Project Stereo

Author: Yongshun Zhang

April 4, 2018

1 Camera Basics

1.1 Implement Zhang's Method

Problems The first report has described some intractable problems I have met, when I try to implement the method of Z.Zhang, and the outcomes I get is not satisfying because of the big error. And now I figure out the bug, it's so stupid, because I forget use the parameter λ.

When I get the intrinsic parameters such as f_x and f_y, I compare them with the intrinsic parameters got by OpenCV function calibrateCamera(). I find that the intrinsic parameter I get is too small, so I focus on f_x and f_y and forget λ. And finally, I figure out that the single values of intrinsic parameters make on sense, the ratio of intrinsic parameters and λ matters, as equation(1) shows [2].

$$\begin{bmatrix} \mu \\ \upsilon \\ 1 \end{bmatrix} = \frac{1}{\lambda} \cdot K \cdot \begin{bmatrix} R & T \end{bmatrix} \cdot \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} \tag{1}$$

Then I modify the code, adding the parameter λ to the transform of coordinates.

Code of the implementation of Z.Zhang's method can be found by link:

https://github.com/zhangyongshun/Project_Stereo/blob/master/python_code/.idea/Z.ZhangCameraCalibration.py

1.2 Estimate the Depth of the Pixels

As what we derive out in Problem 3, each of the points in images coordinates is corresponding to a line in camera 3D coordinates, so we can't use a single camera to estimate the depth of the pixels. Although we can't get the distance of 3D points to the image plane from a single camera, we can get the distance with two cameras, as the following figure show.

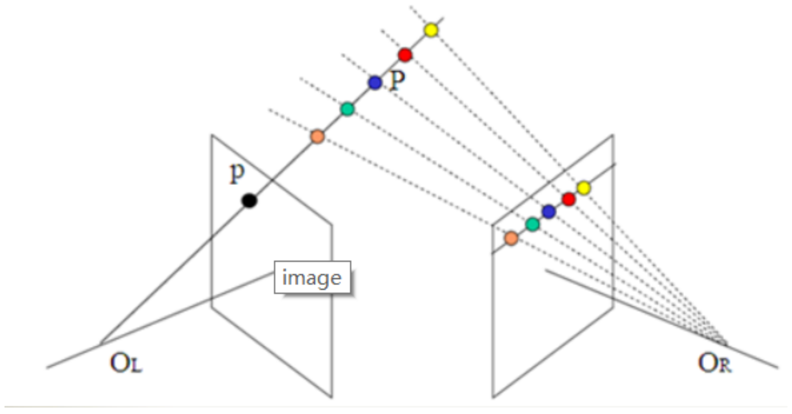


Figure 1: Two Cameras System

From the figure we can see, when a object **P** in 3D coordinates is projected to camera O_L as image 2D point **p**. We suppose is the projection of **P** in camera O_R, so we can get two lines by the points **p** and , and the intersection of two lines is point **P**, which means we get the coordinate of point **P** in 3D coordinates. The detail explanation will be shown in **Binocular Basics**.

2 Binocular Basics

2.1 Projection

Because the 3D world coordinates be aligned with the 3D camera coordinate of the left camera, so 3D world coordinate X also be left camera coordinate.

Projection in left camera planes.

$$\begin{bmatrix} \mu_l \\ \upsilon_l \\ 1 \end{bmatrix} = \frac{1}{X_3} \cdot M_l \cdot \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix} \tag{2}$$

[R | t] is the transform matrix from left camera coordinates to right camera coordinates .Projection of point X in right camera plane is shown in equation(3).

$$\begin{gathered} \begin{bmatrix} X_{1l} \\ X_{2l} \\ X_{3l} \end{bmatrix} = \begin{bmatrix} R & t \end{bmatrix} \cdot \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix} \\ \begin{bmatrix} \mu_r \\ \upsilon_r \\ 1 \end{bmatrix} = \frac{1}{X_{3l}} \cdot Mr \cdot \begin{bmatrix} X_{1l} \\ X_{2l} \\ X_{3l} \end{bmatrix} \end{gathered} \tag{3}$$

2.2 Epipolar Line

Condition Derivations under condition that the 3D world coordinate be aligned with the 3D camera coordinate of the left camera.

As what we derive out in Problem 3, a point x_l in left image coordinates is corresponding to a line in camera 3D coordinates. The derivation is as follows.

