Pokermon Go

Tairui Wang
Andrew ID: tairuiw1
Carnegie Mellon University
5000 Forbes Ave, Pittsburgh, PA 15213

Siwei Zhu
Andrew ID: szhu1
Carnegie Mellon University
5000 Forbes Ave, Pittsburgh, PA 15213

Abstract

One of the reason I want to take this course is the popularity of 'Pokemon GO' . However, according to this article[1], 'Pokemon GO' can not be taken as a real augmented reality(AR) game. In this project, we want to apply a real AR application based on poker cards. Our goals of this project are 1) learn to design computer vision applications with the limitation of mobile device. 2) learn basic knowledge of augmented reality.

1. Introduction

From homework 2, we learned how to project 3-D points to 2-D image. Based on this idea, we want to have a application that cards player scan use an iOS device to have a virtual battle against each other. To achieve this goal, we decide to realize the most three essential parts of this AR application in our project.

As we know from the project name, we used poker cards as our landmarks. After localizing our cards, we identified the poker card. Finally, we projected 3-D objects on the poker card according to card pattern.

To solve above steps, the most challenges we faced are:

1) How to localize the card position in different illumination conditions and detect the cards corners very fast;
2) How to identified the cards quickly, also, robustly;
3) How to create attractive 3-D projections.

Our application has a very good performance on video frame rate. Also, the cards identification shows a good reliability during the card detection. However, there are still some other problems need to be fixed and we will talk about this in result analysis part.

2. Background

The reference book we used is [2]. The two most important ideas of our AR application are: 1) pinhole camera model; 2) perspective transformation and camera pose recovery. The first part can be found in chapter 14. The second part can be found from chapter 15.

The platform of our project is an iPad Air 2 with iOS version 9.3.5. The resolution of our video is 640×480. We derived our intrinsic matrix and distortion parameter from ARToolKit. The sampling frame rate is 30fps. Our development IDE is Xcode with version of 7.3.1. The language we used is objective-C++.

The four frameworks used in our project are:

1) AssetsLibrary
2) AVFoundation
3) CoreMedia
4) opencv2

The first three framework are used to create video stream and give us access to each video frame. OpenCV framework has some very convenient image processing functions for us in the application development. What's more, according to Apple document, AssetsLibrary is not suggested to be used in future development. So, if you want to create a similar application to us, it is better to use Photos frame instead.

Armadillo is an very fast linear algebra library which can be used to deal with floating number calculation. However, Armadillo has some compatibility problems with our first three framework(I am not sure which one).

ARToolKit is a very good AR design library. We borrowed some ideas from [3] for cards localization and identification. Also, we plan to improve our application with this library in the future, such as 3-D animation.

3. Approach

3.1. Card localization

There many ways to localize a given land mark. From previous course projects, such as 'Drawing Book'. They used binary feature description and matching method to get the landmark position. However, to improve their performance, they need to apply a RANSAC algorithm to reduce the wrong matching, which will consume a lot of computational power.

In our project, we designed a method which can give us robust card location estimation and doesn't reduce our application frame rate.

This method contains two parts:

1) Binary image contour tracing

From the algorithm name, we can know that first we need to turn our color image into a binary image. So that there are only two kinds of pixel values in the image: '1' or '0'. After we get the binary image, we define the boundary of a binary image: the pixel with value '1' shares at least one edge or vertex with a pixel with value '0'. After we get the contour pixels, we find all the closed contours and find the contour with the biggest area. In our algorithm, if you give several cards and do not overlap them, it is very convenient to find all the cards by sorting cards' area. However, in our project, we limited the card number as one, because we just want to have a demo of an AR application.

More details about contour tracing can be found from [3] and [4].

