报告

宋晋瑜

> 这份文档记录我在理解、学习、实现给定题目过程中的思考.

思路

这个部分讲我做这些问题时的思路。

绝大部分问题我都是通过查阅资料并学习的。比如最开始的相机基础，我对这块完全不熟悉，因此我先看wiki，后续查阅了一些中文的资料辅助我理解。

编程题我一般是结合文档和博客来搭配着学习完成。

如果有指定参考资料的问题我会先去查看参考资料，如果看不太懂就去搜索不懂内容的中文资料，帮助理解。

整个的学习过程大概是如上所述。

题目部分

1 What are the instrinsics and extrinsics of a camera? What is the camera matrix?

- 内参矩阵 `I` 是将相机坐标系的三维坐标转换为平面坐标的矩阵，形如$\begin{bmatrix} \alpha_x & \gamma & u_0 & 0 \\ 0 & \alpha_y & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$

  其中 $\alpha_x = f \cdot m_x$ $\alpha_y = f \cdot m_y$，f指焦距，m_x m_y指比例系数，这个是用来将距离与像素进行转换，具体就是指x, y方向单位距离下多少个像素。

  $u0, v0$指的是主点的位置，默认是坐标原点，但可能会有偏移，因此需要加上这个偏置。

  γ指的是坐标轴倾斜参数，通常为0。

> “
>
> γ什么场景下用到?

- 外参矩阵 `E` 是将世界坐标系和相机坐标系进行转换的矩阵，外参矩阵表示为$\begin{bmatrix} R_{3\times 3} & T_{3\times 1} \\ 0_{1\times 3} & 1 \end{bmatrix}_{4\times 4}$，其中 `R` 为控制两个坐标旋转的矩阵， `T` 为控制两个坐标平移的矩阵。
- 相机矩阵 `M` 是内参矩阵 `I` 与外参矩阵 `E` 的乘积，即 `M=IE` ，总的效果就是实现了三维世界坐标和二维平面坐标的转换。

2 Camera Imaging

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x m_x & 0 & c_x m_x \\ 0 & f_y m_y & c_y m_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} R & t \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ 1 \end{bmatrix}$$

m_x, m_y是指单位距离有多少像素，计算得到u和v的值即对应该点在平面上的位置

3 Given a 2D image point (u,v), what shape does it correspond to in the 3D camera coordinate? Can you derive its equation?

我理解的是，相机矩阵的过程是 `世界坐标 -> 相机坐标 -> 像素坐标` ，那么反过来进行逆运算即可计算出三维坐标。设内参矩阵为K，旋转矩阵为R, 平移矩阵为T。

1. 转换到相机坐标

$$[X_c \quad Y_c \quad Z_c]^T = K^{-1}[u \quad v \quad 1]^T$$

2. 转换到世界坐标

$$[X_w \quad Y_w \quad Z_w]^T = R^{-1}[X_c \quad Y_c \quad Z_c]^T - R^{-1}T$$

这样就得到了三维坐标。

4 (Distortion) the distortions of the cameras. Given a 2D image point (u,v), can you find its coordinate before distortion?

1. 畸变是指相机透镜透视失真，可分为径向畸变和切向畸变

2. 可以通过标定获得内参、外参以及畸变参数，然后通过计算得到其无畸变的坐标。

具体的计算公式就是：

$$X_{corrected} = X(1 + k_1r^2 + k_2r^4) + [2p_1XY + p_2(r^2 + 2x^2)]$$

$$Y_{corrected} = Y(1 + k_1r^2 + k_2r^4) + [p_1(r^2 + 2Y^2) + 2p_2XY]$$

其中$k1, k2$是径向畸变系数，而$p1, p2$是切向畸变系数。

> 这里 `r` 指什么？好像哪里都没有讲
>
> 答：r^2应该是指$(x + y)^2$

5 (Calibration) Describe what the camera calibration does.