First, transform the image point $(\mu,\ \upsilon)$ from pixel coordinates to world coordinates.

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = Z_c \cdot {M_l}^{-1} \cdot \begin{bmatrix} \mu \\ \upsilon \\ 1 \end{bmatrix} \tag{4}$$

we set $\frac{X_c}{Z_c} = k,\ \frac{Y_c}{Z_c} = t$, k and t are constants, so the line can be describe as $Z_c \cdot (k,\ t,\ 1)$. For each point on the line, we make a projection of it to right image.

$$\begin{bmatrix} X_r \\ Y_r \\ Z_r \end{bmatrix} = \begin{bmatrix} R \\ t \end{bmatrix} \cdot \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \begin{bmatrix} R \\ t \end{bmatrix} \cdot \begin{bmatrix} Z_c * k \\ Z_c * t \\ Z_c \end{bmatrix} \tag{5}$$

Apparently, $[X_r, Y_r, Z_r]^T$ is also a line, we set it as $Z_c * [a, b, 1]^T$, so the projections on right image can be described as follows

$$\begin{bmatrix} \mu_r \\ \upsilon_r \\ 1 \end{bmatrix} = Mr \cdot \begin{bmatrix} X_r \\ Y_r \\ Z_r \end{bmatrix} = Z_c \cdot \left(Mr \cdot \begin{bmatrix} a \\ b \\ 1 \end{bmatrix} \right) \tag{6}$$

which is a line equation called **epipolar line** depended on Z_c.

Figure 1 above clearly shows the relationship between point p=(μ, υ) in left image and its epipolar line in right image.

2.3 Fundamental Matrix

Condition Derivations under condition that the 3D world coordinate be aligned with the 3D camera coordinate of the left camera.

Before I start Problem 11, I first read the blog which introduces the parameters of binocular cameras [1]. The derivation of fundamental matrix uses knowledge of epipolar geometry, which can be found in wikipedia [4] and cnblogs [5].

Before deriving the fundamental matrix, we derive essential matrix first.

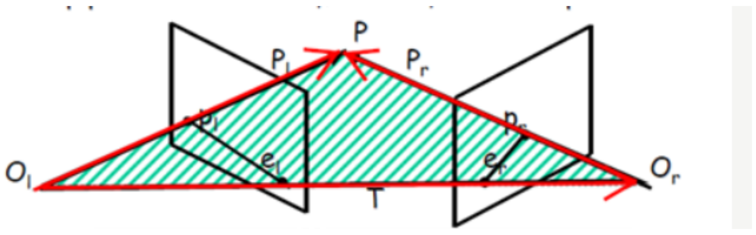


Figure 2: Epipolar Constrain

As Figure 2 shows, $p_l,\ p_r$ is points on image plane. vectors P_1, T and $[P_1 - T]$ are coplanar, which calls **epipolar constraint**, and we can get the following relationship between p_l, p_r, P_1, T and $[P_1 - T]$.

$$p_l = \frac{1}{s_l} \cdot M_l \cdot P_l$$

$$p_r = \frac{1}{s_r} \cdot M_r \cdot P_r \tag{7}$$

$$(P_l - T)^T \cdot T \times P_l = 0, where P_l - T = R^T P_r$$

Then we change the equation as

$$P_r{}^T R \cdot T \times P_l = 0 \tag{8}$$

There is a mistakable recognization of translation vector **T** that we should pay attention to , because it is different from the translation T in pinhole camera calibration [2]. And we change the vector $T \times P_l$ to S • P$_l$.

$$S = \begin{bmatrix} 0 & -T_z & T_y \\ T_z & 0 & -T_x \\ -T_y & T_x & 0 \end{bmatrix}$$
$$P_r{}^T R \cdot T \times P_l = P_r{}^T RSP_l = 0 \tag{9}$$

Let $E = RS$, which is **essential matrix**. As we can see, if we use essential matrix to analyse digital images **E**, we need to know intrinsic parameters of both cameras, but how do we find the relationship between points on two image planes when we don't know the matrices M_l and M_r? So we introduce the fundamental matrix. According to equation (7) and (8), we have

$$\begin{aligned} P_l &= M_l{}^{-1} \cdot p_l \\ P_r &= M_r{}^{-1} \cdot p_r \end{aligned} \tag{10}$$

$$p_r{}^T \cdot (M_r{}^{-1} E M_l{}^{-1}) \cdot p_l = p_r{}^T F p_l = 0, \text{where F is called fundamental matrix.} \tag{11}$$

Epipolar Geometry and Stereo Calibration, Rectification

Before I start to write down answers with following jobs, I'll sum up the knowledge of **stereo calibration, rectification** and **epipolar geometry** firstly, and describe the problems during the process of learning.

