Camera Object detection with 3D Reconstruction

Bo Wu
University of California, Davis
bowu@ucdaivs.edu

Peng Shu Ming
University of California, Davis
shmpeng@ucdavis.edu

ABSTRACT

In camera calibration, the objective is to determine camera parameters which describe the relationship between 3D world coordinates and 2D image coordinates. We can utilize the mapping formulation to determine coordination of the object points. With the rising popularity of using computer vision in smart phones, the performance of 3D reconstruction process by smart phones needs to be inspected and verified. We implemented 3D reconstruction with camera calibration on iOS device and inspect relationship between reconstruction performance and numbers of taken images.

Categories and Subject Descriptors

[**Camera Calibration**]: Stereo Camera; [**Software Engineering**]: Metrics—*complexity measures, performance measures*

General Terms

OpenCV

Keywords

Camera Calibration, OpenCV, C++, Objective-C, Galileo, iPhone

1. INTRODUCTION

In computer vision, 3D reconstruction is the process of determining appearance and location of objects in world coordination. There are different kinds of techniques of completing the reconstruction process [1]. For active method, it actively interferes with the reconstructed object mechanically or radiometrically to obtain the 3D model. The scanners emit radiation or light to detect the beam reflection from the objects. There are limitations for the method. For example, optical techniques are not able to examine the shiny objects [2] and it is not feasible for modeling distant or fast-moving objects [3]. For passive method, optical techniques are utilized for recovering scene and reflectance characteristics from images. A set of images are obtained by observing the object from different viewpoints and illuminations to compute the shape and reflectance at surface points [3]. The techniques tend to be much more affordable compare to active methods.

In our project, we use camera calibration, a passive method, as a technique to reconstruct 3D object points. To achieve our object, we use iPhone 4s back facing camera, OpenCV, an open source computer vision library, and Galileo. The algorithm of camera calibration in OpenCV is based on [4]. In the report, we will first briefly describe information regarding camera calibration and then present implementation steps and results. Section 2 describes the basic formulation of camera calibration model and briefly discusses the camera parameters. The parameters are utilized to obtain the mapping between 3D world coordinates and 2D image coordinates. Section 3 presents stereo camera being utilized in the project and our implementation process is presented in Section 3. Section 4 describes how we can the world points by our process, and section 5 is the improvement of the previous work. Section 6 presents our experimental results by implementing steps described in previous section.

2. CAMERA PARAMETERS

The camera parameters include internal camera parameter, and extrinsic camera parameter. The internal camera parameter is called camera matrix.

2.1 Camera Calibration

The step of our project is finding the camera matrix of iPhone. OpenCV has the sample code to calibrate the camera. It uses the chess board and get the chess board corner as world points. In the image, it can find the image points corresponding to the chess board corner. The outputs of the sample code are including the camera matrix, rotation vector, translation vector, distortion coefficients, and the reprojection error.

From the pin-hole camera model, the camera parameters are the unknown variables, and the known variables are the image points and world points. In each image, it has 49 pairs of points, due to the chess board is 7 by 7. With multiple images, there will be a lot of equations to solve the few unknowns.

2.2 Camera Calibration for iPhone

After running the Camera Calibration sample code, the next step is finding the camera matrix of iPhone. Firstly, we tried to adjust the OpenCV code to the Objective-C vision. However, we had some problems during converting code.

It requires multiple views in order to find the parameters. However, iOS will handle the camera as different events, which means the new event comes it will update the older one and it cannot save it. We had trouble to save the new images points and world points. In order to solve the camera matrix and move on to the next step, we took multiple images by iPhone and use the OpenCV code to solve the camera matrix.

Because of the camera matrix depends on the image size. The size we used is 480X640, and we will continue to use this size for the next step. We took five different images of chess boards,

3. STEREO CAMERA

The propose of our project is finding the object points in the world. It is really hard to find the point by one camera. The solution will be using the stereo camera. Stereo camera requires two cameras. It simulates a pair of human eyes. The position between those two cameras is including some rotation and translation. This means that the first camera rotates and translates by some numbers is the position of the second camera.

In our project, we do not have two cameras to get the stereo camera. The solution is using one camera to take multiple images in the different positions. Those position have the unknown rotations and translations. So that we constructed a stereo vision with only one camera.

3.1 Capture Images

The next step is capturing the images from the iPhone camera. We wrote a iPhone application to do that. The application is based on the OpenCV iOS library and it is developed in Objective-C. OpenCV has two kinds of camera: one is image camera, the other is video camera. These two kinds of camera can save the image or video as Mat format for image processing and video processing. The application can start the camera, capture images, and save the images.

