные данные к работе Среда разработки Visual Studio 2019, язык программирования C++, интерфейс WinAPI, фотографии букв русского алфавита шрифта Arial, текстовый файл с правильно написанными русскими словами. Возможность выбора файла изображения для извлечения из него текста, просмотра справки о приложении, просмотра границ символов, найденных на изображении, возможность распознавания печатных символов из изображения форматов BMP, JPG, PNG с помощью алгоритма сопоставления с шаблоном, сохранение полученного текста в текстовый файл. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

4. Содержание расчетно-пояснительной записки (перечень вопросов, которые подлежат разработке)

Введение. 1. Аналитический обзор литературы. 2. Моделирование предметной области. 3. Разработка программного средства. 4. Тестирование и проверка работоспособности программного средства. 5. Руководство по установке и использованию программного средства. Заключение.

5. Перечень графического материала (с точным обозначением обязательных чертежей и графиков)

Схема алгоритма в формате А1\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

6. Консультант по курсовой работе Шостак Е.В.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

7. Дата выдачи задания 16.09.2021г.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

8. Календарный график работы над проектом на весь период проектирования (с обозначением сроков выполнения и процентом от общего объема работы):

Раздел 1. Введение к 15.09.2021 – 15 % готовности работы;

Разделы 2 к 15.10.2021 – 30 % готовности работы;

Разделы 3 к 15.11.2021 – 60 % готовности работы;

Раздел 4, 5. Заключение к 15.12.2021 – 90 % готовности работы;

Оформление пояснительной записки и графического материала к 26.12.2021 – 100 % готовности работы.

Защита курсового проекта с 27.12.2021г. по 28.12.2021г.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

РУКОВОДИТЕЛЬ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Е.В Шостак

(подпись)

Задание принял к исполнению \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_С.В. Павловец 16.09.2021г.

(дата и подпись студента)

Содержание

[Введение 6](#_Toc91543298)

[1 Аналитический обзор литературы 7](#_Toc91543299)

[1.1 Анализ существующих аналогов 7](#_Toc91543300)

[1.1.1 ABBYY Finereader 7](#_Toc91543301)

[1.1.2 ABBYY Screenshot Reader 8](#_Toc91543302)

[1.1.3 OCR CuneiForm 9](#_Toc91543303)

[1.2 Постановка задачи 10](#_Toc91543304)

[2 Моделирование предметной области 11](#_Toc91543305)

[2.1 ‌Этапы распознавания текста 11](#_Toc91543306)

[2.2 ‌Предобработка 11](#_Toc91543307)

[2.3 ‌Сегментация 12](#_Toc91543308)

[2.4 ‌Классификация 12](#_Toc91543309)

[2.5 ‌Постобработка 13](#_Toc91543310)

[3 Разработка программного средства 14](#_Toc91543311)

[3.1 Описание графических компонентов 14](#_Toc91543312)

[3.1.1 Взаимосвязь графического интерфейса 14](#_Toc91543313)

[3.1.2 Описание графических компонентов главного окна 14](#_Toc91543314)

[3.2 Описание алгоритмов 16](#_Toc91543315)

[3.2.1 Описание алгоритма getProfile 16](#_Toc91543316)

[3.2.2 Описание алгоритма splitBound 17](#_Toc91543317)

[3.2.3 Описание алгоритма groupBounds 19](#_Toc91543318)

[3.2.4 Описание алгоритма getBounds 20](#_Toc91543319)

[3.2.5 Описание алгоритма match 21](#_Toc91543320)

[3.2.6 Описание алгоритма classify 22](#_Toc91543321)

[3.2.7 Описание алгоритма recognise 23](#_Toc91543322)

[3.2.8 Описание алгоритма lev 24](#_Toc91543323)

[4 Тестирование и проверка работоспособности программного средства 25](#_Toc91543324)

[4.1 Сегментация 25](#_Toc91543325)

[4.1.1 Тест 1 25](#_Toc91543326)

[4.1.2 Тест 2 25](#_Toc91543327)

[4.1.3 Тест 3 26](#_Toc91543328)

[4.2 Классификация 27](#_Toc91543329)

[4.2.1 Тест 4 27](#_Toc91543330)

[4.2.2 Тест 5 27](#_Toc91543331)

[4.2.3 Тест 6 28](#_Toc91543332)

[4.2.4 Тест 7 29](#_Toc91543333)

[4.3 Постобработка 30](#_Toc91543334)

[4.3.1 Тест 8 30](#_Toc91543335)

[4.3.2 Тест 9 31](#_Toc91543336)

[4.3.3 Тест 10 32](#_Toc91543337)

[5 Руководство по установке и использованию программного средства 33](#_Toc91543338)

[5.1 Установка 33](#_Toc91543339)

[5.2 Загрузка изображения 33](#_Toc91543340)

[5.3 Закрытие изображения 35](#_Toc91543341)

[5.4 Извлечение текста 35](#_Toc91543342)

[5.5 Сохранение текста в файл 36](#_Toc91543343)

[5.6 Просмотр справки 37](#_Toc91543344)

[Заключение 38](#_Toc91543345)

[Список использованной литературы 39](#_Toc91543346)

[Приложение А 40](#_Toc91543347)

[Приложение Б 55](#_Toc91543348)

[Приложение В 57](#_Toc91543349)

[Приложение Г 63](#_Toc91543350)

[Приложение Д 66](#_Toc91543351)

[Приложение Е 69](#_Toc91543352)

Введение

Большинство информации в повседневной жизни человек получает при помощи своего зрения. Этот обуславливает необходимость автоматизации извлечения информации с различного рода изображений и её последующего анализа, а также развитие теории компьютерного зрения.

Оптическое распознавание символов — одна из областей компьютерного зрения, заключающаяся в механическом или электронном переводе изображений рукописного, машинописного или печатного текста в текстовые данные, использующиеся для представления символов в компьютере.

Первые разработки в этой области были начаты ещё в 1870-х годах [5]. Позднее в 1929 году Густав Таушек получил патент на метод оптического распознавания текста в Германии, после чего за ним последовал Гендель, получив патент на свой метод в США в 1933. В 1935 году Таушек также получил патент США на свой метод. Машина Таушека представляла собой механическое устройство, которое использовало шаблоны и фотодетектор.

Для облегчения задачи распознавания текста ранее были созданы и стандартизированы некоторые моноширинные шрифты OCR-A, OCR-B, которые при этом сохраняют возможность прочтения человеком. Однако позднее стали успешно разрабатываться средства и для более распространённых шрифтов.

Первая коммерчески успешная программа для распознавания кириллического текста – «AutoR», которая была разработана компанией «ОКРУС» и начала распространяться в 1992 году.

Существует множество приложений распознавания текста: сканирование документов, оцифровка книг, считывание символов с бланков, номеров кредитных карт. Ещё одно из применений – чтение вслух текста для слепых. Одним из примеров служит разработанное в 1976 году компанией «Курцвейл компьютер продактс» устройство, состоящее из ПЗС планшетного сканера и синтезатора, преобразующего текст в речь.

Процесс извлечения текста из фотографии во многом зависит от различных факторов, к примеру:

* язык вместе с его алфавитом,
* тип письма: рукописный или печатный,
* стиль шрифта при печатном письме,
* направление письма,
* взаимное расположение текста,
* наличие или отсутствие на картинке изображений помимо текста.

Главными задачами этой работы является создание программного средства для извлечения печатного текста русского алфавита шрифта Arial разного размера с изображений, обработанных от шумов, поворотов и прочих искажений.

# Аналитический обзор литературы

## Анализ существующих аналогов

### ABBYY Finereader

ABBYY Finereader – одна из лучших программ для качественного распознавания текстов на русском языке.

Рисунок 1.1 иллюстрирует интерфейс программы, а рисунок 1.2 – результат распознавания текста с изображения.

Плюсы:

* качественное распознавание текстов;
* поддержка различных языков (больше 170);
* широкая функциональность;
* встроенная проверка правописания;
* экспорт во множество форматов (docx, pdf);
* поддерживается распознавание многостраничных документов;
* в пробной версии полностью функциональна.

Минусы:

* является платным ПО.

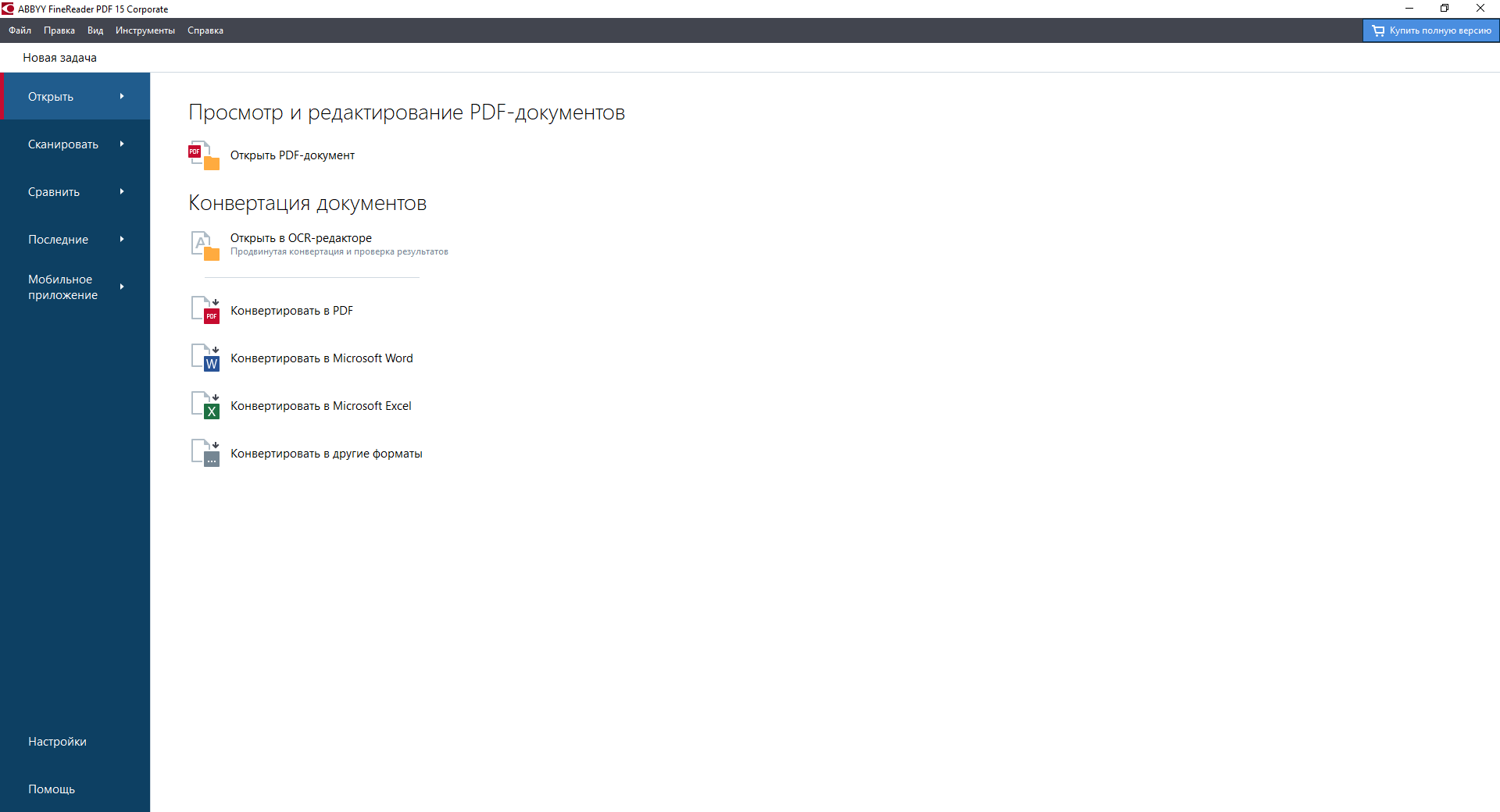


Рисунок 1.1 – Интерфейс программы Finereader

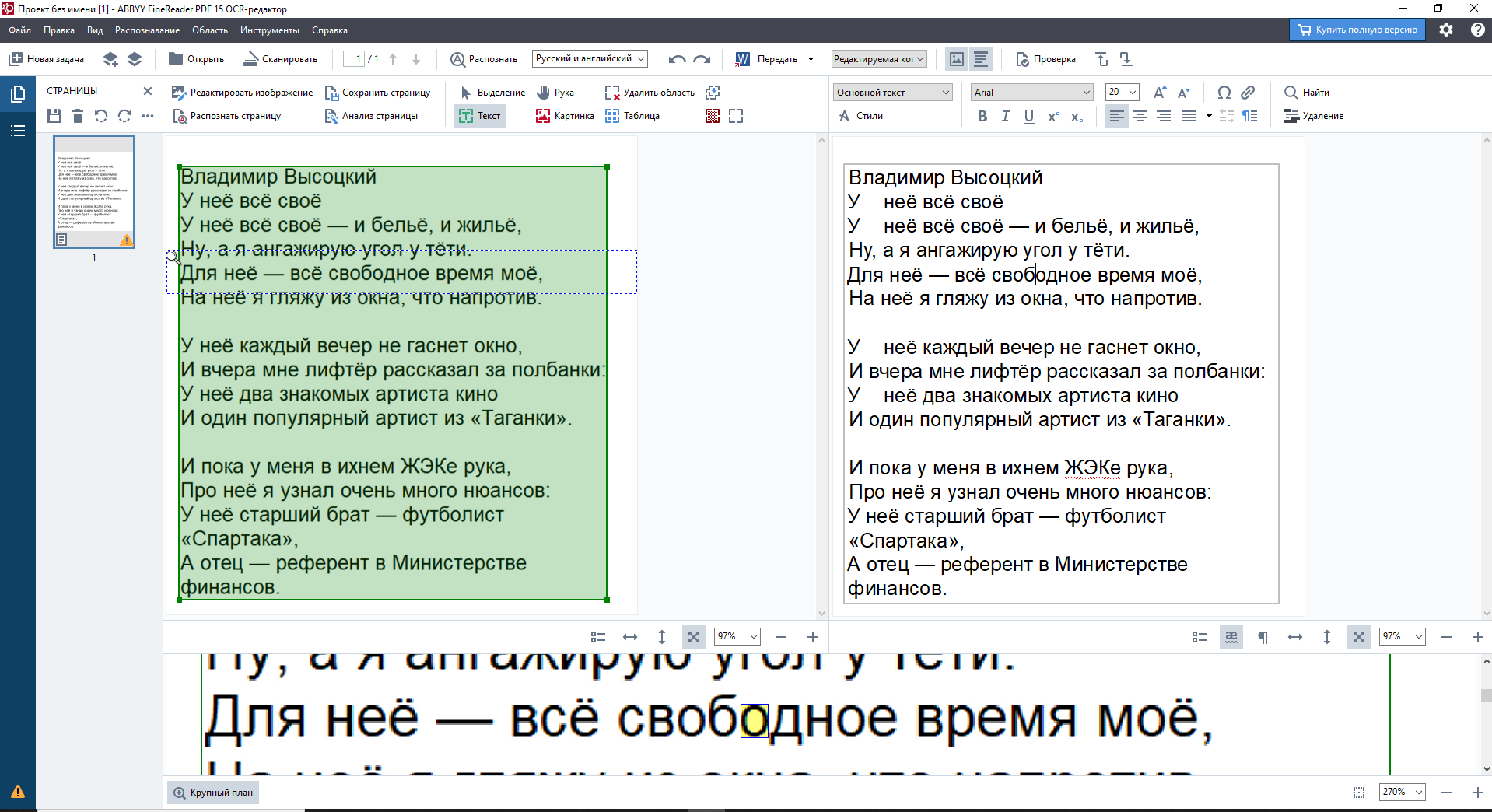


Рисунок 1.2 – Результат работы программы Finereader

### ABBYY Screenshot Reader

ABBYY Screenshot Reader – программа для распознавания текста со снимков экрана.

