**Доклад**

Слайд 2

Мною была подготовлена выпускная квалификационная работа на тему **Разработка объектов дополненной реальности для визуализации туристических объектов Крыма.**

**Целью работы** является разработка программного обеспечения реализующего визуализацию туристических объектов Крыма в дополненной реальности.

**Задачи**, решаемые в работе вы можете увидеть на слайде:

Слайд 3

В ходе подготовки выпускной работы были рассмотрены этапы создания приложения дополненной реальности с распознаванием текста. Вся работа была разделена на две центральные проблемы: **Распознавание текста** и **Дополненная реальность**. И у этих двух проблем был свой список задач, необходимых к решению. Их вы тоже можете увидеть на слайде.

Слайд 4

Первая проблема – это процесс распознавания текста. Он может быть реализован при помощи четырёх основных методов:

1. **Шаблонные методы**

В таких методах необходимо использовать метрику – некое условное значение функции, определяющее положение объекта в пространстве. Если два объекта близко друг к другу, то метрики для таких объектов будут совпадать или быть предельно похожими.

В качестве метрики можно выбрать расстояние Хэмминга, его часто используют при кодировании информации и передачи данных, оно показывает, как сильно объекты не похожи между собой. Таким образом, чтобы определить какая буква изображена на кадре, нужно найти её метрику со всеми готовыми шаблонами. И тот шаблон, расстояние Хэмминга до которого окажется наименьшим, и будет ответом.

1. **Признаковые методы**

Эти методы являются наиболее распространёнными, признак в них определяется как функция от значений, содержащихся в одном или более пикселях, и вычисляется так, что численно выражает некоторую значимую характеристику объекта.

1. **Структурные методы**

Такие методы переводят символ в его топологическое представление, отражающее информацию о взаимном расположении структурных элементов. В качестве структурных единиц могут выступать: линии, пятна, дуги окружностей, углы и т. д.

1. **Нейросетевые методы**

Нейрон, как структурная единица нейронной сети, получает множество входных сигналов. В нашем случае входные сигналы описывают значение пикселя изображения, то есть, если имеется изображение 16х16, входных сигналов у нейрона должно быть 256.

Количество выходных сигналов будет ограничено алфавитом, в нашем случае их 33, без учёта цифр.

Слайд 5

При подготовке к реализации программного продукта, также было проведено исследование, тестирование и разбор современных технологий, позволяющих работать с распознаванием текста.

Для решения этой задачи, в итоге, был выбран подход с использованием библиотеки Tesseract. Так как данная библиотека обладает множеством преимуществ, основные из которых вы можете увидеть на этом слайде.

Слайд 6

Алгоритм работы Tesseract довольно прост, его можно представить следующей схемой.

Верхняя строка – это встроенная в Tesseract предобработка изображения, которая предусматривает бинаризацию изображения и поиск контуров, для отсечения фона. Сам же процесс распознавания можно разделить на следующие этапы:

1. **Поиск строк.** Ключевыми частями этого этапа являются фильтрация объектов изображения и построение линий. Простой фильтр высоты удаляет выпадающие элементы и вертикально соприкасающиеся символы. Средняя высота приближается к размеру текста во всём слове, поэтому можно безопасно отсекать все те объекты, которые меньше средней высоты (знаки препинания, диакритические знаки и шум).
2. **Выбор базовой линии.** После того как строки текста были найдены, устанавливаются ограничительные линии.
3. **Поиск слов.** Данный этап нам не понадобится, так как текст, который мы хотим распознать – это одно слово – название города. С помощью определённых параметров для Tesseract`а можно пропустить этот этап.
4. **Определение фиксированного шага и разделение слова на буквы.** Tesseract проверяет текстовые строки, чтобы определить, как в них распределены буквы. Если буквы расположены с фиксированным шагом, то Tesseract разбивает это слово на символы.
5. **Распознавание слов.** В первую очередь классифицируется исходная сегментация, полученная в результате линейного поиска. Остальная часть распознавания на этом этапе применяется только к тексту без фиксированного шага. То есть, сначала, буквы «С», «и», «л», «ь» будут успешно распознаны, а после определится слово с новым разбиением.
6. **Разделение «слипшихся» символов и распознавание дефектных символов.**
7. **Классификация.** Протекает как двухэтапный процесс. На первом этапе классификатор создает короткий список классов, к которым может принадлежать неизвестный символ. На втором этапе для каждого признака неизвестного символа ищется битовый вектор прототипов данного класса, которому он может соответствовать, а затем вычисляется фактическое сходство между ними.