We used the Galileo to rotate the cellphone in order to have the stereo vision. Galileo is a pan/tilt rotation unit. It can make the iPhone have a 360 degree view. We only use the pan unit to rotation, so the cellphone can rotate in one plane which can make the rotation and translation easier to estimate. The application can also connect to Galileo and control the Galileo's pan rotate 5 degrees.

We took five images. The first image is defined as world coordinate which means the rotation matrix is a identity matrix, and the translation vector is 0. The next image will be taken after the iPhone rotate 5 degrees. This will make us easy to estimate the rotation and translation.

3.2 Rotation Matrix

In order to get the world coordinate points, we need to find the projection matrix. The projection matrix is the product of the camera matrix and the rotation matrix plus translation vector. The next step is finding the rotation matrix and translation vector. There is a matrix called fundamental matrix between two image views which we will discuss in the next section. From the fundamental matrix we can get the translation and rotation between tow images by the corresponding points in the two images of the same world points.

The rotation matrix is easy to estimate. Because the cellphone is rotated by the Galileo, it is really easy to know the rotation matrix.

In three dimensions the basic rotation is a rotation about one of the axes of the coordinate system. There are three kinds of the rotation matrix. They are listed as:

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\theta) & -sin(\theta) \\ 0 & sin(\theta) & cos(\theta) \end{bmatrix}$$

$$R_y(\theta) = \begin{bmatrix} cos(\theta) & 0 & sin(\theta) \\ 0 & 1 & 0 \\ -sin(\theta) & 0 & cos(\theta) \end{bmatrix}$$

$$R_z(\theta) = \begin{bmatrix} cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

In our process, we choose the R_z as our estimation.

3.3 Translation Vector

The hardest part of this project is estimating the translation vector. The translation is between two images are the movement from the first camera center to the second camera center. It will be estimated by geometry. First thing we need is the distance between the center of iPhone and the center of iPhone camera. Because the Galileo holds iPhone and rotates in the centre of iPhone, it is really necessary to know the distance. After rotating the iPhone, we can get triangle with two 2 cm sides, and an angle which is the rotation angle. The translation is the movement from the position of first camera to the position camera which is the third side of the triangle. So we can obtain the translation vector from that triangle.

The translation estimation figure shows how we estimate the translation vector. All we need to find is the length of x and z. Because in this rotation case, y is 0.

