专业：计算机科学与技术
姓名：卢佳盈
学号：3180103570
日期：2020/1/2

浙江大学实验报告

课程名称：计算机视觉 指导老师：宋明黎 成绩：

实验名称：相机定标与鸟瞰图变换

一、实验目的和要求

1. 参考 Learning OpenCV 示例 18-1，利用棋盘格图像进行相机定标，将参数写入 xml 文件保存。棋盘格图像见群文件 Learning OpenCV/LearningOpenCV_Code/LearningOpenCV_Code/calibration

2. 参考示例 19-1，根据求得的内参实现鸟瞰图（俯视）转换，测试图片见群文件 Learning OpenCV/LearningOpenCV_Code/LearningOpenCV_Code/birdseye

二、实验内容和原理

2.1 开发环境

- Windows X64
- Visual Studio 2017
- opencv-3.4.0

2.2 运行方式

```
eyeBird.exe <图像所在文件夹> <棋盘宽的角点数> <棋盘高中的角点数>
如：eyeBird.exe stereoData  9 6
如：eyeBird.exe calibration 12 12
```

2.3 原理介绍

【角点检测】

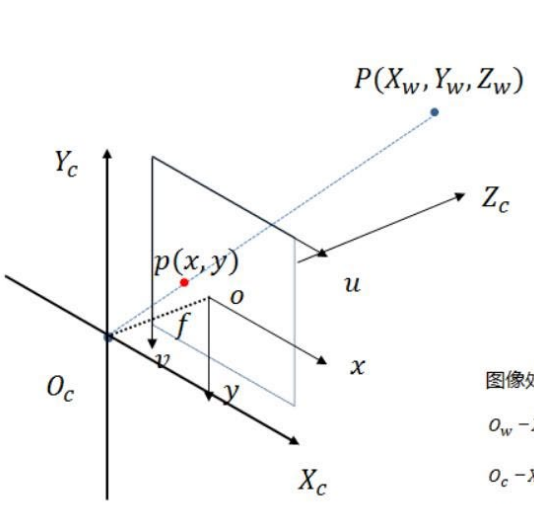
角点就是极值点，即在某方面属性特别突出的点。角点可以是两条线的交叉处，也可以是位于相邻的两个主要方向不同的事物上的点。目前的角点检测算法可归纳为 3 类：基于灰度图像的角点检测、基于二值图像的角点检测、基于轮廓曲线的角点检测。

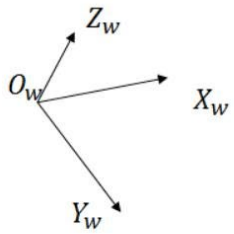
实验中我采用了 openCV 提供的 API：findChessboardCorners，但该检测函数返回的是像素整数坐标，为了是得到的角点坐标更加精确，我们还可以采用基于灰度图像的亚像素化角点的 API：find4QuadCornerSubpix，从而得到浮点型坐标。

【相机标定】

为确定空间物体表面某点的三维几何位置与其在图像中对应点之间的相互关系，必须建立相机成像的几何模型，这些几何模型参数就是相机参数。在大多数条件下这些参数必须通过实验与计算才能得到，这个求解参数的过程就称之为相机标定（或摄像机标定）。

基本原理如下图：





图像处理中涉及到以下四个坐标系：

$O_w - X_w Y_w Z_w$：世界坐标系,描述相机位置,单位m

$O_c - X_c Y_c Z_c$ ：相机坐标系,光心为原点,单位m

$o - xy$：图像坐标系,光心为图像中点,单位mm

uv：像素坐标系,原点为图像左上角,单位pixel

P：世界坐标系中的一点，即为生活中真实的一点；

p：点P在图像中的成像点，在图像坐标系中的坐标为(x, y)，在像素坐标系中的坐标为(u, v);

f ：相机焦距，等于o与O_c的距离，$f = \|o - O_c\|$

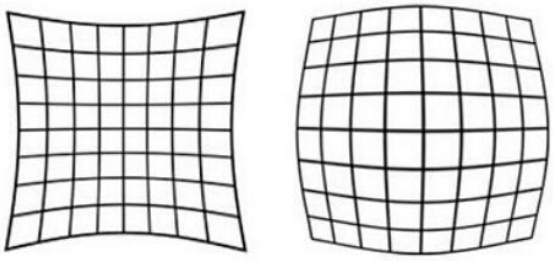
$$Z_c \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{dx} & 0 & u_0 \\ 0 & \frac{1}{dy} & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R & T \\ \vec{0} & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} f_x & 0 & u_0 & 0 \\ 0 & f_y & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R & T \\ \vec{0} & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

