Санкт-Петербургский государственный университет

Кафедра системного программирования

Антропов Игорь Михайлович

Библиотека для приложений дополненной реальности на телефонах

Курсовая работа

Научный руководитель:
ст. преп. Я. А. Кириленко

Санкт-Петербург
2017

Оглавление

1. Введение

Вычислительные мощности современных смартфонов позволяют использовать их для создания систем дополненной реальности. Такие системы требуют постоянного сопоставления позиций реальных и виртуальных объектов в пространстве. Для решения этой задачи существуют два основных подхода, требующих больших вычислительных средств: с использованием маркеров (marker based AR) или без использования (markerless AR).

1.1 Дополненная реальность без маркеров

Возможно несколько подходов к реализации дополненной реальности без маркеров. Один из них, использование гироскопа телефона. Данная реализация позволяет отслеживать координаты объектов и осуществлять корректное взаимодействие с ними, однако большой минус состоит в том, что отсутствует какая либо информация о реальном мире.

Другой вариант это использование метода одновременной локализации и построения карты (SLAM - Simultaneous Localization and Mapping). Данный метод предоставляет нам полную информацию об окружающем мире и возможность отслеживать как свои координаты, так и координаты объектов. Однако из-за отсутствия важных требований, таких как графическая карта, дополнительная камера, датчик расстояния, не представляется возможности использовать его в проектах [1].

1.2 Дополненная реальность с маркерами

Данный подход заключается в использовании специальных, заранее известных изображений - маркеров. Маркеры бывают совершенно разные, хотя чаще всего использую двухмерные бинарные штрихкоды.

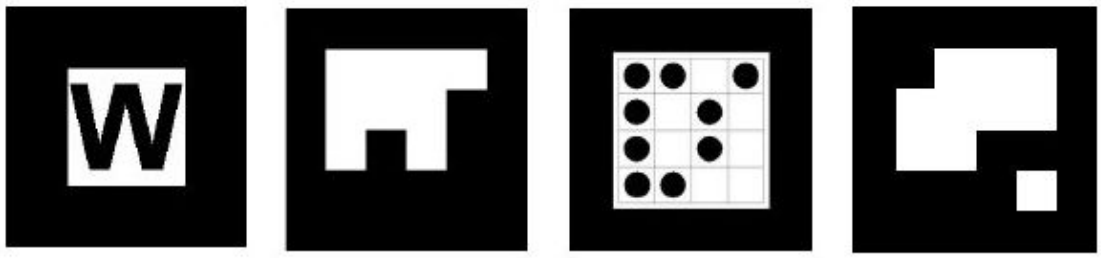

Рис.1 Типы маркеров

Алгоритм распознавания состоит из нескольких последовательный шагов обработки каждого кадра и подсчета координат маркера.

1. Изменения входного изображения

На данном этапе производится изменение изображения для ускорения времени его последующей обработки и уменьшения количества ложных кандидатов.

2. Нахождение контуров кандидатов

Используя один из множества алгоритмов, находим массив точек контура для каждого кандидата.

3. Проверка кандидатов

Сопоставление формы и других признаков кандидата и маркера. В результате получаем координаты углов кандидатов.

4. Подтверждение кандидатов и выявление их номеров

Попытка подсчета информации с кандидата.

5. Преобразование координат

Преобразование координат углов маркера в координаты поворота и переноса.

2. Постановка задачи

Цель данной работы заключается в создании библиотеки для упрощения создания приложений дополненной реальности для телефонов. Для ее достижения поставлены следующие задачи.

- Выбор средства для создания приложений
- Реализация алгоритма нахождения маркеров
- Создание алгоритма перевода координат
- Провести апробацию библиотеки

3. Выбор средств для разработки приложений для телефонов

Так как требовался инструмент для создания приложений одновременно и для телефонов с операционной системой Android и IOS были выбраны несколько кандидатов.

