МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г.ЧЕРНЫШЕВСКОГО»

Кафедра информатики и программирования

РАЗРАБОТКА МОБИЛЬНОГО ПРИЛОЖЕНИЯ-СКАНЕРА С ФУНКЦИЕЙ РАСПОЗНАВАНИЯ ПЕЧАТНОГО ТЕКСТА НА СЛОЖНОМ ФОНЕ

АВТОРЕФЕРАТ МАГИСТЕРСКОЙ РАБОТЫ

Студентки 2 курса 273 группы
направления 02.04.03 Математическое обеспечение и администрирование информационных систем
факультета компьютерных наук и информационных технологий
Талалайкиной Елизаветы Игоревны

Научный руководитель:
зав. кафедрой, к.ф.-м.н., доцент. _____ М.В. Огнева

подпись, дата

Зав. кафедрой:
к.ф.-м.н., доцент _____ М.В. Огнева

подпись, дата

Саратов 2022

ВВЕДЕНИЕ

В современном мире перевод информации, представленной на бумажных носителях, в электронный вид является задачей, актуальной не только для отдельно взятых пользователей, но и целых ведомств и организаций. Электронное представление документов является удобной альтернативной большому количеству бумаг и документов, находящихся по тем или иным причинам на обязательном хранении. Преимуществ у такого подхода достаточно: высокая скорость обработки документов, возможность интеграции с внешними ресурсами, информационными системами, мобильными приложениями, быстрота документооборота между организациями, возможность создания резервных копий документов, а следовательно, меньшая угроза их разрушения со временем или при неправильном хранении. Для отдельно взятых пользователей хранение электронных копий документов, например, на смартфоне означает удобный доступ к ним в отсутствие бумажного оригинала.

Повсеместный переход от бумажного к цифровому документообороту требует инструмента, который бы минимизировал усилия и сокращал временные затраты на перепечатывание, а также помогал исключить ошибки, связанные с необходимостью ручной оцифровки бумажных носителей. Таким инструментом является сканирование – процесс получения цифровой копии бумажного документа. Прототип сканера был создан в XIX веке, и с тех пор технология претерпела множество изменений.

Использование сканера позволяет удобно и быстро перевести бумажный документ в цифровой формат. Сканеры вводят плоское изображение в компьютер лучше, чем цифровые камеры. Однако необходимость наличия сканера создает некоторые трудности процессу – покупка, размещение и настройка устройства отнимает время и деньги. Кроме того, сканеры привязаны к формату документов – на устройстве, подходящем для сканирования документов формата А4, не получится обработать формат А3.

Альтернативой привычному настольному устройству может стать приложение для сканирования на смартфоне. Такое приложение позволяет сфотографировать документ, после чего автоматически определяет его границы, настраивает резкость и четкость изображения и в целом улучшает качество полученного «скана». Пользователь может создавать многостраничные документы в разных форматах и сразу отправлять их по почте или через мессенджеры. Неоспоримыми достоинствами приложения-сканера являются компактность и мобильность технологии – «сканирование» возможно где угодно, необходим только смартфон. Также для приложения не важен формат листа исходного документа.

Однако сканирование предполагает не только получение «фотографии» текста, но и его перевод в редактируемый формат с возможностью поиска – иными словами, оптическое распознавание символов (англ. Optical Character Recognition – OCR).

На данный момент на рынке представлено множество инструментов, поддерживающих данную технологию, для индивидуального пользования как для непосредственно работы с документами, так и в составе программного обеспечения, напрямую не связанного с оцифровкой информации (например, приложения-переводчики с функцией распознавания и перевода сфотографированного или поступающего в режиме реального времени текста). Подобный функционал используется и в специализированных разработках, например, в банковском программном обеспечении (для автоматизации процесса обработки клиентских документов), в почтовых службах (для сортировки писем и посылок), в юридических базах данных (для размещения подписанных документов), в программном обеспечении Госавтоинспекции (для распознавания номерных знаков автотранспорта с помощью камеры контроля скорости). Кроме того, технология оптического распознавания символов встроено во все современное программное обеспечение для работы со сканерами, что позволяет автоматически при сканировании создавать документы с

возможностью копирования из них и поиска по тексту. Однако несмотря на растущую популярность приложений-сканеров для смартфонов далеко не все из них поддерживают данную технологию.

