Расскажу про технологии AR.

*СЛАЙД 1(название доклада)*

Что это такое и для чего нужны.

Расскажу, с помощью чего и как разрабатываются приложения AR.

Затем разберем пример реализации такого приложения.

**Что такое?**

Какие вообще бывают реальности?

* + «Реальная» реальность - реальный мир.

Воспринимаем через органы чувств: зрение, слух, обоняние, вкус, осязание, вестибулярный аппарат(чувство положения в пространстве).

* + Виртуальная реальность.

Cозданный с помощью специальных средств искусственный мир.

Идея спец. средств - воздействовать на наши органы чувств.

В первую очередь это зрение и слух. Потому что на них искусственно воздействовать проще.

**Устройства:** Динамик, издающий звук, лампочка горящая разными цветами, экран монитора.

Дополнительные технологии: шлемы виртуальной реальности, симуляторы для пилотов.

Пример: компьютерные игры.

*СЛАЙД 2(определение АР)*

* + Дополненная реальность. Дополненная реальность это нечто между реальностью и виртуальностью.

Отличие от виртуальной реальности - VR полностью искусственный мир, а в AR реальный мир дополняется виртуальными элементами.

Опять же идея в воздействии на органы чувств. Но в AR берется реальная информация из реального мира и дополняется искусственной.

Устройства: комбинация датчиков и устройств вывода. Например смартфон.

В настоящее время под AR чаще всего понимают дополнение изображения реальных объектов виртуальными. Причина - зрение самое информативное средство восприятия.

**Где применяется?**

Военная техника

*СЛАЙД 3(истребитель)*

Большим толчком для развития технологий дополненной реальности стало использование в военной технике. Если кто-то играл в авиасимуляторы, то наверняка видел такую картинку(на слайде).

Это называется ИЛС(индикатор на лобовом стекле).

Позволяет пилоту не отвлекаться на датчики и оперативнее принимать решения.

Маркетинг

*СЛАЙД 4(BMW)*

Широкое применение в маркетинге.

В журналах часто печатают специальные метки либо метка = страница журнала.

На смартфон специальное приложение и навести камеру на метку можно увидеть заложенный контент - изображения, видео, 3д модель.

Пример: реклама BMW

*СЛАЙД 5(IKEA)*

Пример: IKEA

*СЛАЙД 6(Собор)*

Пример: путеводитель для туристов, информация об объекте на который наведена камера.

*СЛАЙД 7(Google translate)*

Переводчик от google, который переводит текст с камеры.

Навигация

*СЛАЙД 8(Навигация в машине)*

Изображение в правом верхнем углу - очки google glass.

По аналогии с пилотами истребителей позволяет пользоваться навигатором не отвлекаясь от дороги

Развлечения

*СЛАЙД 9(Игра)*

Конечно же индустрия развлечений.

Пример - возможность поиграть в футбол на своем столе.

**Реализация**

Подробнее о реализации.

*СЛАЙД 10(Этапы)*

Если не брать во внимание ввод и вывод, приложение дополненной реальности(визуальной), как правило делает две вещи:

* + Распознавание изображений

Понять, что изображено, чтобы понять, чем дополнять

* + Дополнение изображения новыми элементами

Не всегда нужно распознавание: просто вывод информации, получение информации из других источников таких как высотомер у самолета или GPS-приемник в навигаторе.

Наиболее сложная часть - распознавание изображение.

Для компьютера изображение - всего лишь набор цветных точек.

Как из этого набора точек получить что-то полезное - не простой вопрос.

Существует множество алгоритмов для самых разнообразных задач.

Задачи далеко не ограничиваются дополненной реальностью. Примеры со слайда.

Подходы к распознаванию можно разбить не несколько больших категорий:

* + Перебор видов объекта.

Заранее заготавливаем большое количество образцов для сравнения и поочередно сравниваем их с участками изображения.

Накладное решение.

* + Нахождение контура объекта и исследование его свойств.

Выделяются контуры.

Затем они исследуются, например на размер, форму, количество углов, цвет и т д.

* + Нейронные сети.

Нейронные потому что в подобии человеческому мозгу, конечно в очень упрощенном виде.

Применяются алгоритмы машинного обучения - так называемое обучение на прецедентах.

Выглядит это примерно так: заранее заготавливается большой набор позитивных и негативных примеров. Все это скармливается нейронной сети.

Нейросеть на основе предоставленных примеров выделяет характерные признаки.

Плюс: распознание не только тех объектов которые были в примерах, но и похожих. Качество работы сильно зависит от качества тренировочных примеров

**Маркерная vs безмаркерная**

*СЛАЙД 11(Маркерная vs безмаркерная)*

Технологии AR можно разделить на маркерные и безмаркерные.

Безмаркерная: распознавание образов или более специфичный анализ изображений(поиск плоскостей, анализ контуров).

Маркерная:

проще в реализации, так как на изображении ищется специально заготовленный маркер.

Маркер специально готовится таким образом, чтобы его можно было проще найти, понять его положение в пространстве.