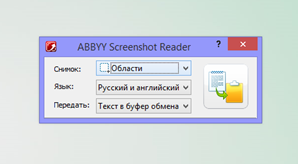


Рисунок 1.3 – Интерфейс программы Screenshot Reader

Плюсы:

* простой и удобный интерфейс;
* использует минимум ресурсов;
* поддержка различных языков;
* возможность сохранения напрямую в буфер обмена;
* множество форматов сохранения.

Минусы:

* является платным ПО

### OCR CuneiForm

CuneiForm обеспечивает быстрое и качественное преобразование бумажных документов и электронных графических файлов в редактируемый текст для последующей работы с ним в офисных программах и текстовых редакторах.

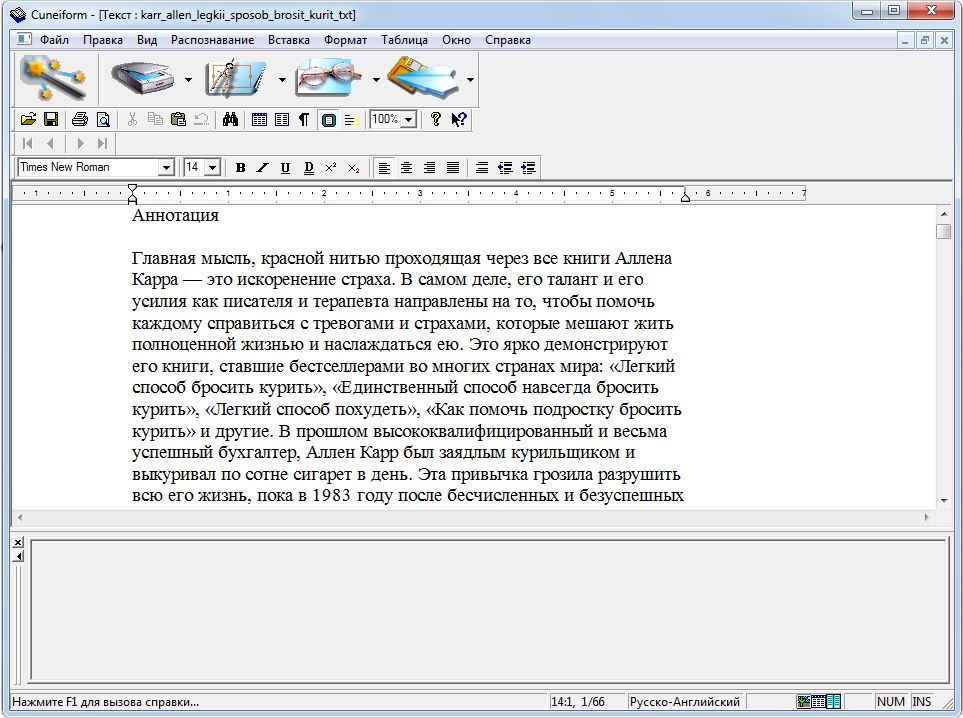


Рисунок 1.4 – Интерфейс программы 1.1.1 OCR CuneiForm

Плюсы:

* поддержка различных шрифтов;
* программу можно скачать бесплатно;
* простой интерфейс;
* текст передается в офисное приложение.

Минусы:

* небольшое количество языков (около 20);
* менее низкая точность распознавания, чем в платных аналогах.

## Постановка задачи

Основной задачей является создание программного средства с графическим интерфейсом, способного извлекать текст шрифта Arial русского алфавита из загруженного из файла изображения формата BMP, JPG, PNG, не содержащего шумы, повороты, картинок, не являющихся буквами и других искажений.

В программе доступны следующие возможности:

* загрузка изображения из файла с его отображением;
* нахождение и отрисовка границ букв;
* распознавание найденных букв русского языка с сохранением пробелов и переходов на новую строку;
* постобработка текста по заданному в текстовом файле словарю;
* сохранение текста в выбранный пользователем файл;
* просмотр справки.

# Моделирование предметной области

## ‌Этапы распознавания текста

В цифровой обработке изображений можно выделить несколько уровней обработки: низкий, средний и высокий. [4]

На низком уровне алгоритмы принимают и возвращают изображения. Их работа заключается в выполнении примитивных операций, таких как уменьшение шума, увеличение резкости, контрастности и т.д.

Следующий уровень принимает изображение и возвращает некие его атрибуты, такие как границы областей, линии контуров, отличительные признаки конкретных объектов и т.д.

Наконец, высокоуровневая обработка включает в себя «осмысление» набора признаков, найденных на среднем уровне.

Аналогично другим задачам цифровой обработки изображения, распознавание теста можно разбить на несколько отдельных блоков: предобработка (низкий уровень), сегментация (средний), классификация (высокий), а также постобработка, относящаяся к области обработки естественных языков.

В дальнейшем рассматриваются только растровые изображения.

## ‌Предобработка

Этап предобработки предполагает избавление от шума, скоса, кусков изображения, не являющегося текстом и т.д.

Наиболее важный этап – применение порогового фильтра (бинаризация). Его работа заключается в сравнении цвета пикселей с пороговым значением и изменение цвета на одно из двух значений в зависимости от результата сравнения.

Бинаризация позволяет отбросить информацию о цвете с целью сосредоточения на форме участков изображения и работы с цветом пикселя как с одним числом.

Большое значение в бинаризации имеет выбор порога. В соответствии с этим бинаризацию подразделяют на статическую (порог - константа) и динамическую (порог зависит от цветовых характеристик изображения), а последнюю - на глобальную (определяется для всего изображения) и локальную (определяется только для некоторой части изображения).

Другие фильтры применяются для избавления от шумов. Они работают с окрестностями пикселей и зачастую основаны на операции свёртки.

Свёртка g матрицы f с ядром w определяется как:

Для получения значений, находящихся за границами матрицы изображения f, прибегают к различным практикам: дублирование значений с границы, получение значений с других концов матрицы и т.д.

Примерами таких фильтров являются фильтр Гаусса, фильтры улучшения чёткости и яркости, эрозия и наращивание. Последние два относятся к группе фильтров морфологических преобразований. [9]

## ‌Сегментация

Сегментация – процесс разделения цифрового изображения на несколько сегментов по определенным критериям.

Для некоторых случаев распознавания текста этот этап может быть упрощён на основе дополнительной информации: о высоте и моноширинности шрифта, ширине отступов, заранее известные места для ввода букв (к примеру, на бланках и банковских картах) и т.д.

Среди методов сегментации можно выделить следующие: основанные на кластеризации, теории графов, контурном анализе, метод водораздела, анализ гистограмм изображения и т.д.

Довольно простыми и эффективными для печатного текста являются методы, основанные на анализе проекционных профилей изображения. [6]

Проекционным профилем изображения называют массив, элементы которого являются суммой яркостей пикселей (числом пикселей переднего плана) соответствующей строки (горизонтальный профиль) или столбца (вертикальный профиль) матрицы изображения.

Эти профили удобно представлять графически в виде гистограмм: на одной оси - координата столбца/строки, на другой – сумма. Гистограммы имеют резкие и длинные перепады в местах отступа между строками, словами и буквами. Именно на этом свойстве основаны алгоритмы сегментации.

Подобный подход, однако, имеет ряд проблем. К примеру, сложность сегментации на уровне букв в письменном тексте, так как, в отличие от печатного, письменные буквы зачастую имеют лигатуры и наклон. К тому же, даже в печатных текстах, у многих шрифтов, расстояние между буквами не фиксировано и прямоугольные области границ могут пересекаться друг с другом, что может отражаться в объединении границ символов.

Анализ гистограммы можно применить и для предобработки изображения, в частности, для нахождения наклона текста. [7]

## ‌Классификация

Этап классификации заключается в сопоставлении частям картинки определённого значения – метки, с помощью которой можно идентифицировать символ в алфавите.

Традиционно, в задачах распознавания текста используют искусственные нейронные сети – системы слоёв, состоящих из отдельных элементов – нейронов, которые принимают, обрабатывают и передают сигналы к другим слоям и соединены связями с определёнными регулируемыми весовыми коэффициентами. Настройка весов называется этапом обучения сети и необходима перед использованием классификатора, для минимизации ошибки. [10]

Другой подход – сопоставление с шаблоном. Каждый раз при необходимости классификации участок изображения попиксельно сравнивается с заранее загруженными в память шаблонами. Область получает ту метку, которой соответствует наибольшая схожесть с шаблоном. [11]

Для нахождения схожести s 2-х изображений I и J с одними размерами (w \* h) наиболее часто используют следующие функции:

## ‌Постобработка

Постобработка – этап, на котором в полученном тексте естественного языка исправляются различные орфографические, морфологические, синтаксические (в т.ч. пунктуационные ошибки) и другие ошибки.

Эти исправления могут проводиться на разных единицах текста: словах, предложениях, словосочетаниях и т.д.

Для исправления орфографических ошибок на уровне слов можно использовать заранее заданный словарь и метрику Левенштейна, определяющей, какое минимальное число удалений, вставок и замен символов в слове необходимо совершить, чтобы получить из него другое.

Эта метрика может быть найдена рекуррентно или по итеративному алгоритму Вагнера-Фишера, использующего вспомогательную матрицу. Алгоритм работает за O (M \* N) и занимает O (M \* N) памяти, где M и N – длины строк. Или O (min {M, N}) памяти при использовании только 2-х строк матрицы на каждом шаге.

# Разработка программного средства

## Описание графических компонентов

### Взаимосвязь графического интерфейса

Для организации графического интерфейса программного средства для распознавания текста было использовано 1 форма – главное окно.

### Описание графических компонентов главного окна

Главное окно предоставляет основной функционал программы: загрузка изображения, нахождение и отрисовка на нём границ букв, распознавания текста с изображения, постобработку текста при распознавании, сохранение текста, просмотр справки.

Окно имеет вид, представленный на рисунке 3.1.

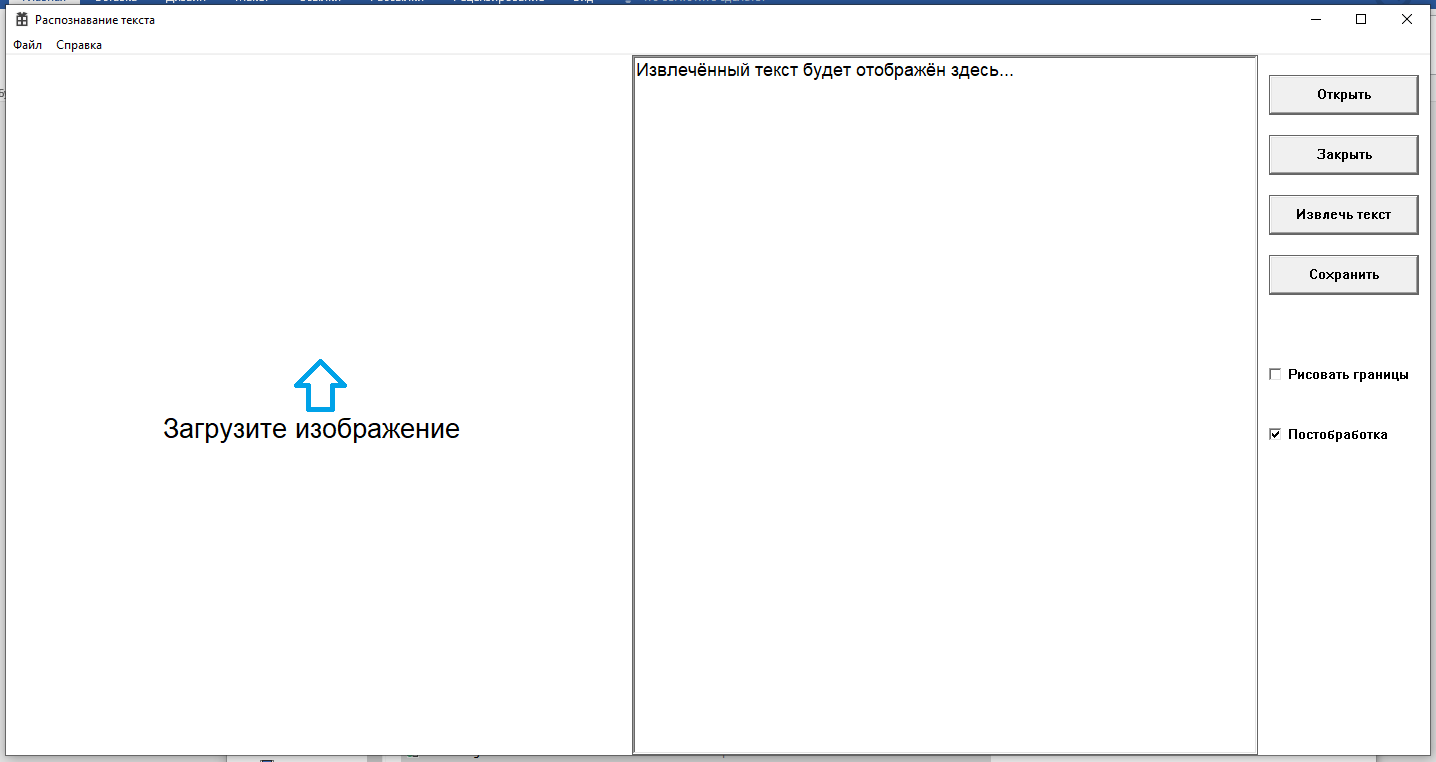


Рисунок 3.1 – Вид главного окна

Потомки главного окна:

* контейнер (статический контрол) для загрузки и отображения выбранного пользователем изображения;
* текстовое поле (rich edit) для вывода текста на изображении после распознавания;
* кнопка «Открыть» для загрузки изображения из файла;
* кнопка «Закрыть» для закрытия изображения, очистки rich edit;
* кнопка «Извлечь текст» для распознавания текста;
* кнопка «Сохранить» для сохранения распознанного текста в указанный пользователем текстовый файл;
* флажок «Рисовать границы» для указания, необходима ли отрисовка границ при следующем извлечении текста;
* флажок «Постобработка» позволяет посмотреть результаты работы алгоритмов без постобработки;
* вкладка меню «Открыть» для загрузки изображения из файла;
* вкладка меню «Закрыть» для закрытия изображения, очистки rich edit;
* вкладка меню «Сохранить» для сохранения распознанного текста в указанный пользователем текстовый файл;
* вкладка меню «Выход» для выхода из приложения;
* вкладка меню «О программе» для вызова справки.