Кроме того, распознавание символов для документов в хорошем качестве, представляющих собой темный текст на однородном светлом фоне, набранный стандартным шрифтом, в настоящий момент является фактически решенной задачей. А вот проблемы распознавания текста на неоднородном фоне и рукописного текста в настоящее время являются предметом активных исследований.

Целью выпускной квалификационной работы является разработка мобильного приложения-сканера с функцией распознавания печатного текста стандартного шрифта на сложном фоне.

Для реализации данной цели необходимо решить следующие **задачи**:

- изучить общие сведения о технологии оптического распознавания символов и ее современное состояние;
- рассмотреть существующие на рынке аналоги разрабатываемому приложению, выделить их особенности, достоинства и недостатки;
- ознакомиться с этапами распознавания текста и существующими алгоритмами;
- выбрать стек технологий для создания приложения;
- реализовать алгоритм распознавания текста на основе метода сравнения моментов контуров букв в тексте с моментами заранее подготовленных шаблонов букв;
- реализовать алгоритм распознавания текста на основе алгоритма масштабного пространства кривизны;
- реализовать алгоритм распознавания текста на основе библиотеки Tesseract OCR;
- провести сравнительный анализ результатов распознавания текста при помощи реализованных алгоритмов;

- выбрать алгоритм, который будет использован в разрабатываемом приложении, при необходимости доработать его;
- разработать мобильное приложение-сканер с функцией распознавания печатного текста стандартного шрифта на сложном фоне.

В теоретической части изложена общая информация о технологии оптического распознавания символов, приведен обзор существующих на рынке мобильных приложений-сканеров с функцией распознавания текста, их достоинств и недостатков, описаны этапы распознавания текста, основные алгоритмы и подходы, используемые на каждом из них, теоретические аспекты реализованных алгоритмов распознавания текста, конфигурация разработанного в рамках выполнения выпускной квалификационной работы приложения и использованные при разработке технологии.

В практической части рассмотрены реализации трех алгоритмов распознавания текста (алгоритм распознавания текста на основе метода сравнения моментов контуров букв в тексте с моментами заранее подготовленных шаблонов букв, алгоритм распознавания текста на основе алгоритма масштабного пространства кривизны, алгоритм распознавания текста на основе библиотеки Tesseract OCR) и приведен их сравнительный анализ, описаны причины выбора для дальнейшей работы алгоритма распознавания текста на основе библиотеки Tesseract и доработка данного алгоритма, позволяющая реализовать поставленную задачу – распознавание печатного текста стандартного шрифта на сложном фоне, описан процесс разработки серверной и клиентской частей приложения, рассмотрены написанные скрипты.

Практическая значимость магистерской работы заключается в том, что разработанное приложение является готовым программным продуктом, который может быть использован на практике. Также приложение показывает более высокие результаты распознавания текста, чем многие аналоги, представленные на рынке.

Структура и объём работы. Магистерская работа состоит из введения, семи разделов, заключения, списка использованных источников и семи приложений. Общий объем работы – 190 страниц, из них 125 страниц – основное содержание, включая 61 рисунок и 6 таблиц, список использованных источников информации – 62 наименования.

КРАТКОЕ СОДЕРЖАНИЕ РАБОТЫ

В первом разделе «Общие сведения об области разработки» в общих чертах рассмотрено современное состояние технологии оптического распознавания символов, а также приведен обзор существующих на рынке мобильных приложений-сканеров с функцией распознавания текста, их достоинств и недостатков.

Оптическое распознавание символов (англ. Optical Character Recognition – OCR) – это технология, преобразовывающая различные типы документов, полученные с помощью сканирования или фотографирования, в редактируемые форматы с возможностью поиска.

В области OCR разработано достаточно много подходов и алгоритмов, показывающих хорошие результаты распознавания. Однако в большинстве прикладных приложений точность распознавания по-прежнему намного ниже человеческого уровня. Распознавание текста на изображениях с неоднородным фоном становится задачей сложнее, собственно, из-за фона, который затрудняет точное детектирование текстовых областей. Сложное форматирование также является причиной ошибок при работе систем распознавания текста.

