**双目视觉立体匹配项目设计文档**

## 一、设计思路与流程简介

本项目使用双目相机或者手机平行移动拍摄同一场景的左、右两幅视点图像，接着使用摄像机标定后的参数进行双目图像的畸变校正和立体校正并得到矫正好的图像对，然后使用SGBM算法获取视差图并进行空洞填充进而获取深度图，最后合成虚拟视点图，具体过程如下：

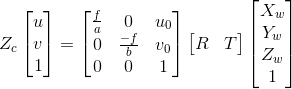
* 摄像机标定（包括内参和外参）
* 双目图像的校正（包括畸变校正和立体校正）
* 立体匹配算法获取视差图并进行空洞填充
* 利用空洞填充后的视差图获取深度图
* 利用深度图进行虚拟视点的合成

## 项目具体流程

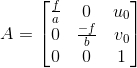
### 2.1摄像头标定

**2.1.1摄像头标定参数简述**

摄像机内参反映的是摄像机坐标系到图像坐标系之间的投影关系，其中世界坐标系和计算机坐标系的直接代数关系如下：



记



IMG_259

A矩阵式刻画摄像机的内部参数，包括焦距f、成像中心的位置、及成像单元尺寸，因此称为内参矩阵。M描述的摄像头的运动关系，所以称为外参矩阵。其中R有三个相关参数，分别是绕x，y，z轴的旋转角度IMG_260。T也有三个相关参数，分别是在三个坐标轴上的平移量Tx、Ty、Tz。

摄像头的畸变是由于成像模型的不精确造成的。人们为了提高光通量用透镜代替小孔来成像，由于这种代替不能完全符合小孔成像的性质，因此畸变就产生了。现在大量使用的透镜为球面镜，原因是其廉价易得。畸变可以分为两大类，径向畸变和切向畸变。详细的畸变介绍可以参考工程光学的相关课程，下面简单介绍相关畸变及其修正。

径向畸变的效应有两种，一种是枕形效应，另一种是桶形效应。径向畸变可用下面公式修正：

IMG_257

切向畸变是由于透镜与成像平面不严格的平行，其可以用如下公式修正：

IMG_258

这样又引入了五个畸变参数：

IMG_259

**2.1.2内参标定**

摄像机内参反映的是摄像机坐标系到图像坐标系之间的投影关系。摄像机的内参包括fx、fy、cx、cy以及畸变系数[k1,k2,p1,p2,k3]，其标定采用张正友标定法，简单易操作，以下为参数含义。

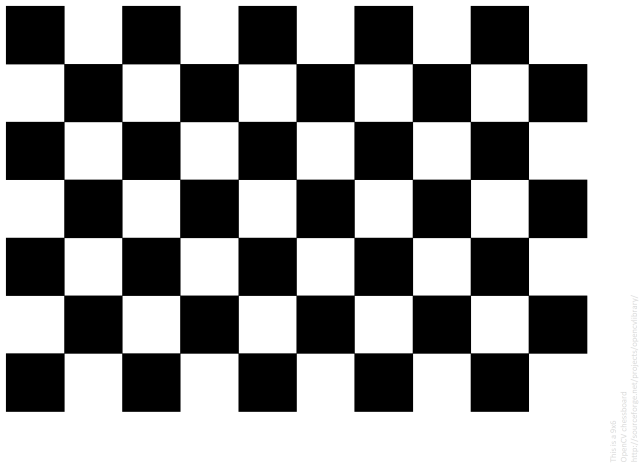
fx、fy：摄像头焦距与像素单元尺寸的比值；

cx、cy：摄像机坐标系原点及摄像头光心在计算机坐标系的位置；

k1、k2、k3：摄像头的切向畸变系数；

p1、p2：摄像头的径向畸变系数。

进行摄像头内参标定时，我们需要使用比较容易提取角点的图像如下图。这里我们采用棋盘格图像，使用双目相机或者手机对着电脑拍摄各个角度的棋盘格图像共9张（8张左右为宜）。棋盘格图像如图所示：