2) Iterative polygon estimation

As we know, a contour will be presented as a sequence of points. So we need to find the best fixed polygon to get vertices of our poker card. The algorithm I used in our code is called 'Ramer–Douglas–Peucker algorithm'. The algorithm recursively divides the line. Initially it is given all the points between the first and last point. It automatically marks the first and last point to be kept. It then finds the point that is furthest from the line segment with the first and last points as end points; this point is obviously furthest on the curve from the approximating line segment between the end points. If the point is closer than ε to the line segment then any points not currently marked to be kept can be discarded without the simplified curve being worse than ε. If the point furthest from the line segment is greater than ε from the approximation then that point must be kept. The algorithm recursively calls itself with the first point and the worst point and then with the worst point and the last point, which includes marking the worst point being marked as kept. When the recursion is completed a new output curve can be generated consisting of all and only those points that have been marked as kept.

More detail can be found from [5] and [6].

After above two steps are being done, we will get four corner points of our poker cards, which can be used for homography estimation and image replacement.

3.2. Card identification and image replacement

First, we scanned several poker cards and turned them into binary images. Then we took these binary images as our templates.

There are four steps for poker cards identification and card replacement:

1) Calculate homography

Here is a brief introduction of homography.

According to [2],if we have two correspondences x and x', we can get:

$$x' = Hx \tag{1}$$

Then we have:

$$x' \times Hx = 0 \tag{2}$$

Since H is a 3 bu 3 matrix with 8 degree of freedom. We need at least 4 correspondences to calculate homography. In our application, we take the four card corners as our correspondences.

2) Project card from the table to our templates

After we get the H, we use (1) to project our poker card on the table to our templates. Then we turn our projected card into binary image.

One thing should be noticed is that, to avoid the holes of our projection, we should search the projected area first and using proper interpolation method to get correct value. So, we should change equation (1) a little bit:

$$x = H^{-1}x' \tag{3}$$

3) Compare cards difference

This step is very simple. We just need to calculate the difference between detected cards and our templates. The best match should be the card with the smallest difference.

4) Poker card replacement

In this step, we want replace our poker card with some interesting monster characters, such as Charizard or Psyduck.

In card replacement, we use the same method from step two. First, we find the homography between monster image and detected card. Then, we use equation (3) to redrawing the detected card.

3.3. 3-D points cloud projection

According to [2], we know the pinhole camera model is:

$$x = K[R \mid t]X \tag{4}$$

where K is camera intrinsic parameter, R is camera orientation, t is the translation between camera frame and reference frame, X is 3-D point.

As we mentioned in introduction, our intrinsic parameter is derived form ARToolKit. However, if you have a very high accuracy requirement of image, you can do the calibration by yourself with OpenCV. More details can be found in [7].

The method to compute R and t is form [2]. Here is a brief introduction:

First we need to make some notes of our variables:

$$H = \begin{bmatrix} h_{11} & h_{11} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{32} & h_{32} & h_{33} \end{bmatrix}$$

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

$$t = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}$$

Before we calculate the extrinsic matrix, we need to eliminate the effect of intrinsic matrix. This gives a new homography that:

$$H' = \begin{bmatrix} h'_{11} & h'_{11} & h'_{13} \\ h'_{21} & h'_{22} & h'_{23} \\ h'_{32} & h'_{32} & h'_{33} \end{bmatrix} = \lambda' \begin{bmatrix} r_{11} & r_{12} & t_x \\ r_{21} & r_{22} & t_y \\ r_{32} & r_{32} & t_z \end{bmatrix}$$

where λ' is the scale.

We compute the SVD of the first two columns of H':

$$\begin{bmatrix} h'_{11} & h'_{12} \\ h'_{21} & h'_{22} \\ h'_{31} & h'_{32} \end{bmatrix} = ULV^T$$

And then set

$$\begin{bmatrix} r_{11} & r_{12} \\ r_{21} & r_{22} \\ r_{31} & r_{32} \end{bmatrix} = U \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} V^T \quad (5)$$

The last column of rotation matrix $\begin{bmatrix} r_{13} & r_{23} & r_{33} \end{bmatrix}^T$ can be derived by taking the cross product of the first two columns.

The final step of rotation matrix estimation is to calculate the determinant of R. If it is -1, we multiply the last column with -1. If it is +1, we do nothing.