Таким образом, проблема распознавания текста в документах со сложным фоном и/или форматированием на сегодняшний день по-прежнему является предметом активных исследований.

В данном разделе было рассмотрено 14 мобильных приложений для сканирования с функцией распознавания текста. Приложения сравнивались по возможностям распознавания текста на сложном фоне (страница журнала) в их бесплатной версии. Только три приложения из рассмотренных дали приемлемый результат при распознавании печатного текста на сложном фоне. Но ограниченный функционал и очень короткий пробный период бесплатных версий этих приложений делает их фактически бесполезными с точки зрения постоянного практического использования.

Таким образом, распознавание текста в документах со сложным фоном и форматированием на сегодняшний день является актуальной задачей в области мобильного сканирования.

Второй раздел «Этапы распознавания текста» описывает этапы распознавания текста, основные алгоритмы и подходы, используемые на каждом из них.

Реализация системы распознавания текста зависит от целей применения системы и устройств, для которых она создается. Однако можно выделить общий алгоритм работы систем распознавания текста:

1. поиск областей, содержащих текст;
2. предварительная обработка локализованной области, улучшение ее качества;
3. сегментация текста на отдельные элементы: строки, слова, символы;
4. выделение признаков каждого символа;
5. распознавание символов;
6. словарная проверка.

В третьем разделе «Теоретические аспекты реализованных алгоритмов распознавания текста» рассматриваются существующие инструменты, позволяющие решить задачу распознавания текста на изображении.

Актуальность задачи распознавания текста привела к появлению программных продуктов, позволяющих создавать собственные OCR-проекты.

OpenCV (англ. Open Source Computer Vision Library, библиотека компьютерного зрения с открытым исходным кодом) – библиотека алгоритмов компьютерного зрения, обработки изображений и численных алгоритмов общего назначения. Реализована на C/C++, также разрабатывается для Python, Java, Ruby, Matlab, Lua и других языков.

Популярна за счёт открытого кода и возможности бесплатно использовать в учебных и в коммерческих целях.

Tesseract OCR – свободно распространяемая библиотека, написанная на языке C++. На данный момент последней версией является Tesseract 4.0, основанная на нейросетях типа LSTM, что улучшает показатели распознаваемости при помощи данной библиотеки.

Так как два из реализованных изначально алгоритмов основываются на контурном анализе, он также подробно рассмотрен в данном разделе.

Любой объект на изображении имеет границы, которые человек воспринимает как резкий перепад яркости между двумя областями. Контурный анализ – важный метод описания, хранения, распознавания, сравнения и поиска объектов на изображении.

При проведении контурного анализа полагается, что контур содержит достаточную информацию о форме объекта, и внутренние точки объекта во внимание не принимаются.

Чтобы сравнить два контура, нужно предварительно рассчитать их моменты. Момент – это характеристика контура, объединённая (суммированная) со всеми пикселями контура.

Один из реализованных вариантов алгоритма распознавания текста основан на алгоритме масштабного пространства кривизны (culvature scale space, CSS), который разглаживает контур буквы с помощью Гауссовой функции ядра и отслеживает её точки перегиба. Данный алгоритм подробно описан в этом разделе. Для расчета кривизны кривых используются вейвлеты Гаусса в алгоритме CSS. На вход для обработки подается текстовая область, представляющая собой слово на сложном фоне, а на выходе генерируется распознанный текст.

Четвертый раздел «Теоретические сведения об используемых в разработке приложения технологиях» описывает конфигурацию разработанного приложения и использованные при разработке технологии.

Архитектура разработанного приложения представляет собой модель клиент-сервер. Клиентом является приложение на смартфоне, написанное в межплатформенной среде разработки Unity. Серверная часть находится на удаленном сервере и написана на Python.

Для создания серверной части приложения были использованы микрофреймворк Flask и библиотека Pillow.

Flask включает в себя базовую функциональность, необходимую всем веб-приложениям, а все остальное может быть предоставлено сторонними расширениями и самим программистом.