使用OpenCV3.4.5+VS2017对双目相机或者手机进行内参标定。标定结果如下，手机镜头不是鱼眼镜头，因此使用普通相机模型标定即可。

上面标定结果顺序依次为fx, fy, cx, cy, k1, k2, p1, p2, k3，保存到文件中供后续使用。

**2.1.3外参标定**

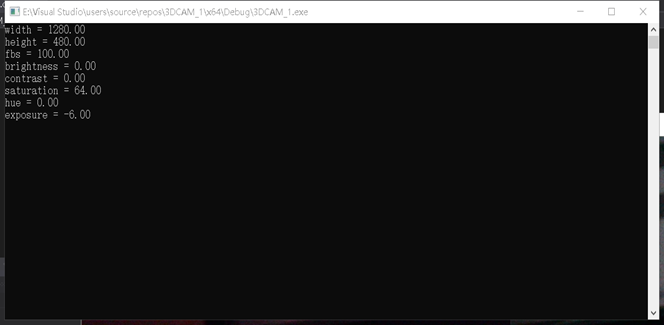
摄像机外参反映的是摄像机坐标系和世界坐标系之间的旋转R和平移T关系。如果两个相机的内参均已知，并且知道各自与世界坐标系之间的R1、T1和R2，T2，就可以算出这两个相机之间的Rotation和Translation，也就找到了从一个相机坐标系到另一个相机坐标系之间的位置转换关系。

摄像机外参标定也可以使用标定板，只是保证左、右两个相机同时拍摄同一个标定板的图像。外参一旦标定好，两个相机的结构就要保持固定，否则外参就会发生变化，需要重新进行外参标定。

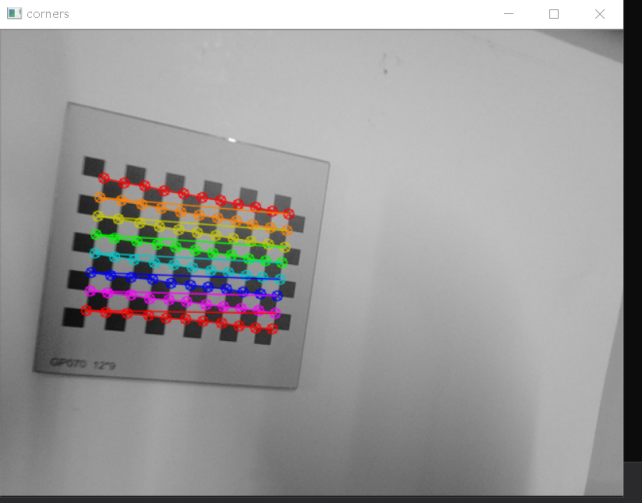
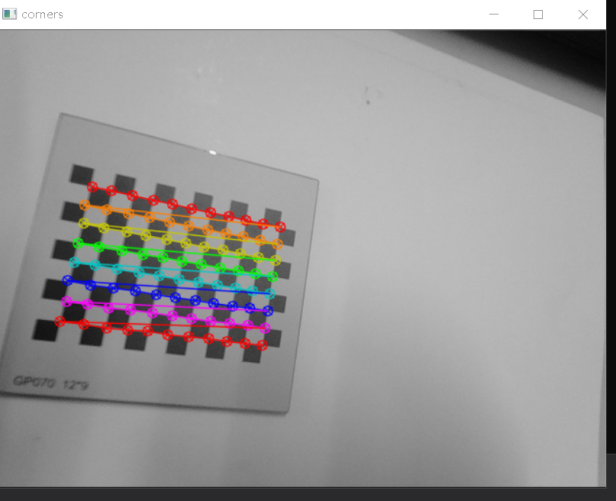
**2.1.4标定的具体过程：**