## Описание алгоритмов

### Описание алгоритма getProfile

Подсчёт проекционного профиля прямоугольной области изображения. Тип профиля определяется по переданной оси: если ось Ox, то считается горизонтальный профиль, иначе – вертикальный.

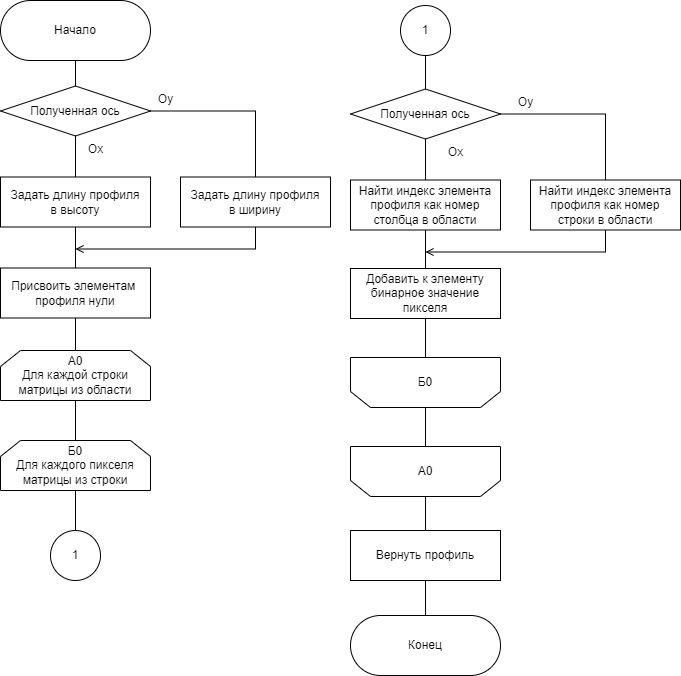


Рисунок 3.1 – Схема алгоритма getProfile

### Описание алгоритма splitBound

Разбиение прямоугольной области r изображения I на подобласти вдоль оси axis. Использует переменную pp для получения проекционного профиля вдоль противоположной разбиению оси. Переменная start и end – абсциссы или ординаты текущей подобласти, изменяются при нахождении перепадов гистограммы от 0 к не 0 и наоборот. Переменная bounds служит для хранения найденных подобластей.

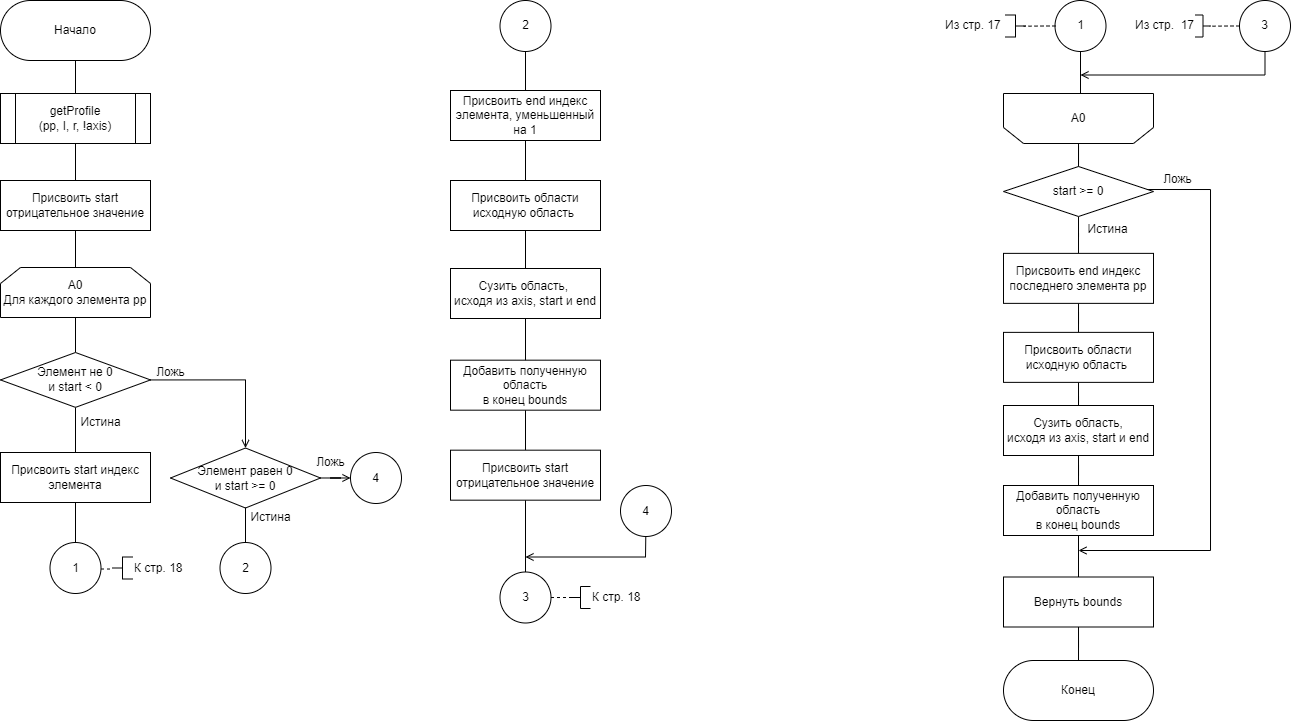


Рисунок 3.2 – Схема алгоритма splitBound (часть 1)

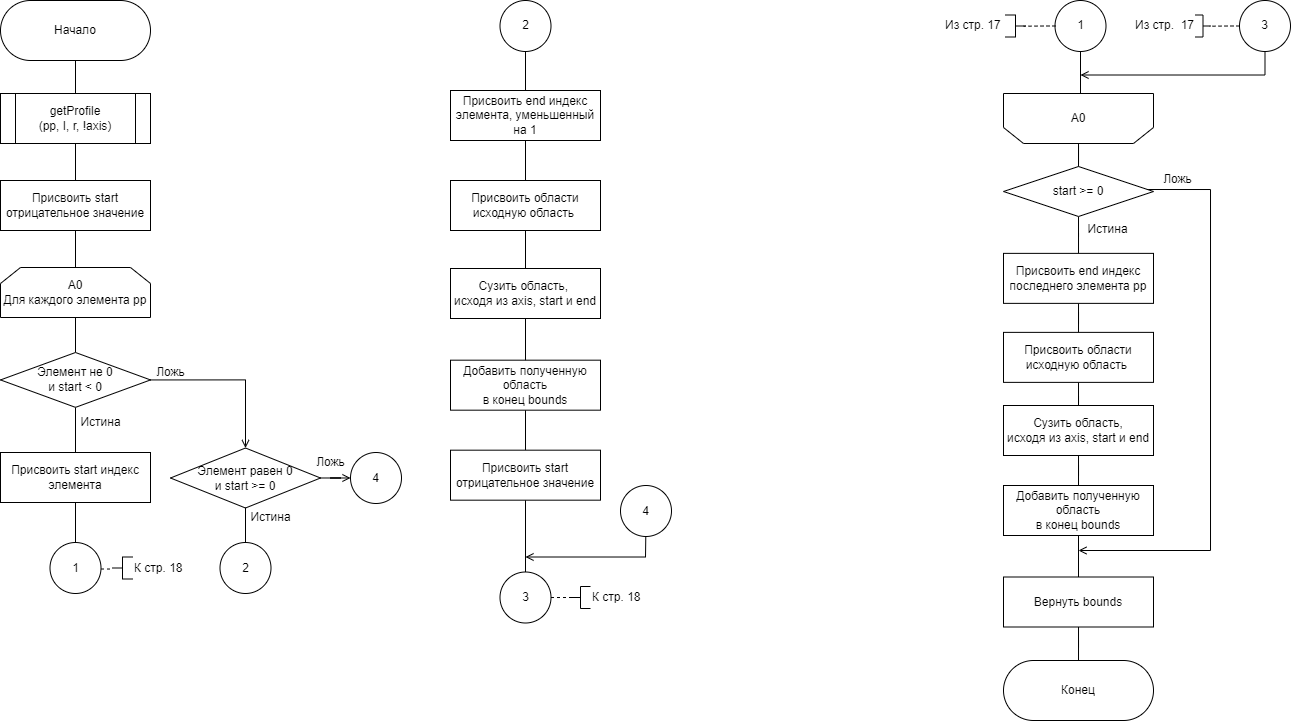


Рисунок 3.3 – Схема алгоритма splitBound(часть 2)

### Описание алгоритма groupBounds

Группирует прямоугольные области bounds в такие последовательности groups, что горизонтальное расстояние следующей области с предыдущей в группе меньше, чем некое заданное значение thresh.

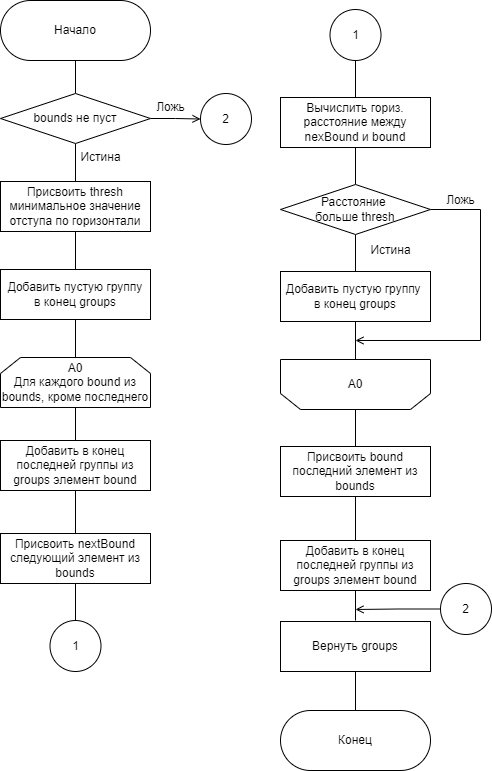


Рисунок 3.4 – Схема алгоритма groupBounds

### Описание алгоритма getBounds

Производит сегментацию изображения I на прямоугольные области bounds. В алгоритме используются переменные lines и letters для сохранения результатов splitBound и words – для сохранения результата groupBounds.

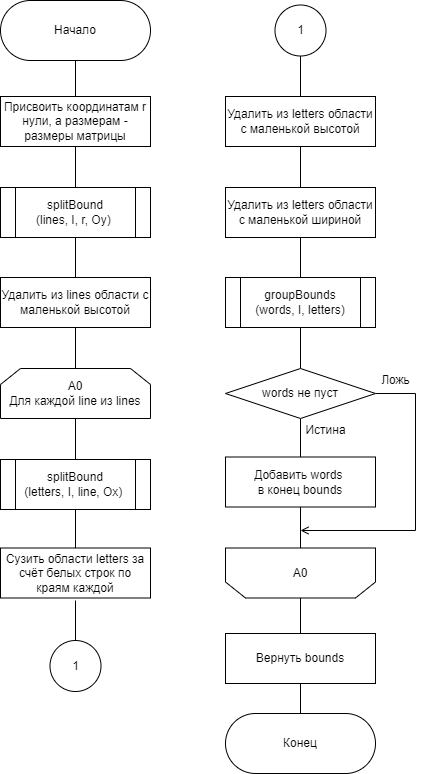


Рисунок 3.5 – Схема алгоритма getBounds

### Описание алгоритма match

Находит долю совпадений пикселей в матрицах изображений X и Y к площади X, учитывая масштаб Y относительно X.

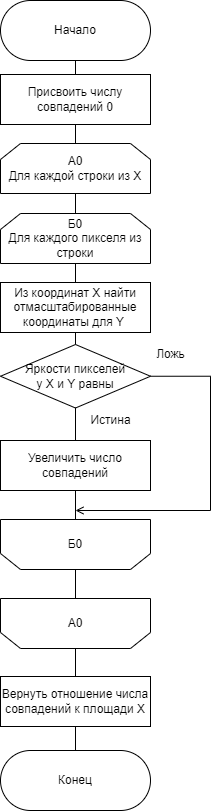


Рисунок 3.6 – Схема алгоритма match

### Описание алгоритма classify

Находит индекс максимального совпадения изображения I с шаблонами из templates, при чём доля совпадения должно быть больше некой эмпирической константы от 0 до 1. Для получения результата из алгоритма match используется переменная similiarity.

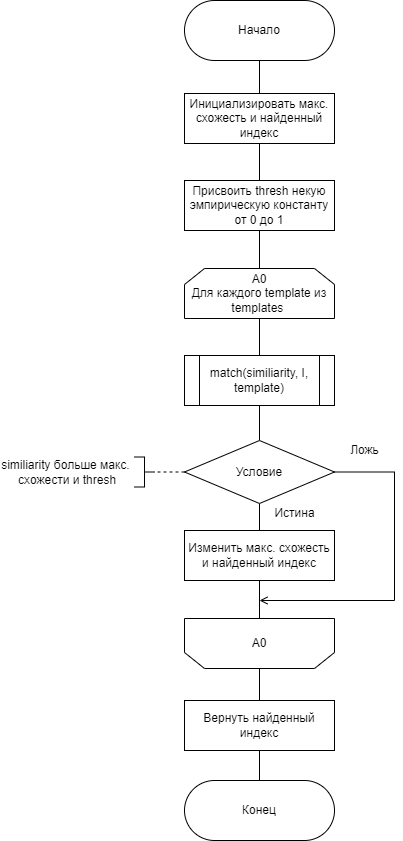


Рисунок 3.7 – Схема алгоритма classify

### Описание алгоритма recognise

Составляет текст, сосредоточенный в областях bounds. Для классификации использует параметры templates и labels, являющиеся шаблонами и соответствующими им метками.