Pillow – это библиотека, предназначенная для работы с растровой графикой. В серверной части разработанного приложения данная библиотека используется для создания PDF-документа из набора переданных на сервер фотографий.

Клиентская часть представляет собой приложение на смартфоне, написанное в межплатформенной среде разработки Unity.

Причин для выбора Unity как платформы для разработки несколько:

- удобство работы с визуальной частью приложения. Все элементы пользовательского интерфейса «перетаскиваются» на сцену, им выставляются нужные свойства и к ним привязываются необходимые скрипты. Кроме того, Unity предлагает разработчику большое количество легко и гибко настраиваемых элементов пользовательского интерфейса;
- в реализуемом приложении необходима связь с сервером, и стандартный Unity-класс UnityWebRequest позволяет составлять HTTP-запросы и обрабатывать HTTP-ответы. Данный класс имеет набор удобных для работы статических функций, которые возвращают объекты UnityWebRequest, правильно настроенные для распространенных сценариев использования;
- в клиентской части приложения необходима сериализация некоторых данных, таких как созданные документы и распознанный текст.

Unity предоставляет разработчику встроенный механизм сериализации данных в разных форматах;

- в клиентской части приложения необходимо делать фотографии документов. Unity предоставляет структуру WebCamDevice, описывающую камеры на устройстве пользователя и позволяющую взаимодействовать с ними без подключения дополнительных инструментов.

В пятом разделе «Реализация алгоритмов распознавания текста и их сравнительный анализ» рассмотрены реализации трех алгоритмов распознавания текста (алгоритм распознавания текста на основе метода сравнения моментов контуров букв в тексте с моментами заранее подготовленных шаблонов букв, алгоритм распознавания текста на основе алгоритма масштабного пространства кривизны, алгоритм распознавания текста на основе библиотеки Tesseract OCR) и приведен их сравнительный анализ, описаны причины выбора для дальнейшей работы алгоритма распознавания текста на основе библиотеки Tesseract и доработка данного алгоритма, позволяющая реализовать поставленную задачу – распознавание печатного текста стандартного шрифта на сложном фоне.

Алгоритм на основе Tesseract OCR не только дал отличные результаты распознавания в случае черного текста на белом и цветном фоне, но и показал более качественные результаты в случае изображений с низкой контрастностью текста и фона. Однако в своем изначальном виде он не подходил для распознавания большого количества слов на изображении, даже если документ представлял собой черный текст на белом фоне.

Причиной возникающих при распознавании текста этим алгоритмом ошибок являлось использование алгоритмом одного подхода при распознавании разных типов изображения. Алгоритм, во-первых, не был рассчитан на выделение блоков текста из изображения, а во-вторых, использовал один способ предобработки для всех поданных на распознавание изображений.

Для решения этих проблем и улучшения результатов распознавания была придумана и реализована доработка алгоритма с помощью обработки входного изображения разными способами и оценки полученного на каждом этапе результата.

Для предобработки изображения используется функционал библиотеки OpenCV, а распознавание осуществляется при помощи Tesseract.

Первый способ предобработки изображения и его последующего распознавания самый простой, он подходит для изображений с контрастным текстом на однотонном фоне.

Второй способ распознавания позволяет распознавать текст на сложном фоне, однако его результаты могут быть не удовлетворительны в случае разрозненных текстовых блоков.

В третьем способе распознавание ведется по отдельным найденным блокам текста, что позволяет распознать текст на сложном фоне, сильно различающемся у разных блоков текста.

Когда результат распознавания удовлетворяет установленному критерию (не меньше заданного количества слов), работа алгоритма заканчивается.

Такой подход к распознаванию текста позволяет получить наилучший результат при оптимальных затратах (так как, например, распознавание каждого блока по отдельности достаточно долго по времени, и к этому не стоит прибегать, если Tesseract дает хорошие результаты при распознавании всего изображения целиком).

Улучшенный алгоритм показал хорошие результаты при распознавании всех типов изображений, в том числе и изображений с разрозненными блоками текста на неоднородном фоне (например, страниц из журналов или рекламных буклетов).

Из результатов тестирования созданного алгоритма был сделан вывод, что он подходит для реализации поставленной задачи.