相机标定就是建立世界坐标与像素坐标对应关系的过程，实质是求解相机的内参，外参以及畸变参数，从而利用标定结果纠正畸变，实现更好的呈像。

6 Provided a series of images taken by the camera, can you calibrate the camera with OpenCV functions?

Done。

首先找到内角点，之后用亚像素检测的方式得到更精确的角点坐标，再用calibrateCamera函数得到内参和外参以及畸变系数。用畸变系数和内参数结合undistort函数来实现畸变的修复。

7 Undistort the images with the calibration results you computed. What functions of OpenCV do you use?

```
dst = undistort(img, cameraMatrix, distortCoeffs)
# OR
initUndistortMap和remap组合
```

> **8 Learn about Zhang’s method [1] for camera cal- ibration. Can you implement Zhang’s method? Report the calibration result with your implemented approach. Compare with the result from Problem 6.**

看了论文只能知道个大概但是写不出来，就先说下思路。总体思路就是带3张以上图像根据$Vb = 0$求出b的全部参数，然后就可以求出A的全部参数，进而通过A求出旋转向量和平移向量，然后就可以得到。我想问的问题是，那个单应性矩阵H怎么求得到的呢...

> **Till now you have been skilled with the tricks of a single camera. Can you use a single camera to estimate the depth of the pixels (the distance of their corresponding 3D points to the image plane)? If not, how will you manage to do that? Explain your reasons. (Hint: Recall Problem 3.)**

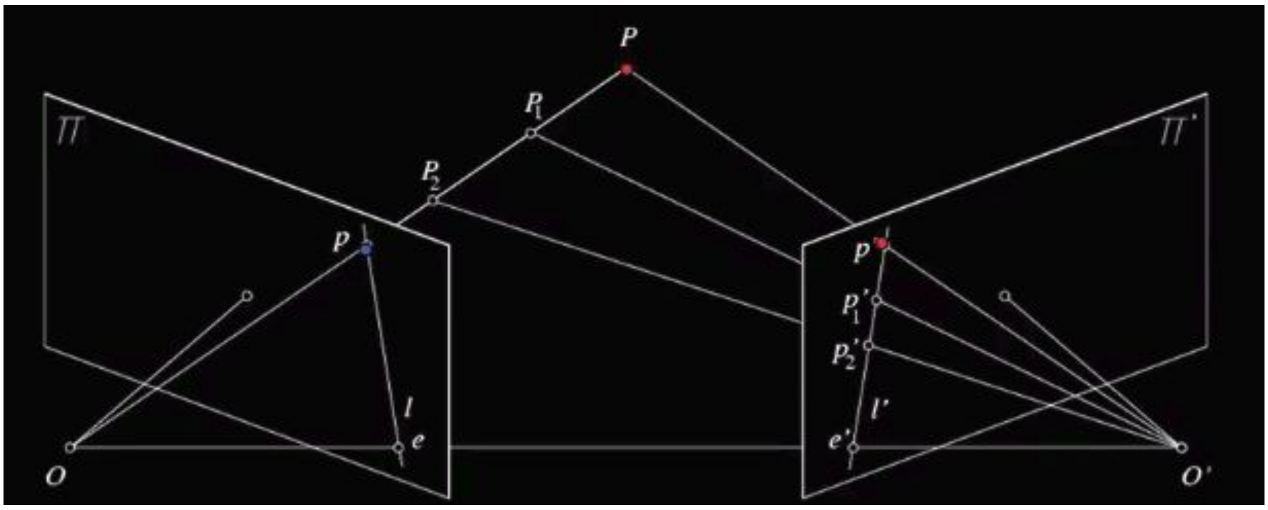
感觉不行，因为沿着光心和像点的连线上有无数个点都满足。

> **9 (Projection) Given a point with 3D world coordinate $X = (X_1, X_2, X_3)$. Write down its projection in left and right camera planes.**

对于左侧的相机，只需要乘以内参矩阵即可得到左侧投影。对于右侧相机，首先要转换左侧坐标系到右侧，再乘以内参矩阵即可得到它在右侧的投影。

> **10 (Epipolar Line) Given a pixel in the left image as $x_l = (u, v)$, its cor- responding 3D point will also project onto the right image plane. Without knowing its precise 3D coordinate, the probable projection point on the right image will have to lie on a line (This is called the epipolar con- straint). This line is called the epipolar line of (u, v). Can you derive the line’s equation? (in the right image’s pixel coordinate) Draw a figure to help you derive.**