**Библиотеки и open source**

В реальном мире "с нуля" ничего не строится.

И чтобы построить дом не нужно знать, как делать кирпичи.

В мире программирования тоже есть «кирпичи».

Алгоритмы реализуются и публикуются в виде подключаемых модулей, библиотек.

Писать самому реализацию каких-то популярных алгоритмов является плохой идеей.

Особенно во все что связанно с безопасностью.

*СЛАЙД 12(библиотеки)*

Вы подключаете такую библиотеку к своему проекту и просто запускаете нужные алгоритмы на своих наборах данных, не задумываясь о деталях их работы.

Библиотеки часто бесплатны.

*СЛАЙД 13(Open source)*

Open source - что это такое? Дословно - открытый исходный код.

Это программное обеспечение(не только библиотеки) код которого открыт для просмотра и изменения.

По сути это переросло в философию.

Есть огромное количество проектов разрабатываемых энтузиастами.

Часто предоставляется возможность поучаствовать в разработке или даже начать собственный проект взяв за основу что-то уже существующее.

Apache, nginx - веб-серверы на которых работает более половины сайтов в интернете.

**OpenCV**

*СЛАЙД 14(OpenCV)*

Одной из таких открытых библиотек является OpenCV - библиотека компьютерного зрения.

Это сборник огромного количества(>2500) реализаций алгоритмов обработки изображений. В том числе необходимых инструментов для распознавания.

Поддерживаемые языки и платформы по слайду.

Разберем пример приложения маркерной дополненной реальности разработанный с помощью OpenCV.

Открываем IDEA.

Все скачивают проект.

Раздаем распечатанные маркеры.

**Практика**

* + Постановка задачи.

Что будем делать?

Будем дополнять нашу реальность котиками.

*СЛАЙД 15(Постановка задачи)*

Будем проецировать картинку кота на маркер.

* + Подробнее рассмотрим маркер.

*СЛАЙД 16(Маркер)*

Это по сути сетка 8 на 8. Где каждая клетка закрашена либо черным либо белым цветом.

У вас разные маркеры(всего 5 вариантов), но между ними есть общие черты.

* + Границы у всех черные. Для того чтобы образовался четырехугольник.

Прямоугольник - простая форма для поиска и анализа(легко пользоваться в декартовой системе координат)

* + Маркеры не симметричны. Это сделано для того, чтобы можно было понять ориентацию маркера в пространстве и на основе этого поворачивать нужным образом картинку.
  + Хранение изображений

*СЛАЙД 17(Хранение изображений)*

Краткий экскурс в то, как изображения хранятся в памяти компьютера.

Все изображение делится на маленькие цветные точки.

Каждая точка кодируется 3 байтами - 3 цветами.

Сочетания оттенков дают любой другой цвет.

Если очень близко посмотреть на монитор, то можно увидеть картинку как на слайде - это физическое отражение принципа.

Каждый цвет может иметь значения от нуля до 255, в двоичном представлении это от 0000 0000 до 1111 1111 - столько вмещает один байт.

Если все цвета имеют значения 0, то цвет черный, если все 1 - белый.

*Запускаем CatAR\_0*

Видим картинку с камеры.

Проект на Java.

Функция processFrame - принимает на вход кадр - это матрица чисел-точек. Кадр на выходе выводится на экран.

Что такое функция, что такое return.

Все что пишем внутри processFrame будет исполняться для каждого кадра.

Сейчас никаких изменений в кадре не происходит поэтому на экран он выводится как есть.

*СЛАЙД 18(План)*

Как я говорил раньше - первый этап - распознавание.

Что нам нужно от него получить?

* + Есть ли в кадре маркер
  + Как он расположен

Как будем действовать:

* + Найдем в кадре контуры - границы.
  + Далее разберем что они из себя представляют(мы ищем четырехугольник)
  + Разберем что внутри контуров(маркер ли это).

* + Первый шаг - найти на изображении контуры. Для этого картинку нужно подготовить.

*Открываем CatAR\_1*

* + Превращение в градации серого.

Сохраняем в переменную чтобы не портить исходный кадр.

В этом виде каждая точка занимает только 1 байт - 0 до 255 градации серого.

* + Блер - размазывание. Это нужно чтобы сгладить мелкие шероховатости и тем самым автоматически отбросить мелкие ненужные нам контуры.
  + Трешолд. Это превращение картинки в бинарную. Бинарная это такая картинка где каждый пиксель может быть либо черным, либо белым. Никаких промежуточных значений.

Трешолд с английского порог. Суть - сравнение с пороговым значением.

Недостаток - работа с различным освещением.

Адаптивный трешолд - порог для каждого пикселя не фиксированный а вычисляется как среднее значение соседних пикселей.

* + Теперь нужно на полученной бинарной картинке найти контуры.