本次实验采用内参和外参同时标定的方法。

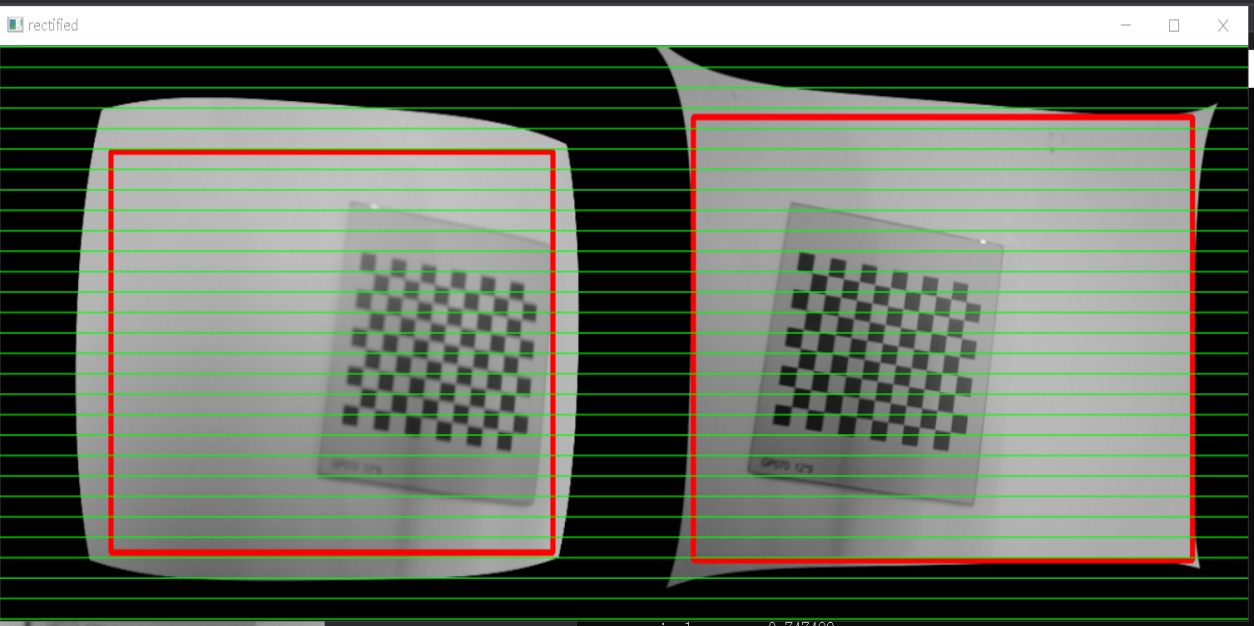
首先获取双目相机的一些基本参数：



其中的参数分别为：获取图像的像素数，帧数，亮度，饱和度，曝光度。之后获取一定量的带有标定板的图片，并进行标定。

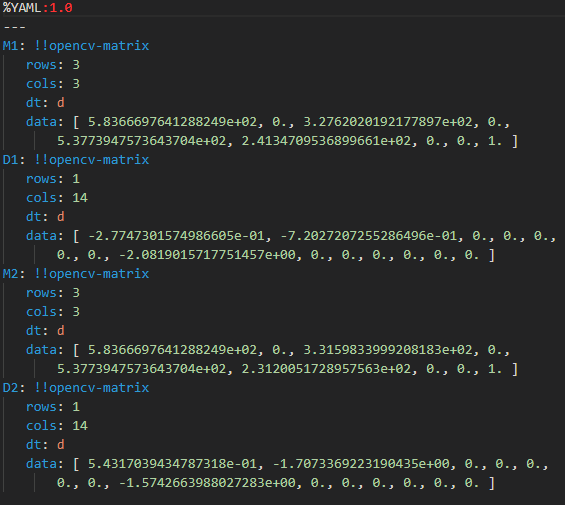


标定结果如下：



其中，红色的框表示有效的摄像区域，绿线标定同一位置的点。得到包含内参和外参的yml文件。

内参yml文件：



其中：

M1,M2分别为左右内参矩阵 ；D1,D2分别为畸变向量；

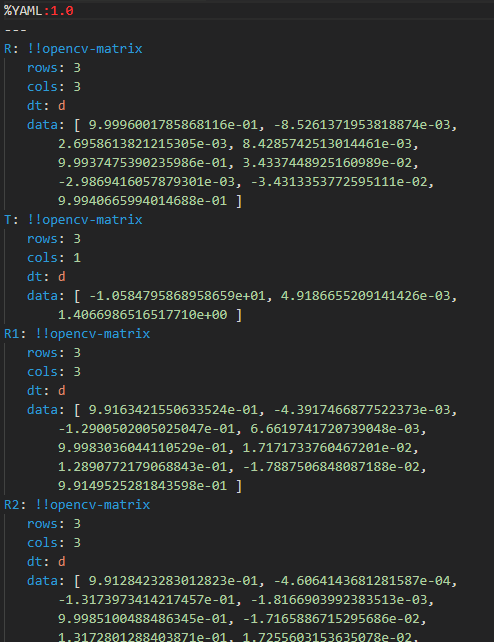
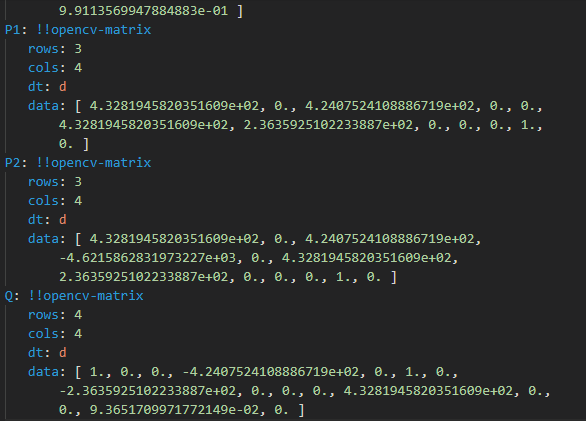
摄像机内参矩阵：{Fx, s, x0,; 0, Fy, y0:0, 0, 1},分别代表的内容如前面所示。

D矩阵代表的为畸变参数，其形式为：

D= { K1,K2,P1, P2{K3{K4,K5,K6}, { S1,S2,S3,S4} } }

k1,k2,k3,k4,k5,k6为径向畸变，p1,p2为切向畸变。

外参yml文件：

其中的参数为：

R——右相机相对左相机的旋转矩阵

T——右相机相对左相机的平移矩阵

R1,R2——左右相机校准变换（旋转）矩阵

P1,P2——左右相机在校准后坐标系中的投影矩阵

Q——视差-深度映射矩阵