相机内参　　相机外参

【透镜畸变】

透镜由于制造精度以及组装工艺的偏差会引入畸变，导致原始图像的失真。镜头的畸变分为径向畸变和切向畸变两类。



• 径向畸变

径向畸变就是沿着透镜半径方向分布的畸变，产生原因是光线在原理透镜中心的地方比靠近中心的地方更加弯曲。 成像仪光轴中心的畸变为 0，沿着镜头半径方向向边缘移动，畸变越来越严重。畸变的数学模型可以用主点（principle point）周围的泰勒级数展开式的前几项进行描述，通常使用前两项，即 k1 和 k2，对于畸变很大的镜头，可以增加使用第三项 k3。

$$x_0 = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$
$$y_0 = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$

• 切向畸变

切向畸变是由于透镜本身与相机传感器平面（成像平面）或图像平面不平行而产生的，这种情况多是由于透镜被粘贴到镜头模组上的安装偏差导致。畸变模型可以用两个额外的参数 p1 和 p2 来描述：

$$x_0 = x + [2p_1 y + p_2(r^2 + 2x^2)]$$
$$y_0 = x + [2p_1 y + p_2(r^2 + 2y^2)]$$

【透镜变换】

透视变换矩阵变换公式为：

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

这是一个从二维空间变换到三维空间的转换，因为图像在二维平面，故除以 Z，（X';Y';Z'）表示图像上的点：

$$\begin{cases} X' = \frac{X}{Z} = \frac{a_{11}x + a_{12}y + a_{13}}{a_{31}x + a_{32}y + a_{33}} \\ Y' = \frac{Y}{Z} = \frac{a_{21}x + a_{22}y + a_{23}}{a_{31}x + a_{32}y + a_{33}} \\ Z' = \frac{Z}{Z} = 1 \end{cases}$$

令 a33=1，展开上面公式，得到一个点的情况：4 个点可以得到 8 个方程，即可解出透视变换矩阵 A。

【生成鸟瞰图】

摄像机斜视拍摄一物体后，形成的图像会发生变形，如果将图像映射到拍摄物体平面上，相当于将相机垂直于拍摄平面，这样就会得到图像的真实形状。由于这种映射相当于将原图重新透视到另一个平面，这种称之为"重投影"。

鸟瞰图的本质就是将图像平面中的信息"重投影"到地平面上，所以，首先要获取两个平面间的投影变换关系 H。在程序中，是通过在地平面上放置标定板图像，然后获得地平面上棋盘格图像上四个顶点的坐标（0，0），（widht-1,0）,(0,height -1),(wdith-1,height-1)；同时，在拍摄的图像平面提取角点，并获得与地平面上四个点对应的角点在图像空间中的坐标值，通过四个坐标点间的对应关系，基于 getPerspectiveTransform（）函数，获得地平面到图像平面间的投影变换关系 H；最后，通过 warpPerspective（）函数对图像进行逆向映射到地平面空间中。

三、 实现过程

3.1 获得文件夹中所有图片的文件名

```
vector<String> get_image_names(string file_path) {
    vector<String>file_names;
```

```
    intptr_t hFile = 0;
    _finddata_t fileInfo;
    hFile = _findfirst(file_path.c_str(), &fileInfo);
    if (hFile != -1) {
        do {
            if ((fileInfo.attrib&_A_SUBDIR)) {
                continue;
            }
            else {
                file_names.push_back(fileInfo.name);
            }
        } while (_findnext(hFile, &fileInfo) == 0);
        _findclose(hFile);
    }
    return file_names;
}
```

3.2 获得所有图像的角点，并进行亚像素精确化

寻找角点：

```
int cvFindChessboardCorners(
    const void* image,
    CvSize pattern_size,
    CvPoint2D32f* corners,
    int* corner_count=NULL,
    int flags=CV_CALIB_CB_ADAPTIVE_THRESH );
```

精确寻找亚像素的内角点:

```
find4QuadCornerSubpix(InputArray img, InputOutputArray corners, Size region_size);
```

