МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

**«КРЫМСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ имени В. И. ВЕРНАДСКОГО»**

(ФГАОУ ВО «КФУ им. В. И. Вернадского»)

**Таврическая академия (структурное подразделение)**

**Факультет математики и информатики**

**Кафедра информатики**

Хоболев Александр Дмитриевич

. **Разработка объектов дополненной реальности при посещении городов Крыма**

Выпускная квалификационная работа

Обучающегося 4 курса

Направления подготовки 01.03.02. Прикладная математика и информатика

Форма обучения очная

Научный руководитель

Доцент кафедры информатики,

Кандидат физико-математических наук М. Г. Козлова

К ЗАЩИТЕ ДОПУСКАЮ:

Заведующий кафедрой,

доктор физико-математических наук,

профессор В. И. Донской

Симферополь, 2019

**Аннотация**

**Хоболев А. Д. Разработка объектов дополненной реальности при посещении городов Крыма.** Выпускная квалификационная работа бакалавра по направлению 01.03.02. Прикладная математика и информатика. Таврическая академия Крымского федерального университета имени В. И. Вернадского.

Целью работы является создание приложения дополненной реальности, использую оптимальное программное обеспечение. Содержание работы состоит в исследовании, тестировании и разборе современных технологий, позволяющих работать с дополненной реальностью. Результатом работы является создание мобильного приложения.

Ключевые слова: дополненная реальность, AR, распознавание, текст, оптические системы распознавания, OCR, бинаризация, макет, маркер, target, объект, модель, моделирование, редактор, программное обеспечение, среда разработки, предобработка, шум, искажение, изображение, методы, классификация, шаг, сегментация, обучение, анализ, приложение, визуализация, город, Крым.

Страниц – 55, таблиц – 0, иллюстраций – 24, приложений – 1, библиографических источников – 37.

Содержание

[Введение 4](#_Toc10713846)

[1. Подходы к распознаванию текста 7](#_Toc10713847)

[1.1. Оптическое распознавание символов 7](#_Toc10713848)

[1.2. Метод распознавания текста №1 11](#_Toc10713849)

[1.3. Метод распознавания текста №2 13](#_Toc10713850)

[1.4. Метод распознавания текста №3 14](#_Toc10713851)

[1.5. Распознавание текста по структуре скелета букв 16](#_Toc10713852)

[1.6. Методы бинаризации изображений 19](#_Toc10713853)

[1.7. Tesseract OCR 21](#_Toc10713854)

[2. Подходы к работе с объектами дополненной реальности 29](#_Toc10713855)

[2.1. Vuforia 32](#_Toc10713856)

[2.2. ARToolKit 34](#_Toc10713857)

[2.3. Подходы к разработке объектов дополненной реальности 35](#_Toc10713858)

[2.3.1. Blender 35](#_Toc10713859)

[2.3.2. 3ds Max 36](#_Toc10713860)

[2.3.3. ZBrush 37](#_Toc10713861)

[3. Разработка комплекса визуализации объектов дополненной реальности при посещении городов Крыма 38](#_Toc10713862)

[3.1. Распознавание текста 38](#_Toc10713863)

[3.2. Создание объектов дополненной реальности 38](#_Toc10713864)

[3.3. Визуализация элементов дополненной реальности 38](#_Toc10713865)

[Приложение 48](#_Toc10713866)

[Список источников 53](#_Toc10713867)

# Введение

На данном этапе развития технологий одним из самых перспективных направлений разработок IT-специалистов является дополненная реальность. Она представляет собой современный способ и представления информации.

Дополненная реальность позволяет легко и визуально получать человеку требуемую информацию. Она является технологией, при помощи которой пользователь может смотреть на предметы с другой стороны, глубже погружаться в процесс и становиться его частью. Дополненная реальность отличается от виртуальной реальности следующим принципом – дополненная реальность заключается в наложении виртуальных объектов на существующие и их использовании в режиме реального времени. Она является взаимодействием окружающего нас мира с техническими устройствами. [1]

Дополненная реальность позволяет вывести процесс взаимодействия пользователя с программами на совершенно новый уровень. Используя программное обеспечение и компьютерное устройство, она наносится на объекты окружающей среды и несет за собой дополнительную информацию и возможности. Сейчас технологии уже позволяют считывать информацию с камер и изображений, а так же дополнять их различными виртуальными объектами. Дополненная реальность уже способна рассказывать нам подробно о тех или иных продуктах, видах одежды и интересующих нас уголках планеты. Можно привести следующие примеры: большой популярностью обладают мобильные приложения с элементами дополненной реальности, теперь любой может завести себе домашнего питомца от кошки до динозавра, кормить его, ухаживать, гулять и играть, что также является хорошей тренировкой для детей перед заведением реального питомца; существуют целые виртуальные магазины продуктов или одежды, в которые можно прийти и увидеть продукты, лежащие на полках, и одежду, висящую на вешалках и уже нет необходимости идти в магазин и мерить одежду, когда уже дома можно посмотреть, как она на тебе сидит. С помощью таких приложений пользователь уже не должен тратить время для того, чтобы добраться до информации, она сама приходит к нему в мобильные устройства.

Технологии дополненной реальности используются и в медицине. [2] Тренажеры и симуляторы такого рода позволяют значительно увеличить качество практических навыков врачей, снизить эмоциональное напряжение при необходимости проведения хирургических операций в реальной жизни и, как следствие, повышается качество медицины во всем мире.

Дополненная реальность может использоваться и в образовательном процессе школ или высших учебных заведений. Уже не нужно будет оборудовать специальные классы для физики, химии или биологии, будет просто достаточно обладать специальным программным обеспечением, которое наглядно покажет различные объекты или процессы.

Безусловно, она будет широко применима в проектировании и строительстве.

Также дополненная реальность уже используется в приложениях, помогающих человеку ориентироваться на местности. [3] Например, при посещении незнакомого города, программа позволяет строить маршруты до необходимых локаций. При посещении музея могут быть доступны такие опции как виртуальный экскурсовод или определение и визуализация маршрута стрелочками до необходимого экспоната.

Причины актуальности использования дополненной реальности:

1. предоставление информации в режиме реального времени;
2. способность обучения с ее помощью. Благодаря наглядности, простоте и способности удивлять дополненная реальность полезна при обучении детей;
3. способность привлекать внимание. Полезно для рекламной индустрии;
4. безграничность сфер использования.

Так как технологии стремительно развиваются и люди уже не могут представить свою жизнь без них, то отличным способом предоставления им визуальной информации о самых красивых и интересных достопримечательностях Крыма будет являться, шагающее в ногу со временем, приложение с элементами дополненной реальности. С помощью таких инновационных программ можно значительно повысить интерес и желание посетить крымский полуостров у людей с любого уголка планеты. Ведь уже здесь и сейчас человек может просмотреть интересующие его достопримечательности и понять, какую из них ему действительно хочется увидеть своими глазами. Разработки такого формата поспособствуют значительному повышению интереса туристов к полуострову и, как следствие, повышению посещаемости.

Целью работы является разработка программного обеспечения визуализации дополненной реальности на примере туристических объектов при посещении городов Крыма. Крым является прекрасным местом для отдыха и туризма в любое время года. Одним из самых успешных способов привлечения гостей в Крым могут стать визуальные приложения, которые покажут, насколько полуостров богат различного рода достопримечательностями, которые подходят абсолютно для каждого – начиная от походов в горы и заканчивая отдыхом у моря. Благодаря способности дополненной реальности удивлять, из-за обращения внимания, как результат, увеличится интерес и посещаемость.

Задачи, решаемые в работе:

1. обзор научных источников по данному вопросу;
2. изучение соответствующих библиотек и программного обеспечения;
3. создание 3D-макетов достопримечательностей;
4. реализация приложения распознавания названия города на изображении (карте);
5. реализация комплекса визуализации.
6. **Подходы к распознаванию текста**

Следует отметить, что задача распознавания текста важна до сих пор даже при большом количестве уже готовых программ. Большое количество задач требует компактных алгоритмов, заточенных под решение конкретно требуемых проблем.

Распознавание текста является оной из важнейших подразделов распознавания образов. Существует большое количество литературы с огромным количеством информации, которая внесла неизмеримый вклад в развитие человечества, которую уже попросту нельзя взять в руки, не повредив книгу. Именно поэтому оцифровка такой литературы столь важна.

Следует отметить, что под распознаванием текста понимается три основных метода:

1. сравнение с подготовленными шаблонами;
2. распознавание, которое использует критерии распознаваемого объекта;
3. распознавание при помощи нейронных сетей, самообучающихся алгоритмов. [4]
   1. **Оптическое распознавание символов**

Оптическое распознавание символов – это механическое или электронное конвертирование текста на изображении в текст машинного кодирования.

Принцип работы оптического распознавания символов заключается в следующих этапах.

1. Процесс предобработки.

Оптические системы распознавания часто прибегают к предобработке изображения с целью улучшения результатов. Существуют такие виды предобработки: выравнивание (de-skew) – это ситуация, когда сфотографированный или отсканированный текст располагается под наклоном, когда нам необходим строго горизонтальный или вертикальный текст; удаление шумов (despeckle) – удаление пятен, сглаживание углов; бинаризация (binarisation) – перевод изображения из цветного в черно-белое; удаление линий (line removal) – удаление нерельефных полей и строк; анализ макета или «зонирование» (Layout analysis or «zoning») – определение столбцов, абзацев, подписей и т. д. как отдельные блоки, это особенно важно в многостолбцовых раскладках и таблицах; обнаружение строк и слов (line and word detection) – установка основной линии для слов и форм символа, разделение слова если необходимо; выделение символов или сегментация (character isolation or «segmentation») – текст изображения разделяется на символы; нормализация пропорций и масштаба изображений (normalize aspect ratio and scale).

2. Распознавание символов.

Здесь присутствуют два основных типа оптического распознавания.

Матричное сопоставление включает в себя сравнение изображения с сохраненным образом на попиксельной основе; это так же известно как «сопоставление с образом», «распознавание образов» или «корелляция изображения». Это основано на том, что входной знак правильно изолирован от остальной части изображения, а сохраненный рельеф имеет аналогичный шрифт того же масштаба. Этот метод работает лучше всего с машинописным текстом и не работает, когда встречаются новые шрифты. Это метод, который применялся в раннем физическом оптическом распознавании на основе фотоэлементов.

Извлечение объектов разбивает знаки на «объекты», такие как линии, замкнутые циклы, направление линий и пересечения линий. Особенности извлечения уменьшают размерность представления и делают процесс распознавания вычислительно эффективным. Эти функции сравниваются с абстрактным векторным представлением символа, которое может привести к одному или нескольким прототипам знаков. Общие методы обнаружения признаков в компьютерном зрении применимы к этому типу OCR, что обычно наблюдается в «интеллектуальном» распознавании рукописного ввода и, действительно, в большинстве современных программ OCR. Классификаторы ближайших соседей, такие как алгоритм k-ближайших соседей, используются для сравнения объектов изображения с сохраненными объектами знака и выбора ближайшего совпадения.

Программное обеспечение, такое как Cuneiform и Tesseract, использует двухфазный подход к распознаванию символов. Второй проход известен как «адаптивное распознавание» и использует формы букв, распознаваемые с высокой достоверностью на первом проходе, для лучшего распознавания оставшихся букв на втором проходе. Это выгодно для необычных шрифтов или сканов низкого качества, где шрифт искажен (например, размыт или выцветает).

3. Постобработка.

Точность распознавания можно повысить, если вывод ограничен лексиконом – списком слов, которые могут встречаться в документе. Это могут быть, например, все слова на английском языке или более технический словарь для конкретной области. Этот метод может быть проблематичным, если документ содержит слова, не входящие в лексикон, например, имена собственные. Tesseract использует свой словарь, чтобы повлиять на шаг сегментации символов, для повышения точности. Выходной поток может быть потоком простого текста или файлом символов, но более сложные системы распознавания могут сохранить исходный макет страницы и создать, например, аннотированный PDF-файл, который включает как исходное изображение страницы, так и текстовое представление с возможностью поиска. «Анализ ближнего соседа» может использовать частоты совпадений для исправления ошибок, отмечая, что некоторые слова часто встречаются вместе. Например, «Вашингтон, округ Колумбия» как правило, гораздо чаще встречается в английском, чем «Вашингтон DOC». Знание грамматики сканируемого языка может также помочь определить, может ли слово быть, например, глаголом или существительным, что позволяет повысить точность.

4. Оптимизация под конкретные приложения.

