Алгоритм распознавания маркера дополненной реальности на примере библиотеки ARToolKit

Р. В. Шириев
Факультет информатики
и робототехники
Уфимский государственный
авиационный технический
университет
Уфа, Россия
e-mail: rvshiriev@gmail.com

А.Ф. Муратов
Факультет информатики
и робототехники
Уфимский государственный
авиационный технический
университет
Уфа, Россия
e-mail: muratov.artur.pro07@gmail.com

Аннотация[1]

В данной статье проведён анализ алгоритма распознавания маркера на изображении при помощи технологии дополненной реальности. В качестве основы принят алгоритм, реализованный в библиотеке ARToolKit.

1. Введение

Дополненная реальность – это технология добавления, внедрения в реальную жизнь, в трехмерное поле восприятия человека виртуальной информации, которая воспринимается как элементы реальной жизни. Реальность расширяется (или дополняется) внедрением в нее виртуальной информации.

Аналитическая компания Markets and Markets спрогнозировала интерес розничной торговли к внедрению решений дополненной реальности[1]. Согласно её заключению, среднегодовой рост рынка приложений дополненной реальности составляет 47,1%.

В данной работе рассматривается, каким образом решается задача распознавания маркера. В качестве базы принята библиотека ARToolKit, распространяемая с лицензией GPLv3.

2. Анализ бизнес-процесса технологии дополненной реальности

На данный момент основными датчиками мобильных устройств являются датчик геолокации, акселератор и камера. Для получения информации о размещении дополняющего объекта самым точным датчиком является камера, так как геолокатор даёт достаточно грубые данные о географическом положении, а акселератор показывает только угол наклона к земной поверхности.

Существуют два принципиальных подхода для создания дополненной реальности на основе изображения, поступающего с камеры: с использованием заранее приготовленного маркера, который нужно распечатывать и без такого. Оба подхода, используя алгоритмы “компьютерного зрения”, распознают объекты в кадре и дополняют их[2]. Но во втором случае программе нужно задать, какие реальные объекты следует дополнять, что часто не эффективно.

В теории маркером может быть любая фигура. Но на практике мы ограничены разрешением камеры телефона, особенностями цветопередачи и вычислительной мощностью оборудования, а потому выбирается обычно черно-белый маркер простой формы. Как правило, это прямоугольник или квадрат с вписанным идентификатором-образом.

Смартфон распознаёт маркер на кружке и дополняет его моделью баскетбольной корзины
Рисунок 1 – Пример реализации дополненной реальности с использованием маркера – игра «ARBasketball»

Блок-схема алгоритма дополнения изображения с камеры устройства при наличии маркера представлена на рисунке 2.

Труды Седьмой всероссийской научной конференции "Информационные технологии интеллектуальной поддержки принятия решений", 28-30 мая, Уфа-Ставрополь, Ханты-Мансийск, Россия, 2019

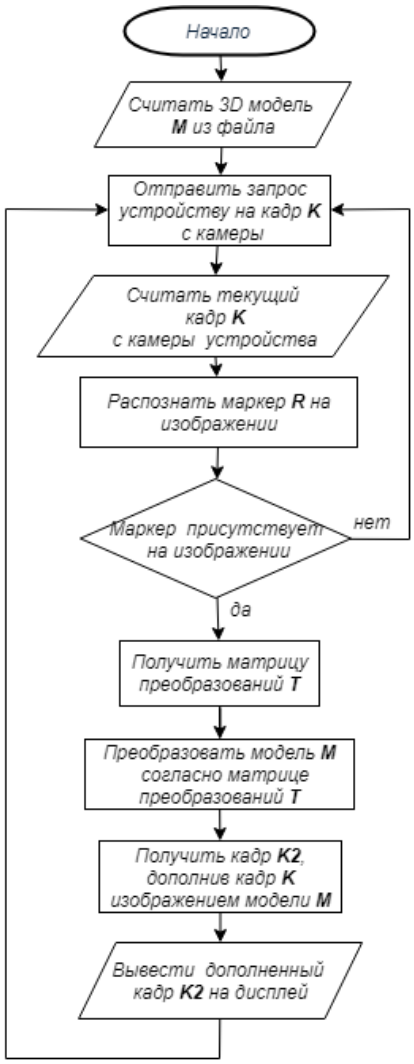


Рисунок 2 – Блок-схема алгоритма дополнения изображения с камеры устройства при наличии маркера