在目标图片上绘制角点的图标：

```
void cv::drawChessboardCorners(
    cv::InputOutputArray image, // 棋盘格图像（8UC3）即是输入也是输出
    cv::Size patternSize, // 棋盘格内部角点的行、列数
    cv::InputArray corners, // findChessboardCorners()输出的角点
    bool patternWasFound // findChessboardCorners()的返回值
);
```

```
if (!findChessboardCorners(imageInput, board_size, image_corners)) {
    exit(1);
}
else {
    Mat imageGray;
    cvtColor(imageInput, imageGray, CV_RGB2GRAY);
    find4QuadCornerSubpix(imageGray, image_corners, board_size);//亚像素精确化
    all_corners.push_back(image_corners);//保存图像角点
```

```
    drawChessboardCorners(imageInput, board_size, image_corners, false);
    imwrite((string)dataDir+"/corners/"+imgList[i], imageInput);//保存标定角点后的图片，保存在原路径下 corners 文件夹中
}
```

3.3 摄像机内外参标定

标定函数：

```
double calibrateCamera(InputArrayOfArrays objectPoints,
                       InputArrayOfArrays imagePoints,
                       Size imageSize,
                       InputOutputArray cameraMatrix,
                       InputOutputArray distCoeffs,
                       OutputArrayOfArrays rvecs,
                       OutputArrayOfArrays tvecs,
                       int flags=0)
```

```
calibrateCamera(object_points, all_corners, image_size, cameraMatrix, distCoeffs, rvecsMat, tvecsMat, 0);
```

其中相机内参矩阵 cameraMatrix、畸变系数 distCoeffs、相机外参矩阵旋转向量 tvecsMat、相机外参矩阵平移向量 rvecsMat 定义如下：

```
Mat cameraMatrix = Mat(3, 3, CV_32FC1, Scalar::all(0)); // 摄像机内参数矩阵
Mat distCoeffs = Mat(1, 5, CV_32FC1, Scalar::all(0)); //摄像机的 5 个畸变系数：k1,k2,p1,p2,k3
vector<Mat> tvecsMat;  //图像的旋转向量数组
vector<Mat> rvecsMat;//图像的平移向量数组
```

3.4 保存参数结果

```
void writeCalibrate() {
    FileStorage fs("intrinsics.xml", FileStorage::WRITE);
    fs << "imageWidth" << image_size.width;
    fs << "imageHeight" << image_size.height;
    fs << "cameraMatric" << cameraMatrix;
    fs << "distCoeffs" << distCoeffs;
    fs.release();
}
```

3.5 矫正图像

图像矫正函数：

```
initUndistortRectifyMap( InputArray cameraMatrix, InputArray distCoeffs,
                         InputArray R, InputArray newCameraMatrix,
                         Size size, int m1type, OutputArray map1, OutputArray map2 );
```

```
initUndistortRectifyMap(cameraMatrix, distCoeffs, R, cameraMatrix, image_size, CV_32FC1, map1, map2);
```

```
Mat imageInput = imread((string)dataDir + "/" + imgList[i]);
Mat correctImage = imageInput.clone();
remap(imageInput, correctImage, map1, map2, INTER_LINEAR);
imwrite((string)dataDir + "/correct/" + imgList[i], correctImage);
```

3.6 透视变换生成鸟瞰图

由四对点计算透射变换：

```
CvMat* cvGetPerspectiveTransform( const CvPoint2D32f*src,
                                  const CvPoint2D32f* dst,
                                   CvMat*map_matrix );
```

对图像进行透视变换：

```
void cvWarpPerspective( const CvArr* src, CvArr* dst,const CvMat* map_matrix,
                        int flags=CV_INTER_LINEAR+CV_WARP_FILL_OUTLIERS,
                        CvScalar fillval=cvScalarAll(0) );
```

```
Mat h = Mat(3, 3, CV_32F, Scalar::all(0));//找到单应矩阵
vector<Point2f> objPts(4);//选定的4对顶点
vector<Point2f> imgPts(4);
int indexArray[4] = {                                   //每对顶点在顶点数组中的index
    0,                                                  //No.1:左上角(0,0)
    board_size.width - 1,                               //No.2:右上角(w-1,0)
    (board_size.height - 1)*board_size.width,           //No.3:左下角(0,h-1)
    board_size.height*board_size.width - 1              //No.4:右下角(w-1,h-1)
};
//给选定的4对顶点赋值：必须是point2f类型，所以objPts只取x,y坐标
for (int j = 0; j < 4; j++) {
    objPts[j].x = object_points[i][indexArray[j]].x;
    objPts[j].y = object_points[i][indexArray[j]].y;
    imgPts[j] = all_corners[i][indexArray[j]];
}

h = getPerspectiveTransform(objPts, imgPts);

Mat imageInput = imread((string)dataDir + "/corners/" + imgList[i]);
Mat imageBird = imageInput.clone();
//使用单应矩阵来remap view
warpPerspective(imageInput, imageBird, h, image_size, CV_INTER_LINEAR + CV_WARP_INV
ERSE_MAP + CV_WARP_FILL_OUTLIERS);
```