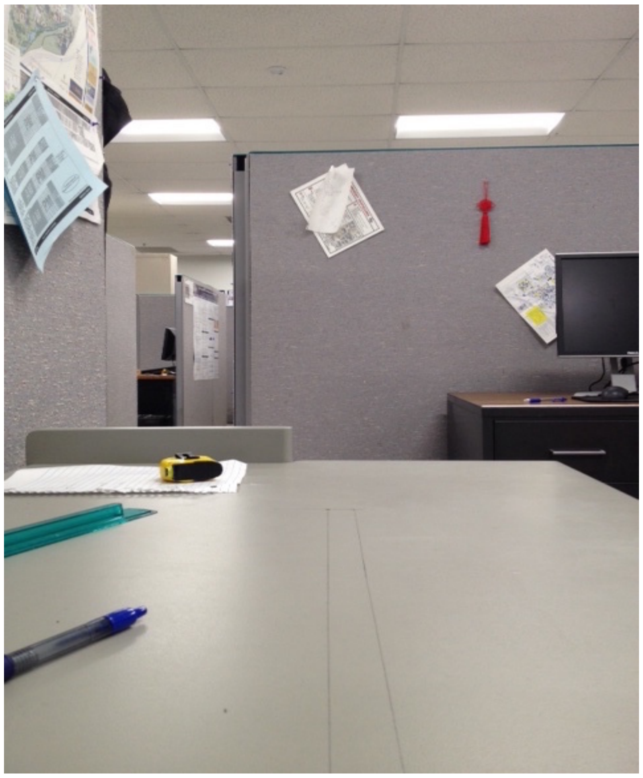


Figure 1: Image 1

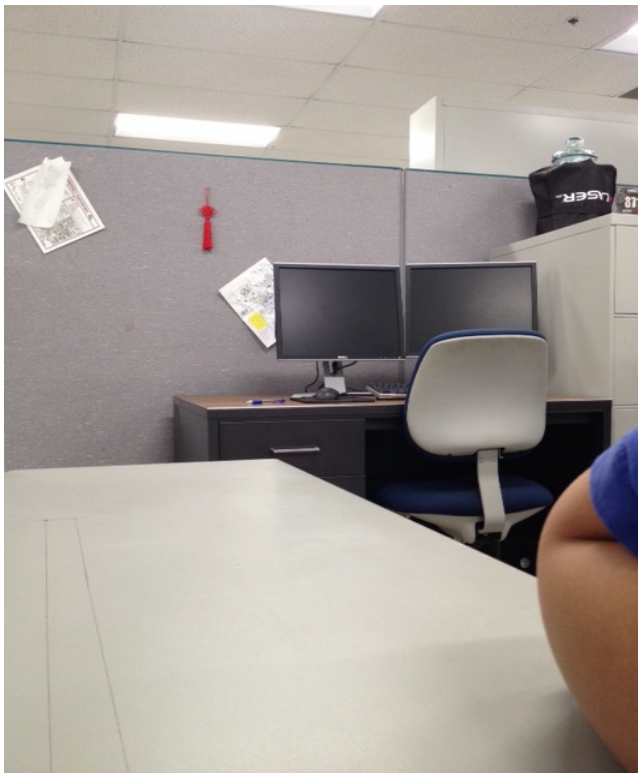
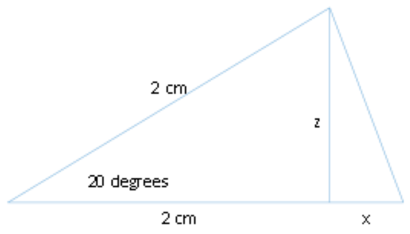


Figure 2: Image 5



The Translation estimation.

4. TRIANGULATION

After obtaining the camera matrix, rotation matrix, and translation vector, we are able to get the projection matrix. The triangulate function in OpenCV requires two projection matrices, two corresponding points from two images as the inputs, and the output 4D points as in the world coordinates.

For the triangulation, we used the first image and the fifth image. Because between those two image, the rotation is 20 degree. This will make the rotation matrix have a lot differents with the identity matrix and the translation vector inclose to 0. Figure 1 and Figure 2 are the first image and second image. There is a black dot in both images which we draw on the table. We defined this point as the world point that we need to find from triangulate.

Because the first image is defined as the world coordinates, this world point should in the line that is perpendicular to the camera. This makes the x of this point is 0. The y of this point is the height of iPhone 4s plus the Galileo which we measured by rule is 14.8 cm. The z is the five times y. We setting the world point this way will help us easy to get the result.

5. FUNDAMENTAL MATRIX

Finding the fundamental matrix is the final step of our project. OpenCV has the function called "findFundamentalMat" The inputs of the function are the two projection matrices from the two cameras, two sets of the corresponding image points from two different views. It requires at least seven points from one view and other seven points from the other.

6. RESULT

In this section, we will discuss our results for each part, and also the improvement of the results.

6.1 Camera Matrix

Because the image size is 480X640, the $C_x = 240$ and $C_y = 320$. The camera matrix we got is:

$$CameraMatrix = \begin{bmatrix} 542.517 & 0 & 239.5 \\ 0 & 542.517 & 319.4 \\ 0 & 0 & 1 \end{bmatrix}$$

From this camera matrix, we know the focal length of x and y are the same. The image center is (239.5, 319.5). This is satisfied the image size.

6.2 Rotation

The rotation angle is 20 degree, so the θ is 20 degree. The rotation matrix just plug in the equation we presents previously.
The rotation matrix is:

$$RotationMatrix = \begin{bmatrix} 0.9396 & -0.342 & 0 \\ 0.342 & 0.9343 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

6.3 Translation

The translation vector is the length of two sides of the right triangle whose hypotenuse is the movement of the camera centre. By geometry the length of two sides are $2 \times sin(20) = 0.684$ and $2 \times (1 - cos(20)) = 0.1206$. So the translation vector is:

$$TranslationVector = \begin{bmatrix} 0.1206 & 0 & 0.684 \end{bmatrix}$$

6.4 Triangulation

In order to get the triangulation work, we got the project matrix by multiply camera matrix and rotation matrix plus translation vector. These are the two projection matrix.

$$ProjectionMatrix_1 = \begin{bmatrix} 542.517 & 0 & 239.5 & \\ 0 & 542.517 & 319.5 & \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$ProjectionMatrix_2 = \begin{bmatrix} 509.748 & -185.540 & 239.5 & 229.245 \\ 185.540 & 509.7489 & 319.5 & 218.538 \\ 0 & 0 & 1 & 0.684 \end{bmatrix}$$