Основные поставщики технологий OCR начали настраивать системы OCR, чтобы лучше справляться с определенными типами ввода. Помимо лексики для конкретного приложения, лучшую производительность можно достичь, принимая во внимание правила, стандартные выражения, или богатую информацию, содержащуюся в цветных изображениях. Эта стратегия называется «Ориентированное на приложение OCR» или «Настраиваемое OCR» и применяется к распознаванию номеров, счетов, снимков экрана, удостоверений личности, водительских прав и производства автомобилей. То есть для улучшения распознавания определяются параметры, которые делают задачу более узкоспециализированной. [5]

Рассмотрим основные трудности при распознавании символов:

* разнообразие языков, символов, и способов изображать их. В некоторых языках присутствуют схожие символы: «N» и «И», «R» и «Я» и др. В некоторых языках присутствуют символы, которые схожи между собой или похожи на цифры. Например, в русском языке: «М», «Н», «И» и «О» и «0», и т.д.;
* искажение изображений символов. Причинами могут представляться огромное количество факторов. От незначительных проблем связанных изображением исходного формата, заканчивая дефектом самого источника;
* при обработке персональных данных, трудности могут возникать из-за методов защиты информации. Например, на паспортах присутствуют невидимые символы;
* так же усложняет задачу большое разнообразие различных шрифтов разных вариаций, размеров и масштабов.

Искажения цифровых изображений текстовых символов могут быть вызваны следующими причинами:

* шумами печати, в частности, не достаточно хорошей печатью (разрывами слитных черт символов), «слипанием» соседних символов, пятнами и ложными точками на фоне вблизи символов и т. п.;
* смещением символов или частей символов относительно их ожидаемого положения в строке;
* изменением наклона символов;
* искажением формы символа за счет оцифровки изображения с низкой степенью детализации;
* эффектами освещения (тени, блики и т. п.) при съемке камерой.

Большинство OCR-систем показывают высокие результаты в случае обработки черного текста на белом фоне. А при анализе не черно-белых изображений полученные данные могут быть некорректны.

Многие проблемы распознавания текста на цветных изображениях можно устранить на этапе предобработки. [6]

OCR-системы должны уметь обозначить на цифровом изображении области, в которых содержится текст, определять в них отдельные строки, выделять символы, распознавать их и быть устойчивыми к различным особенностям печати, расстоянию между символами, строками и т. д.

После изучения трех наиболее популярных приложений по распознаванию символов: ABBYY FineReader, CuneiForm, OCRopus, были сделаны следующие выводы.

1. Большинство из приложений обладают достаточно сложным интерфейсом и большим числом функций, которые рядовому пользователю могут быть попросту не нужны.
2. Все приложения (кроме FineReader) не обладают способностью работы в фоновом режиме.
3. Не все обладают способностью предобработки изображения.
4. Отсутствие графического интерфейса некоторых приложений.
5. Среди них нет программ, которые показывают идеальное распознавание текста. [7]
   1. **Метод распознавания текста №1**

Рассмотрим один из вариантов решения задачи распознавания текста. Предложенный вариант решения данной задачи не определяет расположение текста на изображении, этот этап нивелируется предобработкой изображения.

Здесь используется измененная версия медианного фильтра с увеличением значения компоненты красного цвета. Медианный фильтр – один из фильтров, предназначенных для минимизации уровня шума на изображении и сглаживании острых краев букв. При этом границы букв четко фиксируются.

После обработки изображения производится его сегментация, т. к. предполагается, что текст расположен горизонтально и строки не пересекаются друг с другом, то задача разделения букв не составляет труда. Далее разделяются строки путем нахождения полных горизонтальных полос от левого края изображения до правого, после чего, строки делятся на слова, и далее слова уже делятся на буквы. На выходе получаем текст, представленный изображениями букв.

После процесса разделения изображение подгоняется под размер заготовленных шаблонов. И наступает процесс распознавания. Имеется два варианта – с помощью метрик и с помощью нейронных сетей.

Рассмотри первый случай. Метрикой называется функция, определяющая расстояние в метрическом пространстве. Метрические алгоритмы основаны на понятии сходства объектов. Чем ближе они расположены, тем больше схожи объекты. Например, если две буквы C написаны разным шрифтом, то метрики для таких объектов будут одинаковы либо предельно схожи. В этой задаче выбрана метрика Хэмминга, которая показывает степень различия объектов. Если объекты различаются в двух позициях, то степень непохожести объектов будет равна двум и, как следствие, чем выше степень, тем ниже степень схожести объектов.

Это значит, что для определения написанной буквы, нужно находить метрики и шаблоны, в которых будет самая низкая степень отличия.

В ходе решения этой задачи был сделан следующий вывод: некоторые буквы обладают симметрией, полностью совпадают со своим отражением. И благодаря этому выводу удалось сократить перебор метрик примерно в 6 раз, чего удалость достичь путем вынесения симметричных букв в отдельный класс. Так же некоторые буквы выносятся в отдельный класс из-за своей уникальности.

При создании шаблонов на каком-то конкретном шрифте, сканирование изображения показывает результат примерно в 99% удачного распознавания, а при сканировании шрифта, не являющегося шаблонным, точность падает до 70%.

Во втором случае используется однослойная сеть Кохонена. Работа нейронной сети заключается в том, что, получив на вход изображение, система отвечает импульсом того или иного нейрона. Все нейроны обозначены буквами, из этого следует, что импульс несет ответ.

Каждый вход воспринимается с конкретным коэффициентом в результате. В конце на каждом нейроне скапливается заряд. И тот нейрон, заряд которого больше, и даст импульс.

Процесс обучения: отдельный модуль берет изображение и передает сети, она анализирует все позиции пикселей и выравнивает коэффициенты, минимизирует ошибку и данное изображение сопоставляется определенному нейрону.

Точность распознавания таким методом примерно равна 80%. Это зависит от обучающей выборки, ее величины и качества. [8]

* 1. **Метод распознавания текста №2**

Следующий вариант решения задачи распознавания начинается с поиска текста на изображении. Для этого примера автор использует метод разложения по цвету. Это выполняется путем подсчета всех цветов пикселей. Затем строится гистограмма всех цветов и выделяются основные 3 в порядке убывания: фон, шум, текст.

Далее приступает к работе часть программного кода, который производит бинаризацию изображения (бинарное изображение – это изображение, которое содержит только два цвета, в нашем случае – черный и белый). Здесь создается новое изображение такого же размера, как и изображение которое обрабатывается, с белым фоном, а затем программа осуществляет «проход» по изначальному изображению в поисках цвета, который является текстом на стартовом изображении, и окрашивает пиксели черным цветом на том же месте во втором изображении. Этот процесс можно назвать извлечением текста.

Автором был сделан следующий вывод: если текст написан разным цветом, то предполагается, что самый распространенный цвет – это цвет фона, а, следовательно, остальное – текст.

Следующий шаг – выявление текста на новом изображении. Это будет определяться вектором, содержащим 2 числа – 2 края символа (первое число – левый край символа, второе число – правый). На выходе будет получен набор границ символов (по горизонтали, начало и конец символа). Если границы символов идут последовательно и не пересекаются, это значит, что расположение символов определено верно.

При распознавании таким способом не нужно использовать нейронную сеть, вся работа происходит в векторном пространстве. Векторное пространство обладает следующими преимуществами: нет необходимости в углубленном изучении; есть возможность видеть результат в режиме реального времени; можно добавлять данные или удалять неправильные; легче запрограммировать, чем нейронную сеть; результаты классифицируются; можно добавлять данные, которые не получается распознать. Но есть недостаток – быстродействие ниже в сравнении с нейронными сетями.

Затем, созданное разработчиком векторное пространство принимает 2 словаря и выдает число от 0 до 1, где 0 значит, что они не связаны, а 1 – абсолютно идентичны.

Затем наступает процесс обучения, для которого нужна обучающая выборка, с которой будут сравниваться полученные ранее символы. В качестве обучающей выборки были взяты изображения из интернета, которые эта же программа и разбила на группы до того момента, пока не оказалось хотя бы по одному изображению каждого символа. На этом этапе автором был сделан следующий вывод – чем больше обучающая выборка, тем выше точность распознавания. После этого представлен набор изображений символов.

На последнем шаге производится сравнение полученных данных с обучающей выборкой. Определяем, где находится каждый символ и проверяем его с помощью векторного пространства. Сортируем и печатаем результаты. Уверенность в точности распознавания каждого из объектов находится в диапазоне от 0.928 до 0.983[9]. Рядом с вероятностью находится предполагаемый символ.

Точность данного способа недостаточно высока (22%) [9], но он обладает способностью распознавать не только текст, но и цифры. Однако программа может распознавать каждое пятое изображение за такое время (0.012 секунды), когда человек не смог бы распознать и одно. [9]

* 1. **Метод распознавания текста №3**

Следующий метод распознавания текста основан на сравнении.

Человек с детства смотрит на разные изображения с текстом, но в раннем возрасте еще не осознает, что перед ним конкретная буква, которая определяется однозначно. В процессе взросления, его база образов увеличивается и он уже может различать все буквы однозначно путем сравнения и анализа изображения. Точно так же и компьютер, он получает на вход изображение, определяет, где на нем находится символ и сравнивает с уже имеющейся базой. Если схожий символ найден, символ распознается.

Методы сравнения делятся на следующие категории:

1. сравнение с эталоном;
2. сравнение по критериям.

Эти методы объединяет наличие нейронных сетей в их выполнении.

Сравнение по критерию основано на определении характерных черт символа, это очень трудоемкий процесс, который подвержен проблемам, основанным на сходстве объектов, наличии шума на изображении, его несоответствии и низком разрешении изображения, а так же его минусом является скорость работы.

Сравнение с эталоном является более универсальной задачей. В этой задаче сравнение более простое. Создается эталонное изображение и исходный файл сравнивается с эталонным. Есть множество методов сравнения, таких как попиксельный метод сравнения, наложение, наложение со смещением. В [10] для решения этой задачи используется более простой метод сравнения – попиксельный. Он заключается в следующем, поочередно сравниваются соответствующие пиксели на черно-белых изображениях (исходное и эталонное). Для этого изображения должны быть одинаковых размеров. Сравнение – долгий итерационный процесс, который, при размере изображения 100х100 пикселей, уже совершает 10000 операций. Экспериментальным путем в [10] установлено, что оптимальный размер изображения – 16х16 пикселей.

Перед тем как изменять изображение, нужно найти символы на изображении. Для выполнения этого шага изображение преобразуется в черно-белое и находятся крайние пиксели предполагаемых символов. Вокруг каждого символа строится рамка красного цвета и уже в ней подгоняется размер к оптимальному.

Далее запускается алгоритм сравнения, попиксельно сравниваются два изображения, если на одном из них пиксель черный и на другом соответствующий по расположению пиксель белый, то здесь изображения различны и счетчик ошибки увеличивается на единицу. После чего число ошибок сравнения для каждого эталонного объекта пересчитывается по специальной формуле. Выбирается максимальное значение в массиве и соответствующий ему эталонный символ.

На следующем этапе составляется база (массив записей символа и его графического представления). Создается база русских символов и цифр, где пишется символ на изображении, и запускается программа, которая подгоняет эталонный объект под формат. Делается для того, чтобы сравниваемые изображения были одного размера, цвета (были приближены друг к другу).

Минусы данного способа: при повороте букв, наличии шумов и прочих помехах распознавание слишком часто некорректно; программа не отличает регистр букв и, соответственно, когда буквы нижнего и верхнего регистра схожи, могут допускаться ошибки на этих буквах; программа не распознает такие русские буквы как «Ы» и «Й» из-за того, что они состоят из двух фигур. [10]

* 1. **Распознавание текста по структуре скелета букв**

В решении задачи распознавания текста по структуре скелета букв фигурирует такой термин как «графема». Графема – это единица письменности. Иными словами, графема – это графический примитив, имеющий вид геометрического графа, изображающий начертания букв. Графема обладает следующими свойствами:

1. любые графемы различных символов предельно различны между собой;
2. если символы представляют одну и ту же графему, то различия между этими символами несущественны. Графемы двух одинаковых символов будут сильно похожи, т. е. степень сходства символов определяется степенью сходства их графем.

Стоит заметить, что буквы и графемы не одно и то же. Каждая буква имеет различные способы написания, соответственно, и разные графемы.

Началом в решении задачи данным методом является построения «скелета» буквы (граф бинарного изображения буквы) и удаления лишних ребер, то есть нужно найти некоторый подграф, отображающий характерные свойства буквы. У каждой буквы есть свой эталонный скелет.

Удалением лишних ребер будет заниматься алгоритм, который максимально приближает скелет буквы к эталонному. Это выполняется путем «стрижки» скелета, главная идея: поочередное удаление лишних ребер и сравнение с эталонным графом, пока графы не будут схожи по мере сходства, которая определяется выделением признаков из скелетного графа.

У скелетного графа фигуры вершинами являются центры вписанных пустых кругов (полностью лежащих внутри фигуры кругов, не содержащихся в других вписанных пустых кругах), которые имеют одну или три общих вершины с фигурой. А ребрами являются линии из центров вписанных пустых кругов, которые касаются границы ровно в двух точках.

Стрижка скелетного графа – процесс удаления несущественных ребер. В процессе стрижки скелетного графа количество ребер строго не увеличивается.

После того, как граф построен, следует агрегирование графа. Это процесс, в котором определяются типы вершин графа. Они делятся на три типа: листья – вершины, которые имеют продолжение только в одном направлении, развилки – вершины из которых можно пойти в трех направлениях, ребра – остальные вершины, имеющие 2 направления.