Шестой раздел «Разработка серверной части приложения» описывает скрипт, используемый в серверной части приложения, написанный на языке программирования Python и находящийся на удаленном сервере.

Сервер реализует несколько функций. Во-первых, он обрабатывает полученные от клиента фотографии, приводя их к виду, приближенному к виду сканов, создаваемых с помощью настольного сканера. Обработанные фотографии возвращаются клиенту и сохраняются локально в приложении на смартфоне. Во-вторых, на сервере находится функционал распознавания текста. При необходимости распознать текст на изображении оно отправляется на сервер, клиенту возвращается результат распознавания, который сохраняется локально и доступен даже при отсутствии подключения к серверу. В-третьих, с помощью сервера формируется PDF-документ из полученных от клиента фотографий, входящих в требуемый документ в приложении. В браузере можно открыть сформированный PDF-документ по выводимой в приложении ссылке, откуда его можно загрузить на смартфон или отправить через электронную почту или мессенджер.

Седьмой раздел «Разработка клиентской части приложения» описывает приложение на платформе iOS, разработанное в мультиплатформенной среде разработки Unity, которое является клиентом разработанного клиент-серверного приложения.

С помощью клиентской части приложения можно создавать и хранить многостраничные документы, состоящие из фотографий, обработанных и приведенных к виду, схожему с видом скана, который можно получить обычным настольным сканером. При съемке фотографии можно выделить конкретную область, которая будет добавлена в создаваемый документ. Каждый документ можно редактировать, добавляя и удаляя фотографии-сканы, из которых он состоит, и переименовывать документ. Любую из фотографий в документе можно отправить на распознавание, полученный результат распознавания можно скопировать. Из документа, созданного в

приложении, можно сформировать PDF-документ, который доступен для просмотра через браузер и загрузки на устройство.

Приложение состоит из четырех сцен: «Основной экран», «Камера», «Экран обрезки полученного изображения» и «Экран просмотра файла». Отдельно в разработанном приложении реализована система хранения, которая также описана подробно.

Пользовательская информация, такая как полученные фотографии в виде массива байтов, результаты распознавания текста, названия файлов, сохраняются на устройстве пользователя (смартфоне) с помощью бинарной сериализации.

Все эти данные записываются в поля экземпляра сериализуемого класса системы хранения. В системе хранения также реализованы функции, которые упростят взаимодействие с экземплярами этого класса – фактически, CRUD-операции (создание (англ. create), чтение (англ. read), модификация (англ. update), удаление (англ. delete)).

На сцене «Основной экран» отображаются созданные файлы. При нажатии на любой файл открывается меню, из которого можно удалить файл, открыть его для изменения или создать PDF-документ из этого файла. Также можно создать новый файл, нажав на кнопку внизу экрана.

Сцена «Камера» позволяет пользователю делать фотографии документов, которые будут поданы на распознавание.

Сцена «Экран выделения фрагмента полученного изображения» позволяет выделить фрагмент, который будет добавлен в файл, полученной фотографии документа. Стоит отметить, что это необязательный этап, который может быть пропущен, если пользователя устраивает полученная фотография в первоначальном виде.

На сцене «Экран просмотра файла» отображаются фотографии, находящиеся в файле. Эти фотографии можно удалять из файла, распознавать текст с них, добавлять в файл новые фотографии. Также на этой сцене можно менять имя открытого файла.

ЗАКЛЮЧЕНИЕ

Перевод информации, представленной на бумажных носителях, в электронный вид является задачей, актуальной в современном мире не только для отдельных пользователей, но и целых ведомств и организаций. Основным инструментом перевода бумажных носителей в электронный вид является сканер.

Альтернативой настольному устройству все чаще становится приложение для сканирования на смартфоне. Такое приложение можно установить на любой современный смартфон и сделать быстрым и легким процесс получения скан-копий, даже при отсутствии настольного сканера.

Сканирование с помощью стационарного устройства предполагает не только получение электронной копии документа, но и оптическое распознавание символов текста этого документа. Данный функционал встроен во многие компьютерные приложения для сканирования и позволяет, например, ускорить или даже автоматизировать процесс обработки большого количества документов.