According to Project_stereo's suggestion, I read the csdn blog about parameters which need to be calibrated firstly [1]. We suppose there is a point P in world coordinates, so we can describe the coordinates of P in left camera coordinates and right coordinates respectively [2] [3].

$$\begin{cases} P_l = R_l P_w + T_l \\ P_r = P_r P_w + T_r \end{cases} \tag{12}$$

And the relationship between P_l and P_r is

$$P_r = RP_l + T \tag{13}$$

Matrix R is rotation matrix from left camera coordinates to the right, and vector **T** is translation vector from left camera coordinate to right. We should take notice of the order between R and T, because we rotate the coordinate first and then translate it. Equation of R and T can be derived based equation (12) and (13).

$$\begin{cases} R = R_r R_l{}^T \\ T = T_r - RT_l \end{cases} \tag{14}$$

epipolar geometry

When the matrices M_r, M_l, R_r, R_l, R and translation vectors T_r, T_l, T are calculated out, I think the process of stereo calibration is done, because essential matrix **E** and fundamental matrix can be derived based on the matrices, according to **epipolar geometry**. Because I never know epipolar geometry before, so I first learn the basic knowledge of it on wikipedia [4], and it is easy to understand the concepts of epipolar geometry. Then I find the Xiangxiaren's blog [5] and Robert Collins's courseware about epipolar geometry on computer vision [6], which states clearly about the knowledge and derivations of epiplolar geometry in sterao calibration. The derivations of essential matrix E and fundamental matrix F are as section **2.3 Fundamental Matrix** shows.

RANSAC

When I read the tutorial from the University of Illinois [7], I learn that there are many algorithm to estimate the fundamental matrix, such as 8-point algorithm and 7-point algorithm. The main idea of them are the same: finding the pairs, solving homogeneous equation, just as what Z.Zhang does in calculating homographic matrix H [11], and they can be optimized by normalizing coordinates [8] and using RANSAC to deal with outliers. In order to figure out what RANSAC does in 8-point algorithm, I first read the introduction of RANSAC algorithm in wilipedia [9] and know how RANSAC algorithm work and its applications, but then I still not figure out what the RANSAC is used to do in 8-points algorithm. So I search the related parpers about RANSAC algorithm and 8-points algorithm, and I get Michal Fularz's paper, which states RANSAC algorithm is the most commonly used method of robust estimation, just like incorrectly matched point pairs [10].

Bouguet's Algorithm

The tutorials of stereo image calibration in GuiDo Gerig'course [18] and the book "Learning OpenCV" [19] introduce some algorithms to reproject image planes onto a common plane parallel to the line between camera centers. Bouguet's algorithm introduced in Learning OpenCV [19] and Andrea Fusiello's algorithm [20] adopt the similar idea to find the rectification measure, and Wikipedia also introduces other methods [26] such as what Z.Zhang does [21]. Andrea Fusiello's algorithm derives the homographies from world coordinates and origin camera coordinates to parallel stereo camera system, and Bouguet's algorithm derives the homographies according to the relationship between left and right camera coordinates. I learn the Bouguet's algorithm mostly, because it's realized in OpenCV function stereoRectify().

Given the rotation matrix and translation (R, T) between the stereo images, Bouguet's algorithm for stereo rectification simply attempts to minimize the amount of change reprojection produces for each of the two images (and thereby minimize the resulting reprojection distortions) while maximizing common viewing area by half rotating the left and right camera coordinates based on rotation matrix R, which is realized with rotation vector [22] by function cvConvertScale() in OpenCV source code [23]. But I want to find the strict proof that rotating half could minimize the resulting reprojection distortions, so I read lots of data on wikipedia and other websites [24] [25] [26], but no useful information is found.