### 2.2双目图像的校正

**2.2.1畸变校正**

畸变矫正函数 undistort()；undistort() 是独立的一个畸变矫正函数，一次性可以完成映射矩阵的求解和重新映射。

**2.2.2立体校正**

①得到两个摄像机之间的 Rotation和Translation之后，对两幅图像进行立体对极线校正，这就需要算出两个相机做对极线校正需要的R和T，用R1,T1, R2, T2表示，以及透视投影矩阵P1，P2。

cv::stereoRectify(camK, D, camK, D, imgL.size(), R, -R\*t, R1, R2, P1, P2, Q);

stereoRectify() 的作用是为每个摄像头计算立体校正的映射矩阵。所以其运行结果并不是直接将图片进行立体矫正，而是得出进行立体矫正所需要的映射矩阵。

②得到上述参数后，使用下面的程序进行对极线校正操作，并将校正结果保存到本地。

#校正左图并输出保存

cv::initUndistortRectifyMap(P1(cv::Rect(0, 0, 3, 3)), D, R1, P1(cv::Rect(0, 0, 3, 3)), imgL.size(), CV\_32FC1, mapx, mapy);

cv::remap(imgL, recImgL, mapx, mapy, CV\_INTER\_LINEAR);

cv::imwrite("data/recConyL.png", recImgL);