Такой функционал есть и в некоторых мобильных приложениях для сканирования, однако, как стало понятно по результатам исследования, проведенного в рамках выполнения данной выпускной квалификационной работы, тут он работает не так хорошо. Таким образом, распознавание текста в документах со сложным фоном и форматированием на сегодняшний день является актуальной задачей в области мобильного сканирования.

В ходе выполнения выпускной квалификационной работы были рассмотрены существующие алгоритмы и технологии, позволяющие решить задачу распознавания текста на сложном фоне.

Был выбран стек технологий для создания клиентской и серверной частей приложения.

Были реализованы три алгоритма (алгоритм распознавания текста на основе метода сравнения моментов контуров букв в тексте с моментами заранее подготовленных шаблонов букв, алгоритм распознавания текста на

основе алгоритма масштабного пространства кривизны, алгоритм распознавания текста на основе библиотеки Tesseract OCR). Был проведен сравнительный анализ результатов распознавания текста при помощи реализованных алгоритмов.

Полученные результаты сравнительного анализа позволили выбрать алгоритм распознавания текста (алгоритм на основе библиотеки Tesseract OCR), который был доработан и использован для реализации поставленной задачи – создания приложения-сканера для смартфона с функцией распознавания печатного текста стандартного шрифта на сложном фоне.

Были разработаны клиентская и серверная части приложения для мобильной платформы iOS. При необходимости данное приложение может быть перенесено также на мобильную платформу Android.

Разработанное приложение позволяет создавать и хранить на устройстве пользователя многостраничные документы из фотографий, приведенных к виду скан-копий, переводить эти документы в PDF-формат, загружать их на устройство пользователя, отправлять с помощью электронной почты или мессенджеров, распознавать текст с фотографий, из которых состоит документ, копировать и сохранять результаты распознавания, добавлять и удалять фотографии из документа, менять имя документа.

Таким образом, все поставленные задачи были выполнены, цель работы достигнута.

Отдельные части магистерской работы были опубликованы:

- Талалайкина, Е.И., Лаптев, Ю.В. Обзор технологии оптического распознавания символов / Е.И. Талалайкина, Ю.В. Лаптев // «Научное сообщество студентов XXI столетия. Технические науки»: Электронный сборник статей по материалам С студенческой международной научно-практической конференции. – Новосибирск: Изд. ООО «СибАК». 2021. – № 4(99) – С. 27 – 33.

- Талалайкина, Е.И., Огнева, М.В., Лаптев, Ю.В. Реализация и сравнительный анализ алгоритмов распознавания текста / Е. И. Талалайкина, М. В. Огнева, Ю. В. Лаптев // Информационные технологии в образовании. – 2021. – № 4. – С. 220-224.

Отдельные части магистерской работы были представлены на конференциях:

- выступление с докладом «Реализация и сравнительный анализ алгоритмов распознавания текста» на XIII Всероссийской научно-практической конференции «Информационные технологии в образовании» «ИТО-Саратов-2021» (СГУ, г. Саратов, 30-31 октября 2021 г.);
- выступление с докладом «Реализация алгоритма распознавания печатного текста стандартного шрифта на сложном фоне с использованием функционала библиотеки Tesseract» на Студенческой научной конференции факультета КНиИТ (кафедральный этап) (СГУ, г. Саратов, 21 апреля 2022 г.).

Основные источники информации:

1. Howse, J. Learning OpenCV 4 computer vision with Python 3 / J. Howse, J. Minichino. – Birmingham: Packt Publishing, 2020.
2. Elgendy, M. Deep learning for vision systems / M. Elgendy. – Shelter Island: Manning, 2020.
3. Лутц М. Изучаем Python, 5-е издание / М. Лутц. – Санкт-Петербург.: Символ-Плюс, 2019. – 1280 с.
4. Grinberg M. Flask Web Development. Developing web applications with Python / M. Grinberg. – Sebastopol: O’Reilly Media, 2018 – 312 p.
5. Хокинг Дж. Unity – в действии. Мультиплатформенная разработка на C#. 2-е межд. издание / Дж. Хокинг. – Санкт-Петербург.: Питер, 2019. – 352 с.