圆阵列棋盘平面相机标定法
程序说明文档

赵国栋 SZ1705063

一、程序完成的主要功能

此程序主要完成相机标定功能，采用张正友二维平面标定的方法，但是使用圆阵列棋盘（见图 1、2），结合 OpenCV 函数库编写。OpenCV 库之中有封装好的一整套相机标定函数，如果使用方块棋盘作为标定物，只需连续调用 CV 函数便可以标定相机。本程序保留了最后一步的标定函数即 cvCalibrateCamera2 函数，因此标定原理不变，主要对此函数之前的部分改动，实现功能包括：图像阈值化处理、对阵列圆点棋盘的圆轮廓检测、圆心的确定，图像 2D 点与空间 3D 点对应顺序的处理、将相应的数据输入 cvCalibrateCamera2 函数中使其正常执行以及过程和最终数据的打印与存储。

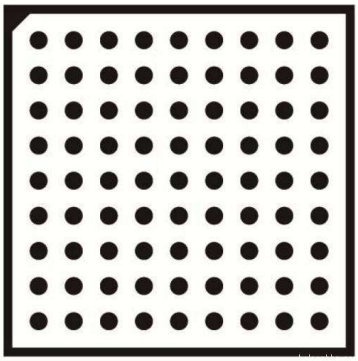

图 1 圆点阵棋盘

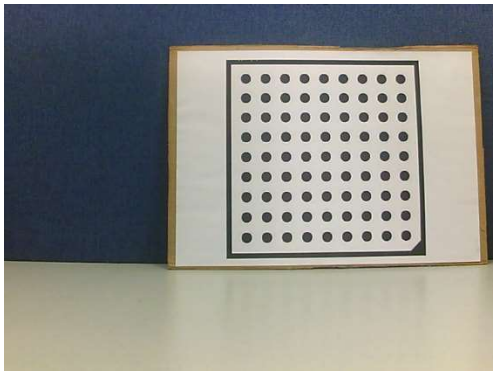

图 2 标定使用的图片之一

二、程序设计与分析

程序流程：

1. 读入 RGB 图像，转换为灰度图像；
2. 将灰度图二值化，使区域最大程度连通成块；
3. 查找轮廓，并且通过面积条件、长宽比条件、最小长度宽度条件将非圆点的轮廓去除；
4. 运用最小外接矩形（可以与图像边缘不平行的那一种）拟合圆轮廓，并求得此矩形的中心作为圆的中心。
5. 将所得圆心数据排序，使其与最终输入的三维点坐标一一对应。
6. 向 cvCalibrateCamera2 函数输入相应的数据、参数，完成标定。

分析：

第 2 步：二值化可以减少轮廓数量，便于剔除不需要的轮廓，否则图中会出现许多细小的轮廓碎片。

第 3 步：主要需要剔除的是大面积的外部边框和细小轮廓碎片，由于圆形在各角度想长宽比不会差太多，所以设置长宽比剔除异样轮廓。

第 4 步：因角度问题，圆在图像中表现为椭圆，最小外接矩形可以使矩形长与椭圆长轴平行，矩形宽与椭圆短轴平行，中心与圆心相同。
第 5 步：三维点坐标采用图片左上到右下的顺序，因为轮廓排序遵循先 y 方向后 x 方向的顺序，且先记录大值，所以一横排圆心坐标连续存储，考前的数据值较大。排序分为两步，第一步每 9 个数据按 x 坐标升序排列；第二步，将每幅图 81 个数据分 9 组按 y 坐标升序排列。
第 6 步：使用张正友二维标定原理。