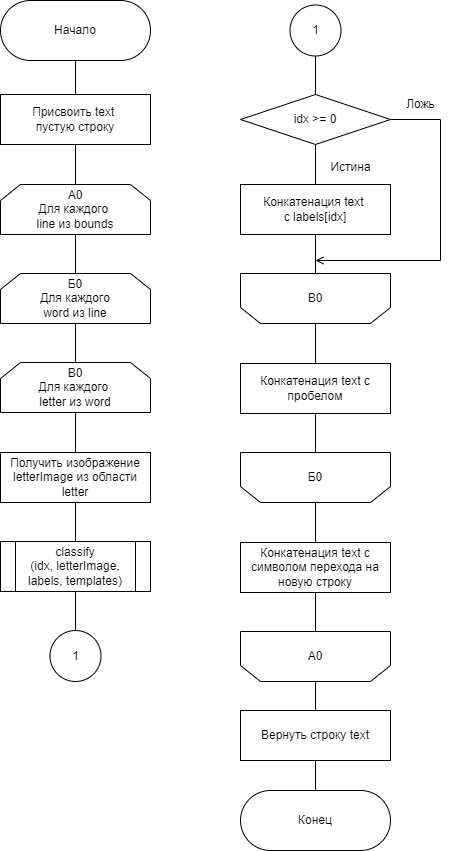


Рисунок 3.8 – Схема алгоритма recognise

### Описание алгоритма lev

Считает метрику Левенштейна по алгоритму Вагнера-Фишера, модифицированному так, что используются только 2 строки вспомогательной матрицы.

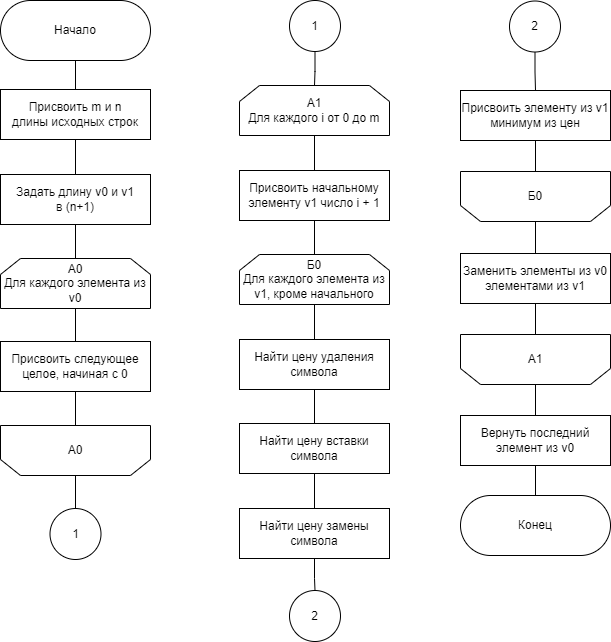


Рисунок 3.9 – Схема алгоритма lev

# Тестирование и проверка работоспособности программного средства

## Сегментация

### Тест 1

Таблица 1 – Тест 1

|  |  |
| --- | --- |
| Тестовая ситуация | На изображении присутствует только 1 буква, обрезанная по бокам |
| Исходный набор данных: | Изображение: |
| Ожидаемый результат: | Обведённая буква в окне с изображением |
| Полученный результат: |  |

### Тест 2

Таблица 2 – Тест 2

|  |  |
| --- | --- |
| Тестовая ситуация | На изображении присутствует несколько слов, все символы которых – буквы русского алфавита |
| Исходный набор данных: | Изображение: |
| Ожидаемый результат: | Обведённые буквы в окне с изображением |

Продолжение таблицы 2

|  |  |
| --- | --- |
| Полученный результат: |  |

### Тест 3

Таблица 3 – Тест 3

|  |  |
| --- | --- |
| Тестовая ситуация | На изображении присутствует тонкие маленькие символы, не входящие в русский алфавит (точка, тире) и диакритические знаки |
| Исходный набор данных: | Изображение: |
| Ожидаемый результат: | Обведённые буквы русского алфавита, не обведённые диакритические знаки, точка и тире в окне с изображением |
| Полученный результат: |  |

## Классификация

### Тест 4

Таблица 4 – Тест 4

|  |  |
| --- | --- |
| Тестовая ситуация | На изображении литературный стих |
| Исходный набор данных: | Изображение: |
| Ожидаемый результат: | Схожий текст без знаков препинания, написанный строчными буквами в окне с текстом |
| Полученный результат: |  |

### Тест 5

Таблица 5 – Тест 5

|  |  |
| --- | --- |
| Тестовая ситуация | На изображении отрывок технической литературы |
| Исходный набор данных: | Изображение: |
| Ожидаемый результат: | Схожий текст без знаков препинания, написанный строчными буквами в окне с текстом |

Продолжение таблицы 5

|  |  |
| --- | --- |
| Полученный результат: |  |

### Тест 6

Таблица 6 – Тест 6

|  |  |
| --- | --- |
| Тестовая ситуация | На изображении отрывок научной литературы |
| Исходный набор данных: | Изображение: |
| Ожидаемый результат: | Схожий текст без знаков препинания, написанный строчными буквами в окне с текстом |
| Полученный результат: |  |

### Тест 7

Таблица 7 – Тест 7

|  |  |
| --- | --- |
| Тестовая ситуация | На изображении отрывок песни |
| Исходный набор данных: | Изображение: |
| Ожидаемый результат: | Схожий текст без знаков препинания, написанный строчными буквами в окне с текстом |
| Полученный результат: |  |

## Постобработка

### Тест 8

Таблица 8 – Тест 8

|  |  |
| --- | --- |
| Тестовая ситуация | Словарь содержит слова, не находящиеся на изображении |
| Исходный набор данных: | Изображение: |
| Ожидаемый результат: | Схожий текст без знаков препинания, написанный строчными буквами в окне с текстом; много ошибок на уровне слов, но сами слова написаны без ошибок |
| Полученный результат: |  |

### Тест 9

Таблица 9 – Тест 9

|  |  |
| --- | --- |
| Тестовая ситуация | Словарь содержит слова, не находящиеся на изображении |
| Исходный набор данных: | Изображение: |
| Ожидаемый результат: | Схожий текст без знаков препинания, написанный строчными буквами в окне с текстом; много ошибок на уровне слов, но сами слова написаны без ошибок |
| Полученный результат: |  |

### Тест 10

Таблица 10 – Тест 10

|  |  |
| --- | --- |
| Тестовая ситуация | Словарь содержит слова, не находящиеся на изображении |
| Исходный набор данных: | Изображение: |
| Ожидаемый результат: | Схожий текст без знаков препинания, написанный строчными буквами в окне с текстом; много ошибок на уровне слов, но сами слова написаны без ошибок |
| Полученный результат: |  |

# Руководство по установке и использованию программного средства

## Установка

Для установки программного средства требуется запустить «setup.exe». Далее откроется окно установщика, представленное на рисунке 5.1.

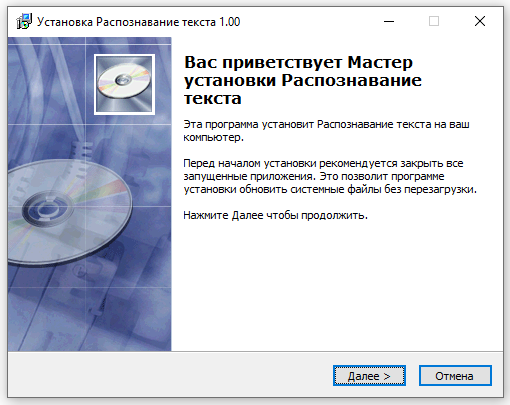


Рисунок 5.1 – Вид главного окна установщика

После установки по указанному пути появится 2 файла: «Распознавание текста.exe» и «dictionary.txt». 1-й файл – непосредственно приложение, 2-й – словарь, используемый для постобработки, в который можно по желанию добавить часто используемые слова через пробелы или запятую. Слова в файле сортируются при запуске приложения.

## Загрузка изображения

Для того, чтобы загрузить изображение пользователь может:

* нажать кнопку «Открыть» слева на главном окне;
* в меню главного окна выбрать «Файл»-«Открыть»;
* сделать двойное нажатие по контейнеру с изображением (при открытии приложения – изображение с надписью: «Загрузите изображение».

После этого откроется диалоговое окно с возможность выбрать файл формата BMP, JPG, PNG по указанному пути, что проиллюстрировано на рис 5.2.

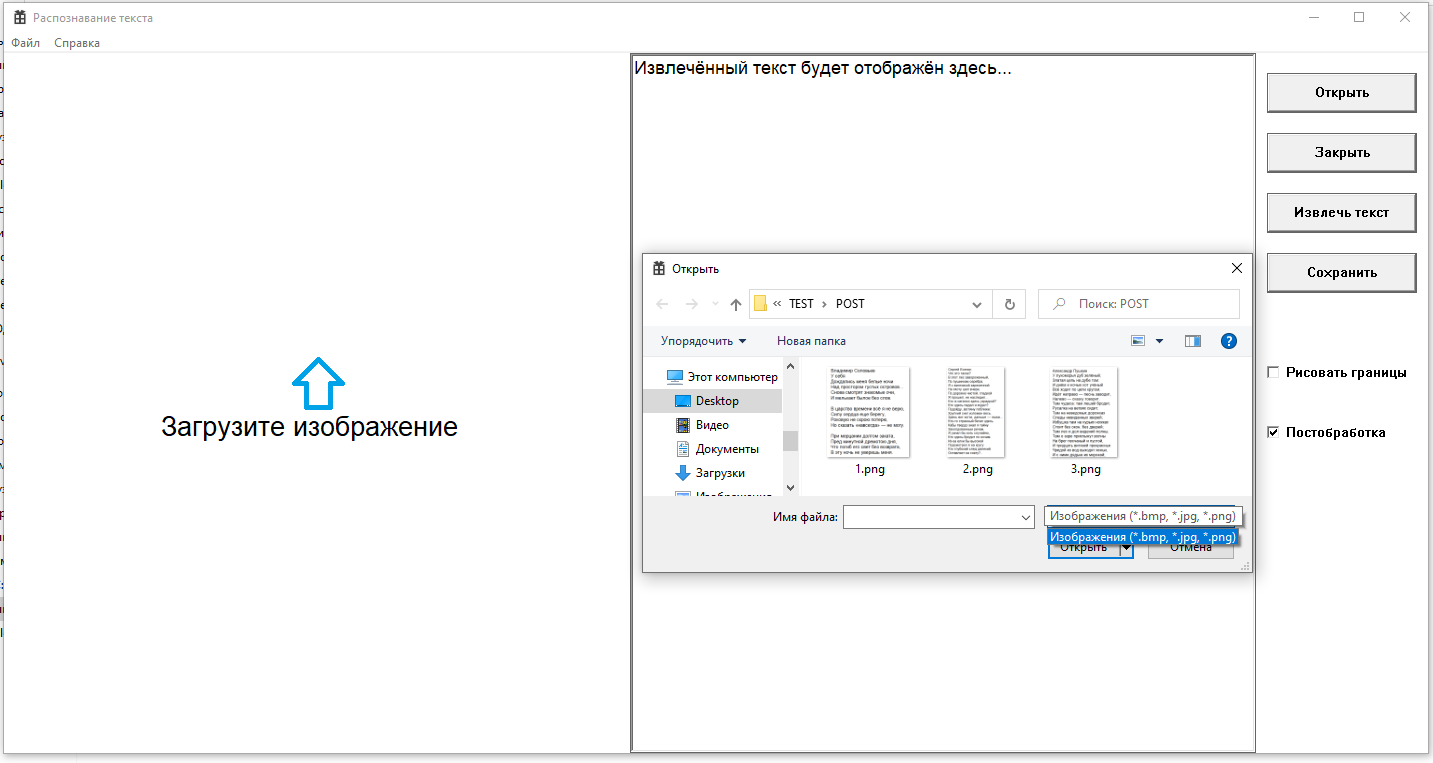


Рисунок 5.2 – Диалог выбора изображения

После подтверждения файла изображение отобразиться в контейнере слева, что показано на рис. 5.3.

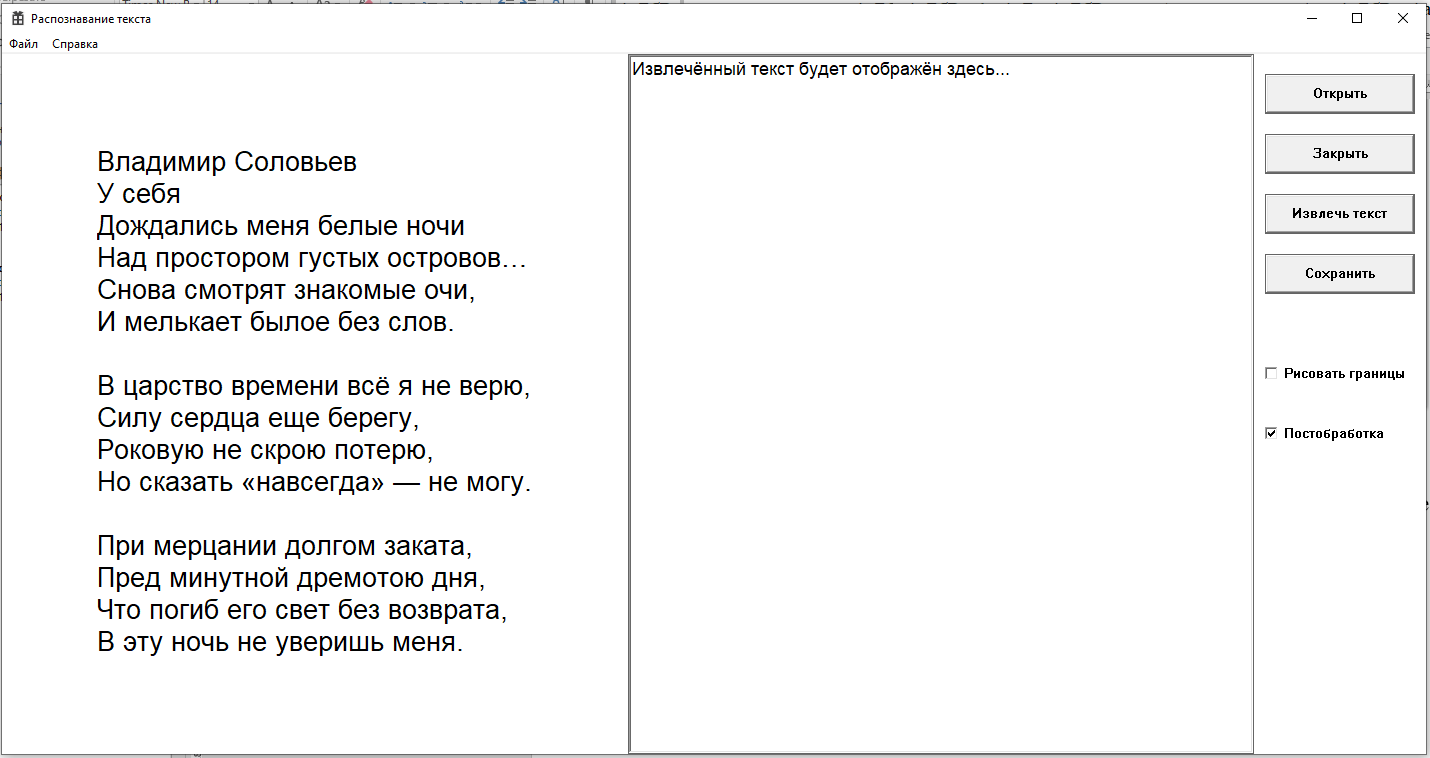


Рисунок 5.3 – Открытое изображение

## Закрытие изображения

Для того, чтобы закрыть выбранное изображение пользователь может:

* нажать кнопку «Закрыть» справа на главном окне;
* в меню главного окна выбрать «Закрыть»

После этого в контейнере с изображением и текстовом поле отобразятся стандартные изображения и текст, что отражено на рис. 5.6. Подобное действие также происходит при открытии другого изображения.