Этап распознавания маркера (5-ый блок на блок-схеме) может решаться различными способами, так как он представляет собой задачу компьютерного зрения. Детализируем далее этот этап.

3. Алгоритм распознавания маркера

На вход подаётся растровое изображение с камеры мобильного телефона или другого устройства.

На выходе формируются координаты точек, в которые в дальнейшем необходимо вписать 3D модель.

Шаги алгоритма:

1. **Выделение маркера на изображении.** На вход подаётся растровое изображение с камеры мобильного телефона или другого устройства. На выходе формируются координаты крайних точек фигуры, похожей на маркер, если она была найдена.
2. **Выравнивание внутренней области маркера и сравнение её с шаблоном.** Область между координатами выравнивается и сравнивается с по заданному алгоритму с шаблоном маркера. Это нужно, для того чтобы отсекать другие фигуры, похожие на маркер.
3. **Получение координат маркера в системе координат устройства.** В результате по этим координатам 3D-модель переносится в положение экрана.

Выделение маркера на изображении, в рамках библиотеки ARToolKit, делится на следующие этапы:

1. Бинаризация изображения
2. Определение замкнутых областей
3. Определение контуров области
4. Определение углов и линий области

Приведём алгоритм каждого из этапов.

Алгоритм бинаризации изображения

На входе имеем изображение размера W на H. Каждый пиксель представлен в стандартном 3-цветном формате RGB (красный, зелёный и синий соответственно). Обозначим это изображение как P.

$$P_{ij} \rightarrow (R_{ij}, G_{ij}, B_{ij})$$
$$\text{где } i \in [1, W], j \in [1, H], \quad R_{ij}, G_{ij}, B_{ij} \in [0,255]$$

Суть бинаризации в общем случае состоит в разделении всех пикселей изображения на 2 цвета. В данном случае требуется разделить изображение на чёрный и белый цвета. Для этого чаще всего изображение сначала переводят в изображение в градациях серого (полутоновое изображение).

Способы определения полутонового пикселя GS[2]:

- Lightness: $GS = (\max(R_{ij},G_{ij},B_{ij}) + \min(R_{ij},G_{ij},B_{ij}))/2$
- Luminosity: $GS = 0.21 \times R_{ij} + 0.72 \times G_{ij} + 0.07 \times B_{ij}$
- Среднее: $GS = (R_{ij} + G_{ij} + B_{ij}) / 3$

Чаще всего используется метод Luminosity, так как его коэффициенты соответствуют восприятию человеком конкретных цветов.

После этого процесса у полутонового изображения определённым образом выбирают пороговую яркость T. Пиксели ярче данного порога считаются белыми, темнее – чёрными.

Рассмотрим основные виды порога T:

- Жёстко фиксированный – для всех кадров в программе используется один заранее введённый порог (T = const);
- Глобально-вычисляемый – для каждого кадра вычисляется один порог для всех пикселей ($T = f(P_{GS})$). Например: использование средней яркости изображения; Метод Оцу – порог, уменьшающий дисперсию внутри класса, которая определяется как взвешенная сумма дисперсий двух классов;

– Локально-вычисляемый – для каждого пикселя вычисляется порог на основе его окружения ($T = f(P_{GS}, i, j)$).

При жёстко фиксированном пороге не тратится время на его расчёт, но эффективным данный метод является только в условиях постоянной освещённости помещения и яркости камеры.
Глобально-вычисляемый порог не имеет данного недостатка, но из рисунка 3 видно, что в условиях, когда яркость непостоянна в пределах одного кадра, следует использовать локально-вычисляемый порог.