После проделанных шагов на графе могут содержаться шумовые ребра. Это возникает, если на шрифте, которым написан символ, есть засечки. Для человека эти засечки помехой не являются, чего нельзя сказать про компьютер. На следующем этапе будет производиться удаление засечек. В процессе удаления засечек, автором были отмечены следующие признаки: если засечки в скелетном графе присутствуют, то их не меньше двух; для любой засечки характерны следующие признаки: одна вершина ребра является листовой и вторая ее вершина является развилкой; длина ребра не превышает некоторого порога.

Таким образом, алгоритм состоит в следующем:

1. нахождение множества шумовых ребер в графе по имеющимся признакам;
2. удаление лишних ребер.

На следующем этапе производится выделение признаков полученного скелетного графа для построения классификатора. Т. к. скелетный граф каждой буквы имеет собственную структуру, то выделить общие признаки невозможно. Значит, нужно разбить модели графем на такие классы, в которых эти признаки выделить можно, они будут лежать в одинаковых признаковых пространствах, которые не совпадают с признаковыми пространствами двух различных классов.

Признаки разделяются на признаки верхнего уровня и признаки нижнего уровня. На этапе определения признаков признаки верхнего уровня определяются путем разбиения изображения на прямоугольные области, в которых подсчитываются все листья и развилки. Признаковое описание верхнего уровня фиксируется. Признаки нижнего уровня будут определять признаки полученных прямоугольников на изображении.

После чего наступает этап обучения алгоритма.

Для каждого признака верхнего уровня обучается свой классификатор – каждый на своем признаковом пространстве, соответственно своему нижнему признаку. Строится словарь классификации по всей выборке.

Алгоритм классификации состоит из следующих шагов:

1. определение математической модели графемы;
2. определение признаков верхнего уровня;
3. проверка наличия признака верхнего уровня в словаре. Если он отсутствует, то отказ в классификации, иначе – переход к следующему шагу;
4. построение признаков нижнего уровня, применение обученного классификатора, получение ответа.

Данный подход обладает следующими достоинствами: он не зависит от размера, шрифта и начертания буквы; выделяется структура (графема) буквы, которая позволяет работать и с буквами других шрифтов; признаки интерпретируемы. Недостатки: наличие отказа в классификации; зависимость от качества изображения и его бинаризации. [11]

* 1. **Методы бинаризации изображений**

Процесс бинаризации – это процесс перевода цветного изображения в черно-белое (двухцветное). Главным параметром сравнения является порог яркости *t*, с которым сравнивается значение яркости каждого пикселя. Если значение выше параметра *t*, то пиксель становится черным, иначе – белым. Существует два метода бинаризации: глобальный и локальный. В первом порог определяется однозначно для всего изображения, во втором – порог изменяется для разных областей изображения. Основной целью бинаризации является сокращение информации на изображении, что обуславливается упрощением последующих операций над ним. А удачное сокращение информации в работе с изображениями часто сказывается только положительно, т. к. часто приходится работать с изображениями, на которых большое количество шума, бликов, присутствует неравномерный фон и т. д. эта лишняя информация только усложняет работу с текстом или числами на изображении. Но с другой стороны в процессе бинаризации часто могут происходить искажения именно требуемой нам информации, которую мы хотим синтезировать. Могут быть такие ошибки: разрыв линий, потеря мелких деталей, слипание близлежащих символов, появление ложных объектов. Каждый алгоритм бинаризации обладает своими минусами и плюсами. Приведем известные алгоритмы бинаризации.

1. Алгоритм Бредли.

Данный алгоритм основан на локальном оценивании. Сравнение яркости пикселей не является сложной задачей, но при локальном оценивании следует выбирать такой порог, при котором символы будут отделяться наилучшим образом. Здесь, для определения порога, используется метод интегральных изображений. Интегральные изображения – это быстрый способ нахождения среднего значения яркости на выбранной области. Однажды составив для всего изображения интегральную матрицу, можно без проблем считать среднее значения яркости на любой выбранной области. Далее изображение разбивается на равные области (8 областей), на которых вычисляется пороговая величина. За изначальную величину t выбирается число, которое составляет 15% от средней яркости на всех частях изображения. А изменяется порог в зависимости от средней яркости этой части.

Достоинствами является простота реализации, скорость выполнения, возможность не подбирать параметры для большинства случаев и достойные показатели при работе с неоднородным фоном. Недостатки: низкая чувствительность к деталям с низким контрастом. [12]

Методы бинаризации изображения отличаются лишь способами выбора пороговых значений.

1. Бинаризация с нижним порогом: этот алгоритм является наиболее простым и распространенным алгоритмом бинаризации. В нем все пиксели сравниваются с пороговым значением яркости *t*. Если значения пикселя выше, то пиксель становится белым, если ниже – черным.
2. Бинаризация с верхним порогом является процессом обратным бинаризации с нижним порогом и получается негатив.
3. Бинаризация с двойным ограничением. Данный метод подходит для продуктивного использования в случае, когда интересует какой-то определенный диапазон пикселей.

Плюсами алгоритмов 2)-4) является простота их реализации и скорость выполнения. Минусом является узость применения, т. к. часто предоставляются недостаточно контрастные изображения и уже сложно подобрать оптимальный порог *t*. Для таких задач существуют следующие алгоритмы бинаризации:

1. Алгоритм Оцу.

Он является алгоритмом локальной пороговой обработки. Задачей метода является минимизация дисперсии внутри области. Для этого строится гистограмма распределения яркости пикселей на всём изображении. Она строится на основе значений 𝑝𝑖=𝑛𝑖/𝑁, где 𝑁 – число всех пикселей, а 𝑛𝑖 – число пикселей яркости i. Полученный набор делится на два класса порогом 𝑡 ∈ [0. . 𝐿]. В каждом классе вводятся относительные частоты 𝜔0 и 𝜔1.

1. Алгоритм Ниблэка.

Он основан на понятии о том, что порог яркости бинаризации изменяется на основании локального значения отклонения. Порог яркости для каждого 𝑝𝑖 считается формулой: 𝑝𝑖=µ𝑖+𝑘𝑠𝑖*,* гдеµ𝑖 – это среднеарифметическое, а 𝑠𝑖 – среднеквадратичное отклонение для точки i в некоторой ее окрестности. Слишком малая окрестность будет повышать значение шума на изображении, а слишком большая – понижать степень детализации. Параметр *k* показывает, какую область изображения считать за сам объект. [13]

* 1. **Tesseract OCR**

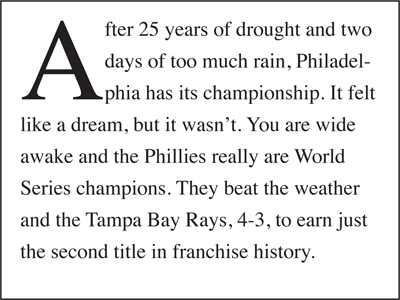
Рассмотрим принцип работы оптической системы распознавания Tesseract. Распознавание происходит как двухпроходный процесс. При первом проходе делается попытка распознать каждое слово сразу. Каждое удовлетворительное слово передается как данные обучения. Адаптивный классификатор получает шанс более точно распознавать остальной текст. Так как адаптивный классификатор, возможно, узнал что-то полезное слишком поздно, чтобы сделать вклад в обучающую выборку, второй проход проходит по странице, в котором слова, которые не были распознаны достаточно хорошо, распознаются снова. Финальный этап удаляет случайные пробелы, и находит текст, написанный малыми прописными буквами.

Основная часть данной работы состоит в разработке приложения с использованием системы OCR Teseract. Рассмотрим основные процессы работы Tesseract.

1. Метод нахождения строк.

Алгоритм поиска строк является одной из компонент работы Tesseract. Алгоритм нахождения строк создан так, что перекошенная страница может быть распознана без удаления перекоса, тем самым предотвращая потерю качества изображения. Ключевые части процесса – фильтрация и построение линий. Предполагая, что к анализу макета страницы уже предоставляются области текста примерно одинакового размера текста. Средняя высота приближается к размеру текста в области, поэтому объекты, которые меньше, чем некоторая средняя высота, скорее всего, являются шумом. Отфильтрованные объекты с большей вероятностью подойдут к модели непересекающихся, параллельных, но наклонных линии. Сортировка и обработка объектов по координате *X* делает возможным определять точки к уникальной текстовой строке, а отслеживание наклона по всей странице позволяет значительно уменьшить опасность присвоения неправильной текстовой строки при наличии перекоса. После того как отфильтрованные точки были присвоены линиям, наименьшее среднее из квадратов используется для установки базовых строк и отфильтрованные контуры ставятся обратно в соответствующие строки.

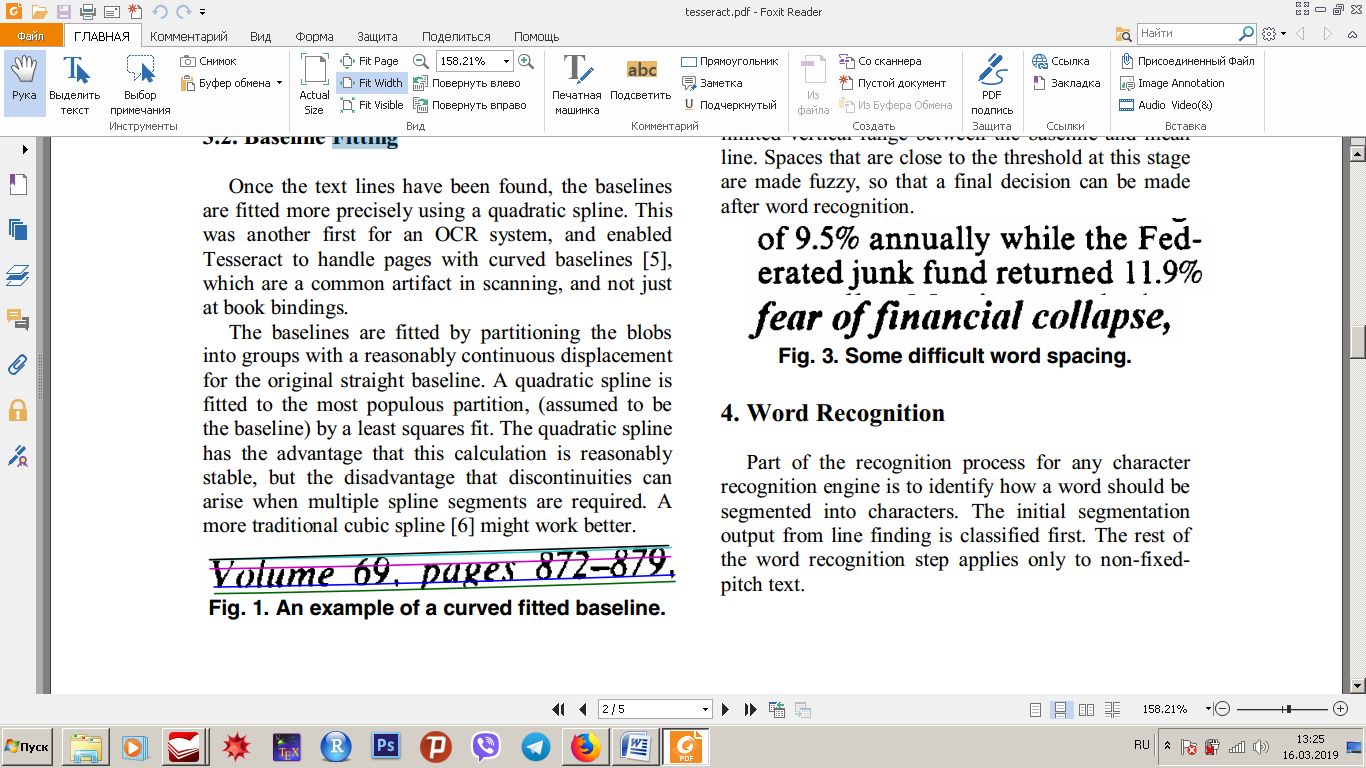
Последний этап процесса установки линий объединяет контуры, которые перекрыли друг друга по горизонтали хотя бы на половину, накладывая корректную базу и корректно соотнесенные части поломанных символов.



*Рис. 1.1. Текст в области*

1. Метод выбора базовой линии.

Как только текстовые строки были найдены, базовые линии установлены более точно с помощью квадратичного сплайна. Это позволяет Tesseract являться программой для обработки страниц с изогнутыми базовыми линиями, которые являются распространенной ситуацией при сканировании.