*Открываем CatAR\_2.*

* + Запускаем алгоритм поиска контуров из OpenCV и сохраняем контуры в переменную. Контур - просто набор точек. Точка - пара координат х, y. Для того чтобы понять что у нас получается давайте нарисуем найденные контуры поверх кадра. Это делается с помощью функции drawContoursOnFrame.
  + С подобным набором точек тяжело работать, поэтому их нужно преобразовать в полигоны - многоугольники. Нарисуем их другим цветом.
  + Полигоны получили. Теперь надо разбираться что это за полигоны и в их содержимом.

*Открываем CatAR\_3*.

* + Первым шагом - отфильтруем лишнее по количеству вершин и длине.
  + Далее нам нужно каждый полигон развернуть в квадрат, чтобы была возможность его проанализировать.

СЛАЙД 19(Трансформации)

Ориентироваться в квадрате проще чем произвольном полигоне.

Для подобный задач используются трансформации.

Достаточно популярно аффинное преобразование.

Вкратце выглядит это так - искажение задается матрицей. Матрица координат каждой точки умножается на эту матрицу.

К примеру матрица которая ничего не делает выглядит так(главная диагональ 1).

Из формул можно увидеть, что a и e - растягивания. c, f - паралельный перенос.

В аффинном преобразование паралелльные прямые всегда остаются паралелльными.

В перспективном добавляется еще 3 элемента. Особенность перспективного - прямые всегда остаются прямыми, но паралельные прямые не обязательно параллельны после трансформации - актуально для 3д мира.

Матрицами такие вещи делать удобно потому что можно серию преобразований «спрятать» в одной матрице и потом всего одной операцией преобразовывать координаты всех точек.

OpenCV сделает все за нас.

Получаем матрицу трансформации - участка изображения ограниченого контуром в квадрат 80 на 80.

Применяем эту матрицу к бинарному изображению(далее будет понятно почему именно к нему)

Для наглядности showMat.

* + Развернули в квадрат, теперь нужно разобрать его содержимое и понять маркер ли это.
  + Вспомним что наш маркер это сетка 8 х 8. Такую сетку можно представить в виде матрицы, где не закрашенная(белая) клетка - 0, а закрашенная(черная) - 1.

*Открываем файл MarkerEthalon* и вводим туда свой маркер. Далее из этого эталона получаются повернутые варианты.

* + Теперь нужно содержимое полигона попытаться превратить в такой же формат матрицы для сравнения.

*Открываем CatAR\_4*.

* + В матрице 8 строк и 8 столбцов, нужно также разбить развернутую картинку на 8 х 8 кусочков.

Циклом идем по строкам от 0 до 7 а внутри каждой строки от 0 до 7 столбца.

Каждый раз "вырезаем" клеточку из картинки размером 10 на 10(вспомним что развернули маркер в квадрат 80 х 80). С помощью библиотечной функции считаем там не нулевые пиксели(именно поэтому картинка бинарная). Вспомним что 0 - белый, 1 - черный.

Для наглядности printMatrix.

* + Остается только сравнить полученную матрицу с эталоном.

*Открываем CatAR\_5.*

* + Поочердено сравниваем маркер с повернутыми эталонами. Если находим совпадение то печатаем в консоль Up или Right. Допишем для оставшихся положений.

Собственно с первым этапом - распознанием закончили. Переходим ко второму.

*СЛАЙД 20(План второго этапа)*

Теперь нужно дорисовать на найденные маркеры котов.

* + Подготовим повернутые картинки для каждого положения.

*Открываем CatImage.*

Up загружена из файла. Нужно повернуть на нужный угол(по часовой стрелке) rotateImage.

*Открываем CatAR\_6*

дописываем присваивание image.

После выбора картинки нужно исказить ее чтобы она совпала с маркером в кадре. Тут опять же нужна трансформация. Получаем матрицу трансформации картинки в полигон. И применяем ее.

* 1. Остается только совместить искаженное изображение с исходным кадром. С этим помогут логические операции.

*СЛАЙД 21(Логические операции)*

* 1. Вспомним таблицы истинности для операций И ИЛИ. Если всмотреться в таблички то можно увидеть, что И принимает значение 0 если хотя бы один из операндов 0. ИЛИ также с 1. С помощью этого свойства можно делать следующее - маски на слайде.

Как это поможет нам? Вспомним как хранятся в памяти изображения. К ним также можно применять логические операции.

СЛАЙД 22(Кот на черном фоне)

Применяя трансформацию к картинке кота мы получаем картинку А. Черные пиксели это нули.

Если подготовить картинку «фона» с черной «дыркой» по форме кота и применить к этим двум картинка операцию ИЛИ, то они совместятся и получим фон с котом вписанным в нужную область.

*Открываем CatAR\_7*

Теперь нужно подготовить такую картинку с белой «дыркой».

Берем белую картинку(все 1) по размеру кота и искажаем. Получаем B.

Применяем отрицание. Получаем C.

Далее применяем к полученной картинке и изначальному кадру операцию И. Получаем D. Картинку с черной «дыркой».

Остается только применить к этой картинке и к коту на черном фоне операцию ИЛИ.