我的理解是这样的，如下图。



现在知道左侧的相机原点，也知道$(R|t)$，等于知道右侧的相机原点，那么可以通过M_l得出左侧的相机坐标系下的三维坐标，这样，等于有三个点是清楚的，因此这个对极面是确定的，那对极面与相机呈像平面的交线就是对极线，设对极面方程为$F(x,y,z)$,右侧相机呈像平面方程为$G(x,y,z)$，则$L_{epipolar}$表达式为

$$\begin{cases} F(x,y,z)=0, \\ G(x,y,z)=0 \end{cases}$$

11 (Fundamental Matrix) Generally, given a fixed two camera system. There is a matrix F that for any 3D point, its projections in two images xl and xr will satifsy xTl F xr = 0. F is called the fundamental matrix. Can you derive the matrix?

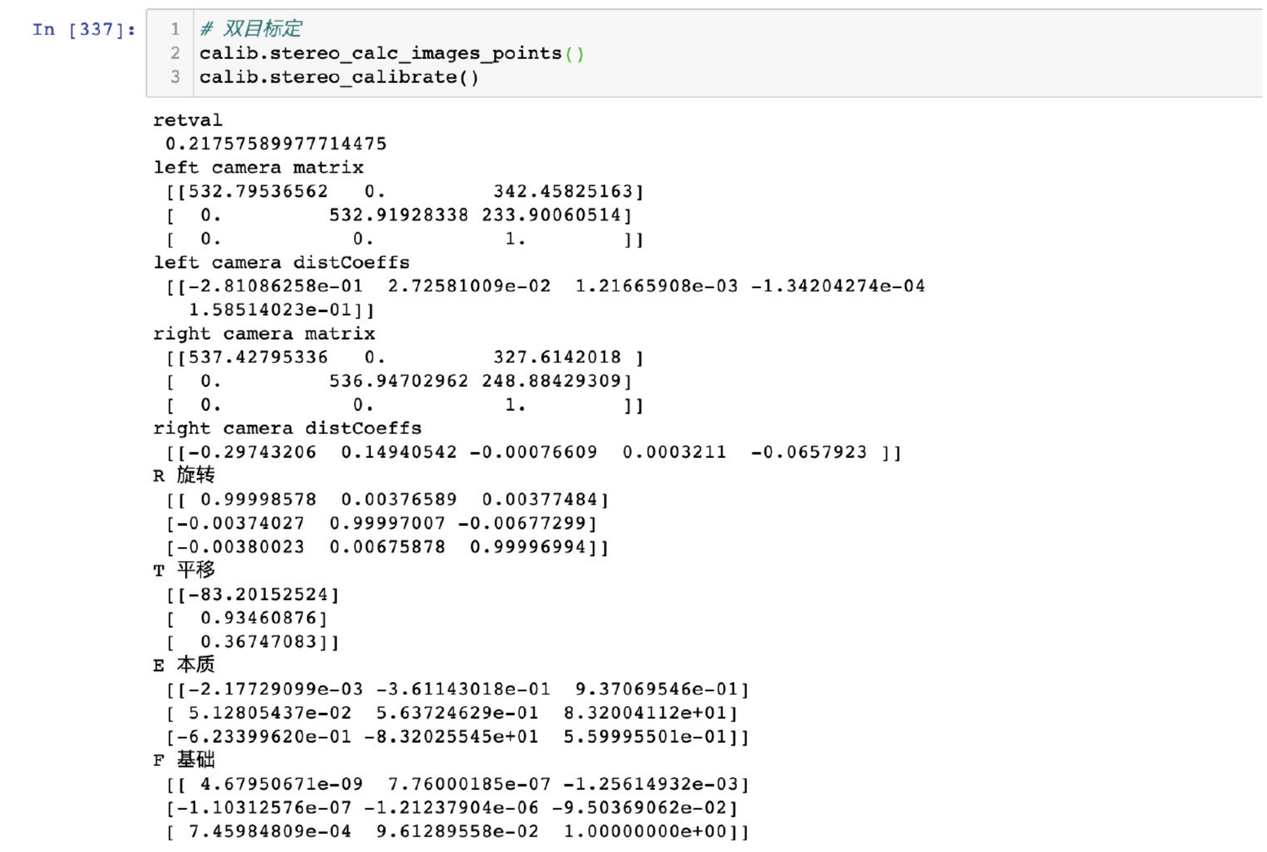
$F=(K_{intrinsic,left}^{-1})^T E K_{intrinsic,right}^{-1}$，其中$E$为本质(征)矩阵，K为内参矩阵

推导过程学习自参考[5].