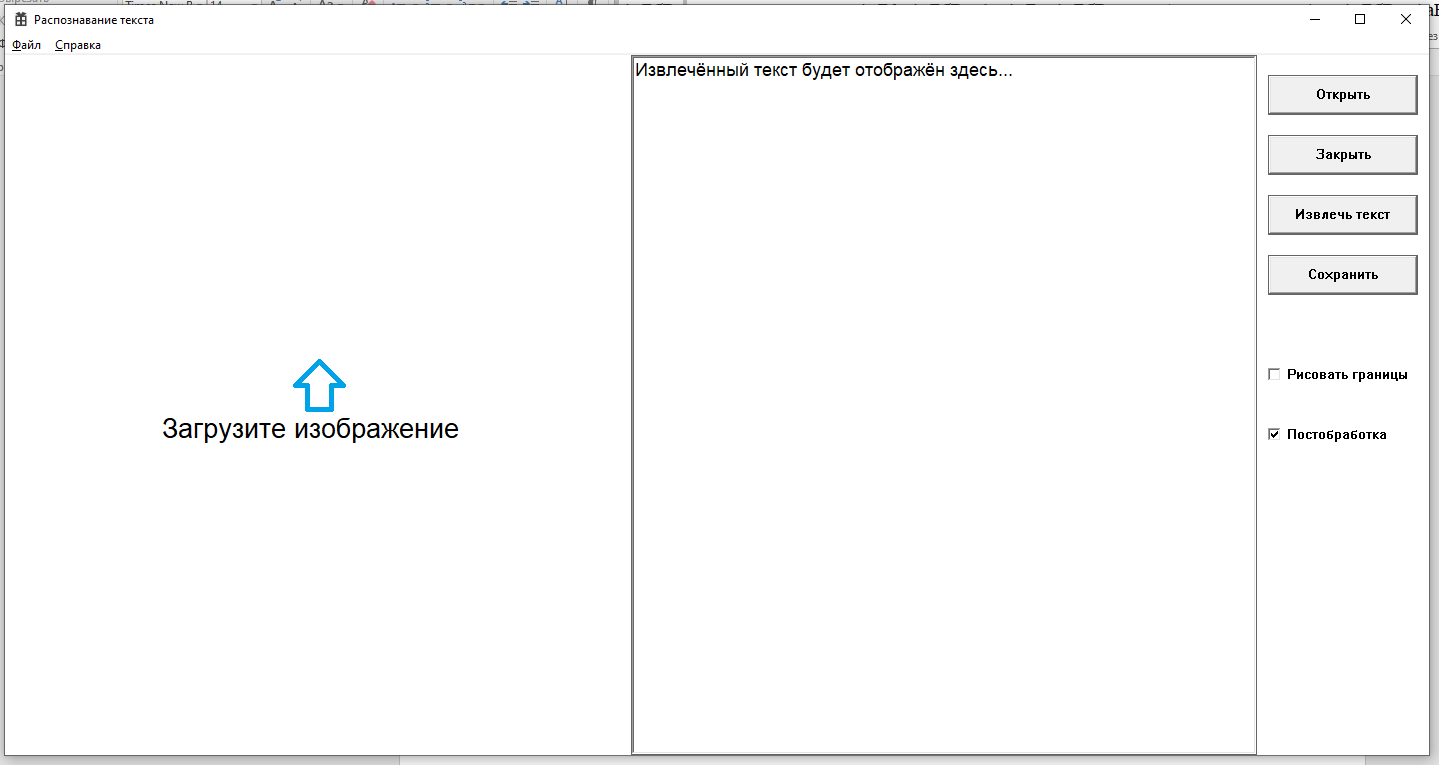


Рисунок 5.4 – Вид окна после закрытия изображения

## Извлечение текста

Для того, чтобы извлечь текст необходимо нажать кнопку «Извлечь текст» справа на главном окне. После этого в текстовом поле появится извлечённый из изображения текст, который можно скопировать или сохранить в файл.

При выбранном флажке «Рисовать границы», вокруг каждой найденной на изображении буквы, будет нарисован окаймляющий её прямоугольник.

При отключенном флажке «Постобработка» в текстовом поле будет выведен результат сегментации и классификации без дополнительной обработки текстовой строки.

Вид главного окна после извлечения текста представлен на рис. 5.7.

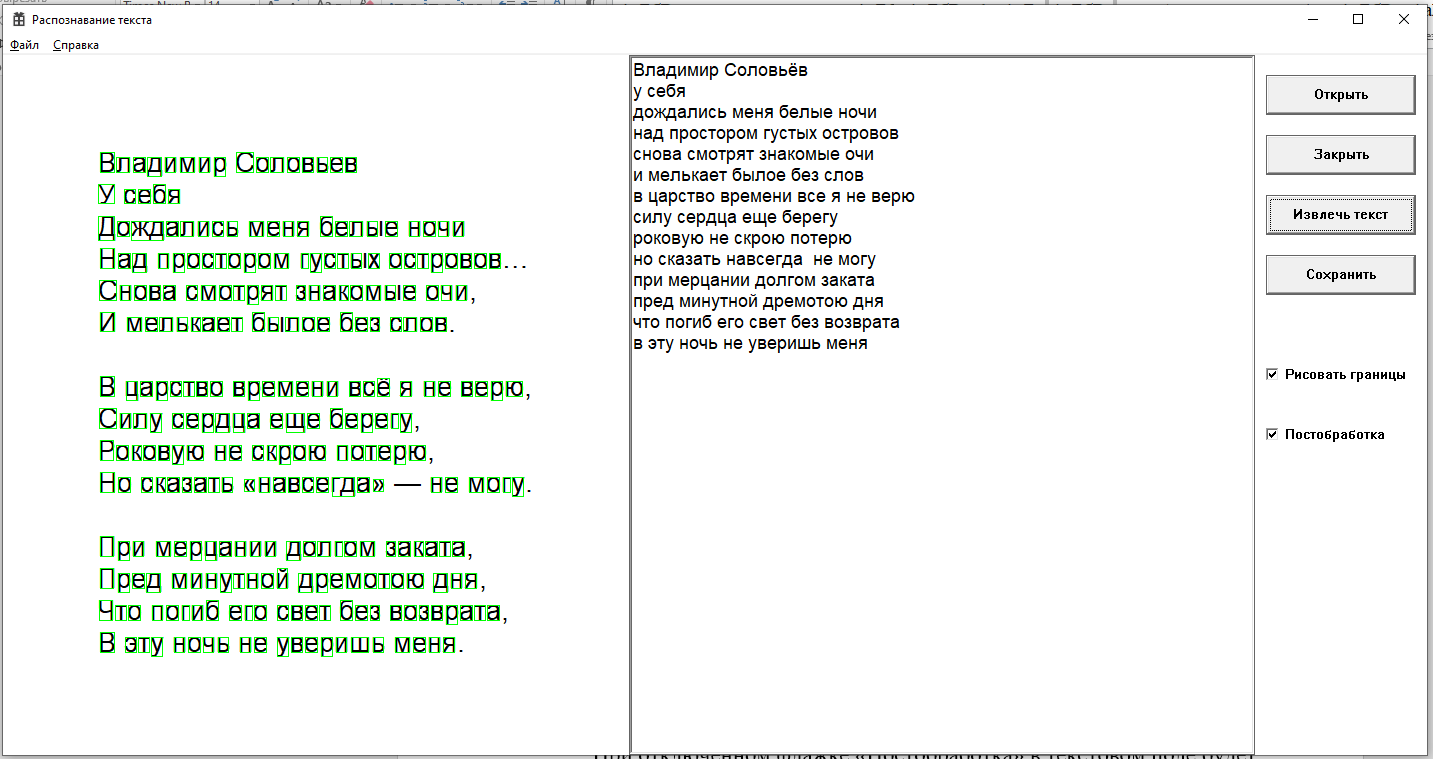


Рисунок 5.5 – Вид главного окна после извлечения текста

## Сохранение текста в файл

Чтобы сохранить извлеченный текст в .txt-файл в формате UTF-8 пользователь может:

* нажать кнопку «Сохранить»;
* в меню главного окна выбрать «Файл»-«Сохранить».

После этого, при наличии в текстовом поле извлечённого текста, появится диалоговое окно, предлагающее выбрать файл для перезаписи (создания). Это можно увидеть на рис 5.6.

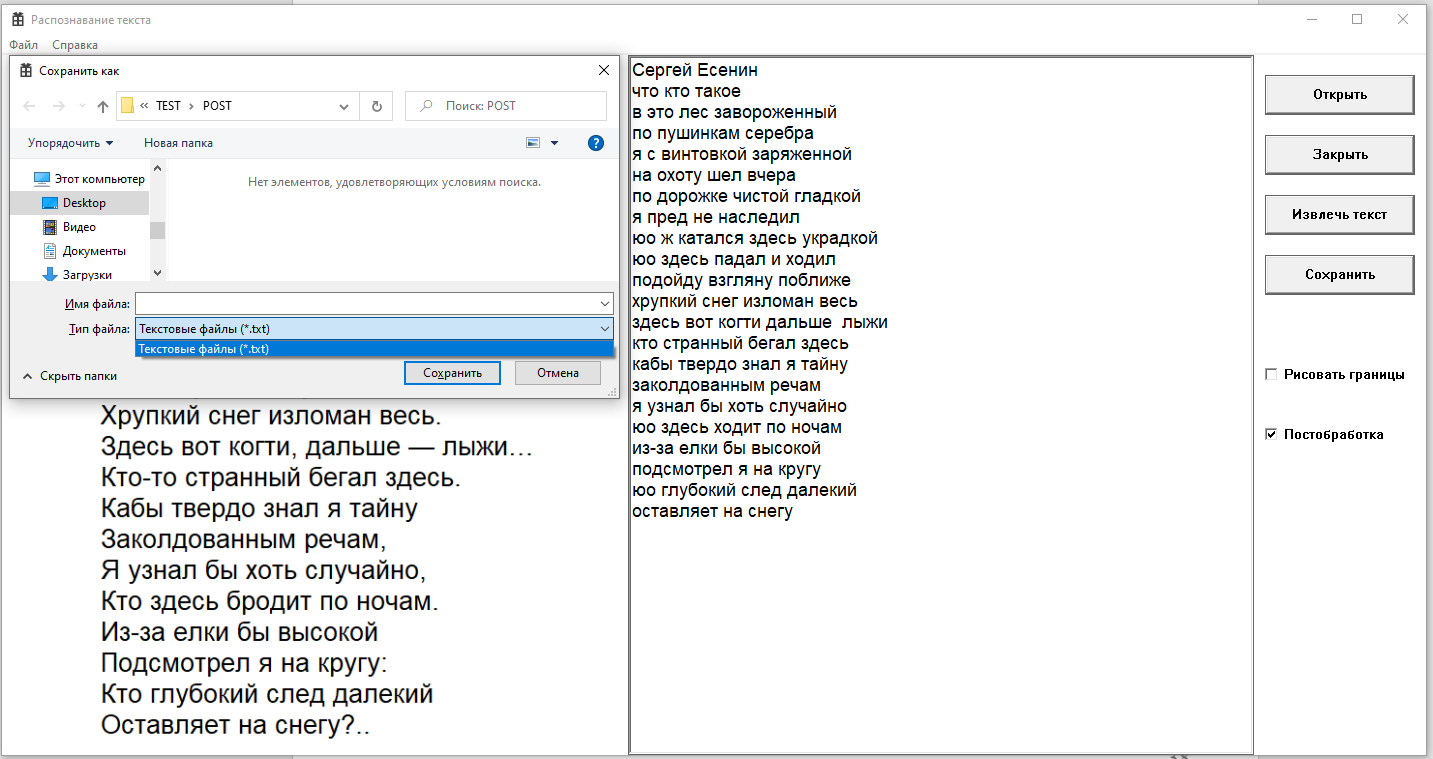


Рисунок 5.6 – Сохранение в файл

## Просмотр справки

Для того, чтобы просмотреть справку о приложении необходимо в меню главного окна выбрать «Справка»-«О программе». Результат можно увидеть на рис. 5.7.

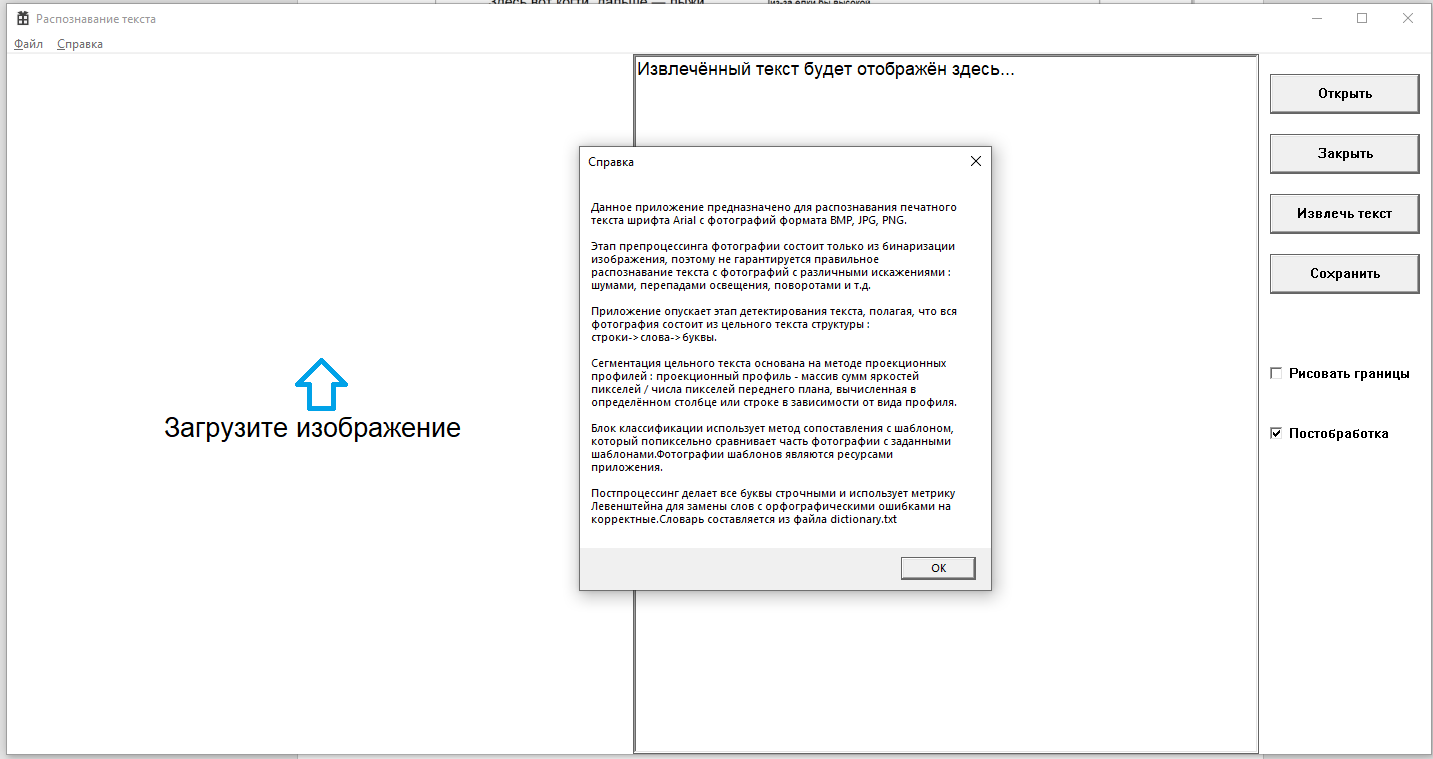


Рисунок 5.7 – Справка программы

Заключение

По итогу работы над курсовым проектом было разработано программное средство для распознавания текста, имеющее графический интерфейс.