Слева направо: изображение в градациях серого; бинарное изображение, полученное методом Оцу (глобально-вычисляемый порог); бинарное изображение, полученное методом Sauvola Thresholding (локально-вычисляемый порог);
Рисунок 3 – Сравнение двух методов бинаризации изображения

В данной работе для бинаризации изображения будем использовать метод Sauvola Thresholding. Порогом в этом случае является средняя яркость в квадрате с ребром $2r$ (1).

$$T = \frac{1}{2r}\sum_{i,j=-r}^{r} P(i,j), r = const \qquad (1)$$

Для того чтобы вычисление порога для пикселя занимало константное число операций, суммирование происходит при помощи интегрального изображения II (2), которое вычисляется для каждого кадра за квадратичное число операций.

$$II(x,y) = \sum_{i=0}^{x}\sum_{j=0}^{y} P(i,j) \qquad (2)$$

Тогда порог яркости будет вычисляться следующим образом (3).

$$T(i,j) = II_{i+r,j+r} + II_{i-r,j-r} - II_{i-r,j+r} - II_{i+r,j-r} \qquad (3)$$

Таким образом алгоритм бинаризации можно описать так:

Вход:

– Изображение $P_{ij} \to (R_{ij}, G_{ij}, B_{ij})$, где $i \in [1,W], j \in [1,H]$, $R_{ij}, G_{ij}, B_{ij} \in [0,255]$

– Радиус вычисления порога r, где $r \in [1, \min(H,W)]$

Выход:

– Изображение Bin_{ij}, где $i \in [1,W], j \in [1,H]$, $Bin_{ij} \in [0,1]$

Шаги алгоритма:

1. Для каждого пикселя P_{ij} на изображении размером W на H рассчитать серый пиксель GS_{ij}. Как $GS_{ij} = 0.21\times R_{ij} + 0.72\times G_{ij} + 0.07\times B_{ij}$.
2. Для каждого пикселя GS_{ij} на изображении размером W на H рассчитать пиксель интегрального изображенияII_{ij}. Как $II_{ij} = II_{i,j-1} + L_{i-1,j}$, где сумма значений от $II_{0,j}$до $II_{i-1,j}$
3. Для каждого пикселя II_{ij} на изображении размером W на H рассчитать бинарный пиксель Bin_{ij} следующим образом: если $P_{ij} > T_{ij}$, то присвоить 1; иначе – 0.

Алгоритм выделения замкнутых областей

Пиксели изображения после бинаризации образуют множество чёрных и белых областей. Целью данного этапа является выделение областей и их отсеивание по характеристикам. Значащими будем считать только области чёрных пикселей. Каждой области присваивается числовой идентификатор. Это делается путем прохода по всем пикселям слева направо сверху вниз и сохранения идентификаторов областей в буфер для каждого пикселя.

Идентификатор текущего пикселя (если он чёрный) определяется следующим образом:

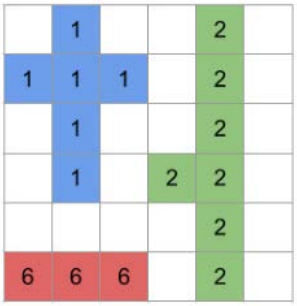
Шаг 1. Если пиксель выше текущего чёрный, то в буфере для этого пикселя размещается идентификатор области, в которую входит пиксель выше.

Шаг 2. В противном случае, если пиксель правее и выше текущего чёрный, выполняются некоторые проверки, чтобы определить, требуется ли объединить две области (в настоящее время) с разными идентификаторами. Если, по крайней мере, один из пикселей слева и/или слева сверху от текущего пикселя чёрный, то эти две области действительно связаны. Идентификаторы двух пересекающихся областей принимаются эквивалентными, просто записывая тот факт, что эти две метки эквивалентны, - это позволяет не делать переприсваивание значений в буфере для всех пикселей области (это было бы весьма неэффективно).