After that, all we need to do is plugging those parameters in the function to get the world coordinates point. The output is:

$$P = \begin{bmatrix} -0.005 & -0.1788 & -0.9766 & 0.1189 \end{bmatrix}$$

After normalization the point is:

$$P_n = \begin{bmatrix} -0.0431 & -1.5045 & -8.2141 & 1 \end{bmatrix}$$

6.5 Fundamental Matrix

All of our previous work is based on estimating the rotation and translation. This will make our output point not close to the correct result. in order to avoid this kind of error we move on to the next step, our final step, using the fundamental matrix to get the rotation and translation from two views. To get the fundamental matrix, we need to find at least seven points from each image. We found eight points and figure 3 show the eight points which show as the green dots we found and figure 4 shows the corresponding points in the image 5 which show as the red dots. The points in the figure 3 are all on right side of the image, because after rotating they will become on the left side of the image. We measured all of sixteen points in pixel in order to plug them in the function.

OpenCV has one function to solve the fundamental matrix, but there are three kinds of algorithm implementation. We tried all of them and get two different kinds of fundamental matrix, and they are:

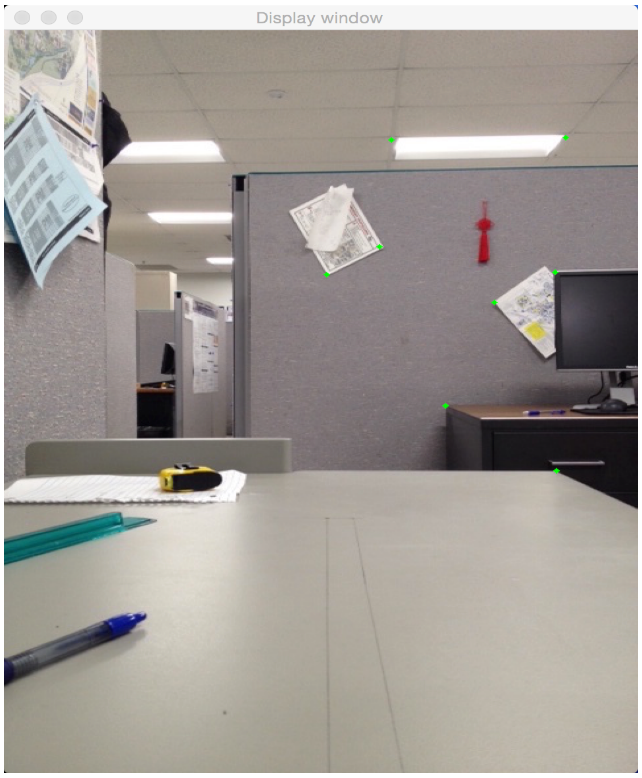


Figure 3: Image 1 with eight points

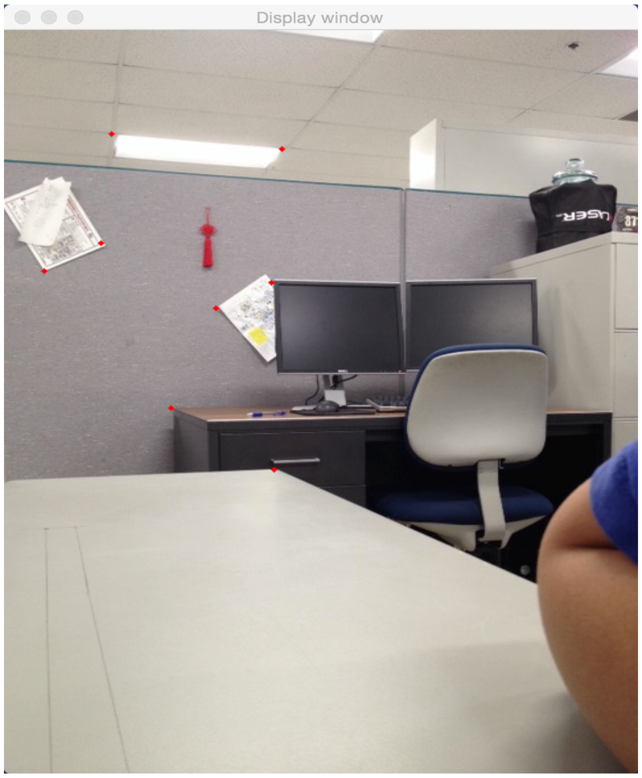


Figure 4: Image 5 with eight points

Table 1: Results of Eight Points

Point 1	Point1'	World Point
(244, 210)	(30, 234)	(-0.0106, 0.1847, -0.9277, 1)
(371, 234)	(160, 239)	(-0.2189, 0.1509, -0.9309, 1)
(417, 208)	(202, 217)	(-0.2684, 0.1777, -0.8451, 1)
(293, 94)	(81, 89)	(-0.0674, 0.2802, -0.6748, 1)
(284, 186)	(83, 183)	(0.15907, 0.28347, -0.81149, 1)
(334, 323)	(126, 325)	(0.1702, 0.2931, -0.8336, 1)
(418, 379)	(204, 378)	(0.1691, 0.2846, -0.8149, 1)
(425, 92)	(210, 102)	(0.143, 0.268, -0.7789, 1)