Приложение прошло все этапы тестирования, в результате которых были устранены все неполадки. Приложение имеет относительно высокую скоротать работы.

В ходе тестирования обнаружилось, что:

* при сегментации часто возникают такие ошибки, как слияние границ букв (сочетания «жд», «гд» и т.д), буква «ы» делится на 2 символа и подобные слова нуждаются в постобработке, часто диакритических знаки у букв «ё» и «й» отделяются от самих букв;
* при классификации из-за схожести (в плане отношения высоты к ширине) многих печатных букв с их строчными вариантами, возникает необходимость постобработки или дополнительных предположений о размерах шрифта;
* при замене слов на этапе постобработки стоит дать пользователю возможность выбора из нескольких близких вариантов, так как при сильном разрастании словаря часто происходят ненужные замены.

В дальнейшем возможны такие улучшения, как поддержка других операционных систем, возможность работы с искажёнными изображениями (шум, скос, затемнение и т.д.) за счёт улучшения блока предобработки, возможность распознавания рукописного текста, для чего необходима замена классификатора, улучшение процесса постобработки за счёт проверок на уровне предложений, словосочетаний и т.д.

Список использованной литературы

[1] Серебряная Л.В., Марина И.М. Структуры и алгоритмы обработки данных: учебное пособие/ Л.В. Серебряная – Минск: БГУИР, 2013 – 29 с.

[2] Вирт Н. Алгоритмы и структуры данных: учебное пособие / Н.Вирт – М.: Мир, 1989 – 245 с.

[3] Ахо А. Структура данных и алгоритмы: учебное пособие / А. Ахо, Дж. Хопкрофт, Ульман Дж. – М.: Вильямс, 2003 – 77 с.

[4] Гонсалес Р., Вудс Р. Цифровая обработка изображений Издание 3-е, исправленное и дополненное / Р. Гонсалес, Р. Вудс - М: Техносфера, 2012. – 1104 с.

[5] Schantz H. F. The history of OCR: optical character recognition / H. F. Schantz – Recognition Technologies Users Association, 1982 – 114 p.

[6] Segmentation in OCR [Электронный ресурс]. – Режим доступа: <https://towardsdatascience.com/segmentation-in-ocr-10de176cf373>/. – Дата доступа: 25.12.2021.

[7] Deskewing scanned documents using horizontal projections [Электронный ресурс]. – Режим доступа: <https://muthu.co/deskewing-scanned-documents-using-horizontal-projections/>. – Дата доступа: 25.12.2021.

[8] Распознавание текста в ABBYY FineReader [Электронный ресурс]. – Режим доступа: <https://habr.com/ru/company/abbyy/blog/225215/>. – Дата доступа: 25.12.2021.

[9] Математическая морфология [Электронный ресурс]. – Режим доступа: <https://habr.com/ru/post/113626/>. – Дата доступа: 25.12.2021.

[10] Backpropagation [Электронный ресурс]. – Режим доступа: <https://brilliant.org/wiki/backpropagation/>. – Дата доступа: 25.12.2021.

[11] Optical Character Recognition By Using Template Matching [Электронный ресурс]. – Режим доступа: https://www.semanticscholar.org/paper/Optical-Character-Recognition-By-Using-Template-Nadira-Kamariah/b5ac80e654108b898a9fcc827eadf1580d500bcc/. – Дата доступа: 25.12.2021

Приложение А

Исходный код основного модуля

(main.h и main.cpp)

// MAIN.H

#pragma once

#include "resource.h"

#include "RichEdit.h"

#include "preprocessing.h"

#include "segmentation.h"

#include "recognition.h"

#include "postprocessing.h"

#include <fstream>

#include <codecvt>

#include <commdlg.h>

#define MAX\_LOADSTRING 100

#define MIN\_WND\_WIDHT 800 // Min width of the main window

#define MIN\_WND\_HEIGHT 600 // Min height of the main window

#define S\_PBOX 0.44// Scale of the picture box

#define S\_TEXT 0.44// Scale of the text field

#define S\_OPTIONS 0.12// Scale of the panel

#define BTN\_WIDTH 150 // Width of buttons

#define BTN\_HEIGHT 40 // Height of buttons

#define BTN\_GAP 20 // Height of buttons' gaps

#define WIDTH(r) ((r).right - (r).left)

#define HEIGHT(r) ((r).bottom - (r).top)

#define TXTFLD\_STYLE WS\_BORDER | WS\_CHILD | WS\_VISIBLE |\

WS\_VSCROLL | ES\_LEFT | ES\_MULTILINE|\

ES\_AUTOVSCROLL | ES\_READONLY

#define PBOX\_STYLE WS\_VISIBLE | WS\_CHILD |SS\_WHITERECT|\

SS\_NOTIFY | SS\_BITMAP |SS\_CENTERIMAGE

// Function prototypes

ATOM MyRegisterClass(HINSTANCE hInstance);

BOOL InitInstance(HINSTANCE, int);

LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);

void onCreate();

void onSize();

void onLoad();

void onUnload();

void onExtract();

void onSave();

void onAbout();

wstring loadString(UINT);

vector<Mat> loadTemplates(UINT, UINT);

vector<wstring> loadDictionary(wstring);

void rewriteDictionary(const vector<wstring>&, wstring);

void setDefaultImage();

void setDefaultText();

// MAIN.CPP

#include "framework.h"

#include "main.h"

HINSTANCE hInst;

WCHAR szTitle[MAX\_LOADSTRING];

WCHAR szWindowClass[MAX\_LOADSTRING];

// Descriptors

HWND hMainWnd, // Main window

hPctBox, // Picture box to load image

hTxtFld, // Text field for recognized text

hBtnLoad, hBtnUnload, hBtnExtract, hBtnSave, // Buttons

hCbDrawBounds, hCbPostProcess; // Checkboxes

Bitmap\* image = nullptr; // Image to extract text from

Bounds\* bounds = nullptr;

vector<Mat> templates; // Templates to match with

wstring labels; // Templates' labels

vector<wstring> dict; // Dictionary for postprocessing

int APIENTRY wWinMain(\_In\_ HINSTANCE hInstance,

\_In\_opt\_ HINSTANCE hPrevInstance,

\_In\_ LPWSTR lpCmdLine,

\_In\_ int nCmdShow)

{

UNREFERENCED\_PARAMETER(hPrevInstance);

UNREFERENCED\_PARAMETER(lpCmdLine);

// Init GDI+

GdiplusStartupInput gdiplusStartupInput;

ULONG\_PTR gdiplusToken;

if (GdiplusStartup(&gdiplusToken, &gdiplusStartupInput, NULL) != Ok)

return FALSE;

// Init COM

HRESULT hr = CoInitializeEx(NULL, COINIT\_APARTMENTTHREADED | COINIT\_DISABLE\_OLE1DDE);

if (!SUCCEEDED(hr))

{

GdiplusShutdown(gdiplusToken);

return FALSE;

}

LoadStringW(hInstance, IDS\_APP\_TITLE, szTitle, MAX\_LOADSTRING);

LoadStringW(hInstance, IDC\_TEXTRECOGNITION, szWindowClass, MAX\_LOADSTRING);

MyRegisterClass(hInstance);

if (!InitInstance(hInstance, nCmdShow))

return FALSE;

HACCEL hAccelTable = LoadAccelerators(hInstance, MAKEINTRESOURCE(IDC\_TEXTRECOGNITION));

MSG msg;

while (GetMessage(&msg, nullptr, 0, 0))

{

if (!TranslateAccelerator(msg.hwnd, hAccelTable, &msg))

{

TranslateMessage(&msg);

DispatchMessage(&msg);

}

}

// Close COM, GDI+

CoUninitialize();

GdiplusShutdown(gdiplusToken);

return (int)msg.wParam;

}

// Register main window class

ATOM MyRegisterClass(HINSTANCE hInstance)

{

WNDCLASSEXW wcex;

wcex.cbSize = sizeof(WNDCLASSEX);

wcex.style = CS\_HREDRAW | CS\_VREDRAW;

wcex.lpfnWndProc = WndProc;

wcex.cbClsExtra = 0;

wcex.cbWndExtra = 0;

wcex.hInstance = hInstance;

wcex.hIcon = LoadIcon(hInstance, MAKEINTRESOURCE(IDI\_TEXTRECOGNITION));

wcex.hCursor = LoadCursor(nullptr, IDC\_ARROW);

wcex.hbrBackground = (HBRUSH)(COLOR\_WINDOW + 1);

wcex.lpszMenuName = MAKEINTRESOURCEW(IDC\_TEXTRECOGNITION);

wcex.lpszClassName = szWindowClass;

wcex.hIconSm = LoadIcon(wcex.hInstance, MAKEINTRESOURCE(IDI\_SMALL));

return RegisterClassExW(&wcex);

}

// Initialise app's instance and create main window

BOOL InitInstance(HINSTANCE hInstance, int nCmdShow)

{

hInst = hInstance;

hMainWnd = CreateWindowW(szWindowClass, szTitle,

WS\_OVERLAPPEDWINDOW, CW\_USEDEFAULT, 0, CW\_USEDEFAULT, 0,

nullptr, nullptr, hInstance, nullptr);

if (!hMainWnd)

return FALSE;

ShowWindow(hMainWnd, nCmdShow);

UpdateWindow(hMainWnd);

return TRUE;

}

LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)

{

switch (message)

{

case WM\_CREATE:

{

hMainWnd = hWnd;

onCreate();

break;

}

case WM\_SIZE:

{

onSize();

break;

}

case WM\_GETMINMAXINFO:

{

// Set min size of the window

LPMINMAXINFO lpMMI = (LPMINMAXINFO)lParam;

lpMMI->ptMinTrackSize.x = MIN\_WND\_WIDHT;

lpMMI->ptMinTrackSize.y = MIN\_WND\_HEIGHT;

break;

}

case WM\_CTLCOLORSTATIC:

{

int id = GetDlgCtrlID((HWND)lParam);

if (id == IDC\_CB\_DRAW || id == IDC\_CB\_POSTPROC || id == IDC\_PCT\_BOX)

return (LRESULT)GetStockObject(WHITE\_BRUSH);

break;

}

case WM\_COMMAND:

{

int id = LOWORD(wParam);

int code = HIWORD(wParam);

switch (id)

{

case IDM\_ABOUT:

onAbout();

break;

case IDM\_EXIT:

DestroyWindow(hWnd);

break;

case IDM\_OPEN:

onLoad();

break;

case IDM\_SAVE:

onSave();

break;

case IDM\_CLOSE:

onUnload();

break;

case IDC\_BTN\_LOAD:

if (code == BN\_CLICKED)

onLoad();

break;

case IDC\_BTN\_UNLOAD:

if (code == BN\_CLICKED)

onUnload();

break;

case IDC\_BTN\_EXTRACT:

if (code == BN\_CLICKED)

onExtract();

break;

case IDC\_BTN\_SAVE:

if (code == BN\_CLICKED)

onSave();

break;

case IDC\_PCT\_BOX:

if (code == STN\_DBLCLK)

onLoad();

break;

default:

return DefWindowProc(hWnd, message, wParam, lParam);

}

}

break;

case WM\_DESTROY:

onUnload();

PostQuitMessage(0);

break;

default:

return DefWindowProc(hWnd, message, wParam, lParam);

}

return 0;

}

void redrawImage(const Bounds\* bounds = nullptr)

{

if (image)

{

RECT r;

GetClientRect(hPctBox, &r);

int w = image->GetWidth(), h = image->GetHeight();

// Image must fit biggest size of picture box

double scaleX = WIDTH(r) / (double)w;

double scaleY = HEIGHT(r) / (double)h;

double scale = min(scaleX, scaleY);

if (scale < 1)

{

// Picture doesn't fit

w \*= scale;

h \*= scale;

}

else

scale = 1; // Picture fits & there are spaces

// Scale image

Bitmap output(w, h);

Graphics g(&output);

g.ScaleTransform(scale, scale);

g.DrawImage(image, Rect(0, 0, image->GetWidth(), image->GetHeight()));

// Draw bounds

if (bounds && IsDlgButtonChecked(hMainWnd, IDC\_CB\_DRAW))

{

Pen letterPen(Color(255, 0, 255, 0), 1);

for (const auto& line : \*bounds)

for (const auto& word : line)

for (const auto& letter : word)

g.DrawRectangle(&letterPen, letter);

}

// Reset image on static control

output.SetPixel(0, 0, Color(255, 255, 255));

HBITMAP hBmp;

if (output.GetHBITMAP(0, &hBmp) == Ok)

SendMessageW(hPctBox, STM\_SETIMAGE, IMAGE\_BITMAP, (LPARAM)hBmp);

else

{

onUnload();

}

}

}

// WM\_CREATE handler: creates windows, loads recognition data

void onCreate()

{

// Load recognition data

labels = loadString(IDS\_LABELS);

templates = loadTemplates(IDB\_BITMAP1, labels.size());

dict = loadDictionary(L"dictionary.txt");

rewriteDictionary(dict, L"dictionary.txt");

// Create picture box

hPctBox = CreateWindowW(L"STATIC", nullptr, PBOX\_STYLE,

0, 0, 0, 0, hMainWnd, (HMENU)IDC\_PCT\_BOX, hInst, nullptr);

setDefaultImage();

// Create text field

LoadLibraryW(L"Msftedit.dll");

hTxtFld = CreateWindowExW(0, MSFTEDIT\_CLASS, nullptr, TXTFLD\_STYLE,

0, 0, 0, 0, hMainWnd, nullptr, hInst, nullptr);

setDefaultText();