四、实验结果

【数据储存】

stereoData 库 intrinsics.xml：

```
<?xml version="1.0"?>
<opencv_storage>
<imageWidth>640</imageWidth>
<imageWidth>imageHeight</imageWidth>
<imageWidth>480</imageWidth>
<cameraMatric type_id="opencv-matrix">
  <rows>3</rows>
  <cols>3</cols>
  <dt>d</dt>
  <data>
    5.3462073064668232e+02 0. 3.3168833821176571e+02 0.
    5.3441680319697468e+02 2.4055945689700553e+02 0. 0. 1.</data></cameraMatric>
<cameraMatric>distCoeffs</cameraMatric>
<cameraMatric type_id="opencv-matrix">
  <rows>1</rows>
  <cols>5</cols>
  <dt>d</dt>
  <data>
    -3.0592485621749138e-01 1.7734397899770143e-01
    -7.5780380165622189e-06 -1.0614675096782724e-03
    -9.4816891545943166e-02</data></cameraMatric>
</opencv_storage>
```

手机相机内参数：

$$\begin{bmatrix} 5.3462073064668232e+02, \\ 0, \\ 3.3168833821176571e+02, \\ 0, \\ 5.3441680319697468e+02, \\ 2.4055945689700553e+02, \\ 0, \\ 0, \\ 1 \end{bmatrix}$$

畸变系数：

$$\begin{bmatrix} -3.0592485621749138e-01, \\ 1.7734397899770143e-01, \\ -7.5780380165622189e-06, \\ -1.0614675096782724e-03, \\ -9.4816891545943166e-02 \end{bmatrix}$$

calibration 库内参 intrinsics.xml：

```
<?xml version="1.0"?>
<opencv_storage>
<imageWidth>1600</imageWidth>
<imageWidth>imageHeight</imageWidth>
<imageWidth>1200</imageWidth>
<cameraMatric type_id="opencv-matrix">
  <rows>3</rows>
  <cols>3</cols>
  <dt>d</dt>
  <data>
    1.7658255778186283e+03 0. 8.2221720650245106e+02 0.
    1.7639201436073170e+03 6.6592875724543728e+02 0. 0. 1.</data></cameraMatric>
<cameraMatric>distCoeffs</cameraMatric>
<cameraMatric type_id="opencv-matrix">
  <rows>1</rows>
  <cols>5</cols>
  <dt>d</dt>
  <data>
    5.2421468818907475e-02 -4.3755723645950101e-01
    1.1118728842369264e-02 3.0921778697701394e-03 1.3526792978294593e+00</data></cameraMatric>
</opencv_storage>
```

手机相机内参数：

$$\begin{bmatrix} 1.7658255778186283e+03, \\ 0, \\ 8.2221720650245106e+02, \\ 0, \\ 1.7639201436073170e+03, \\ 6.6592875724543728e+02, \\ 0, \\ 0, \\ 1 \end{bmatrix}$$

畸变系数：

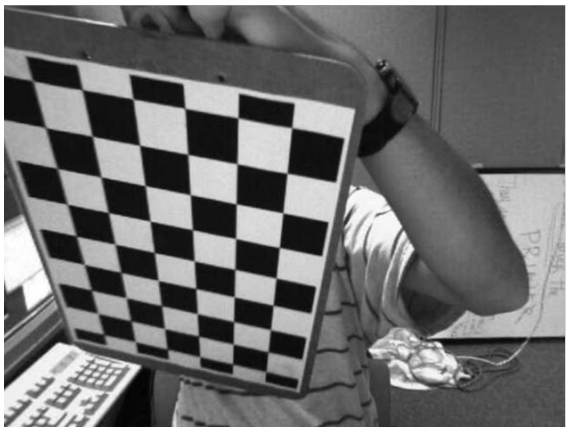
$$\begin{bmatrix} 5.2421468818907475e-02, \\ -4.3755723645950101e-01, \\ 1.1118728842369264e-02, \\ 3.0921778697701394e-03, \\ 1.3526792978294593e+00 \end{bmatrix}$$