三、关键程序代码

```
cvThreshold( src, src,160, 255, CV_THRESH_BINARY );
```
使用双边二值化，阈值 160 以下归 0，以上归 255，阈值的确定是通过参数调试得到，对此次标定的图像，在 160 附近可以最大程度地将图像连成区域，减少轮廓数目。

```
cvFindContours( src, storage, &contour, sizeof(CvContour), CV_RETR_LIST, CV_CHAIN_APPROX_SIMPLE );
```
查找轮廓函数，将轮廓以链的结构存储在 storage 内。

```
double maxarea=500;//面积上限
double minarea=100;//面积下限
double tmparea=fabs(cvContourArea(contour));//获得轮廓面积
if(tmparea < minarea) {
    cvSeqRemove(contour,0); //删除面积小于设定值的轮廓
    continue;}
if(tmparea > maxarea) {
    cvSeqRemove(contour,0); //删除面积大于设定值的轮廓
    continue;}
```
面积主要用于删除外框轮廓、细小的轮廓

```
CvRect aRect = cvBoundingRect( contour, 0 ); //拟合最小外接矩形
if(aRect.height<8){
    cvSeqRemove(contour,0); //删除高度小于设定值的轮廓
    continue;}
if ((aRect.width/aRect.height)>2){
    cvSeqRemove(contour,0); //删除宽高比例大于设定值的轮廓
    continue;}
```
外接矩形用于删除面积条件无法剔除的轮廓

```
box=cvMinAreaRect2(contour,0);//拟合可旋转的最小外接矩形
float x=box.center.x;//得到矩形中心坐标，作为圆心坐标
float y=box.center.y;

cvCalibrateCamera2(object_points.cvmat,//三维空间点矩阵
```

```
image_points.cvmat,//二维图像点矩阵
point_counts.cvmat,//每幅图点数
image_size,//图像尺寸
intrinsic_matrix.cvmat,//内参矩阵
distortion_coeffs.cvmat,//畸变系数矩阵
rotation_vectors.cvmat,//旋转矩阵
translation_vectors.cvmat,//平移矩阵
0);
```

相机标定函数及各参数

四、程序运行结果分析

本程序运行的结果（具体结果保存在 caliberation_result.txt 中）：

内参矩阵： {1225.98,0,357.299}
{0,1230.89,285.295}
{0,0,1}

畸变系数向量： {-0.287842,0.652812,0.0151536,-0.0086691}

为验证结果的正确性，作者使用 MATLAB 软件以及张氏黑白方块棋盘的标定工具箱 TOOLBOX_calib 对同一个相机拍摄的黑白方块棋盘进行标定（因为网传 MATLAB 比 OpenCV 相机标定精度高）。

MATLAB 标定的结果如下（具体结果保存在 Calib_Results.txt 中）：

内参矩阵： {1175.943825±19.215869,0,366.531684±26.793159}
{0,1178.324143±17.202079,244.884491±21.368023}
{0,0,1}

畸变系数向量： {-0.133916, -0.146815, 0.006616, -0.009399}

结果分析:

可以看到结果大体接近，但偏差还是不小的，举例来讲，第一个参数的偏差比率：

$$\frac{1225.98-11\quad.943825}{1175.943825}\times 100\,\% = 4.25\,\%$$

作者认为偏差产生的主要原因在于圆心确定的一步，没有进行亚像素处理，从小数点后位数也可以看出，虽然采用浮点数计算，但取得的圆心坐标最小分辨率为 0.25，而 MATLAB 标定中小数位则持续到小数点后十几位。由于此课程的另一个作业正是亚像素圆检测，作者在此程序里就不进一步优化，把此步交给另一半同学吧。

作者在查阅资料过程中找到一种思路，将外边框检测出来，拟合四边形，以四边形四个顶点为参照将图片进行透视变换变成正对相机的平面，此时圆变成正常圆形非椭圆，采用霍夫圆变换检测圆及圆心，此种方法比拟合外接矩形精度高，再将圆心坐标做反透视变换变到原图上作为真正的圆心坐标。

程序的另一个问题是鲁棒性能较差，即此程序不具备广泛适用性。由于程序设计是针对这一批图像的，因此在算法选择上采用简单化的原则，换一批图像进行测试可能无法得到好的结果。

例如二值化阈值 160，对另一批图像来说不一定为最佳，可以考虑采用动态阈值的方法；再如轮廓的过滤采用面积加长宽组合，使用其他图像就可能有一些形状奇怪但面积符合阈值范围的轮廓被保留；又如图像 2D 数据的排序，对于倾斜较大的图片，并不能保证同一排的 9 个数据被存储在相连的 9 个存储单位里，最终会导致 2D 与 3D 数据不相对应。

若是想要进行优化达到普遍适用，需要进行更多图像数据的测试，并对以上三个问题进行改进，使其满足各类情况。