// Create buttons

hBtnLoad = CreateWindowW(L"BUTTON", L"Открыть",

WS\_VISIBLE | WS\_CHILD | BS\_DEFPUSHBUTTON,

0, 0, 0, 0, hMainWnd, (HMENU)IDC\_BTN\_LOAD, hInst, nullptr);

hBtnUnload = CreateWindowW(L"BUTTON", L"Закрыть",

WS\_VISIBLE | WS\_CHILD | BS\_DEFPUSHBUTTON,

0, 0, 0, 0, hMainWnd, (HMENU)IDC\_BTN\_UNLOAD, hInst, nullptr);

hBtnExtract = CreateWindowW(L"BUTTON", L"Извлечь текст",

WS\_VISIBLE | WS\_CHILD | BS\_DEFPUSHBUTTON,

0, 0, 0, 0, hMainWnd, (HMENU)IDC\_BTN\_EXTRACT, hInst, nullptr);

hBtnSave = CreateWindowW(L"BUTTON", L"Сохранить",

WS\_VISIBLE | WS\_CHILD | BS\_DEFPUSHBUTTON,

0, 0, 0, 0, hMainWnd, (HMENU)IDC\_BTN\_SAVE, hInst, nullptr);

// Create checkboxes

hCbDrawBounds = CreateWindowW(L"BUTTON", L"Рисовать границы",

WS\_VISIBLE | WS\_CHILD | BS\_AUTOCHECKBOX,

0, 0, 0, 0, hMainWnd, (HMENU)IDC\_CB\_DRAW, hInst, nullptr);

hCbPostProcess = CreateWindowW(L"BUTTON", L"Постобработка",

WS\_VISIBLE | WS\_CHILD | BS\_AUTOCHECKBOX,

0, 0, 0, 0, hMainWnd, (HMENU)IDC\_CB\_POSTPROC, hInst, nullptr);

SendMessageW(hCbPostProcess, BM\_SETCHECK, BST\_CHECKED, 0);

}

// WM\_SIZE handler: moves windows

void onSize()

{

// Initialise coordinates

RECT r;

GetClientRect(hMainWnd, &r);

int x, y = 0, w, h = HEIGHT(r);

int dw = max((BTN\_WIDTH - S\_OPTIONS \* WIDTH(r)) / 2, 0);

// Move picture box

x = 0;

w = (int)(S\_PBOX \* WIDTH(r)) - dw;

MoveWindow(hPctBox, x, y, w, h, TRUE);

// Move text field

x += w;

w = (int)(S\_TEXT \* WIDTH(r)) - dw;

MoveWindow(hTxtFld, x, y, w, h, TRUE);

// Move buttons

x += w + (WIDTH(r) - x - w - BTN\_WIDTH) / 2;

y = BTN\_GAP;

w = BTN\_WIDTH;

h = BTN\_HEIGHT;

MoveWindow(hBtnLoad, x, y, w, h, TRUE);

y += BTN\_GAP + h;

MoveWindow(hBtnUnload, x, y, w, h, TRUE);

y += BTN\_GAP + h;

MoveWindow(hBtnExtract, x, y, w, h, TRUE);

y += BTN\_GAP + h;

MoveWindow(hBtnSave, x, y, w, h, TRUE);

y += BTN\_GAP + h + h;

MoveWindow(hCbDrawBounds, x, y, w, h, TRUE);

y += BTN\_GAP + h;

MoveWindow(hCbPostProcess, x, y, w, h, TRUE);

redrawImage(bounds);

}

// hBtnLoad click handler: call open dialog and loads image

void onLoad()

{

// Init input and output structure for dialog

OPENFILENAME ofn = { 0 };

wchar\_t szFilename[256] = L"";

ofn.lStructSize = sizeof(OPENFILENAME);

ofn.hwndOwner = hMainWnd;

ofn.lpstrFilter = L"Изображения (\*.bmp, \*.jpg, \*.png)\0\*.bmp;\*.jpg;\*.png\0";

ofn.lpstrTitle = L"Открыть";

ofn.lpstrFile = szFilename;

ofn.nMaxFile = 256;

ofn.Flags = OFN\_PATHMUSTEXIST | OFN\_FILEMUSTEXIST;

// Open file dialog and get file name

if (GetOpenFileName(&ofn))

{

onUnload(); // Reset text and image

image = new Bitmap(ofn.lpstrFile); // Load image

redrawImage();

}

}

// hBtnUnload click handler: reset text and image

void onUnload()

{

if (image)

{

// Load default image and text

setDefaultImage();

setDefaultText();

// Reset pointers

delete image;

image = nullptr;

if (bounds)

{

delete bounds;

bounds = nullptr;

}

}

}

// hBtnExtract click handler: recognises text from image

void onExtract()

{

if (image)

{

wstring text;

try

{

auto mat = getGrayscale(\*image);

threshold(mat, 175);

if (bounds)

{

delete bounds;

bounds = nullptr;

}

bounds = new Bounds(getBounds(mat));

// Optional output

if (IsDlgButtonChecked(hMainWnd, IDC\_CB\_DRAW))

redrawImage(bounds);

else

redrawImage(nullptr);

text = recognise(mat, \*bounds, labels, templates);

if (IsDlgButtonChecked(hMainWnd, IDC\_CB\_POSTPROC))

text = correctText(text, dict);

}

catch (exception& e)

{

text = L"Ошибка при распознавании...";

}

SendMessageW(hTxtFld, WM\_SETTEXT, 0, (LPARAM)text.c\_str()); // Output

}

}

// hBtnSave click handler: saves text from field to file

void onSave()

{

if (image)

{

// Init input and output structure for dialog

OPENFILENAME ofn = { 0 };

wchar\_t szFilename[256] = L"";

ofn.lStructSize = sizeof(OPENFILENAME);

ofn.hwndOwner = hMainWnd;

ofn.lpstrFilter = L"Текстовые файлы (\*.txt)\0\*.txt\0";

ofn.lpstrTitle = L"Сохранить как";

ofn.lpstrFile = szFilename;

ofn.nMaxFile = 256;

ofn.Flags = OFN\_HIDEREADONLY | OFN\_PATHMUSTEXIST;

ofn.lpstrDefExt = L"txt";

// Open file dialog

if (GetSaveFileName(&ofn))

{

// Create stream and set conversion

wofstream fs(ofn.lpstrFile);

fs.imbue(locale(locale::empty(), new codecvt\_utf8<wchar\_t>));

// Get text from field

WCHAR\* buffer = 0;

int nLen = GetWindowTextLength(hTxtFld);

if (nLen >= 0)

{

buffer = new WCHAR[nLen + 1];

buffer[nLen] = L'\0';

if (GetWindowText(hTxtFld, buffer, nLen + 1))

{

// Save to file without '\r'

wstring text(buffer);

text.erase(remove(text.begin(), text.end(), L'\r'), text.end());

fs << text;

}

delete[] buffer;

}

fs.close();

}

}

}

// IDM\_ABOUT handle: calls helper

void onAbout()

{

wstring x = L"Данное приложение предназначено для распознавания печатного текста шрифта Arial с фотографий формата BMP, JPG, PNG.";

x += L"\n\nЭтап препроцессинга фотографии состоит только из бинаризации изображения, поэтому не гарантируется правильное распознавание текста с фотографий с различными искажениями : шумами, перепадами освещения, поворотами и т.д.";

x += L"\n\nПриложение опускает этап детектирования текста, полагая, что вся фотография состоит из цельного текста структуры : строки->слова->буквы.";

x += L"\n\nСегментация цельного текста основана на методе проекционных профилей : проекционный профиль - массив сумм яркостей пикселей / числа пикселей переднего плана, вычисленная в определённом столбце или строке в зависимости от вида профиля.";

x += L"\n\nБлок классификации использует метод сопоставления с шаблоном, который попиксельно сравнивает часть фотографии с заданными шаблонами.Фотографии шаблонов являются ресурсами приложения.";

x += L"\n\nПостпроцессинг делает все буквы строчными и использует метрику Левенштейна для замены слов с орфографическими ошибками на корректные.Словарь составляется из файла dictionary.txt";

MessageBoxW(hMainWnd, x.c\_str(), L"Справка", MB\_OK);

}

// Load string from resource

wstring loadString(UINT ID)

{

// Get pointer to readonly resource

WCHAR\* pBuf = nullptr;

int len = LoadStringW(hInst, ID, reinterpret\_cast<LPWSTR>(&pBuf), 0);

// Type conversion

if (len)

return wstring(pBuf, len);

else

return wstring();

}

// Load bitmaps from resources and converts to bin matrices

vector<Mat> loadTemplates(UINT startId, UINT templatesCount)

{

vector<Mat> templates(templatesCount);

// Resources must have sequential identifiers

for (UINT i = 0; i < templatesCount; i++)

{

Bitmap\* pBmp = Bitmap::FromResource(hInst, MAKEINTRESOURCE(startId + i));

templates[i] = getGrayscale(\*pBmp);

threshold(templates[i], 175);

delete pBmp;

}

return templates;

}

// Load words from utf8 file and sorts them in

// lexicographic order

vector<wstring> loadDictionary(wstring fileName)

{

// Create stream and set conversion

wifstream ifs(fileName);

ifs.imbue(locale(locale::empty(), new codecvt\_utf8<wchar\_t>));

// Read file

vector<wstring> dict;

wstring word;

while (ifs >> word)

dict.push\_back(word);

ifs.close();

// Sort strings

sort(dict.begin(), dict.end());

return dict;

}

// Writes dictionary to file in utf8

void rewriteDictionary(const vector<wstring>& dict, wstring fileName)

{

// Create stream and set conversion

wofstream ofs(fileName);

ofs.imbue(locale(locale::empty(), new codecvt\_utf8<wchar\_t>));

for (auto& word : dict)

ofs << word << endl;

ofs.close();

}

// Set default image

void setDefaultImage()

{

// Load default image

Bitmap\* pBmp = Bitmap::FromResource(hInst,

MAKEINTRESOURCE(IDB\_BITMAP1 + templates.size()));

// Get handler

HBITMAP hBmp;

if (pBmp->GetHBITMAP(0, &hBmp) == Ok) // Extract handle

SendMessageW(hPctBox, STM\_SETIMAGE, IMAGE\_BITMAP, (LPARAM)hBmp); // Set image

delete pBmp; // Free memory

}

// Set default text and font

void setDefaultText()

{

// Set font

HFONT hFont = CreateFontW(20, 0, 0, 0,

FW\_NORMAL, FALSE, FALSE, FALSE, RUSSIAN\_CHARSET,

OUT\_OUTLINE\_PRECIS, CLIP\_DEFAULT\_PRECIS, CLEARTYPE\_QUALITY,

VARIABLE\_PITCH, L"Arial");

SendMessageW(hTxtFld, WM\_SETFONT, (WPARAM)hFont, (LPARAM)FALSE);

DeleteObject(hFont); // Free memory

// Set text

WCHAR defaultText[] = L"Извлечённый текст будет отображён здесь...";

SendMessageW(hTxtFld, WM\_SETTEXT, 0, (LPARAM)defaultText);

}

Приложение Б

Исходный код модуля предобработки

(preprocessing.h и preprocessing.cpp)

// PREPROCESSING.H

#pragma once

#include "framework.h"

/\* Extracts grayscale matrix from bitmap

\*

\* param bitmap: Source bitmap

\* returns: 2D matrix of grayscale intensities

\*/

Mat getGrayscale(Bitmap& image);

/\* Binarize image matrix

\* set 1 for all that lesser than threshold and

\* 0 for all that are higher or equal

\*

\* param m: Grayscale matrix of image

\* param thresh: Threshold for binarization

\*/

void threshold(Mat& I, byte thresh);

//PREPROCESSING.CPP

#include "preprocessing.h"

Mat getGrayscale(Bitmap& bitmap)

{

// Init data

int pixelSize = 3,

w = bitmap.GetWidth(),

h = bitmap.GetHeight();

Rect area(0, 0, w, h);

// Lock bits from bitmap area into system memory

BitmapData bmpData;

if (bitmap.LockBits(&area, ImageLockModeRead, PixelFormat24bppRGB, &bmpData) != Ok)

return Mat(0);

// Loop over image and convert to grayscale matrix

Mat grayscale(h, vector<byte>(w));

for (int i = 0; i < h; i++)

{

// Stride can be negative for bottom-up bitmaps

byte\* bytes = (byte\*)bmpData.Scan0 + (i \* bmpData.Stride);

// Loop over scan line

for (int j = 0; j < w; j++)

{

//Get each colour component

byte r = bytes[pixelSize \* j];

byte g = bytes[pixelSize \* j + 1];

byte b = bytes[pixelSize \* j + 2];

// Conver to grayscale

byte intensity = round(0.299 \* r + 0.587 \* g + 0.114 \* b);

// Assign grayscale matrix

grayscale[i][j] = intensity;

}

}

// Unlock bitmap from system memory

bitmap.UnlockBits(&bmpData);

return grayscale;

}

void threshold(Mat& I, byte thresh)

{

for (auto& row : I)

for (auto& px : row)

if (px < thresh)

px = 1;

else

px = 0;

}

Приложение В

Исходный код модуля сегментации

(segmentation.h и segmentation.cpp)

// SEGMENTATION.H

#pragma once

#include "framework.h"

/\* Extract letter bounding boxes from image

\* param I: Matrix of image

\* returns: 3D vector of bounding boxes of letters

\*/

Bounds getBounds(const Mat& I);

// SEGMENTATION.CPP

#include "segmentation.h"

// Axises of image

typedef bool Axis;

#define Ox true

#define Oy false

/\* Get projection profile of image region

\* Ox - horizontal profile(sums along rows),

\* Oy - vertical profile(sums along columns)

\*

\* param I: Matrix of image

\* param r: Region of image

\* param axis: Direction of summation

\* returns: Profile projection

\*/

vector<int> getProfile(const Mat& I, Rect r, Axis axis)

{

// Init profile vector

int size = axis == Ox ? r.Height : r.Width;

vector<int> pp(size);

int m = r.Y + r.Height, n = r.X + r.Width;

// Loop along y-axis

for (int i = r.Y; i < m; i++)

{

// Loop along x-axis

for (int j = r.X; j < n; j++)

{

// Get index of current element in vector

int idx = axis == Ox ? i - r.Y : j - r.X;

// Update sum

pp[idx] += I[i][j];

}

}

return pp;

}

/\* Segmentation along given axis

\* param I: Matrix of image

\* param r: Image bounds

\* param axis: Axis

\* returns: Bounds of found images

\*/

vector<Rect> splitBound(const Mat& I, Rect r, Axis axis)

{

Rect bound;

vector<Rect> bounds;

// Get profile projection

auto pp = getProfile(I, r, !axis);

// Init coordinates of text segment and size of profile

int start = -1, end, n = pp.size();

// Loop over profile projection

for (int i = 0; i < n; i++)

{

if (pp[i] && start == -1)

start = i; // Save start coordinate of segment

else if (pp[i] == 0 && start != -1)