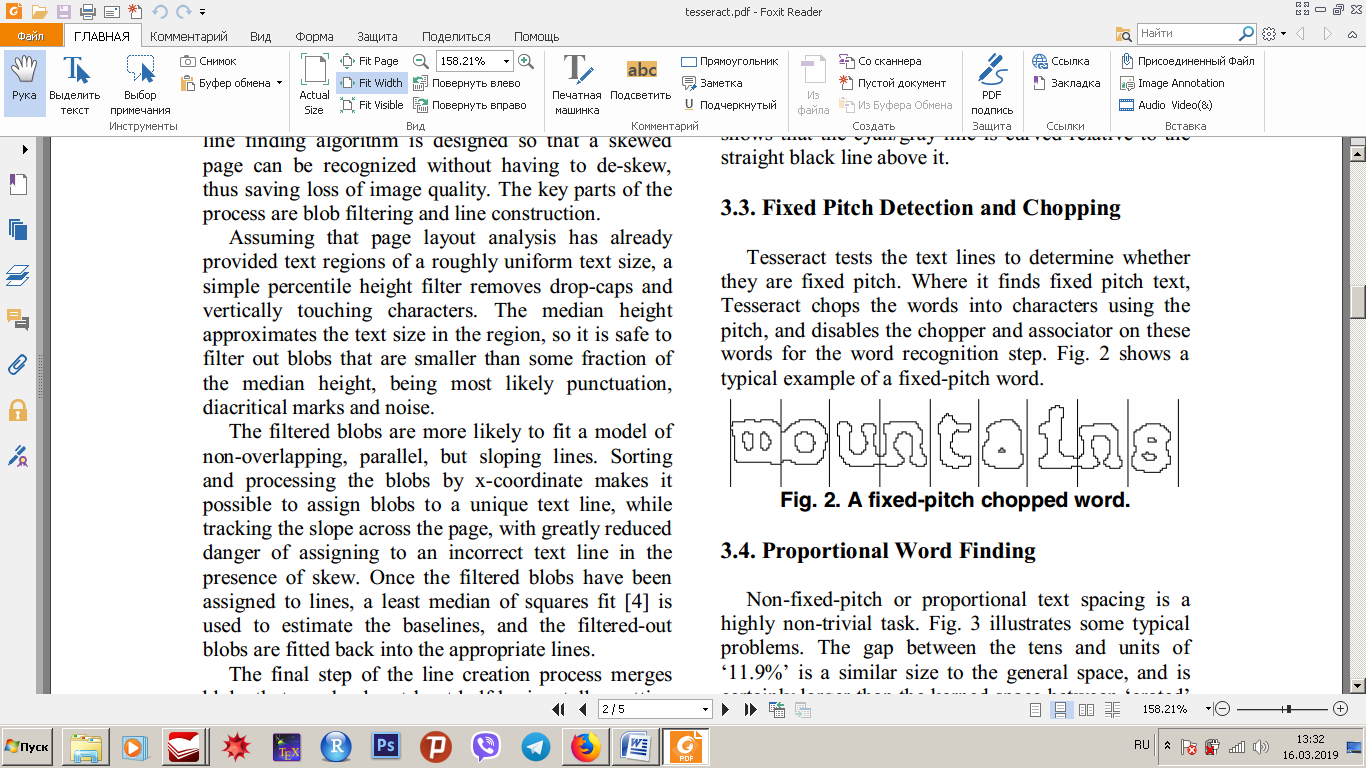
*Рис. 1.2. Базовые линии*

Базовые линии устанавливаются путем разделения точек на группы. Квадратичный сплайн установлен в наиболее густонаселенной области, (предполагается, что это базовая линия) по методу наименьших квадратов. Квадратичный сплайн имеет преимущество в том, что этот расчет является стабильным, но недостаток в том, что разрывы могут возникать, когда требуется несколько сплайновых сегментов. Более традиционный кубический сплайн может работать лучше.

На рис. 1.2. представлены настроенные базовые линии. Все эти линии параллельны (разделение по *Y*) и незначительно изогнуты. Изогнутость можно заметить при сравнении с черной линией, которая точно прямая.

1. Определение фиксированного шага и сегментация слов.

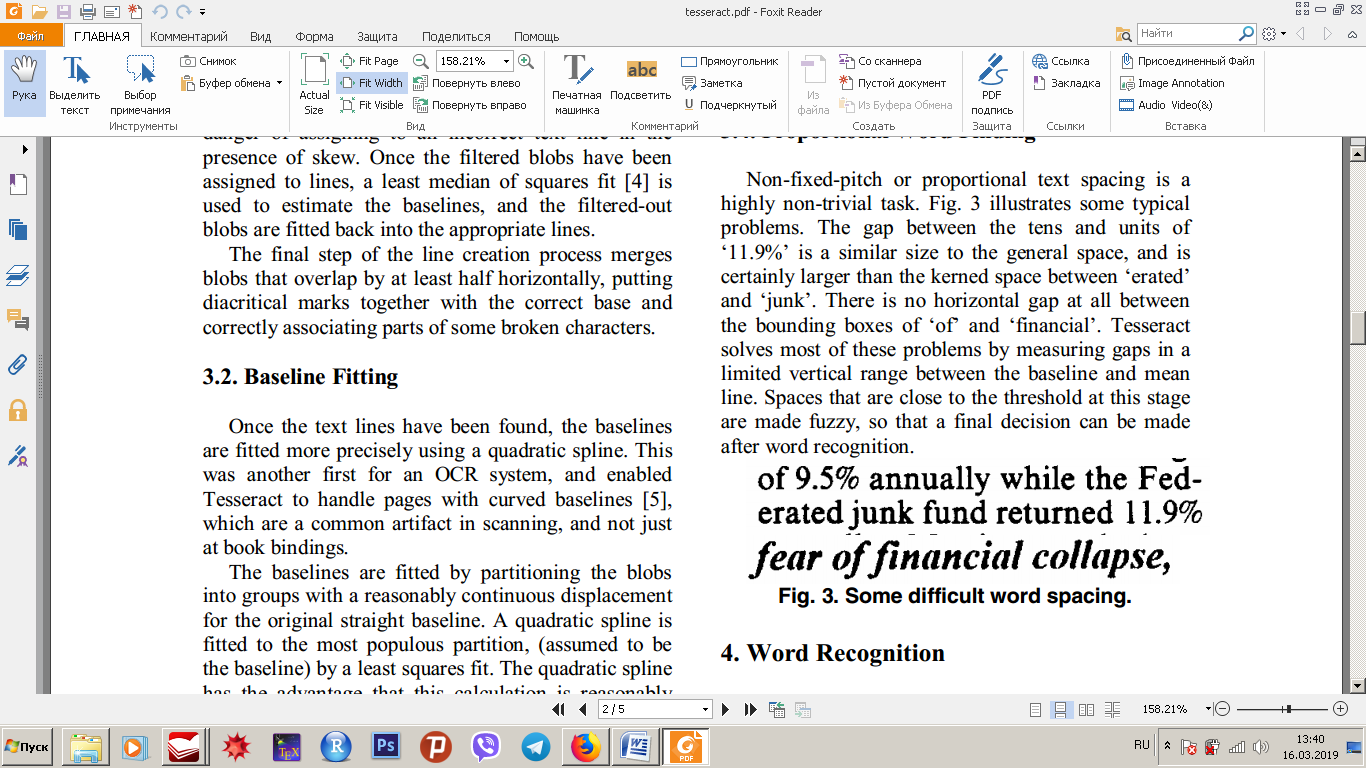
Tesseract проверяет текстовые строки, чтобы определить, с фиксированным ли буквы шагом. Где он находит текст с фиксированным шагом, Tesseract разбивает слова на символы.



*Рис. 1.3. Типичный пример слова с фиксированным шагом*

1. Обнаружение слов.

Нефиксированный шаг или пропорциональный интервал текста весьма нетривиальная задача. [14] Рис. 1.4. иллюстрирует некоторые типичные проблемы. Разрыв между десятыми и единицами «11,9%» такой же как и размер общего пространства и определенно больше, чем пространство между «erated» и «junk». Там нет горизонтального разрыва вообще между ограничительными рамками слов «of» и «financial». Tesseract решает большинство из этих проблем путем измерения зазоров между базовой линией и средней линией. Пространства, которые близки к порогу, сделаны так, что окончательное решение может быть принято после распознавания слова.



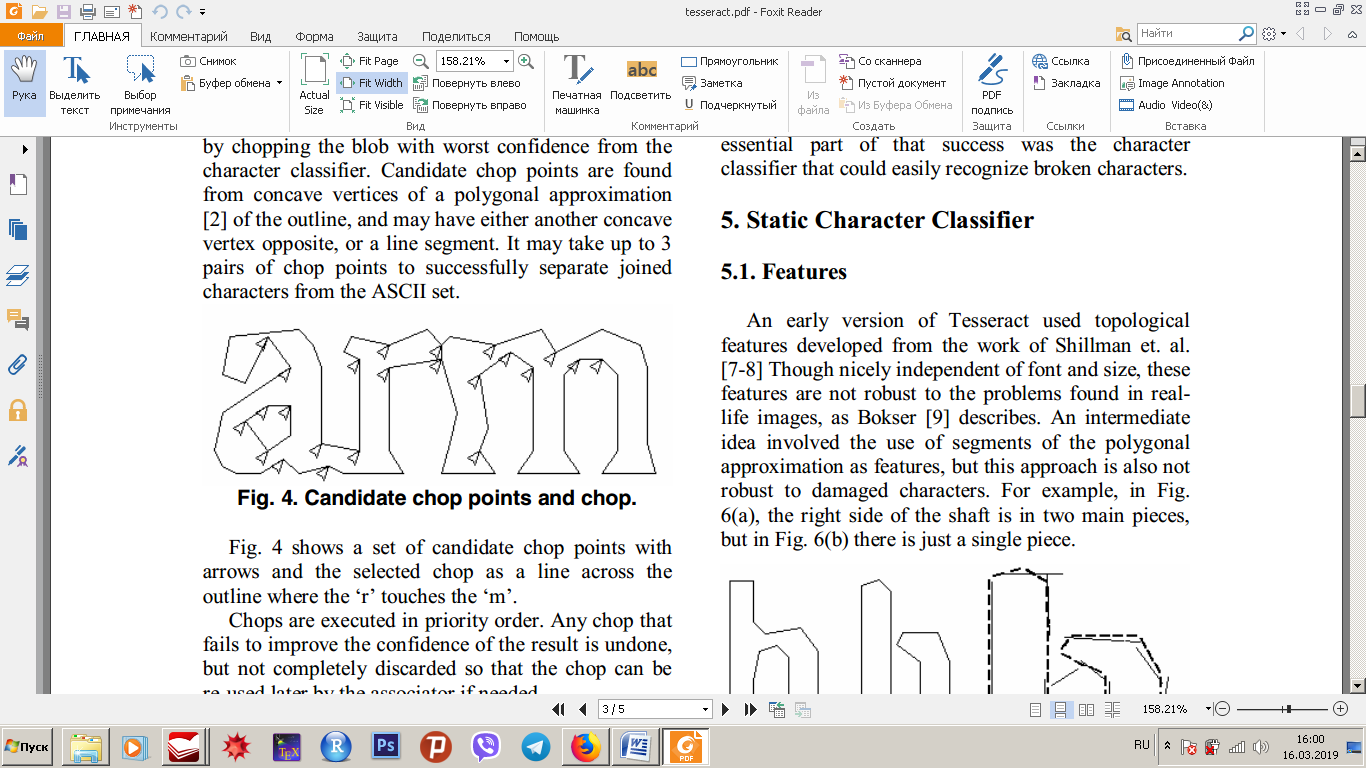
*Рис. 1.4. Текст с нефиксированным шагом*

1. Распознавание слов.

Задача распознавания текста у любого метода распознавания образов заключается в определении того как слово должно быть разбито на символы. Результат сегментации, который был проведен в первую очередь, классифицируется первым. Оставшаяся часть распознавания на этом этапе применяется исключительно к тексту с нефиксированным шагом пробела.

1. Разделение соединенных символов.

Когда результат распознавания слова неудовлетворителен, Tesseract пытается улучшить результат, нарезая символ с худшей классификацией. Кандидатуры точек разделения находятся из вогнутых вершин полигональной аппроксимации границ, и могут быть как другими вершинами, так и сегментами линии. Для успешного разделения связанных символов из «ASCII» набора, необходимы три пары таких точек.