- Cocos2d
  Плюсы:
  - Бесплатный
  - Открытый исходный код

  Минусы:
  - Преимущественно для двухмерных приложений
  - Малое количество пользователей

- Marmalade
  Плюсы:
  - Скорость работы

  Минусы:
  - Проблемы со сторонними библиотеками

- Unreal engine
  Плюсы:
  - Бесплатная пробная версия
  - Поставляется с открытым исходным кодом

  Минусы:
  - Проблемы с документацией
  - Проблемы с приложениями на операционной системе IOS

- Unity
  Плюсы:
    - Большая и основательная документация
    - Легкий для новичков

  Минусы:
    - Закрытый исходный код

4. Существующие решения

4.1 Vuforia

Комплект средств разработки приложений дополненной реальности для мобильных устройств. Предоставляет интерфейсы программирования приложений на языках C++, Java, Objective-C и через интеграцию с системой Unity. Vuforia позволяет создавать набор собственных маркеров и отслеживать их. Однако большими минусами являются невозможность конфигурировать маркеры и цена продукта.

4.2 Emgu

Кросс-платформенное дополнение библиотеки OpenCV. Позволяет реализовать полностью работу с маркерами, однако содержит в себе всю библиотеку OpenCV, что в свою очередь сказывается на размере. Так же появляется необходимость реализовывать перевод координат каждый раз, так как в OpenCV такого модуля нет.

5. Создание библиотеки

5.1 Правила построения маркера

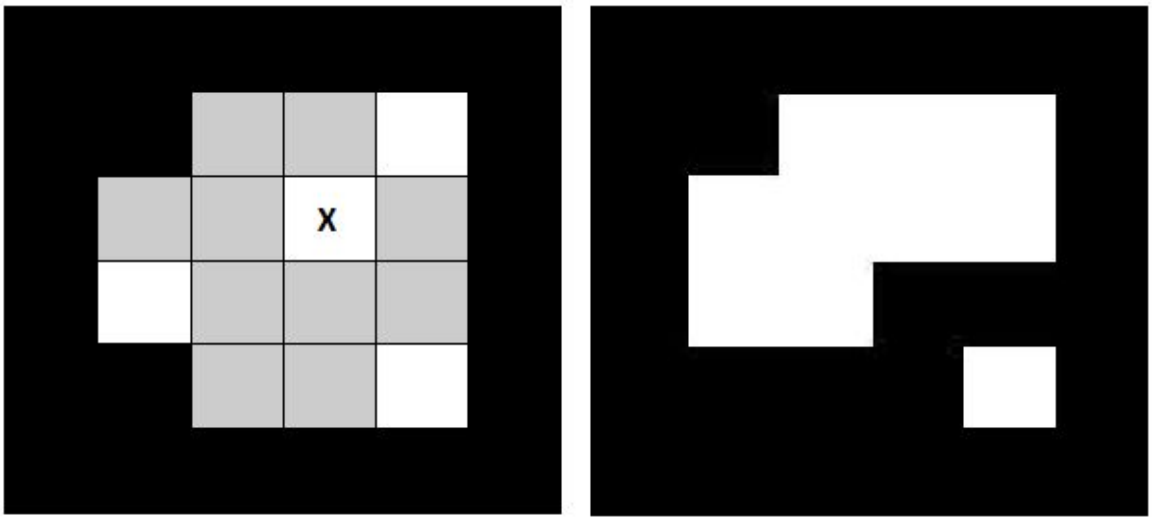

Рис. 2 Пустой маркер и пример

На рисунке показан каркас маркера и пример маркера под номером 15. Серые ячейки отвечают за кодирование информации. Ячейка с символом X является контрольным битом. Она удовлетворяет правилу: если количество белых ячеек нечетное, то X должна быть белой. Такое построение маркера обеспечивает одни из лучших результатов [2].

5.2 Подробный алгоритм работы с маркером

- Изменение входного изображения

Основная цель библиотеки — работа с телефонами. Во избежания сильной нагрузки первым шагом изменяется размер

изображения. Для скорости изменения была выбрана интерполяция методом ближайшего соседа.

Следующим шагом является приведение изображения в оттенки серого. Для него был выбран алгоритм перевода изображения по формуле

$$\mathrm{RGB[A]\ to\ Gray:}\quad Y \leftarrow 0.299 \cdot \mathrm{R} + 0.587 \cdot \mathrm{G} + 0.114 \cdot \mathrm{B}$$

Далее для повышения контраста изображения был применен метод адаптивной эквализации гистограммы с ограничением CLAHE.