Шаг 3. В противном случае, берётся идентификатор пикселя, находящегося левее и выше текущего, если он чёрный.

Шаг 4. В противном случае, берётся идентификатор пикселя, находящегося левее текущего, если он чёрный.

Шаг 5. Наконец, если ни одно из вышеперечисленных условий не было выполнено, то была найдена новая область. Новый идентификатор присваивается буферу пикселя.



Было использовано 6 идентификаторов, но областей всего 3.

Рисунок 4 – Результат работы алгоритма выделения областей

Для каждой области собираются следующие данные:

– Число пикселей, входящих в область;

– Сумма x-координат, а также сумма y-координат каждого пикселя, входящего в область;

– Крайние точки областей.

На основе этих данных происходит анализ формы области и принимается решение о том, может ли она представлять маркер[4].

Алгоритм выделения контуров

В данной работе используется метод определения контуров, известный как метод окрестностей Мура. Результатом алгоритма является список вершин образующих контур области.

Он состоит из следующих шагов:

Шаг 1. На предыдущем этапе были получены крайние точки для каждых областей. На этом этапе ищется стартовый пиксель V_{strart}. За него принимается самый верхний пиксель. Он находится при помощи линейного поиска, слева направо. Изначально в списке пикселей контура находится только V_{start}.

Шаг 2. $P_{current} = V_{start}$. За P_{check} возьмём пиксель, находящийся над стартовым пикселем.

Шаг 3. Если P_{check} не является черным (т.е. его идентификатор равен 0), то за P_{check} принимается следующий пиксель в направлении по часовой стрелке, относительно $P_{current}$. Так повторяется до тех пор, пока не будет найден черный пиксель. Этот черный пиксель принимается за $P_{current}$и добавляется в список.

Шаг 4. Если $P_{current}$ равен V_{start}, то алгоритм останавливается. В противном случае, происходит переход к шагу 5.

Шаг 5. Пусть P_{last} - предпоследний пиксель в списке. За P_{check} принимается следующий после P_{last} пиксель в направлении по часовой стрелке, относительно $P_{current}$. Затем происходит переход к шагу 3.

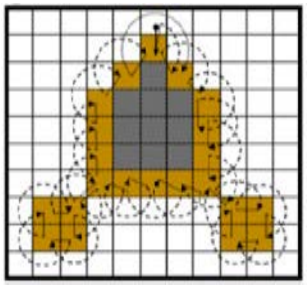


Рисунок 5 – Нахождения контура области методом окрестностей Мура

Таким образом, после применения этих действий ко всем областям получаем их контуры[5].

Алгоритм определения линий и углов

Опишем алгоритм поиска вершин и граней маркера.

Первый этап - оценка расположения вершин. У нас уже есть первая вершина (V_{start}). Найти ещё одну тоже относительно просто. Начиная с первой точки, в цепочке ищется точка, которая наиболее удалена (вычисляется наибольшая гипотенуза) от первой вершины. Из этого могут возникнуть две ситуации, если предположить, что рассматриваемая область на самом деле является маркером:

– Была найдена противоположная вершина четырёхугольника;

– Была найдена одна из соседних вершин четырёхугольника.

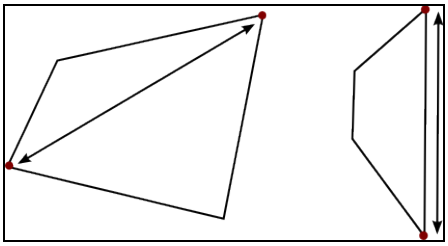


Рисунок 6 – Варианты расположения двух самых удалённых друг от друга вершин четырёхугольника

Для того чтобы найти две оставшиеся вершины, требуются максимизировать длину каждого ребра четырёхугольника. Поэтому для каждой точки имеющегося сегмента цепи (между двумя найденными точками) вычисляется расстояние от данной точки до прямой, проходящей через изначально найденные вершины. Именно эта длина должна быть максимальной.