Слайд 7

Для моделирования 3D моделей достопримечательностей Крыма был выбран 3D-редактор 3ds Max, так как он имеет большие преимущества перед другими инструментами в области моделирования архитектурных сооружений.

Распознавание AR маркеров и выведение 3D модели на опорную поверхность проводилось без использования готовых инструментов по следующим причинам:

1. отсутствие у представленных решений полностью бесплатных пакетов;
2. личная заинтересованность в самостоятельной реализации всех этапов визуализации 3D моделей в дополненной реальности;
3. отсутствие материально-технической базы для использования профессиональных продуктов для создания AR приложений.

Слайд 8

Немного о алгоритме работы приложения. Алгоритм реализуется на языке Python и состоит из трёх скриптов: *main.py* – реализация интерфейса и распознавание текста, *render.py* – визуализация 3D модели, *objloader.py* – парсинг obj файла для разбиения модели на треугольники.

Все города, достопримечательности которых будут проецироваться в видеопоток, объединены в отдельный класс – *City*. При создании каждого объекта этого класса он инициализируется изображением опорной поверхности, вычисленными для неё дескрипторами и ключевыми точками, а также 3D моделью достопримечательности соответствующую определённому городу.

60 раз в секунду происходит захват кадра из видеопотока, после чего он передаётся в функцию *pasteModelIntoFrame*, которая также в качестве аргументов принимает текущий город и предыдущий успешно распознанный кадр. Данная функция проводит бинаризацию изображения и вычисляет по два ближайших дескриптора для каждой точки, отсеивает наименее похожие точки и несимметричные соответствия. После вычисляет матрицу гомографии и получает из неё и, заранее вычисленной, матрицы камеры матрицу проекции (трансформации).

Данная матрица трансформации включает в себя поворот и перенос, благодаря которым исходная модель будет перенесена на опорную поверхность и спроецирована на кадр. Проектирование модели на кадр осуществляется при помощи функции *render,* которая для каждого треугольника заданной модели проецирует все его точки на опорную поверхность и рисует полученный треугольник на сохранённой копии текущего кадра.

После отрисовки модели в кадре кадр возвращается в видеопоток. В случае недостаточного совпадения дескрипторов исходной опорной поверхности и дескрипторов кадра кадр возвращается в видеопоток без изменений.

При реализации приложения у меня возникла следующая проблема: так как процесс распознавания текста с картинки занимает приблизительно 600 миллисекунд, а кадр с видеопотока мы извлекаем 60 раз в секунду, то есть приблизительно каждые 17 миллисекунд, то отсюда следует, что распознавать текст с каждого кадра не представляется возможным. Решением этой проблемы стала поддержка языком Python многопоточности. Таким образом функция, которую я назвал *getTextWithTesseract*, которая осуществляет предобработку полученного кадра, распознавание текста с него и последующую постобработку распознанного текста, запускается в отдельном потоке раз в секунду. Такой подход позволяет процессу распознавания работать независимо от процесса вывода кадра в видеопоток, но распознавая при этом не все полученные кадры.

В зависимости от распознанного текста, в качестве текущего города назначается город соответствующий распознанию, а соответственно текущей 3D моделью становится модель достопримечательности этого самого города.