Заключительным изменением входного изображения для уменьшения количества ложных кандидатов является использование метода фильтрации изображения с помощью функции Гаусса.

- Нахождение контуров кандидатов

Для нахождения контуров кандидатов сначала применяется алгоритм детектора границ Канни[3]. После чего применяется алгоритм нахождения контуров, который выдает информацию о координатах контуров и их иерархии.

- Проверка кандидатов

После применения функции поиска контуров, получаем всех кандидатов и появилась необходимость проверить их на соответствие реальным маркерам.

Первым шагом, используя знания иерархии, проверяется, что у кандидата есть вложенный в него контур. То есть контур внутри нашего контура. Это свойство достигается путем из-за наличия у маркера полностью одноцветной границы по периметру. Это самая первая проверка и отсеет большинство ложных вариантов.

Далее используется алгоритм Дугласа-Пекера, позволяющий уменьшить число точек кривой, аппроксимированной большей

серией точек. После чего следует проверки на подходящий размер и соответствие форме маркера. В результате получаем координаты четырех углов кандидата.

- Подтверждение кандидатов и выявление их номеров

После выполнений предыдущих манипуляций, были получены координаты кандидатов больше всего похожих на маркеры. Для этого изображение кандидата разбивается на 6 столбцов и 6 строк. Из полученного разбиения строится матрица 6 на 6, где если в изображении преобладает черный цвет, то в матрице в этой ячейке будет 1, иначе 0. Далее следует проверка правил построения маркера и контрольного бита. Следующим шагом находится номер маркера.

- Преобразования координат

Последним этапом необходимо решить проблему перспектива и точка. Для этого используем повторяющийся алгоритм основанный на оптимизации Левенберга - Марквардта.

5.3 Изменение координат для использования в Unity

После обработки маркеров были получены вектор поворота и вектор переноса.

Для использования вектора переноса, пользователь должен указать масштаб соответствия координат реального мира и сцены юнити.

Вектор поворота приведем к матрице поворота, используя алгоритм Родригеса и преобразуем в кватернион по алгоритму Кена Шоемака [4], который и будем использовать в Unity.

5.4 Использование библиотеки

От пользователя требуется добавить скрипт на тот объект, который будет отображать текущее изображение камеры. Если в приложении не подразумевается наличие изображения с камеры, то добавить необходимо на пустой объект. После этого координаты поворота и переноса будет высчитываться автоматически для дальнейшего использования. Для гибкости использования пользователю также придется следить за соответствием координат результатов и сцены и устанавливать масштаб переносов. В дополнение к этому, в алгоритме используются данные калибровки камеры, которые также должны быть предоставлены.

6. Апробация библиотеки

Для проверки работоспособности библиотеки было создано приложение, использующее координаты поворота и переноса маркера для корректировки положений активных объектов на сцене. Сразу после записка приложение будет накладывать на маркер объект “портал” и производить рейкастинг.

Заключение

В рамках данной работы были выполнены следующие задачи.

- Был произведен выбор средства разработки мобильных приложений.
- Реализован алгоритм нахождения маркера и перевода его координат.
- Создана библиотека[1].
- Создан пример работы с библиотекой.

[1] https://github.com/Antropovi/Markers-Unity

Список литературы

[1] Monocular SLAM for Real-Time Applications on Mobile Platforms / Mohit Shridhar, Kai-Yuan Neo // Stanford University, June 2015.

[2] Visual Marker Detection and Decoding in AR Systems: A Comparative Study / Xiang Zhang, Stephan Fronz, Nassir Navab // ISMAR '02 Proceedings of the 1st International Symposium on Mixed and Augmented Reality, 97, 2002.

[3] A Computational Approach to Edge Detection / J.Canny // IEEE Trans. on Pattern Analysis and Machine Intelligence, 8(6): 679-698, 1986.

[4] Quaternion Calculus and Fast Animation / Ken Shoemake // SIGGRAPH course notes, 1987.