After we split the rotation matrix R into r_l and r_r, we can puts the cameras into coplanar alignment but not into row alignment. So we need other rotation matrix R_{rect} to the left camera' s epipole to infinity and align the epipolar lines horizontally, which is based on translation vector T.

$$R_{rect} = \begin{bmatrix} (e_1)^T \\ (e_2)^T \\ (e_3)^T \end{bmatrix}$$

$$e_1 = \frac{T}{||T||} \tag{15}$$

$$e_2 = \frac{[-T_y, T_x, 0]}{({T_x}^2 + {T_y}^2)^{\frac{1}{2}}}$$

$$e_3 = e_1 \times e_2$$

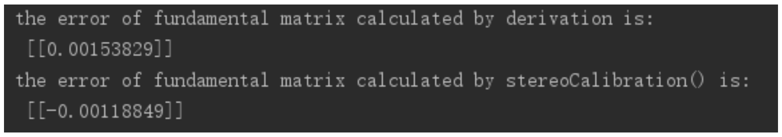
The row alignment of the two cameras is then achieved by setting

$$\begin{aligned} R_l &= R_{rect} r_l \\ R_r &= R_{rect} r_r \end{aligned} \tag{16}$$

2.4 Stereo Calibration

Firstly, I calibrate the left camera and right camera with images in left.zip and right.zip respectively, and I get the rotation matrices R_l, R_r and translations T_l, T_r with function cameraCalibrate(). Then I follow the derivations in **Section 2.3**, so matrices R, T, E, F are calculated out.

In the program, I use the opencv function stereoCalibrate() [27] to get those matrices cvE, cvF in order to make comparison with matrices E, F calculated out according to **Section 2.3**, and calculate the expression: ${x_r}^T F x_l$ with F and cvF, which is expected to be zero. The outcomes are as Figure 3 shows, which errors is zero approximately.



```
the error of fundamental matrix calculated by derivation is:
 [[0.00153829]]
the error of fundamental matrix calculated by stereoCalibration() is:
 [[-0.00118849]]
```

Figure 3: the output error

Then I choose a point in the left image and drop the epipolar line which is calculated with fundamental matrix, in the right image.

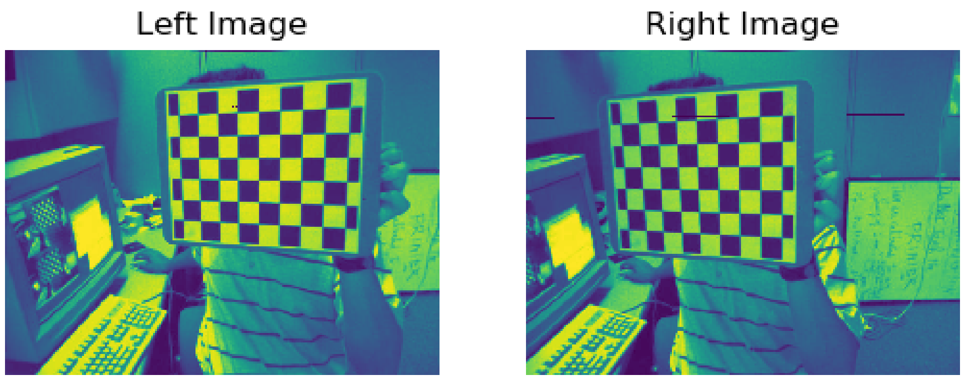


Figure 4: epipoplar line

The code of stereo calibration can be found by link

https://github.com/zhangyongshun/Project_Stereo/blob/master/python_code/binocular_basics/opencvStereoCalibration.p

Problem

The most difficult problem I met is the **wrong form** with matrix S in tutorial [1], which is shown below.

$$S = \begin{bmatrix} 0 & -T_x & T_y \\ T_z & 0 & -T_x \\ -T_y & T_x & 0 \end{bmatrix} \tag{17}$$

Matrix S is define by the equation(15)

$$\begin{cases} (P_l - T_r)^T (P_l \times T) = 0 \\ P_l \times T_r = SP_l \end{cases} \tag{18}$$

and the right form of S should be

$$S = \begin{bmatrix} 0 & -T_z & T_y \\ T_z & 0 & -T_x \\ -T_y & T_x & 0 \end{bmatrix} \tag{19}$$

This error takes me a few days to figure out. When I learn the tutorial, I don't derive the form of S by myself, which I should do.

So when I calculate out the fundamental matrix F based with wrong matrix S, I get the wrong epipolar line, which as Figure 5 shows.