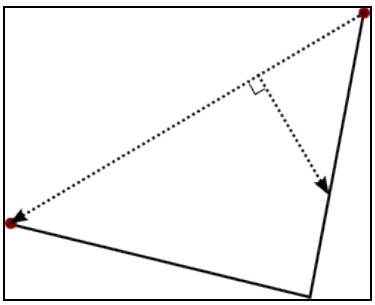


Рисунок 7 – Поиск расстояния от крайних вершин до точек сегмента цепи

Если максимальное расстояние от точек сегмента цепи до прямой на концах сегмента цепи больше чем k*(длина отрезка на концах сегмента цепи), где k - некий оптимальный коэффициент, то эта точка также считается вершиной маркера, иначе считается, что сегмент представляет собой ребро.

Подобные действия выполняются рекурсивно для каждого вновь образующегося сегмента. Если при выполнении алгоритма вершин становится больше 4, то контур бракуется, так как не является четырёхугольником.

После завершения алгоритма, имеются оценки положения 4 вершин. На практике эти оценки могут быть весьма плохими, когда речь идет о маркере. На данный момент все вершины искались из имеющейся цепочки контура, что не совсем корректно. Рассмотрим ситуацию, когда на заднем плане за маркером есть темный объект, который заставляет его выглядеть так, что вершина находится дальше от центра маркера. Для определения правильного положения вершины, следует использовать остальные пиксели с каждой стороны найденного приближения вершины, чтобы значительно улучшить оценку[6][7].

Алгоритм получения проективного преобразования

В данной работе используется итеративный метод решения задачи получения проективного преобразования[3].

$$\begin{bmatrix} h\hat{x}_{si} \\ h\hat{y}_{si} \\ h \end{bmatrix} = CT_{cm} \begin{bmatrix} x_{mi} \\ y_{mi} \\ z_{mi} \\ 1 \end{bmatrix}, i = 1,2,3,4 \quad (3)$$

Суть метода заключается в минимизации функции ошибки:

$$err = \frac{1}{4}\sum_{i=1}^{4}((x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2) \quad (4)$$

Начальные значения для алгоритма могут выбираться следующими способами:

– При помощи геометрических преобразований над 4-мя крайними точками маркера;

– Используя матрицу из предыдущего кадра.

4. Заключение

В рамках данной работы были рассмотрены:

- Методы компьютерного зрения, применимые для решения задачи распознавания маркера;
- Алгоритм распознавания маркера для дальнейшего дополнения изображения.

В статье изучены различные тонкости расчёта и их оптимизаций, что позволяет лучше понимать внутреннюю структуру инструментов дополненной реальности.

Список используемых источников

1. Augmented Reality Market worth 61.39 Billion USD by 2023. https://www.marketsandmarkets.com/PressReleases/augmented-reality.asp (дата обращения: 08.04.2019г.).
2. Распознавание маркера дополненной реальности / Geektimes. @shadoof, 3 января 2012. URL: https://geektimes.ru/post/135659 (дата обращения: 28.05.2017г.).
3. Inside ARToolKit. Tutorial at ART 02 by Hirokazu Kato. URL: https://www.hitl.washington.edu/artoolkit/Papers/ART02-Tutorial.pdf (дата обращения: 28.05.2017г.).
4. Dismantling ARToolkit Part 1: Labelling / xGoat. Rob Spanton. 8мая 2011г. URL: https://xgoat.com/wp/2011/05/08/dismantling-artoolkit-part-1-labelling/ (дата обращения: 28.05.2017г.).
5. Edge Detection Techniques: Evaluations and Comparisons. Ehsan Nadernejad, Sara Sharifzadeh, Hamid Hassanpour. Applied Mathematical Sciences, Vol. 2, 2008, no. 31, 1507 – 1520.
6. Dismantling ARToolkit Part 2: Contour Detection. Chris Kirkham. 27 июня 2011г.
7. Dismantling ARToolkit Part 3: Locating Vertices. Chris Kirkham. 14 июля 2011г.