After we have the rotation matrix, we need to estimate the scale factor λ':

$$\lambda' = \frac{\Sigma_{m=1}^{3} \Sigma_{n=1}^{2} h'_{mn} / r_{mn}}{6} \quad (6)$$

This allows us to estimate the translation vector as:

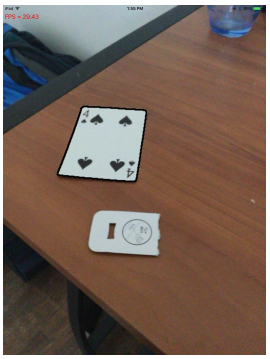
$$t = \begin{bmatrix} h'_{13} & h'_{23} & h'_{33} \end{bmatrix}^T / 6 \quad (7)$$

Finally we achieve the extrinsic matrix $\begin{bmatrix} R \mid t \end{bmatrix}$.

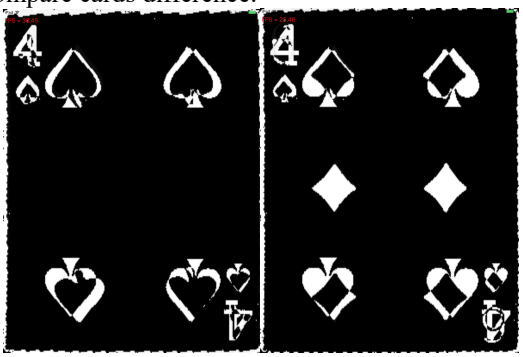
Once we have intrinsic and extrinsic matrix of our camera, we can project 3-D points to 3-D image with equation (4).

4. Result

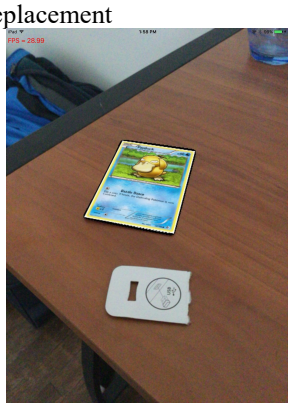
1) Find poker card contour:



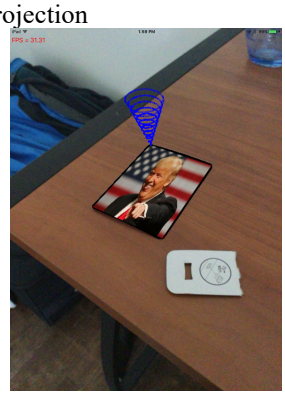
2) Compare cards difference:



3) Poker card replacement



4) 3-D points projection



Basically we finished our goal to create a real AR application. Our method has a good performance on card localization and card pattern recognition. Our application video frame rate is as high as the sampling rate, which means there is computation redundancy further application development. However, there still some problems need to be fixed in our application:

1) Our application is sensitive to video rotation. This problem can be solved by rotating image and sorting corners order.

2) If there too many card templates, calculating difference of each template might not be as fast as feature matching algorithms, such as BRISK.

3) We are not focusing on computation acceleration because of the fps we have. Once we apply some more complicated animation, the computation load might be too heavy for iPad CPU.

5. List of work

Equal work was performed by both project members.

6. GitHub Page

https://github.com/TairuiWang/Pokermon_Go

References

[1] http://venturebeat.com/2016/07/14/stop-referring-to-pokemon-go-as-augmented-reality/

[2] Simon J.D Prince. Computer Vision: Models, Learning, and Inference,2012

[3] http://www.imageprocessingplace.com/downloads_V3/root_downloads/tutorials/contour_tracing_Abeer_George_Ghuneim/intro.html

[4] Satoshi Suzuki. KeiichiA be. Topological structural analysis of digitized binary images by border following, doi:10.1016/0734-189X(85)90016-7

[5] https://en.wikipedia.org/wiki/Ramer%E2%80%93Douglas%E2%80%93Peucker_algorithm

[6] Urs Ramer, An iterative procedure for the polygonal approximation of plane curves, doi:10.1016/S0146-664X(72)80017-0

[7] http://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html