12 (Stereo Calibration) We've already learnt how to calibrate a sin- gle camera. For a binocular camera system, calibration does more than calibrate each of them. The transform (R|t) between the left and the right cameras also needs calibration. To know more about stereo calibration, you may refer to http://blog.csdn.net/xuelabizp/article/details/50417914. Now please use OpenCV to calibrate the binocular cameras with images in left.zip and right.zip respectively. Report the results. (Make sure you understand the geometrical meanings of the parameters)

Done。

过程是先进行单目标定，再使用stereoCalibrate函数进行双目标定，得到的结果如下图，包含左侧相机的相机矩阵，畸变系数，右侧相机矩阵，相机系数，左右相机转换的旋转矩阵和平移矩阵，以及本质矩阵和基础矩阵。



```
In [337]:
# 双目标定
calib.stereo_calc_images_points()
calib.stereo_calibrate()
```

```
retval
 0.21757589977714475
left camera matrix
 [[532.79536562   0.         342.45825163]
 [  0.         532.91928338 233.90060514]
 [  0.           0.           1.        ]]
left camera distCoeffs
 [[-2.81086258e-01  2.72581009e-02  1.21665908e-03 -1.34204274e-04
   1.58514023e-01]]
right camera matrix
 [[537.42795336   0.         327.6142018 ]
 [  0.         536.94702962 248.88429309]
 [  0.           0.           1.        ]]
right camera distCoeffs
 [[-0.29743206  0.14940542 -0.00076609  0.0003211  -0.0657923 ]]
R 旋转
 [[ 0.99998578  0.00376589  0.00377484]
 [-0.00374027  0.99997007 -0.00677299]
 [-0.00380023  0.00675878  0.99996994]]
T 平移
 [[-83.20152524]
 [  0.93460876]
 [  0.36747083]]
E 本质
 [[-2.17729099e-03 -3.61143018e-01  9.37069546e-01]
 [ 5.12805437e-02  5.63724629e-01  8.32004112e+01]
 [-6.23399620e-01 -8.32025545e+01  5.59995501e-01]]
F 基础
 [[ 4.67950671e-09  7.76000185e-07 -1.25614932e-03]
 [-1.10312576e-07 -1.21237904e-06 -9.50369062e-02]
 [ 7.45984809e-04  9.61289558e-02  1.00000000e+00]]
```

> **13 Given left-image pixel p_l, write down its epipolar line. Pick an arbitrary point on the epipolar line as the projection of the 3D point. Can you derive the 3D coordinate of the point?**