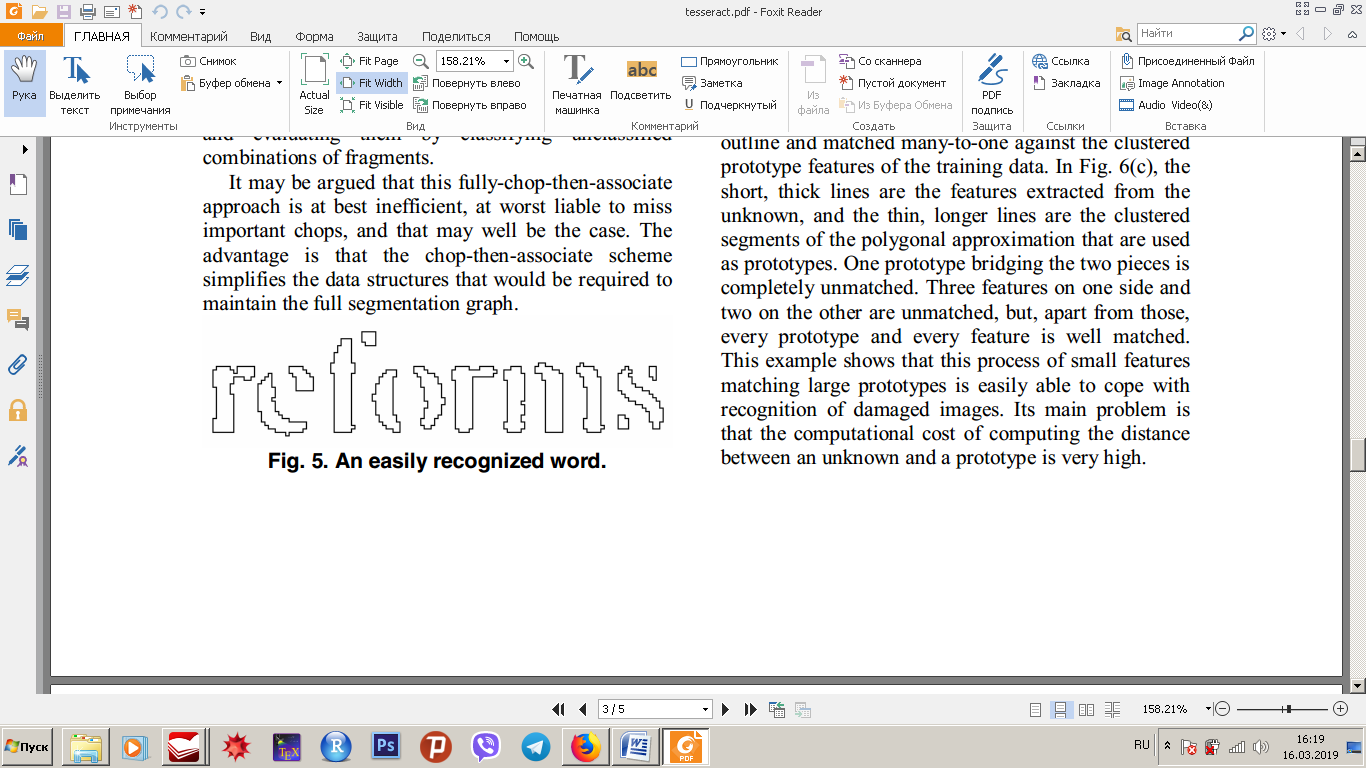


*Рис. 1.5. Возможные точки разреза*

На рис. 1.5. показан набор возможных точек разреза и выбран разрез, где «r» касается «m». Разделение выполняется по порядку. Каждый разрез, который не улучшает результат классификации, не будет выполнен, но он остается памяти, поэтому в дальнейшем в случае необходимости он может быть использован.

1. Распознавание поврежденных символов.

Когда потенциальные разрезы были исчерпаны, если слово все еще недостаточно хорошо распознано, оно передается ассоциатору. Данный модуль совершает «А\*» поиск сегментированных символов в возможной комбинации максимально обрезанных контуров, чтобы найти символ максимально похожий на обрубленный элемент. Данный алгоритм находит маршрут с наименьшей стоимостью от одной вершины (начальной) к другой (целевой, конечной). Он предоставляет кандидату новые состояния из приоритетной очереди и оценивает их, классифицируя неклассифицированные комбинации фрагментов. Можно утверждать, что этот подход в лучшем случае неэффективен, а в худшем случае очень вероятна потеря важных сегментов разделения. Преимущество заключается в том, что эта схема упрощает структуры данных, которые потребуются для поддержки полного графа сегментации.

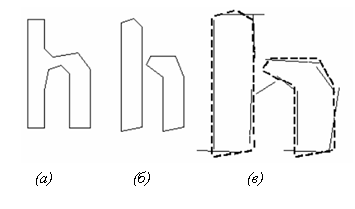


*Рис. 1.6. Пример поврежденных символов*

Когда «А\*» поиск был впервые разработан в 1989 году, точность Tesseract для обработки искаженных символов была наголову выше, чем коммерческие разработки того времени. Значительная часть успеха работы заключается в классификаторе образов, который способен с легкостью распознавать поврежденные элементы.

1. Статический классификатор символов.

Промежуточная идея заключалась в использовании сегментов полигонального приближения в качестве признаков, но этот подход также не устойчив к поврежденным объектам. Например, на рис. 1.7 (а), фигура состоит из одной части, но на рисунке (б) она состоит из двух частей.

**

*Рис. 1.7. (a) Нетронутая «h», (б) сломанная «h»,*

*(в) сопоставление свойств прототипов.*

Прорывным решением является идея о том, что неизвестные свойства не обязательно должны совпадать со свойствами в данных обучения. Во время обучения сегменты полигонального приближения используются для обнаружения особенностей, но при распознавании, свойства небольшой фиксированной длины (в нормализованных единицах) извлекаются из контура и сопоставляются многие к одному с кластерными свойствами прототипов данных обучения. На рис. 1.7 (с) короткие, толстые линии – это элементы, извлеченные из неизвестного, а тонкие, более длинные линии – это кластерные сегменты полигонального приближения, которые используются в качестве прототипов.

Один прототип не совпадает с двумя связанными признаками, но в этом прототипе находит свое отражение каждое из этих свойств по отдельности. Получается, что при сравнении малых признаков с более крупными прототипами, можно добиться хороших результатов при обработке поврежденных символов.

Его главная проблема заключается в том, что затраты на вычисление сходства между неизвестным и прототипом очень высоки. Таким образом, элементы, извлеченные из неизвестного символа, являются 3-мерными (x position, y position, angle), обычно с 50-100 признаками в символе, а элементы-прототипы являются 4-мерными (x position, y position, angle, length) обычно с 10-20 свойствами в конфигурации прототипа.

1. Классификация.

Классификация проходит в два этапа. На первом этапе создается короткий список классов символов, которым может соответствовать неизвестный символ.

Каждый признак выбирается из битового вектора классов, которым этот признак может соответствовать, и битовых векторов просуммированных по всем свойствам. Классы с наибольшим значением (после корректировки ожидаемого количества объектов) становятся окончательным списком для следующего шага.

Каждая особенность неизвестного элемента ищет битовый вектор прототипов выбранного класса, которому он может соответствовать, и затем вычисляется фактическое сходство между ними. Каждый класс символов прототипа представлен логическим выражением суммы продукта, причем каждый термин называется конфигурацией, поэтому в процессе расчета расстояния ведется запись общего сходства каждого элемента в каждой конфигурации, а также каждого прототипа. Наилучшее сходство, которое рассчитывается на основе суммированных признаков прототипа, является наилучшим из всех сохраненных конфигураций класса.

1. Обучение данных.

Из-за того, что классификатор обладает способностью распознавания поврежденных символов, не следует, что классификатор был обучен на поврежденных символах. Фактически, классификатор был обучен только на 20 выборках из 94 символов 8 и различных шрифтов, но с 4 атрибутами (обычный, жирный, курсив, жирный курсив), что составило в общей сложности 60160 обучающих выборок. Это существенно отличается от других опубликованных классификаторов, таких как классификатор Calera с более чем миллионом выборок и классификатор Baird с 100 шрифтами с 1175000 элементами обучения.

1. Лингвистический анализ.

Когда начинается распознавание по новой области, модуль лингвистики выбирает наилучшее слово из доступных в следующих категориях: слово из словаря, часто встречающееся слово, часто встречающаяся цифра, часто встречающееся слово верхнего регистра, часто встречающееся слово нижнего регистра. Среди них классификатор выбирает слово, которое больше всех похоже на слово, которое было подано изначально на вход. А уже окончательное решение по выбранной области будет слово с наименьшим дистанционным рейтингом, в котором каждая из категорий умножается на некоторую константу.

1. Адаптивный классификатор.

Адаптивные классификаторы очень полезны при использовании в системах оптического распознавания. Статичный классификатор достаточно успешно справляется с любыми видами шрифтов, но его способность различать символы и не–символы недостаточно развита. Классификаторы, у которых чувствительность к шрифту выше и которые обучены на основе выходных данных статического классификатора, обладают лучшим умением распознавания символов в документе, где количество шрифтов ограниченно.

Шаблонным классификатором Tesseract не пользуется, но пользуется функциями статического классификатора. Адаптивный классификатор совершает нормализацию по базовым линиям, это и является основным отличием от статического классификатора, если не брать во внимание обученные данные. Такая нормализация делает классификатор более стойким к шумам и делает обнаружение символов верхнего и нижнего регистра проще. А главным достоинством нормализации является отсутствие зависимости от шрифтов, а так же от зависимости от ширины символа. Изображение снизу показывает пример нормализации по базовой линии и форму нормализации по моменту. [14]



*Рис. 1.8. Слово в базовых линия, и нормализованные по моменту буквы*

1. **Подходы к работе с объектами дополненной реальности**

Полагаясь на результаты исследований из статьи [15] на сайте HackerNoon, можно сказать, что к 2025му году объем дополненной и виртуальной реальности в сфере здравоохранения составит примерно 5 миллиардов долларов. А так же огромный прорыв в скором времени произойдет в сфере туризма, т. к. около 84% пользователей со всего мира будут заинтересованы в использовании приложений дополненной реальности в путешествиях.

Опираясь на эти и другие положительные факторы компании-гиганты, такие как Apple и Google, вкладывают большие средства в разработку ARkit и ARCore соответственно. Системы дополненной реальности по виду представляемой информации делятся на [16] :

1. визуальные системы. Основаны на зрительном восприятии человека;
2. аудио системы. Основаны на слуховом восприятии человека;
3. аудиовизуальные системы. Основаны на объединении визуальной и аудио системы.

За основу технологии дополненной реальности берется система оптического трекинга [17]. Маркеры в реальном мире распознаются камерой, она «переносит» маркеры в виртуальную среду, после чего накладывает один слой реальности на другой. Данным образом создается дополненная реальность.

Основные отрасли развития технологии оптического трекинга:

1. «безмаркерная» AR технология.

Данная технология функционирует благодаря специальным методам распознавания местности, где происходит процесс наложения виртуальной «сетки» на ландшафт вокруг нас, который снимается камерой. Далее алгоритмы программы находят опорные точки на «сетке» и уже по ним определяется место, где виртуальный объект будет располагаться. Данная модель обладает следующим преимуществом: нет необходимости создавать специальные идентификаторы на местности для камеры, так как объекты реального мира являются маркерами сами по себе.

1. AR технология, которая базируется на маркерах.

Специально подготовленные метки или маркеры безусловно легче находятся камерой и в следствии они дают более жесткую привязку для объекта виртуальной реальности к месту его расположения. Данная технология работает намного стабильнее в сравнении с «безмаркерной» системой [18].

*Рис. 2.1. Пример маркера*

1. «пространственная» технология.

Данная технология основана на расположении объекта в пространстве. Здесь используются данные GPS-системы, которая встроена в устройство. Место отображения объекта виртуальной реальности зависит от места расположения устройства и использует данные пространственных координат. Программа активируется, когда координаты, которые заложены в программе, совпадают с местоположением устройства.

Работа с дополненной реальностью требует следующего оборудования [19]:

1. графическая станция. Ею может являться телефон, планшет, ноутбук и так далее;
2. дисплей. Экран графической станции;
3. камера устройства. Камера служит средством, которое соединяет окружающий нас мир с виртуальным;
4. маркеры, метки. Точки, к которым объекты виртуальной реальности «прикрепляются»;
5. программное обеспечение. Специальные алгоритмы, которые дают камере возможность распознания метки, определения конкретной модели, которая привязана к этой метке и расположения объекта виртуальной реальности на области.

Технология дополненной реальности по своей сути является программным обеспечением, которое объединяет вышеперечисленные элементы в единую систему, в которой каждая часть выполняет свои собственные функции.

Одной из основных задач системы является определение положения метки в реальном мире по снимку, который был получен через камеру. Распознавание метки происходит в несколько этапов. Изображение, которое было снято камерой, распознается программой на наличие на нем заданного шаблонна – метки. После нахождения маркера задачей системы является построение объекта виртуальной реальности в системе координат изображения камеры и на последнем этапе происходит его привязка к метке, что позволяет виртуальной модели следовать за меткой, это дает камере возможность не быть постоянно статичной.

Однако маркерные технологии так же обладают минусами. Иногда в случае движения камеры устройства объект может «оторваться» от метки или пропасть с экрана. Данная проблема обусловливается тем, что камера больше не распознает метку на изображении. Есть ряд основных причин, по которым камера может прекратить распознавание метки:

1) очень большая или очень низкая степень освещенности метки напрямую влияет на камеру;

2) расположение камеры в пространстве по отношению к метке может быть недостаточно удачно, эта проблема возникает, если маркер попадает в поле зрения камеры только частично или пользователь слишком быстро перемещает камеру, в этом случае большинство камер просто не может уследить перемещение метки из-за ограниченной частоты кадров в секунду;

3) низкое разрешение камеры дает серьезные искажения изображения и метка в жизни не совпадает по внешним характеристикам с меткой, которая была получена из изображения камеры;

4) программное обеспечение, которое было разработано некорректно, серьезно влияет на успешность работы программы.

Аппаратная часть, которая выполняет основные функции дополненной реальности, должна решать следующие основные задачи: получать видеопоток высокого качества, обладать возможностью обработки полученного видеопотока, дополнения объектами виртуальной реальности и вывода результата на устройство для конечного пользователя.