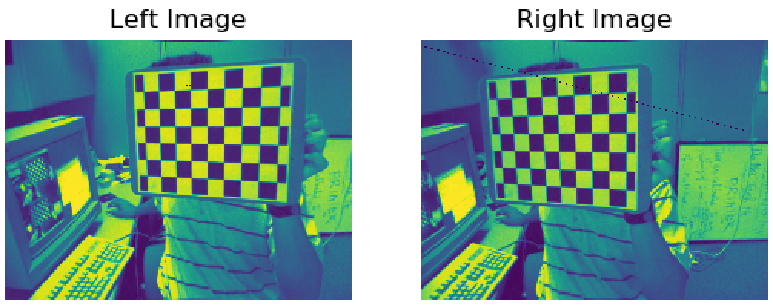


Figure 5: Wrong Epipolar Line

Then I use the opencv function stereoCalibrate() to get matrix F as cvF, and the epipolar line is right with cvF, but I don't konw why. So I review the knowledge and derivations of camera calibration including single camera calibration, to figure out if I make mistakes of some knowledge or not, and I read some papers [11] [21] [12] and websites [13] [14] [15] [16] about the distortion and fundamental matrix in order to find some missing knowledge, and nothing gotten. Lastly I compare my code with the original code of opencv function stereoCalibrate() [17] and I figure out the wrong form of matrix S given by tutorial. I should read the original code of opencv function stereoCalibrate() firstly!

2.5 Derive 3D Coordinate Based on Points p_l and p_r

Let p_l, p_r represent the point in pixel coordinates of left and right image, $P_l X_l, Y_l, Z_l]$, $P_r = [X_r, Y_r, Z_r]$ represent the point in left and right camera coordinates, According to equation (1), we have the following form of P_l and P_r

$$\begin{aligned} P_l &= \frac{1}{Z_l}(M_l{}^{-1} p_l) \\ P_r &= \frac{1}{Z_r}(M_r{}^{-1} p_r) \end{aligned} \tag{20}$$

And P_l and P_r represnt the lines related to p_l and p_r, then we solve the resulting equation(18) simultaneously, we get the relationship between Z_l and Z_r

$$\frac{Z_r}{Z_l} = (R_l{}^{-1} M_l{}^{-1} p_l - T_l)^T (R_r{}^{-1} M_r{}^{-1} p_r - T_r) \tag{21}$$

And According to equation(13) and equation(18), we can get another relationship between Z_r and Z_l

$$\frac{1}{Z_r}(M_r{}^{-1} p_r) = \frac{1}{Z_l}(R M_l{}^{-1} p_l) + T \tag{22}$$

Solve the resulting equation(19) and (20) simultaneously and we get the value of Z_l and Z_r, then we take the value of Z_l and Z_r into equation(18) to get the 3D coordinate.

2.6 Rectification

According to the derivations about stereo recitification before, we calculate the rotation matrices r_l, r_r and R_rect, r_l and r_r put the cameras into coplanar alignment, R_rect puts cameras into the row alignment.

The program can be found by link

https://github.com/zhangyongshun/Project_Stereo/blob/master/python_code/binocular_basics/opencvStereoRectify.py

In the program, I calculate the rotation matrices r_l, r_r, R_rect, then I rectify the cameras with the rotation matrices and draw parallel lines to check the outcomes. Besides, I use the OpenCV function stereoRectify() [27] to make comparison. The output figure is as follows.