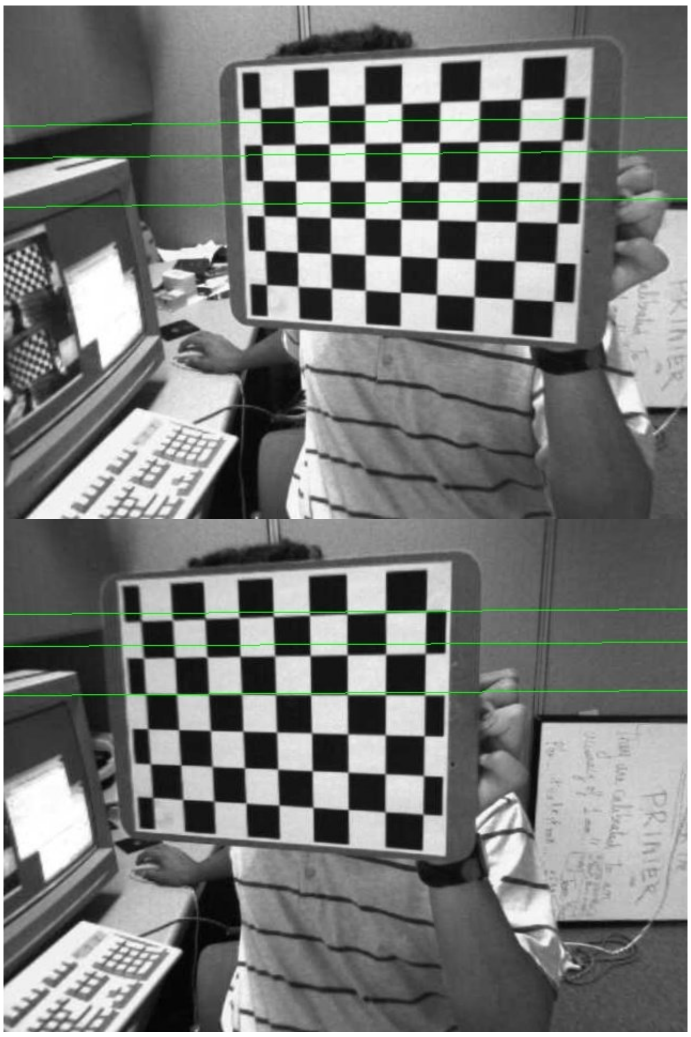
设基本矩阵为F，则左侧点p_l在右侧呈现平面上的对极线表达式为$l = F^T p_l$

任选一点p_r，左侧相机光心O_l，右侧相机光心O_r，因此可以得到$O_l p_l$和$O_r p_r$两个空间曲线表达式，设为$F(x, y, z)$和$G(x, y, z)$联立即可得到原点的三维坐标。

> **14 (Rectification) The pose of the two cameras are usually arbitrary. It is usually necessary to make the two cameras parallel so that the epipolar line will become horizontal. This is achieved by adding a Rotation transformation to one of the cameras. (See http://blog.csdn.net/gdut2015go/article/details/48391949. You may also refer to the tutorial 3 for more details). Now use OpenCV to rectify the left and the right images with the calibration results obtained from Problem 13. Display the images and check with your eyes to see if they are really rectified. (A pair of the images is enough. Remember to undistort the images)**

Done，见Jupyter Notebook。

图像在 `rectify` 如下目录，这里放一个对第一个图片处理之后左边和右边的效果，图像上的线是对极线，可以看到水平了。



> 15 Once the images are rectified, the epipolar lines will become parallel to the image axes. For such a binocular camera system, the transformation between them will be simplified to $(I|t)$. The rotation matrix will become unit matrix since their axes are parallel. The translation matrix t will become $(b, 0, 0)$ with the coordinates defined in the OpenCV document Can you derive the baseline b from the results in Problem 14?