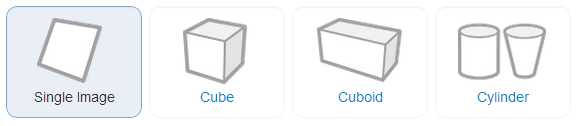
* 1. **Vuforia**

Vuforia [20] — платформа, которая позволяет создавать приложения дополненной реальности для мобильных устройств на операционных системах Android и iOS. Для дальнейшей работы потребуется определить следующее понятие. Мишень – это некоторый реальный объект, который приложение знает и благодаря этому может расставить виртуальные объекты в соответствующих местах.

В качестве ядра системы выступает библиотека QCAR, которая написана на языке C++, которая предоставляет возможности от определения мишени до рендеринга картинки. Кроме QCAR данная система так же содержит Vuforia SDK для Android и iOS; Target Manager – система, которая занимается созданием и управлением мишенями; Vuforia Web Services – позволяет вынести хранение мишеней и включает в себя прочий функционал.

Мишенью называется некоторый объект, за который будет «цепляться» объект виртуальной реальности.

Vuforia предоставляет возможность располагать объекты виртуальной реальности на следующих мишенях:



*Рис. 2.2. Мишени в Vuforia*

1. image target (Single image) – это базовая мишень, которая представляет собой обычную плоскость: лист бумаги, обложка книги, поверхность стола или напечатанное фото и т. д. Такие объекты выступают в роли двумерной плоскости. По ней будет определяться, какой именно объект попал в объектив камеры, его размеры и положение в пространстве. Однако, не любая картинка может являться хорошей мишенью. А удачной мишенью будут являться изображения с контрастными деталями. Ведь на них будет строиться опорная матрица для распознавания мишеней;
2. simple 3D target (Cube и Cuboid) – это мишени, которые представляют собой прямоугольные параллелепипеды. Примером могут послужить: коробка из-под чего-либо, спичечный коробок и т. д. Любая мишень такого типа состоит не из одной, а уже из шести плоскостей. И здесь уже понадобится шесть картинок каждой плоскости;
3. cylinder target – это мишени, которые представляют собой цилиндр или усеченный конус. Для задания мишеней такого типа необходимо указать диаметры оснований и высоту, изображения обоих оснований и еще изображение боковой поверхности;
4. frame markers – данная мишень представляется в виде заранее подготовленной рамки. Такой подход используется в случае, если изображение, которое поместится в рамку слабо детализировано, обладает низкой контрастностью и оно не подходит под формат image target;
5. text (word target) – в систему Vuforia также встроена система распознавания символов, но пока она работает только с латиницей. Поэтому уже мишенью может служить любое слово из английского языка. Текстовые мишени будут полезны при создании приложений для обучения иностранным языкам.

Все мишени, которые необходимы приложению, могут храниться в Device Database. Это позволяет работать приложению в offline режиме, но данный метод хранения мишеней больше подходит при работе с небольшим объемом мишеней, в другом случае, когда хранить базу всех мишеней на устройстве не является целесообразным, помогает Cloud Database – это сервис из Vuforia Web Services, который предназначен для удаленного хранения базы и предназначен для определения мишеней, на основании присланных данных устройством.

Также мишени могут быть созданы и из самого приложения, беря за основу изображение, полученное с камеры устройства.

Следует отметить, что QCAR не требует попадания в кадр абсолютно всей мишени. Для ее распознавания достаточно попадания в поле зрения лишь ее части.

* 1. **ARToolKit**

ARToolKit [21] является набором программных библиотек, широко используемых в работе с дополненной реальностью. Открытый исходный код является одной из особенностей этой системы, так как большинство современных систем, предназначенных для работы с дополненной реальностью, не дают пользователям возможности увидеть код. Однако документация все равно достаточно ограничена. На таких платформах как Windows, Linux, iOS и Android идёт постоянная поддержка. ARToolKit является системой, которая поддерживает одну из самых популярных сред разработки кроссплатформенных игр – Unity. ARToolKit обладает рядом особенностей, которые включают в себя: стабильную систему отслеживания маркера, поддержку калибровки камеры, отслеживание нескольких объектов одновременно, способность программирования на нескольких языках, оптимизация под мобильное устройство и абсолютная поддержка Unity3D.

Процесс работы ARToolKit напрямую связан с наложением виртуальных объектов на реальные, путем нахождения квадратных маркеров. Квадратный маркер обведен толстой черной границей и содержит внутри себя уникальное изображение, определенное разработчиком заранее, что и делает маркеры уникальными. Такие изображения называются шаблонами. Каждый квадратный маркер распознается и отслеживается, а после он используется для расчета места расположения объектов дополненной реальности в пространстве.

Принцип работы ARToolKit:

1. происходит захват видео полем зрения камеры;
2. отправка видео компьютеру;
3. программное обеспечение рассматривает каждый кадр видео и ищет на нем квадратные формы, которые могут являться маркерами;
4. когда маркер, содержащий в себе шаблон, найден на каком-то из кадров и он идентифицирован в программе, то наступает процесс расчета положения маркера и его ориентации относительно камеры с помощью математических формул программы;
5. далее происходит отрисовка объекта виртуальной реальности, основываясь на полученных ранее результатах расчета положения и ориентации;
6. отрисовка модели осуществляется на переднем плане видео и сопоставляется с движением фонового видео, что позволяет модели «закрепиться» на общей картинке;
7. конечный пользователь видит модель на экране дисплея и создается иллюзия, что объект находится внутри пространства, но на самом деле он видит модель над видео потоком в реальном времени.

Так же были рассмотрены следующие фреймворки: Wikitude, Lay AR, Kudan AR, Metaio SDK, String, Catchroom, Infinity AR, Semapedia, ARTag, 2Nowa, Ailowe, Wi2Geo. Все вышеперечисленные системы обладают способностью дополнять окружающий нас мир дополнительной информацией и основываются на безмаркерной, маркерной или пространственных технологиях [22].

* 1. **Подходы к разработке объектов дополненной реальности**

Одной из основных составляющих в работе с дополненной реальностью являются сами объекты дополненной реальности. Ведь именно на них пользователь смотрит через экран своего устройства. И сила воздействия на пользователя напрямую зависит от качества и привлекательности элементов дополненной реальности. Это значит, что перед каждым разработчиком стоит вопрос выбора оптимального редактора для создания элементов дополненной реальности. В дальнейшем элементами дополненной реальности будут являться 3D модели достопримечательностей городов Крыма, а это значит, что предстоит рассмотреть различные 3D редакторы и определить, какой из них будет больше подходить для решения поставленных задач.

* + 1. **Blender**

Проект Blender [23] – это бесплатная альтернатива профессиональным коммерческим 3D редакторам. Одной из его особенностей является возможность абсолютно любого разработчика присоединиться к его усовершенствованию и добавлению новых опций. Многие из инструментов, которые существуют в Blender, были добавлены пользователями для решения своих конкретных задач и нашли широкое применение и среди остальных пользователей. Так же стоит отметить, что, в сравнении с другими профессиональными редакторами, Blender занимает всего несколько десятков мегабайт памяти компьютера. И одним из самых главных преимуществ этого редактора является кроссплатформенность. Работает на операционных системах Windows, Linux и Mac OS. Так же в его минимальных требованиях прописаны достаточно низкие показатели систем, которые могут с ним функционировать, однако быстродействие программы сильно зависит от системных показателей.

Программа включает в себя широкий спектр инструментов создания трехмерной графики. Например, есть возможность подвергать анимированные объекты воздействию ветра, контролировать отдельный вес каждого элемента, выбирать направляющие анимации, создавать эффекты жидких тел, моделировать поведение мягких тел и много другое. Интерфейс программы изначально кажется перегруженным, но после короткого отрезка работы с программой такое ощущение пропадает.

* + 1. **3ds Max**

Редактор 3ds Max [24] достаточно популярен и ориентирован на создание объектов архитектуры. Главной его особенностью является большая база уже готовых объектов, которые пользователь имеет возможность использовать в своем проекте. Также в программу встроена возможность просмотра объекта в сильно приближенном к реальности формате, этого удалость добиться путем интеграции фотореалистичного визуализатора. Но кроме встроенных визуализаторов пользователь также может использовать и сторонние средства визуализации – V-ray и Maxwellrender.

Кроме того, функциональный спектр программы сильно увеличивается за счет большой базы плагинов. Плагин Dreamscape позволяет моделировать различные природные ландшафты, акватории и многое другое, а плагин Afterburn дает пользователю возможность моделирования очень реалистичных взрывов. Но и без подключения специальных плагинов пользователь самостоятельно может писать скрипты для анимации моделей.

* + 1. **ZBrush**

ZBrush [25] отличается от остальных 3D редакторов тем, что здесь в моделировании 3D-объектов используется только техника трехмерной лепки. Данный метод заключается в том, что пользователь, используя виртуальные инструменты, словно лепит из пластилина объекты, придавая ему нужную форму, вытягивая или вдавливая нужные ему области поверхности объекта. Данный метод очень практичен и прост для освоения. На сегодняшний день данное приложение можно считать лидером в области трехмерной лепки. Кроме того, ZBrush поддерживает импорт и экспорт файлов в формате .obj, что позволяет переносить объекты из других 3D редакторов и дорабатывать их в режиме лепки. Однако для создания анимации объектов этот редактор не подходит, так как тут попросту отсутствуют инструменты добавления анимации объектам.

Так же были рассмотрены такие программы 3D редактирования как: Cinema 4D, Maya, Vue9.5 xStream. Но каждая из этих программ направлена на решение своих конкретных задач: работа с видео, мультипликацией и работа с ландшафтами соответственно [26] .

1. **Разработка комплекса визуализации объектов дополненной реальности при посещении городов Крыма**
   1. **Распознавание текста**

Был выбран способ распознавания текста при помощи библиотеки Tesseract. Tesseract обладает множеством преимуществ, опираясь на которые эта оптическая система и была выбрана:

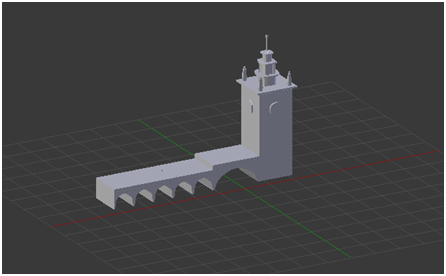
1. Обладает открытым исходным кодом.
2. Показывает отличные результаты на черно-белых изображениях.
3. Высокая скорость и простота реализации задачи распознавания текста на своей базе.
4. Имеет возможность работать с библиотеками компьютерного зрения (openCV).
5. Это не коммерческий продукт.
   1. **Создание объектов дополненной реальности**

Для разработки объектов дополненной реальности было выбрано программное обеспечение для создания трёхмерной компьютерной графики Вlender. Преимуществами выбранного программного обеспечения является широкий спектр возможностей, простота использования и отсутствие необходимости платить за его использование.

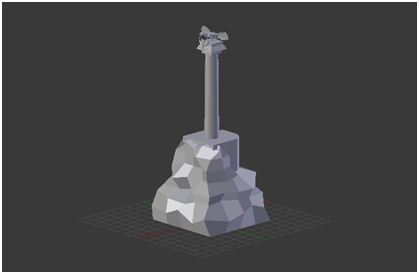
* 1. **Визуализация элементов дополненной реальности**

Для расположения 3D объектов в камере устройства была выбрана библиотека Vuforia.