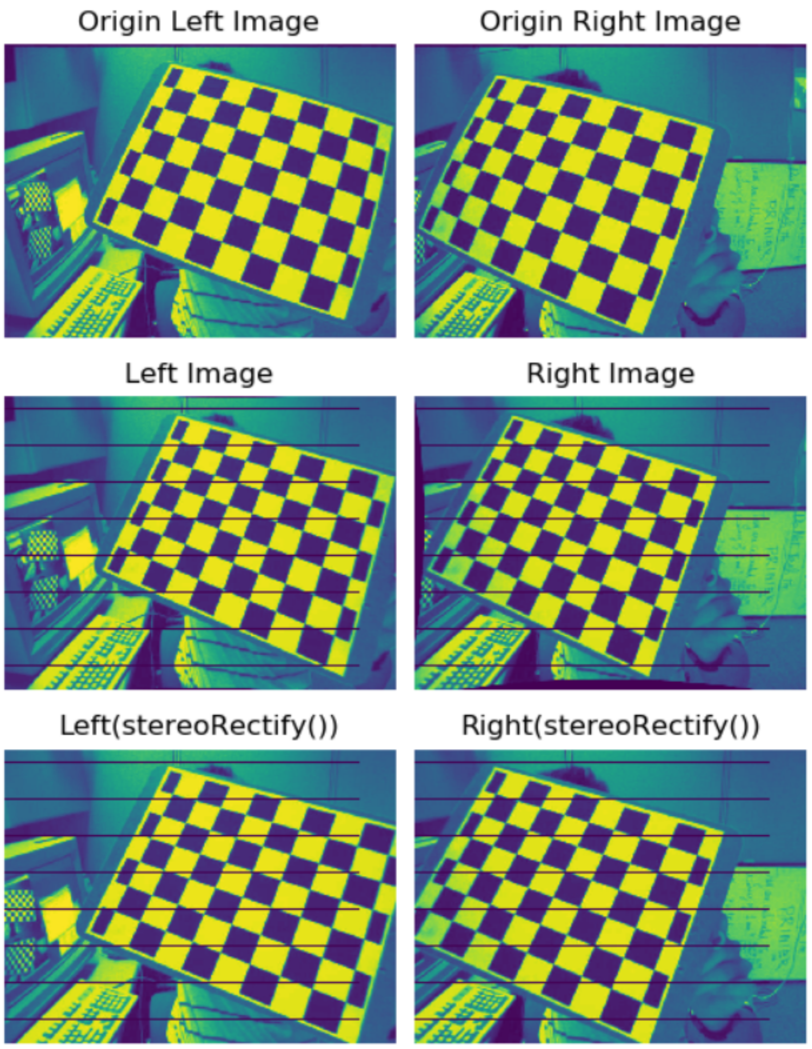


Figure 6: Image Rectification

2.7 Derivation of Translation b

It's easy to do this base on vector T$_r, whichreperesentstheoriginofrightcameracoordinatesinleftcameracoordinates, beforer$

Although the left and right camera coordinates have been rotated, the distance between the origins of both camera coordinates aren't changed. So we can derive the distance, and get vector b.

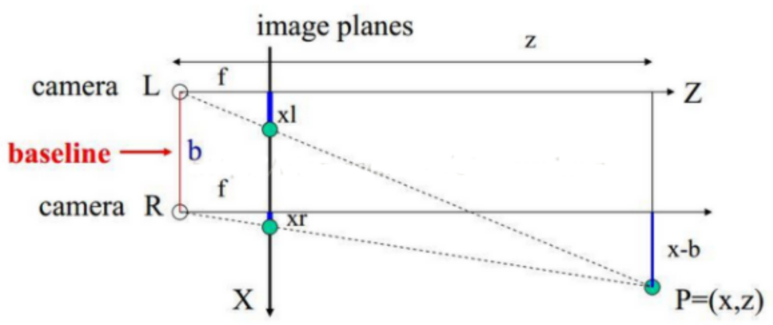
$$T_r = [T_x, T_y, T_z]$$

$$d = \sqrt{{T_x}^2 + {T_y}^2 + {T_z}^2} \tag{23}$$

$$b = [d, 0, 0]$$

2.8 Depth-Disparity

Given point P in the 3D world coordinates, P = $[X, Y, Z]^T$, then we have baseline b, and focal length f, and we can discrbe the relationship between 3D world coordinates and camera coordinates with the figure below.



From the figure, we have

$$\frac{Z}{f} = \frac{X}{x_l} = \frac{X-b}{x_r} \tag{24}$$

So we get the equation of X

$$X = \frac{x_l * b}{x_l - x_r} \tag{25}$$

Then we take X into equation(24), we get the equation of Z

$$Z = \frac{b * f}{x_l - x_r} = \frac{b * f}{d} \tag{26}$$

In the learning process, I get the information of OpenCV functions from OpenCV website [27], and learn some numpy functions from Numpy Reference [28]. I also get some knowledge of linear algebra [31] [30] and search for functions to draw on images on OpenCV website [29].