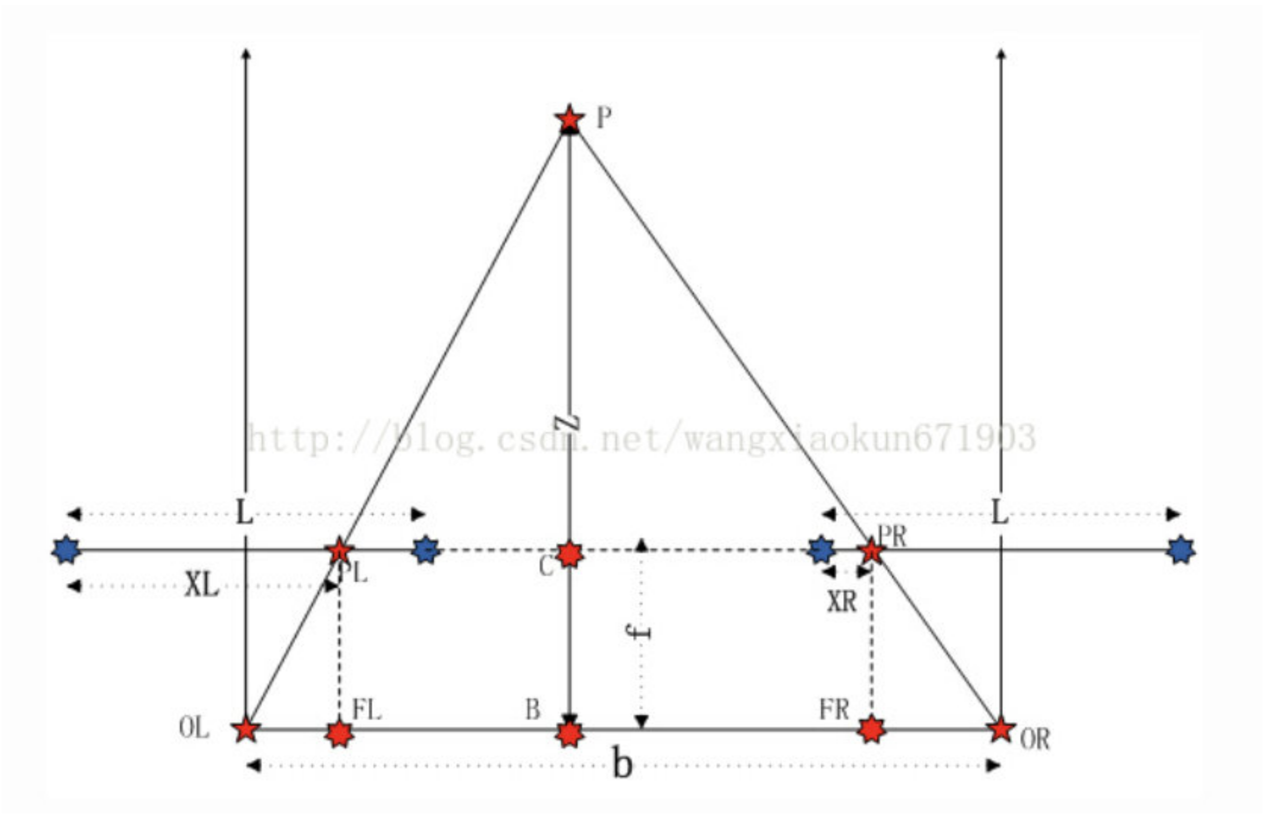
基线的话方向向量应该就是$(b, 0, 0)$吧，以左侧相机坐标系为世界坐标系，表达式就是

$$\begin{cases} x = 0, \\ z = 0 \end{cases}$$

即y轴。

如果这个问题是问基线距的话我理解那应该是提前测量好的吧。

16 (Depth-Disparity) For a calibrated binocular camera system, the epipolar lines can be made parallel with rectification. For a pixel on the left image with coordinate (x_l, y), its matching point (the projection of the 3D point on the right image) must have coordinate of the form $(x_l - d, y)$. $d = x_l - x_r$ is called the pixel's disparity. Can you derive the 3D point's coordinate given baseline b and the camera matrices? If the camera has indentical vertical and horizontal focal lengths, can the depth z (Z coordinate value) of a pixel be written as $z = bf/d$? Write down your derivation.