【生成图像】

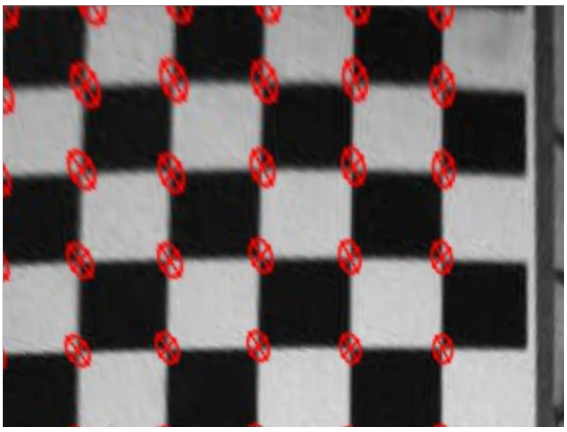
stereoData：（right01.jpg）

原图

标定角点

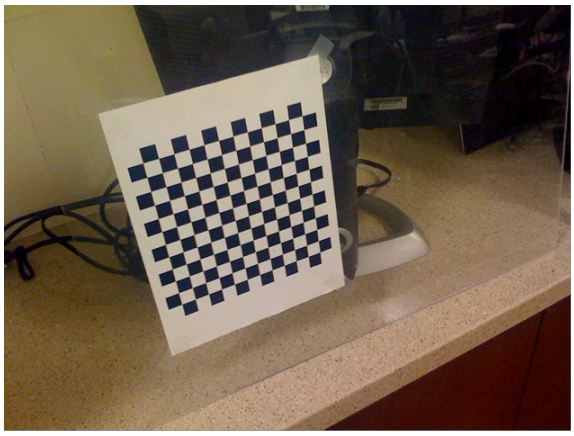


修正畸变

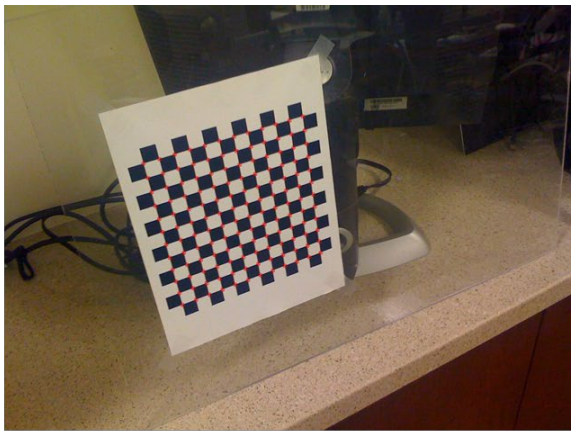


鸟瞰图

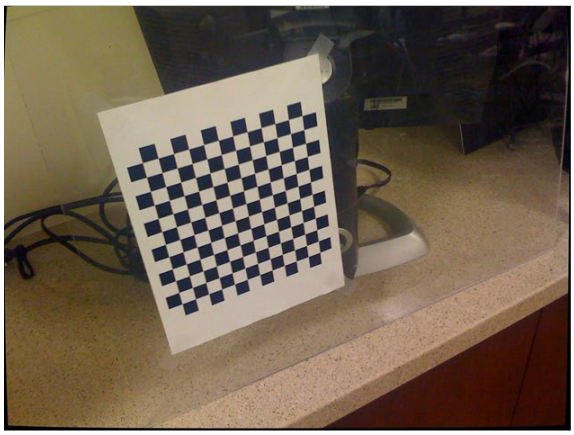
calibration：（IMG_0191.jpg）



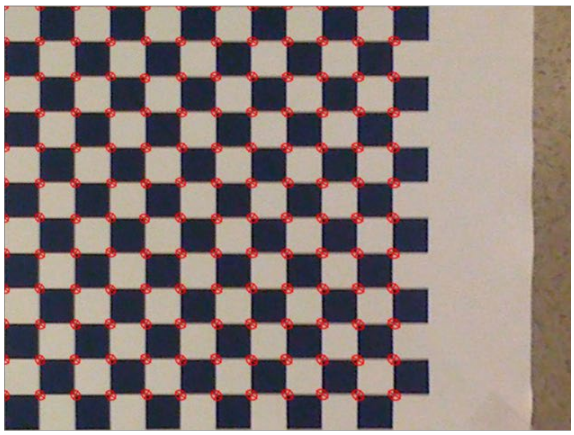
原图



标定角点



修正畸变



鸟瞰图