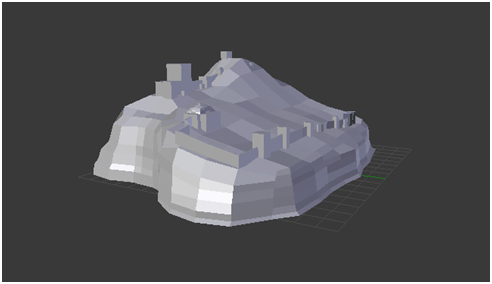
Первым этапом создания приложения было создание 3D моделей достопримечательностей для четырех городов – вокзал Симферополь-Пассажирский города Симферополь, Памятник затопленным кораблям города Севастополь, Генуэзская крепость города Судак и Храм Святого Иоана Предтечи города Керчь. Достопримечательности были выбраны таким образом, чтобы показать различные виды достопримечательностей: архитектурный, военный, исторический и религиозный памятники соответственно. Были получены следующие результаты:

**

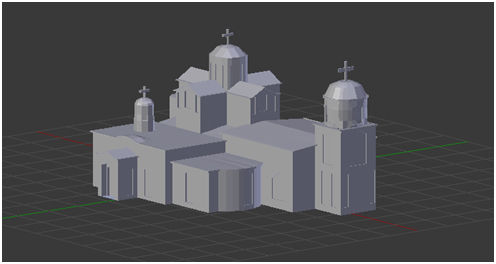
*Рис. 3.1. 3D-модель Симферополь*

**

*Рис. 3.2. 3D-модель Севастополь*

**

*Рис. 3.3. 3D-модель Судак.*

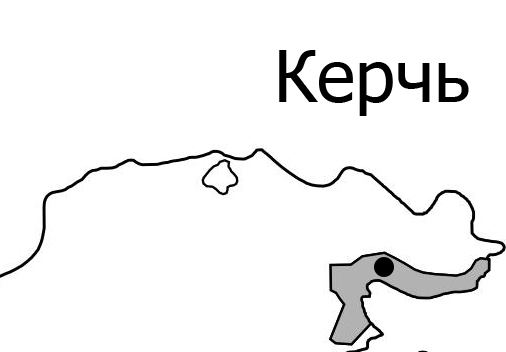
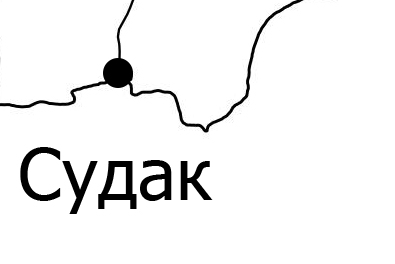
**

*Рис. 3.3. 3D-модель Керчь.*

Вторым шагом создания приложения была реализация задачи распознавания текста на изображении. Изображение карты было выбрано специальным образом, одним из критериев выбора изображения являлась его цветовая палитра, наивысшим процентом удачного распознавания обладают черно-белые изображения, исходя из этого соображения, карта была выбрана именно такого цвета. Так же небольшой корректировке подверглись и некоторые линии на карте (при помощи программы PhotoShop), которые, являясь шумами, уменьшали вероятность успешного распознавания. Для лучших результатов распознавания карта была разбита на 4 изображения (т. к. работа идёт с четырьмя городами) с одним названием города на одном участке карты.

*Рис. 3.5. Карта, Симферополь*

*Рис. 3.6. Карта, Севастополь*

*Рис. 3.7. Карта, Керчь*

*Рис. 3.8. Карта, Судак*

Процесс распознавания реализован следующим образом. После наведения камеры на изображение с текстом необходимо сделать фотографию области сканирования и после этого на экране появится отсканированный текст.

Приложение для распознавания реализовано двумя скриптами, отвечающими за разные задачи. Один создает фотографию области текста, и передает ее второму скрипту для распознавания текста на сфотографированном изображении. Первый скрипт работает следующим образом: из массива подключенных камер (может быть не одна камера) выбирается порядковый номер камеры, которую будем использовать. Инициализируем текстуру изображением с первой камеры. Каждый раз вызывается метод, который обновляет изображение выводимое на экран камерой. При нажатии на кнопку «Photo» вызывается метод «TakePhoto», который не заканчивается пока не будет выполнен снимок. Снимок делается следующим образом: создается текстура по высоте и ширине как используемая камера, сохраняются в текстуру пиксели с текущего изображения камеры, создается массив байтов, куда перекодируется изображение в png, запись массива байтов по указанному пути, вызывается у второго скрипта метод GetText, куда передаётся путь к сохранённому фото.

Принцип работы второго скрипта: создаётся экземпляр Tesseract'a и инициализируется данными для распознавания на русском языке. В переменную Pix p сохраняется изображение по переданному ранее пути. В результате обработки изображения данные записываются в переменную Page page. Получим первое слово распознанного текста, запишем это слово в текстовое поле (надпись) на экране, вернём это слово.

После написания программы распознавания будем использовать ее как базу для создания конечного приложения.

Последним шагом является добавление дополненной реальности в приложение. Для этого выбрана среда разработки Unity вместе с библиотекой Vuforia, которую можно установить через Unity Hub вместе с GameObjects. Создаем сцену и удаляем с неё камеру, т. к. будет использоваться AR камера с GameObjects, которые были скачаны ранее.

AR камера – это виртуальная камера, которая имитирует камеру смартфона. Далее импортируем скрипты Vuforia Core Samples из Asset Store. Для последующей работы с Vuforia требуется зарегистрироваться на сайте https://developer.vuforia.com/ для получения ключа, который необходимо ввести в настройках Vuforia.

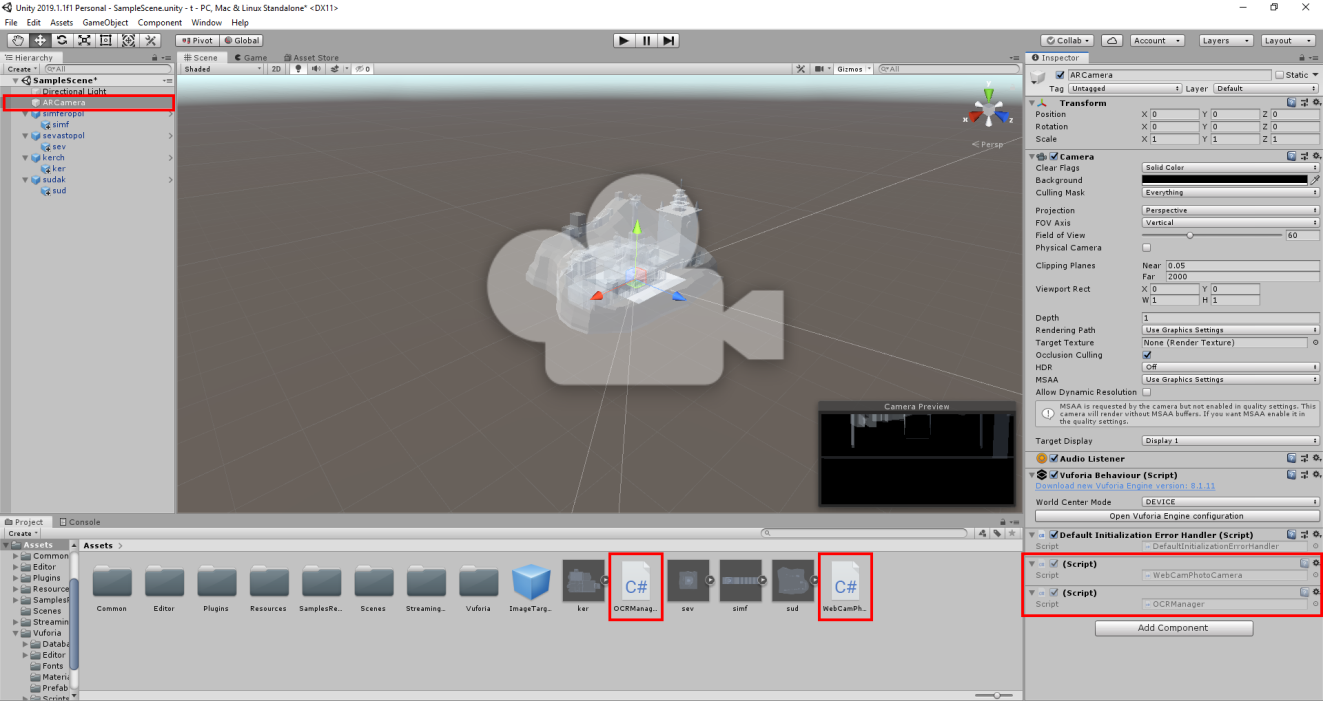
После чего выбирается камера, с которой будет вестись работа в процессе разработки, а так же указывается частота ее кадров в 30 fps. На сцену добавляются 3D объекты. Для их добавления файлы из Вlender сохраняются в формате .fbx в папке проекта и переносятся на сцену Unity. 3D объекту также можно задать размер, координаты и материал. На сцене сразу может быть расположено несколько объектов.

Далее создается Prefab. Это специальный тип ассетов, который позволяет хранить GameObject со всеми значениями свойств и компонентами. Эта опция очень полезна когда требуется, чтобы все экземпляры отдельно взятого объекта имели одинаковые значения свойств, чтобы при редактировании одного объекта такого типа в сцене не приходилось повторно вносить те же изменения во все остальные копии.

*Рис. 3.9. Prefab*

Далее делается Prefab из Image Target. Каждый Prefab именуется названием города и потом от распознанного текста вызывается нужный Prefab. И в качестве Image Target будет считаться сам текст. После чего располагаем в Unity 3D модель относительно Image Target. Vuforia считывает текст как изображение, определяет его как Image Target после чего находит центр Image Target, относительно которого будет происходить возведение 3D модели.

В значении поля Database ставится значение “EMPTY”, это делается для того чтоб оно не искало никаких баз данных с таргетными изображениями. После чего в коде скрипта OCRManager, который был написан ранее, подключается using’и, которые позволяют работать с Vuforia (using System.Collections.Generic и using Vuforia) и добавляется конструкция switch/case, которая дает возможность, в зависимости от параметра result (результат распознавания), обращаться к нужному Prefab’у. Полученный результат распознавания в case записываем в нижнем регистре (result = result.ToLower()), чтобы распознанный текст с большей вероятностью подошел под условие. В скрипте WebCam так же добавляются using’и для работы с Vuforia. Далее эти два скрипта, в которых реализован видеопоток и частота обновления кадров, прикрепляются к AR камере.

*Рис. 3.10. Скрипты, прикрепленные к камере*

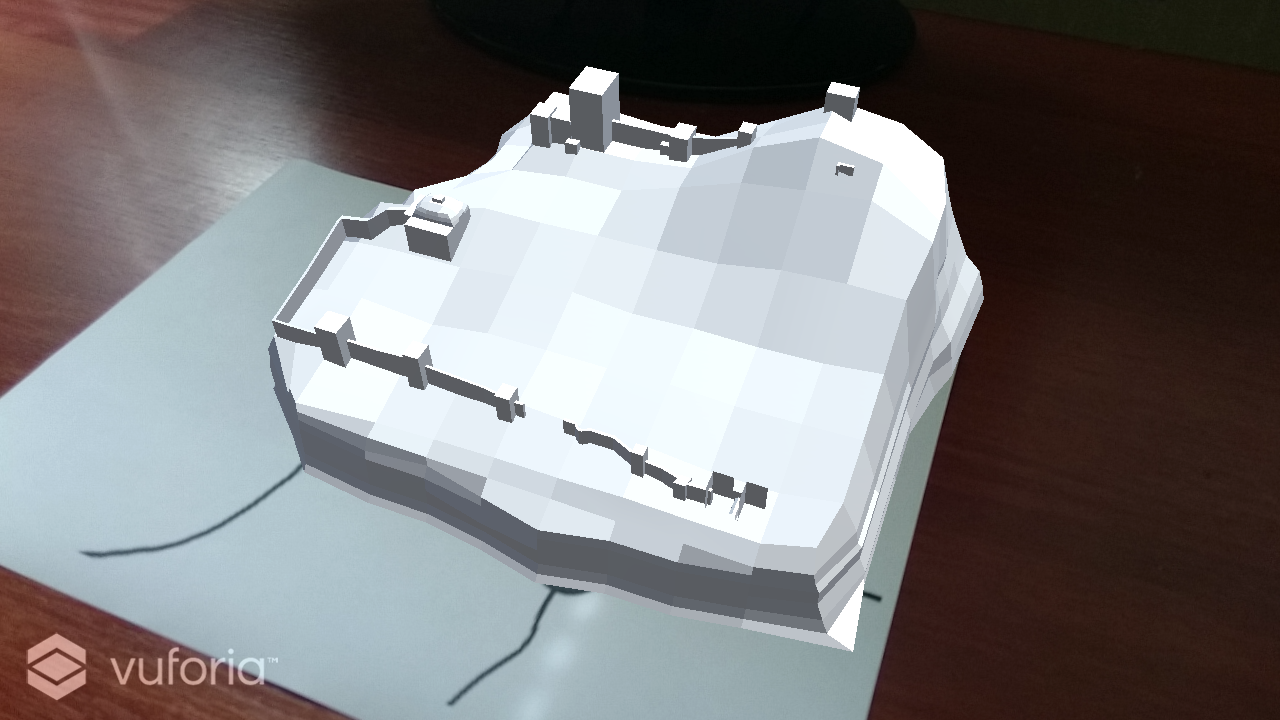
После выполнения всех перечисленных действий, проект был сохранен в формате .apk для последующей возможности использования приложения с помощью телефона.