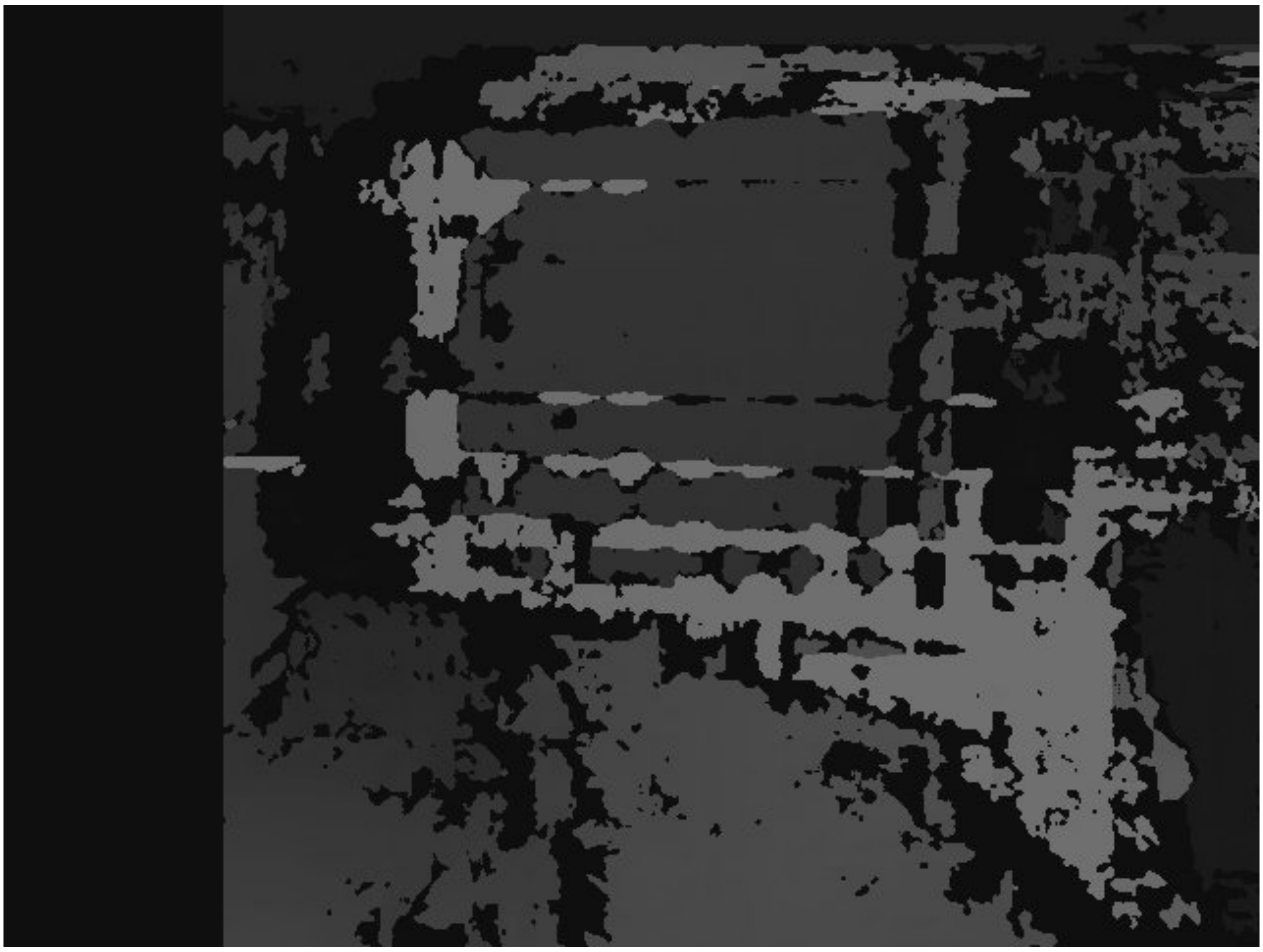
$(b - ((X_l - L/2) + (L/2 - X_r)))/b = (z - f)/z$

推导后即可得到 $z = bf/d, d = X_l - X_r$

17 (SGBM) Can you use OpenCV to compute the disparity maps for the images you used for stereo calibration? Visualize several disparity results and check with your eyes to see if the results are reasonable.

Done.见Jupter Notebook。

大部分还是准确的，不过有些区域好像看起来有点问题，以下图片是针对left01做的，感觉好像有些问题。



18 (Unsupervised) Do you think that supervised deep learning approaches can be easily used for stereo matching? Why or why not? Recently unsupervised deep learning based approaches have been proposed. Can you find out these approaches from the evaluation page?

我认为有监督的方法可能不是很好，我个人理解有监督的学习大多还是标记过的，那么标记意味着要对各种场景都建模，但是现实中比如自动驾驶场景，对驾驶场景全部建模和标记我认为是非常困难的。并且数据集也比较少，难以训练。总的来说缺点就是

- 数据少，训练困难
- 泛化差，无法对全部场景进行建模

看到的一些无监督的方法有：

- SUW_Stereo: SUW-Learn: Joint Supervised, Unsupervised, Weakly Supervised Deep Learning for Monocular Depth Estimation
- OASM-Net: OASM-Net
- Unsupervised Learning of Stereo Matching ***Chao Zhou, Hong Zhang, Xiaoyong Shen, Jiaya Jia***; Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2017, pp. 1567-1575

19 Find out the most efficient deep learning based approach and compare it with SGM. In real time scenarios like autonomous driving, which approach will you prefer? Explain your reason.