#校正右图并输出保存

cv::initUndistortRectifyMap(P2(cv::Rect(0, 0, 3, 3)), D, R2, P2(cv::Rect(0, 0, 3, 3)), imgL.size(), CV\_32FC1, mapx, mapy);

cv::remap(imgR, recImgR, mapx, mapy, CV\_INTER\_LINEAR);

cv::imwrite("data/recConyR.png", recImgR);

 ③对极线校正结果如下所示，查看对极线校正结果是否准确，可以通过观察若干对应点是否在同一行上。

**2.2.3用到的其它的函数**

映射变换计算函数 initUndistortRectifyMap()，该函数功能是计算畸变矫正和立体校正的映射变换。获取原图如下：

经过校正后的图如下：

### 2.3立体匹配获取视差图并进行填充

**2.3.1 SGBM算法获取视差图**

得到立体校正后的左右两幅图像后，匹配点是在同一行上的，可以使用OpenCV中的BM算法或者SGBM算法计算视差图。由于SGBM算法的表现要远远优于BM算法，因此采用SGBM算法获取视差图。SGBM中的参数如下：

int numberOfDisparities = ((imgSize.width / 8) + 15) & -16;

cv::Ptr<cv::StereoSGBM> sgbm = cv::StereoSGBM::create(0, 16, 3);

sgbm->setPreFilterCap(32);

int SADWindowSize = 9;

int sgbmWinSize = SADWindowSize > 0 ? SADWindowSize : 3;

sgbm->setBlockSize(sgbmWinSize);

int cn = imgL.channels();

sgbm->setP1(8 \* cn\*sgbmWinSize\*sgbmWinSize);

sgbm->setP2(32 \* cn\*sgbmWinSize\*sgbmWinSize);

sgbm->setMinDisparity(0);

sgbm->setNumDisparities(numberOfDisparities);

sgbm->setUniquenessRatio(10);

sgbm->setSpeckleWindowSize(100);

sgbm->setSpeckleRange(32);

sgbm->setDisp12MaxDiff(1);

int alg = STEREO\_SGBM;

if (alg == STEREO\_HH)

sgbm->setMode(cv::StereoSGBM::MODE\_HH);

else if (alg == STEREO\_SGBM)

sgbm->setMode(cv::StereoSGBM::MODE\_SGBM);

else if (alg == STEREO\_3WAY)

sgbm->setMode(cv::StereoSGBM::MODE\_SGBM\_3WAY);

sgbm->compute(imgL, imgR, disp);

默认计算出的是左视差图，如果需要计算右视差图，则将上面加粗的三条语句替换为下面前三条语句。由于视差值计算出来为负值，disp类型为16SC1，因此需要取绝对值，然后保存：

sgbm->setMinDisparity(-numberOfDisparities);

sgbm->setNumDisparities(numberOfDisparities);

sgbm->compute(imgR, imgL, disp);

disp = abs(disp);

SGBM算法得到的左、右视差图如下，左视差图的数据类型为CV\_16UC1，右视差图的数据类型为CV\_16SC1（SGBM中视差图中不可靠的视差值设置为最小视差（mindisp-1）\*16。因此在此例中，左视差图中不可靠视差值设置为-16，截断值为0；右视差图中不可靠视差值设置为（-numberOfDisparities-1）\*16，取绝对值后为（numberOfDisparities+1）\*16，所以两幅图会有较大差别）：