References

[1] **双目摄像机标定参数说明** CSDN http://blog.csdn.net/xuelabizp/article/details/50417914

[2] **Camera resectioning** Wikipedia https://en.wikipedia.org/wiki/Camera_resectioning

[3] **Camera Matrix** Wikipedia https://en.wikipedia.org/wiki/Camera_matrix

[4] **Epipolar geometry** Wikipedia https://en.wikipedia.org/wiki/Epipolar_geometry

[5] **计算机视觉基础 4 —— 对极几何 (Epipolar Geometry)** CNblogs http://www.cnblogs.com/gemstone/archive/2011/12/20/2294551.html

[6] **Stereo Vision** Robert Collins, CSE486 Lecture 8 http://www.cse.psu.edu/%7Ertc12/CSE486/lecture08.pdf

[7] **Epipolar Geometry and Stereo Vision** Derek Hoiem, CD 543/ECE 549 University of Illinois https://courses.engr.illinois.edu/cs543/sp2011/lectures/Lecture%2023%20-%20Epipolar%20Geometry%20and%20Stereo%2 %20Vision_Spring2011.pdf

[8] R. Hartley. "In defence of the 8-point algorithm," In Proceedings of the 5th International Conference on Computer Vision, pages 1064–1070, Boston, MA, June 1995. IEEE Computer Society Press.

[9] **Random sample consensus** Wikipedia https://en.wikipedia.org/wiki/Random_sample_consensus

[10] Michal Flularz, "FPGA implementation of the robust essential matrix estimation with RANSAC and the 8-point and the 5-point method,", Facing the Multicore-Challenge*II* Pages 60-71. Spring-Verlag Berlin, Heidelberg. 2012

[11] Z. Zhang, "A flexible new technique for camera calibration", IEEE Trans- actions on pattern analysis and machine intelligence, vol. 22, no. 11, pp. 1330-1334, 2000.

[12] J.P. Barreto, K. Daniilidis, "Fundamental Matrix for Cameras with Radial Distortion", Proc. Int'l Conf. Computer Vision, pp. 625-632, 2005.

[13] undistort image before estimating pose using solvePnP stackoverflow https://stackoverflow.com/questions/34550499/undistort-image-before-estimating-pose-using-solvepnp?rq=1

[14] **How to rectify points, stereo camera calibration** OpenCV Q&A http://answers.opencv.org/question/96250/how-to-rectify-points-stereo-camera-calibration/

[15] **Essential matrix** Wikipedia https://en.wikipedia.org/wiki/Essential_matrix

[16] **Distortion(optics)** Wikipedia https://en.wikipedia.org/wiki/Distortion(optics)

[17] **Original Code of stereoCalibrate** OpenCV https://github.com/opencv/opencv/blob/master/modules/calib3d/src/calib

[18] **Image Rectification (Stereo)** GuiDo Gerig, CS 6320 Spring 2012, http://www.sci.utah.edu/%7Egerig/CS6320S2012/Materials/CS6320CVF2012Rectification.pdf

[19] Gray Bradski, Adrian Kaehler, Learning OpenCV[M] Chapter 12:Projection and 3D Vision, O'Rerilly Media, 2008

[20] Andrea Fusiello, Emanuele Trucco, Alessandro Verri, "A compact algorithm for rectification of stereo pairs", Machine Vision and Applications, Spring-verlag, 2000.

[21] Z. Zhang, "Computing Rectifying Homographies for Stereo Vision", Microsoft Research Microsoft Corporation One Microsoft Way Redmond, WA 98052, 1999

[22] **刚体在三维空间的旋转 (关于旋转矩阵、DCM、旋转向量、四元数、欧拉角)** CSDN https://blog.csdn.net/mulinb/article/details/51227597

[23] **Original Code of stereoRectify** OpenCV https://github.com/opencv/opencv/blob/master/modules/calib3d/src/calibrat

[24] **Bouguet 极线校正的方法** CSDN https://blog.csdn.net/wangxiaokun671903/article/details/38039383

[25] **立体标定与立体校正** CSDN https://blog.csdn.net/onthewaysuccess/article/details/40736849 1330-1334, 2000.

[26] **Image rectification** Wikipedia https://en.m.wikipedia.org/wiki/Image_rectification

[27] **Geometric Image Transformations** Opencv3
https://docs.opencv.org/3.3.1/da/d54/group___imgproc___transform.html#ga55c716492470bfe86b0ee9bf3a1f0f7e

[28] **Numpy Functions Reference** Numpy https://docs.scipy.org/doc/numpy-dev/reference/index.html

[29] **Drawing Functions** Opencv https://docs.opencv.org/2.4/modules/core/doc/drawing_functions.html

[30] 郑宝东, 王忠英线性代数与空间解析几何 [M]. 第 4 版. 高等教育出版社, 2013 .04

[31] David C. Lay. 线性代数及其应用 [M]. 刘深泉, 洪毅等译. 第 3 版. 机械工业出版社, 2005 .08