LEAStereo的表现最好，当然是选这个方法，首先运行速度快，其次错误率非常低，自动驾驶场景需要非常高速的识别速度以及非常高的准确率，如果不能保证以60fps以上的速度去运行其实都还是不可商用的。我看到SGM算法速度普遍都比较慢，是无法应付自动驾驶场景的。

else

剩余题目受限于现在的能力无法解答 T^T ...

参考

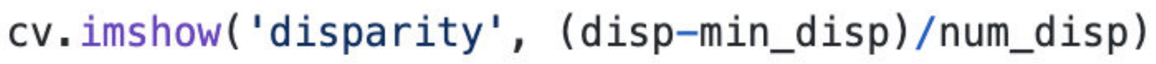
Camera Calibration and 3D Reconstruction	https://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html	**07-20**
标题	地址、文献引用	时间
相机矩阵(Camera Matrix)	https://blog.csdn.net/gwplovekimi/article/details/90172544	0718-15
相机矩阵	https://zhuanlan.zhihu.com/p/33406665?utm_source=qq&utm_medium=social&utm_oi=683679535938015232	07-19
视觉检测——单目相机的标定	https://blog.csdn.net/Kano365/article/details/90721424	07-20

张正友标定算法理论及算法实现	https://blog.csdn.net/u010418035/article/details/47617909	07-21
对极几何与基本矩阵	https://zhuanlan.zhihu.com/p/33458436	07-21
opencv/opencv	https://github.com/opencv/opencv/	07-21
张正友标定法翻译	https://blog.csdn.net/heroacool/article/details/50286677?utm_medium=distribute.pc_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-1.nonecase&depth_1-utm_source=distribute.pc_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-1.nonecase	07-21
How to calculate an epipolar line with a stereo pair of images in Python OpenCV	https://stackoverflow.com/questions/51089781/how-to-calculate-an-epipolar-line-with-a-stereo-pair-of-images-in-python-opencv	07-21
张正友相机标定Opencv实现以及标定流程	https://blog.csdn.net/dcrmg/article/details/52939318	07-21
[机器视觉]张氏相机标定论文解析	https://xgyopen.github.io/2019/04/06/2019-04-06-imv-calibrate-camera-paper/	07-21
StereoRectify()函数定义及用法畸变矫正与立体校正	https://blog.csdn.net/qq_36537774/article/details/85005552	07-21
OpenCV双目标定的流程	https://blog.csdn.net/u011722133/article/details/79422942	07-21
基本矩阵、本质矩阵和单应矩阵	https://blog.csdn.net/kokerf/article/details/72191054	07-23
第七节、双目视觉之空间坐标计算	https://www.cnblogs.com/zyly/p/9373991.html	07-23
[机器视觉]张氏相机标定论文解析	https://xgyopen.github.io/2019/04/06/2019-04-06-imv-calibrate-camera-paper/	07-23
第六节、双目视	https://www.cnblogs.com/zyly/p/9366080.html	07-21

觉之相机标定		
vs2015 + opencv3 双目摄像头标定（C++实现）	https://blog.csdn.net/u013289254/article/details/99200881	07-21
张正友标定法	https://blog.csdn.net/tiantangzixue/article/details/79878996	07-21
对极几何 Epipolar Geometry	https://zhuanlan.zhihu.com/p/79845576	07-21
How to calculate an epipolar line with a stereo pair of images in Python OpenCV	(https://stackoverflow.com/questions/51089781/how-to-calculate-an-epipolar-line-with-a-stereo-pair-of-images-in-python-opencv)	07-21

问题

1. 内参γ什么场景下用到?
2. 单应是什么？这块没看明白
3. 单应矩阵怎么求呢?
4. SGBM怎么调参数呢？感觉生成的视差图不够准确而且有一些小问题。
5. 输出视差图是直接输出么？看到opencv/samples里面这样一行代码但是输出的结果就特别小，图片非常黑



```
cv.imshow('disparity', (disp-min_disp)/num_disp)
```

6. 后续有问题还会及时邮件问您