$$F_1 = \begin{bmatrix} 3.166e-6 & 4.54e-6 & 0.02 \\ -3.456e-6 & 3.12e-8 & 0.017 \\ -0.04 & -0.00122 & 1 \end{bmatrix}$$

$$F_2 = \begin{bmatrix} -0.0005 & ?0.0028 & 0.2807 \\ 0.00265 & 7.25e-5 & -07639 \\ -0.0358 & 0.1978 & 1 \end{bmatrix}$$

Both of them are right. Because they are satisfied the definition of the fundamental matrix.

Next, we need to find the essential matrix which is equal to the transpose of camera matrix times the fundamental matrix times the camera matrix. We need to do the singular value decomposition(SVD) for the essential matrix. Because the essential matrix also equal to:
$E = U\Sigma V^T$
There are few properties about the essential matrix, The first is $\Sigma = diag(s, s, 0)$.
In our result, we got $\Sigma = \begin{bmatrix} 3.15 & 2.94 & 0 \end{bmatrix}$, which has a little bit errors. To get the rotation matrix we need to defined a matrix:
$W = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ The rotation matrix is equal to $R = UWV^t$ or $R = UW^tV^t$
The translation vector is equal to $V = U_3$ or $V = -U_3$
From these we can get the rotation, translation and projection matrix. So we can rerun the triangulatePoints to get the world points of those eight points and the one we used previously.
The first point is $\begin{bmatrix} -0.003 & -0.1653 & -0.9163 & 0.3646 \end{bmatrix}$
After normalization is $\begin{bmatrix} -0.0092 & -0.4535 & -2.5131 & 1 \end{bmatrix}$. From this result, x is close to 0, and z is five times of y. This shows a really good result.
Table 1 shows the eights objects in the images points and the world points from the triangulation.

The results are really hard to understand, because these numbers doesn't make sense at first time. However, look them carefully, we can find out the first point has the almost same x-value with the black dot we draw on table. Both of their x value is close to 0. Point 1 and point 2 they are on the same z plane, and their z-value are really similar. In conclusion, if those point has some common value, for example they are on the same plane, they will have one value of their coordinate is the same.

7. ERRORS

Finally, we will talk about the errors we have in our projects. The first error is from the estimation of rotation and translation. The rotation matrix is the basic rotation on 3D. This is depends on the Galileo having a really perfect rotation which we are not 100 percent sure. The translation is estimated from the geometry, which can have a really huge error. Because everything we measure from a ruler. After using the fundamental matrix, we can get some really reasonable rotation and translation results.
However, there is an error we cannot avoid is finding the image points. We found the image points by click from the left top of the image to the object in the image, and the Preview will show the image points in pixel unit. If my hand shakes, the image point will have a 4 or 5 pixels error. From figure 3 and figure 4, it is easy to see some object points are not in the same position. I think this is the reason why the essential matrix's singular values have 0.2 different.

8. REFERENCES

[1] 3D reconstruction
http://en.wikipedia.org/wiki/3D_reconstruction

[2] Cao-long HUYNH, ?How to improve the laser scanning of shiny surfaces which can give unwanted reflections.? CVMT Aalborg University: Department of Electronic System 2010.

[3] Seitz, S.M., "An Overview of Passive Vision Techniques" in Siggraph 99 Course on 3D Photography (1999).

[4] Z. Zhang., ?A Flexible New Technique for Camera Calibration? IEEE Transactions on Pattern Analysis and Machine Intelligence, 22(11):1330-1334, 2000.

[5] Camera Calibration and 3D Reconstruction?OpenCV http://docs.openc
uction.html

[6] Geometric Camera Parameters
http://www.cse.unr.edu/ bebis/CS791E/Notes/CameraParameters.pdf

[7] Camera Calibration OpenCV
http://docs.opencv.org/trunk/doc/py_tutorials/py_calib3d/py_calibration
ation.html#calibration

[8] EpipolarGeometry OpenCV
http://docs.opencv.org/trunk/doc/py_tutorials/py_calib3d/py_epipolar_g
y_epipolar_geometry.html#epipolar-geometry