{

end = i - 1; // Save end coordinate of segment

// Init bounding box

bound = r;

if (axis == Ox)

{

// Change x-coordinates

bound.X = start;

bound.Width = end - start;

}

else

{

// Change y-coordinates

bound.Y = start;

bound.Height = end - start;

}

// Push to collection of all bounds

bounds.push\_back(bound);

// Reset start of text segment

start = -1;

}

}

// Image ends with black pixels

if (start != -1)

{

end = n - 1; // Set end coordinate of segment

// Init bound box

bound = r;

if (axis == Ox)

{

// Change x-coordinates

bound.X = start;

bound.Width = end - start;

}

else

{

// Change y-coordinates

bound.Y = start;

bound.Height = end - start;

}

// Push to collection of all bounds

bounds.push\_back(bound);

}

return bounds;

}

/\*Calculates mean size of bounds along given axis

\* param images: 1D vector of bounds

\* param axis: Given axis

\* returns: Mean height for Oy and width for Ox

\*/

int getMeanSize(const vector<Rect>& bounds, Axis axis)

{

int sum = 0, size;

for (auto& bound : bounds)

{

// Get size along axis

size = axis == Ox ? bound.Width : bound.Height;

sum += size; // Add size to sum

}

// Max with 1 is used for zero division

return sum / max(1, bounds.size());

}

/\*Group bounding boxes

\* param I: Matrix of image

\* param letters: 1D vecotr of bounds

\* returns: 2D vector of bounds

\*/

vector<vector<Rect>> groupBounds(const Mat& I,

const vector<Rect>& bounds)

{

// Init groups

vector<vector<Rect>> groups;

if (bounds.size() == 0)

return groups;

groups.push\_back(vector<Rect>());

// Get threshold of distance (min space size)

int thresh = (int)(0.7 \* getMeanSize(bounds, Ox));

// Iterate over bounds until pre-last image

for (auto it = bounds.begin(); (it + 1) != bounds.end(); it++)

{

// Get distance between bounds

Rect bounds = \*it, nextBounds = \*(it + 1);

int distance = nextBounds.X - bounds.X - bounds.Width;

// Add another bound to the group

groups[groups.size() - 1].push\_back(\*it);

if (distance >= thresh)

{

// Create a new group

groups.push\_back(vector<Rect>());

}

}

// Add last bound to the group

groups[groups.size()-1].push\_back(bounds[bounds.size() - 1]);

return groups;

}

/\*Filters blank rows from bounding boxes

\* param I: Matrix of image

\* param rect: Bounding boxes

\*/

void cropRows(const Mat& I, vector<Rect>& rects)

{

for (auto& r : rects)

{

// Get projection profile

auto hpp = getProfile(I, r, Ox);

// Skip top white rows

int i = 0;

while (i < hpp.size() && hpp[i] == 0)

{

r.Y++;

r.Height--;

i++;

}

// Skip bottom white rows

i = hpp.size() - 1;

while (i >= 0 && hpp[i] == 0)

{

r.Height--;

i--;

}

}

}

/\* Erases small bounds from vector

\* param bounds: Bounds to filter

\* param axis: Axis to get size

\* para, threshold: Threshold of size filtering

\*/

void filterBounds(vector<Rect>& bounds,

Axis axis,

int threshold)

{

for (auto it = bounds.begin(); it != bounds.end();)

{

int size = axis == Ox ? it->Width : it->Height;

if (size < threshold)

it = bounds.erase(it); // Filter

if (it != bounds.end())

{

// Avoid incrementing vector's end

it++;

}

}

}

Bounds getBounds(const Mat& I)

{

Bounds bounds;

if (I.size() == 0 || I[0].size() == 0)

return bounds;

// Extract lines, words and letters

auto lines = splitBound(I, Rect(0, 0, I[0].size(), I.size()), Oy);

filterBounds(lines, Oy, 0.3 \* getMeanSize(lines, Oy));

for (auto& line : lines)

{

auto letters = splitBound(I, line, Ox);

cropRows(I, letters);

int meanWidth = getMeanSize(letters, Ox);

auto words = groupBounds(I, letters);

for (auto& subLetters : words)

{

filterBounds(subLetters, Ox, 0.6 \* meanWidth);

filterBounds(subLetters, Oy, 0.6 \* meanWidth);

}

if (words.size() > 0)

bounds.push\_back(words);

}

return bounds;

}

Приложение Г

Исходный код модуля классификации

(recognition.h и recognition.cpp)

// RECOGNITION.H

#pragma once

#include "framework.h"

/\* Recognises letters from image regions and

\* combines them into 1 string

\* param I: Matrix of image

\* param bounds: Bounds of letters

\* param labels: Labels of each template

\* param templates: Templates to compare image with

\* returns: Extracted text

\*/

wstring recognise(const Mat& I,

const Bounds& bounds,

wstring labels,

const vector<Mat>& templates);

// RECOGNITION.CPP

#include "recognition.h"

/\* Creates new matrix from submatrix

\* param src: Source matrix to get info from

\* param r: Region to copy

\* returns: Copied submatrix

\*/

Mat copyMat(const Mat& src, Rect r)

{

Mat dst(r.Height, vector<byte>(r.Width));

for (int i = r.Y; i < r.Y + r.Height; i++)

for (int j = r.X; j < r.X + r.Width; j++)

dst[i - r.Y][j - r.X] = src[i][j];

return dst;

}

/\* Pixelwise comparison of 2 images

\* Images can have different sizes:

\* scaling is done in place when accessing

\* another pixel

\* param X: 1st matrix

\* param Y: 2nd matrix

\* returns: Similiarity rate in [0, 1] section

\*/

double match(const Mat& X, const Mat& Y)

{

int h = X.size(), w = X[0].size(), matches = 0;

double yScale = Y.size() / (double)h,

xScale = Y[0].size() / (double)w;

for (int i = 0; i < h; i++)

{

for (int j = 0; j < w; j++)

{

if (X[i][j] == Y[(int)(yScale \* i)][(int)(xScale \* j)])

matches++;

}

}

return (double)matches / w / h;

}

/\* Classify image by given templates and labels

\* param I: Matrix of image

\* param labels: Labels of each template

\* param templates: Templates to compare image with

\* returns: Index of max match in labels or -1 if it was low

\*/

int classify(const Mat& I,

wstring labels,

const vector<Mat>& templates)

{

double max = 0;

int maxIdx = -1;

for (int i = 0; i < templates.size(); i++)

{

double similiarity = match(I, templates[i]);

if (similiarity > max && similiarity > 0.6)

{

maxIdx = i;

max = similiarity;

}

}

return maxIdx;

}

wstring recognise(const Mat& I,

const Bounds& bounds,

wstring labels,

const vector<Mat>& templates)

{

wstring text;

for (int i = 0; i < bounds.size(); i++)

{

for (int j = 0; j < bounds[i].size(); j++)

{

for (int k = 0; k < bounds[i][j].size(); k++)

{

Mat letterImage =

copyMat(I, bounds[i][j][k]);

int idx =

classify(letterImage, labels, templates);

if (idx >= 0)

text += labels[idx];

}

text += L" ";

}

text += L"\n";

}

return text;

}

Приложение Д

Исходный код модуля постобработки

(postprocessing.h и postprocessing.cpp)

// POSTPROCESSING.H

#pragma once

#include "framework.h"

/\* Correct given text by dictionary

\* param text: Given text with misspellings

\* param dict: Given dictionary with correct words

\* returns: Corrected text

\*/

wstring correctText(wstring text, const vector<wstring>& dict);

//POSTPROCESSING.CPP

#include "postprocessing.h"

#define minimum(a, b, c) ( min( (a), min( (b), (c) ) ) )

#include <cwctype>

/\* Calculate Levenstein distance of 2 strings

\* param s1: First string

\* param s2: Second string

\* returns: Edit distance

\*/

int lev(wstring s1, wstring s2)

{

int m = s1.size(), n = s2.size(); // Sizes of matrix

vector<int> v0(n + 1), v1(n + 1); // Previous and current rows

// Init 1st row of matrix

for (size\_t i = 0; i < n + 1; i++)

v0[i] = i;

for (size\_t i = 0; i < m; i++)

{

v1[0] = i + 1; // Init 1st column of matrix

for (size\_t j = 0; j < n; j++)

{

// Calculate costs of transformations

int del = v0[j + 1] + 1;

int ins = v1[j] + 1;

int sub = v0[j] + (s1[i] != s2[j]);

// Calculate next element

v1[j + 1] = minimum(del, ins, sub);

}

swap(v0, v1); // Set previous row to current

}

return v0[n];

}

/\* Split string into tokens by delimiter

\* param text: Given string

\* param delim: Given delimiter

\* returns: 1D vector of tokens

\*/

vector<wstring> split(wstring text, wchar\_t delim)

{

wstring token;

vector<wstring> tokens;

wistringstream ss(text);

while (getline(ss, token, delim))

tokens.push\_back(token);

return tokens;

}

/\* Change each letter's case in lower case

\* param word: Given string to convert

\* returns: Same string in lowercase

\*/

wstring towlower(wstring s)

{

for (auto& letter : s)

letter = towlower(letter);

return s;

}

wstring correctText(wstring text, const vector<wstring>& dict)

{

setlocale(LC\_ALL, "ru-RU"); // Locale to change case

wstring correctedText; // Output

vector<wstring> words, lines; // Substrings of text

lines = split(text, L'\n'); // Split text into lines

for (const auto& line : lines)

{

words = split(line, L' '); // Split into words

for (auto& word : words)

{

word = towlower(word); // Convert to lower

// Find min Levenshtein distance

int minDist = MAXINT, minIdx = -1;

for (int k = 0; k < dict.size(); k++)

{

int dist = lev(word, towlower(dict[k]));

if (dist < minDist)

{

minDist = dist;

minIdx = k;

}

}

// Correct word if distance is sufficient

if (minIdx >= 0 && minDist <= word.size() / 2)

word = dict[minIdx];

correctedText += word + L" ";

}

correctedText += L"\n";

}

return correctedText;

}

Приложение Е

Исходный код других заголовочных файлов

(framework.h, Resource.h, targetver.h)

// FRAMEWORK.H

#pragma once

#include "targetver.h"

#define WIN32\_LEAN\_AND\_MEAN

// WinApi

#include <windows.h>

// Files and strings

#include <string>

#include <sstream>

// Collections and algorithms

#include <vector>

#include <algorithm>

// COM

#include <objidl.h>

#include <shobjidl.h>

// GDI+

#include <gdiplus.h>

#pragma comment (lib, "Gdiplus.lib")

// Namespaces

using namespace Gdiplus;

using namespace std;

typedef vector<vector<byte>> Mat;

typedef vector<vector<vector<Rect>>> Bounds;

// RESOURCE.H

#define IDC\_MYICON 2

#define IDC\_BTN\_LOAD 3

#define IDC\_BTN\_UNLOAD 4

#define IDC\_BTN\_EXTRACT 5

#define IDC\_PCT\_BOX 6

#define IDC\_BTN\_SAVE 7

#define IDC\_CB\_DRAW 8

#define IDC\_CB\_POSTPROC 9

#define IDD\_TEXTRECOGNITION\_DIALOG 102

#define IDS\_APP\_TITLE 103

#define IDD\_ABOUTBOX 103

#define IDM\_ABOUT 104

#define IDS\_STRING104 104

#define IDM\_EXIT 105

#define IDI\_TEXTRECOGNITION 107

#define IDI\_SMALL 108

#define IDC\_TEXTRECOGNITION 109

#define IDS\_LABELS 110

#define IDM\_OPEN 111

#define IDM\_SAVE 112

#define IDM\_CLOSE 113

#define IDR\_MAINFRAME 128

#define IDB\_BITMAP1 356

#define IDB\_BITMAP2 357

#define IDB\_BITMAP3 358

#define IDB\_BITMAP4 359

#define IDB\_BITMAP5 360

#define IDB\_BITMAP6 361

#define IDB\_BITMAP7 362

#define IDB\_BITMAP8 363

#define IDB\_BITMAP9 364

#define IDB\_BITMAP10 365

#define IDB\_BITMAP11 366

#define IDB\_BITMAP12 367

#define IDB\_BITMAP13 368

#define IDB\_BITMAP14 369

#define IDB\_BITMAP15 370

#define IDB\_BITMAP16 371

#define IDB\_BITMAP17 372

#define IDB\_BITMAP18 373

#define IDB\_BITMAP19 374

#define IDB\_BITMAP20 375

#define IDB\_BITMAP21 376

#define IDB\_BITMAP22 377

#define IDB\_BITMAP23 378

#define IDB\_BITMAP24 379

#define IDB\_BITMAP25 380

#define IDB\_BITMAP26 381

#define IDB\_BITMAP27 382

#define IDB\_BITMAP28 383

#define IDB\_BITMAP29 384

#define IDB\_BITMAP30 385

#define IDB\_BITMAP31 386

#define IDB\_BITMAP32 387

#define IDB\_BITMAP33 388

#define IDB\_BITMAP34 389

#define IDB\_BITMAP35 390

#define IDB\_BITMAP36 391

#define IDB\_BITMAP37 392

#define IDB\_BITMAP38 393

#define IDB\_BITMAP39 394

#define IDB\_BITMAP40 395

#define IDB\_BITMAP41 396

#define IDB\_BITMAP42 397

#define IDB\_BITMAP43 398

#define IDB\_BITMAP44 399

#define IDB\_BITMAP45 400

#define IDB\_BITMAP46 401

#define IDB\_BITMAP47 402

#define IDB\_BITMAP48 403

#define IDB\_BITMAP49 404

#define IDB\_BITMAP50 405

#define IDB\_BITMAP51 406

#define IDB\_BITMAP52 407

#define IDB\_BITMAP53 408

#define IDB\_BITMAP54 409

#define IDB\_BITMAP55 410

#define IDB\_BITMAP56 411

#define IDB\_BITMAP57 412

#define IDB\_BITMAP58 413

#define IDB\_BITMAP59 414

#define IDB\_BITMAP60 415

#define IDB\_BITMAP61 416

#define IDB\_BITMAP62 417

#define IDB\_BITMAP63 418

#define IDB\_BITMAP64 419

#define IDB\_BITMAP65 420

#define IDB\_BITMAP66 421

#define IDB\_BITMAP67 422

#define IDC\_STATIC -1

// Next default values for new objects

#ifdef APSTUDIO\_INVOKED

#ifndef APSTUDIO\_READONLY\_SYMBOLS

#define \_APS\_NO\_MFC 1

#define \_APS\_NEXT\_RESOURCE\_VALUE 424

#define \_APS\_NEXT\_COMMAND\_VALUE 32771

#define \_APS\_NEXT\_CONTROL\_VALUE 1000

#define \_APS\_NEXT\_SYMED\_VALUE 110

#endif

#endif

// TARGETVER.H

#pragma once

#include <SDKDDKVer.h>