Ниже будут представлены скрин-шоты работы приложения:

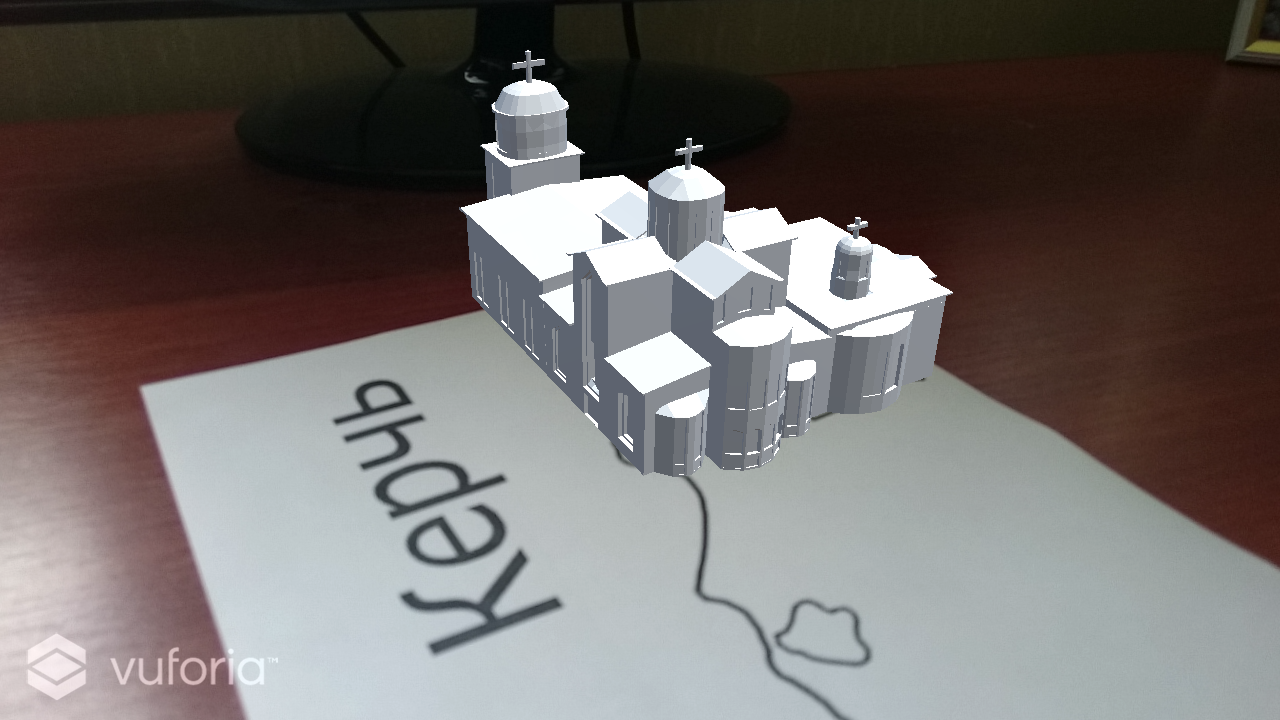
*Рис. 3.11. Результат, Симферополь*

****

*Рис. 3.12. Результат, Севастополь*

****

*Рис. 3.13. Результат, Судак*

****

*Рис. 3.14. Результат, Керчь*

**Заключение**

В результате работы было разработано приложение, которое хорошо работает на черно-белых картах. Для усовершенствования предстоит включить в модуль, который будет заниматься предобработкой изображения, что значительно повысит точность распознавания названий городов и на цветных картах. Так же предстоит расширить спектр городов, достопримечательности которых так же будут находиться на карте. Некоторые города имеют несколько характерных достопримечательностей, из этого следует, что предстоит разработка нескольких 3D-моделей города и будущий интерфейс пользователя может получать информацию о типах достопримечательностей, которые хочет посетить пользователь и предоставлять ему только те, которые соответствуют данной категории, а если в городе несколько достопримечательностей одной категории, то будет возможен их поочередный просмотр.

Технология дополненной реальности в применении к туристической отрасли имеет огромный спектр возможностей. Окончательным вариантом приложения будет программа, которая, кроме визуализации достопримечательностей, будет способна проложить путь к необходимому месту с возможностью выбора способа передвижения пользователя и провести на нем экскурсию.

У модуля оптического распознавания Tesseract есть недостаток – это относится к поддержке русского языка, она есть, но данные обучены недостаточно в сравнении с интернациональными языками. Но при необходимости данные обучения можно создать и самостоятельно. Или через некоторое время разработчики могут их усовершенствовать.

# Приложение

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using System.IO;

using UnityEngine.UI;

using Vuforia; //подключение библиотеки Vuforia

public class WebCamPhotoCamera : MonoBehaviour

{

//изображение с веб-камеры

WebCamTexture webCamTexture;

//UI элемент, на который выводится изображение с камеры

[SerializeField]

RawImage rICamera;

//переменная типа Texture2D

Texture2D curPhoto;

[SerializeField]

OCRManager ocrManager;

//путь к сделанному фото

string path;

//вызывается один раз при запуске сцены

void Start()

{

path = Application.streamingAssetsPath;

ocrManager = GetComponent<OCRManager>(); //получить ссылку на экземпляр OCRManager

WebCamDevice[] devices = WebCamTexture.devices; //массив подключённых камер

webCamTexture = new WebCamTexture(devices[0].name); //инициализируем текстуру с изображением первой камерой

webCamTexture.Play(); //метод, начинающий получать изображение с камеры

}

//вызывается каждый кадр

void Update()

{

rICamera.texture = webCamTexture; //вывод на экран изображения с камеры

}

//вызывается при нажатии на кнопку "Photo"

public void Photo()

{

StartCoroutine(TakePhoto()); //запуск корутины

}

//метод (корутина), делающий фото

IEnumerator TakePhoto()

{

yield return new WaitForEndOfFrame(); //не закончит выполнение метода, пока не сделается снимок

Texture2D photo = new Texture2D(webCamTexture.width, webCamTexture.height); //создаём текстуру, по высоте и ширине как наша камеры

photo.SetPixels(webCamTexture.GetPixels()); //сохраняем в эту текстуру пиксели с текущего изображения камеры

photo.Apply();

byte[] bytes = photo.EncodeToPNG(); //создали массив байтов, куда перекодировали сделанное фото

File.WriteAllBytes(Path.Combine(path, "photo.png"), bytes); //записали массив байт по указанному ранее пути (папка StreamingAssets в проекте)

ocrManager.GetText(Path.Combine(path, "photo.png")); //вызываем у OCRManager'a метод GetText, куда передаём путь к сохранённому фото

curPhoto = photo; //в переменную "текущее фото" (curPhoto) присваеваем сделанное фото

}

}

*Листинг 1.1. Скрипт 1*

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using System;

using System.IO;

using Tesseract; //подключение библиотеки Tesseract

using UnityEngine.UI;

using Vuforia; //подключение библиотеки Vuforia

public class OCRManager : MonoBehaviour

{

//UI элемент (надпись), куда будет сохраняться распознанный текст

[SerializeField]

public Text resultText;

string dataPath;

void Start()

{

dataPath = Path.Combine(Application.streamingAssetsPath, "tessdata"); //запишем путь к данным для инициализации Tesseract'a

}

//распознаёт текст на изображении по пути path и возвращает этот текст

public string GetText(string path)

{

var ocr = new TesseractEngine(dataPath, "rus", EngineMode.Default); //создаём экземпляр Tesseract'a и инициализируем его данными для распознавания на русском языке

Pix p = Pix.LoadFromFile(path); //сохраняем в переменную Pix p картинку по переданному ранее пути

Page page = ocr.Process(p); //запишем данные в результате обработки картинки в переменную Page page

var result = page.GetText().Split(' ', '\n')[0]; //получим первое слово распознанного текста

result = result.ToLower(); //записываем слово в нижнем регистре

return result; //вернём это слово

switch (result)

{

case "симферополь": ApplicationId.GameObject.Find("simferopol");

case "севастополь": ApplicationId.GameObject.Find("sevastopol");

case "керчь": ApplicationId.GameObject.Find("kerch");

case "судак": ApplicationId.GameObject.Find("sudak");

}

}

}

*Листинг 1.2. Скрипт 2*

**Список источников**

* + - 1. Макогон М. О. Дополненная реальность и ее применение / М. О. Макогон, А. С. Долгих. – Иркутск. 2016. – С. 173 – 175.
      2. Виртуальная и дополненная реальность в медицине [Электронный ресурс] – Режим доступа: URL: https://make-3d.ru/articles/virtualnaya-i-dopolnennaya-realnost-v-medicine/.
      3. Мобильный гид с дополненной реальностью [Электронный ресурс] – Режим доступа: URL: https://4pda.ru/2018/01/15/349175/.
      4. Optical character recognition [Электронный ресурс] – Режим доступа: URL: [https://en.wikipedia.org/wiki/Optical\_character\_recognition#Techniques](https://en.wikipedia.org/wiki/Optical_character_recognition" \l "Techniques).
      5. NicomSoft [Электронный ресурс] – Режим доступа: URL: <https://www.nicomsoft.com/optical-character-recognition-ocr-how-it-works/>.
      6. Методы распознавания текста [Электронный ресурс] – Режим доступа: URL: <https://habr.com/ru/post/220077/>.
      7. Распознаем текст, используя расстояние Хэмминга [Электронный ресурс] – Режим доступа: URL: <https://habr.com/ru/post/90867/> .
      8. Декодирование капчи на Python [Электронный ресурс] – Режим доступа: URL: <https://habr.com/ru/post/149091/> .
      9. Разве Tesseract распознаёт медленно? [Электронный ресурс] – Режим доступа: URL: <https://habr.com/ru/company/smartengines/blog/300990/>.
      10. Лапкина, А. Л. Распознавание текста по структуре скелета букв/ А. Л. Лапкина. – М. 2018. – С. 3 – 19.
      11. Текстовые капчи легко распознаются нейронными сетями глубокого обучения [Электронный ресурс] – Режим доступа: URL: <https://habr.com/ru/post/343222/>.
      12. Пешков, А. О. Анализ методов бинаризации изображений / А. О. Пешков, В. К. Крят. – Ростов-на-Дону. 2015. – С. 2 – 7.
      13. Smith, R. An Overview of the Tesseract OCR Engine / – С. 1 – 5.
      14. Quora [Электронный ресурс] – Режим доступа: URL: <https://www.quora.com/How-does-the-Tesseract-API-for-OCR-work>.
      15. HackerNoon [Электронный ресурс] – Режим доступа: URL: <https://hackernoon.com/predictions-for-the-future-of-augmented-reality-63c7b8c9d794>.
      16. Смолин, А. А. Системы виртуальной, дополненной и смешанной реальности / А. А. Смолин, Д. Д. Жданов, И. С. Потемин, А. В. Меженин, В. А. Богатырев. – СПб. 2018. – С. 45 – 54.
      17. Funreality [Электронный ресурс] – Режим доступа: URL: <https://funreality.ru/technology/augmented_reality/>.
      18. Шакиров, И. Ш. «Дополненная реальность»: инновационная технология организации образовательного процесса по информатике / И. Ш. Шакиров. – Лесосибирск. 2016. – С. 10 – 36.
      19. Константинова, К. Э. Исследование возможностей и опыта использования технологий дополненной реальности в экспозиционно-выставочной деятельности музеев / К. Э. Константинова. – М. 2018. – С. 6 – 13.
      20. Vuforia: немного магии в нашей реальности [Электронный ресурс] – Режим доступа: URL: <https://habr.com/ru/post/198862/>.
      21. Dgng [Электронный ресурс] – Режим доступа: URL: <http://dgng.pstu.ru/conf2017/papers/125/>.
      22. Методы распознавания текстов [Электронный ресурс] – Режим доступа: URL: <https://habr.com/ru/post/112442/>.
      23. 3D редакторы, плюсы и минусы [Электронный ресурс] – Режим доступа: URL: <https://habr.com/ru/post/136350/>.
      24. Оптическое распознавание символов в Linux [Электронный ресурс] – Режим доступа: URL: <https://habr.com/ru/post/153617/>.
      25. Бинаризация изображений: алгоритм Брэдли [Электронный ресурс] – Режим доступа: URL: <https://habr.com/ru/post/278435/>.
      26. Сегментация изображения [Электронный ресурс] – Режим доступа: URL: <https://habr.com/ru/post/128768/>.
      27. **What's OCR?** [Электронный ресурс] – Режим доступа: URL: <http://www.dataid.com/aboutocr.htm>.
      28. Использование Open Source OCR библиотеки Tesseract в Android на примере простого приложения [Электронный ресурс] – Режим доступа: URL: <https://habr.com/ru/post/282582/>.
      29. Как создать приложение дополненной реальности с помощью ARCore [Электронный ресурс] – Режим доступа: URL: <https://habr.com/ru/post/439190/>.
      30. Что такое ARCore? Всё, что вам нужно знать [Электронный ресурс] – Режим доступа: URL: <https://habr.com/ru/post/437378/>.
      31. Создание вашего первого ARCore-приложения [Электронный ресурс] – Режим доступа: URL: <https://habr.com/ru/post/438178/>.
      32. Создание AR-игры с помощью Vuforia [Электронный ресурс] – Режим доступа: URL: <https://habr.com/ru/post/440592/>.
      33. Azoft [Электронный ресурс] – Режим доступа: URL: http://www.azoft.ru/ blog/biblioteki-dopolnennoj-realnosti-dlya-ar-prilozhenij/.
      34. Афонасенко, А. В. Обзор методов распознавания структурированных символов / А. И. Елизаров. – М.: Технические науки, 2008. – С. 83 – 87.
      35. Вахрушева, Т. С. Применение технологий дополненной реальности в образовании / Т. С. Вахрушева. – СПб. 2017. – С. 33 – 40.
      36. Меженин, В. А. Богатырев. – СПб. 2018. – С. 45 – 54.
      37. GOLOS [Электронный ресурс] – Режим доступа: URL: <https://golos.io/vox-populi/@vp-webdev/raspoznavanie-teksta-na-izobrazhenii>.