**2.3.2空洞填充**

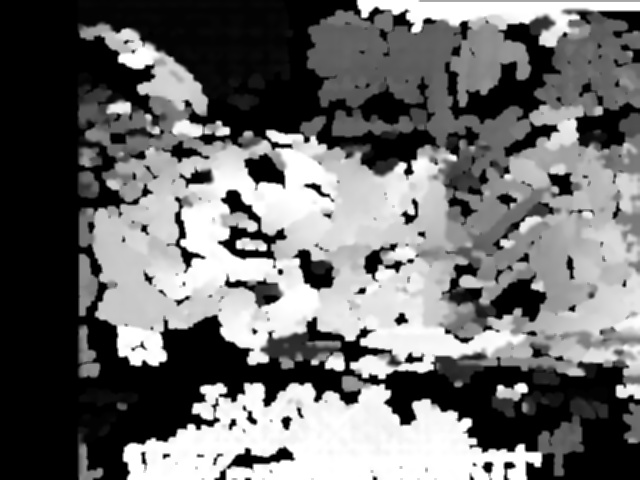
视差图中视差值不可靠的视差大多数是由于遮挡引起，或者光照不均匀引起，所以本项目会进行空洞填充来还原更为真实的视差图。空洞填充也有很多方法，在这里先检测出空洞区域，然后用附近可靠视差值的均值进行填充，具体步骤如下：

①以视差图dispImg为例。计算图像的积分图integral，并保存对应积分图中每个积分值处所有累加的像素点个数n（空洞处的像素点不计入n中，因为空洞处像素值为0，对积分值没有任何作用，反而会平滑图像）。

②采用多层次均值滤波。首先以一个较大的初始窗口去做均值滤波（积分图实现均值滤波就不多做介绍了，可以参考我之前的一篇博客），将大区域的空洞赋值。然后下次滤波时，将窗口尺寸缩小为原来的一半，利用原来的积分图再次滤波，给较小的空洞赋值（覆盖原来的值）；依次类推，直至窗口大小变为3x3，此时停止滤波，得到最终结果。

③多层次滤波考虑的是对于初始较大的空洞区域，需要参考更多的邻域值，如果采用较小的滤波窗口，不能够完全填充，而如果全部采用较大的窗口，则图像会被严重平滑。因此根据空洞的大小，不断调整滤波窗口。先用大窗口给所有空洞赋值，然后利用逐渐变成小窗口滤波覆盖原来的值，这样既能保证空洞能被填充上，也能保证图像不会被过度平滑。

空洞填充具体代码见附录，填充后的视差图如下：



### 2.4将空洞填充后的视差图转化为深度图

视差的单位是像素（pixel），深度的单位往往是毫米（mm）表示，视差与深度的转换公式如下：

depth = ( f \* baseline) / disp

depth——深度值；

f——归一化的焦距，也就是内参中的fx；

baseline——两个相机光心之间的距离，称作基线距离；

disp——视差值。

在上面我们用SGBM算法获取了视差图，接下来转换为深度图，具体代码见附录。其中，我们将深度图保存为png格式图片，这样数据为ushort类型及CV\_16UC1，其方位为0-65535，如果是mm为单位，则最大能表示约65米的深度，足够使用。最终空洞填充后的深度图，如下，左边为没有填充的视差图生成的深度图，右边为填充视差图生成的图：



也可以采用相对深度，即通过设置baseline等于1，然后将数值从0-65525映射到0-255进而得到相对深度图，这样就能够将其保存为uchar类型，但是精度没有那么高。

### 2.5虚拟视点的合成

利用视差图合成虚拟视点，视差图一般通过图像匹配获取，主要有两种方法。

**2.5.1正向映射**

简单的利用左视点原图和视差图进行视点合成，取每一个像素点处的视差值，然后计算新图像中像素点位置，然后赋值。但是这种方法容易出现边缘锯齿现象。利用没有填充的视差图得到的效果如下：



利用填充后的视差图得到的效果如下：



**2.5.2反向映射**

先根据左视点视差图生成虚拟视点的视差图，然后反向映射得到每一个像素点在原图像中的浮点位置，利用线性插值获取最终颜色值。这种方法为反向映射结合线性插值方法，但是如果处理的原视差图未经过空洞填充处理，容易出现裂纹现象，故本项目采用反向映射+空洞填充+双线性插值。填充空洞后生的虚拟视点图如下：

