CV 作业三报告

姓名：秦昇　学号：3120000060　课程：计算机视觉

软件开发说明：

1. 基于 OpenCV 库，对输入的一组棋盘图像进行标定。
2. 输出根据图像标定所得相机的 4 个内参和 5 个畸变参数。
3. 对输入的一幅图像，根据所得相机标定参数进行还原并输出。
4. 输出该图像的 bird_eye 俯瞰视角变换图。

算法具体步骤

1. **图像的读入和棋盘角点检测**

   为了便于读入，将输入的图片名称列表放在 lists.txt 文件中。利用如下函数来检测棋盘内部的角点，这里需预先设置棋盘内角点数为每行每列 12 个。这里将图像转化为灰度图像后，可以通过 cvFindCornerSubPix 函数来精确检测角点，得到像素级别的位置。

```
int found = cvFindChessboardCorners(
                image,
                board_sz,
                corners,
                &corner_count,
```

```
            CV_CALIB_CB_ADAPTIVE_THRESH |
CV_CALIB_CB_FILTER_QUADS
            );
       cvCvtColor(image, gray_image, CV_BGR2GRAY);
       cvFindCornerSubPix(gray_image, corners, corner_count,
           cvSize(11, 11), cvSize(-1, -1),
cvTermCriteria(CV_TERMCRIT_EPS + CV_TERMCRIT_ITER, 30, 0.1));
```

在测试时我们可以利用 cvDrawChessboardCorners 和 cvShowImage 函数来输出将检测到角点画出后的图像。

2. 相机标定与参数输出

需要注意的是，在标定的时候，并非每一张图像都能够检测出完整的角点，例如给定的第一组 listss1.txt 中，23 幅图像只有 22 幅检测出了所有角点，所以预先需要将角点的矩阵的大小变为正确的形式

```
       CvMat* object_points2 = cvCreateMat(successes*board_n, 3,
CV_32FC1);
       CvMat* image_points2 = cvCreateMat(successes*board_n, 2,
CV_32FC1);
       CvMat*   point_counts2   =   cvCreateMat(successes,   1,
CV_32SC1);
       //TRANSFER THE POINTS INTO THE CORRECT SIZE MATRICES
       for (int i = 0; i<successes*board_n; ++i){
           ///      CV_MAT_ELEM(*image_points2, CvPoint2D32f,0,i)
= CV_MAT_ELEM(*image_points, CvPoint2D32f,0,i);
           ///      CV_MAT_ELEM(*object_points2,CvPoint3D32f,0,i)
= CV_MAT_ELEM(*object_points,CvPoint3D32f,0,i);
           CV_MAT_ELEM(*image_points2,    float,    i,    0)    =
CV_MAT_ELEM(*image_points, float, i, 0);
           CV_MAT_ELEM(*image_points2,    float,    i,    1)    =
CV_MAT_ELEM(*image_points, float, i, 1);
           CV_MAT_ELEM(*object_points2,    float,    i,    0)    =
CV_MAT_ELEM(*object_points, float, i, 0);
           CV_MAT_ELEM(*object_points2,    float,    i,    1)    =
CV_MAT_ELEM(*object_points, float, i, 1);
           CV_MAT_ELEM(*object_points2,    float,    i,    2)    =
CV_MAT_ELEM(*object_points, float, i, 2);
```

```
}
```

然后调用 cvCalibrateCamera2 函数进行标定，结果在屏幕输出的同时也可以输出到 Intrinsics.xml 和 Distortion.xml 中，方便以后载入并使用

```
CV_MAT_ELEM(*intrinsic_matrix, float, 0, 0) = 1.0f;
CV_MAT_ELEM(*intrinsic_matrix, float, 1, 1) = 1.0f;
cvCalibrateCamera2(
    object_points2,
    image_points2,
    point_counts2,
    cvGetSize(image),
    intrinsic_matrix,
    distortion_coeffs,
    NULL,
    NULL,
    0//CV_CALIB_FIX_ASPECT_RATIO
    );
// Save our work
cvSave("Intrinsics.xml", intrinsic_matrix);
cvSave("Distortion.xml", distortion_coeffs);
```

3. 对输入图像进行畸变的消除

在载入根据输入图像进行标定的参数后，我们只需调用如下函数，来实现我们消除畸变时所需 mapx 和 mapy 矩阵。然后调用 cvRemap 函数，重绘图像并输出。

```
cvInitUndistortMap(
    intrinsic,
    distortion,
    mapx,
    mapy
    );
```

```
 IplImage *t = cvCloneImage(image);
cvRemap(t, image, mapx, mapy);
```

4. **实现 bird_eye 俯瞰视角变换**

将高度 Z 初始值设为 25，用 cvGetPerspectiveTransform 函数可以得到透视投影的变换矩阵，利用该矩阵绘图，可得透视投影图，按 u 和 d 键可以改变 Z 值的大小实现投影图像的实时变化。

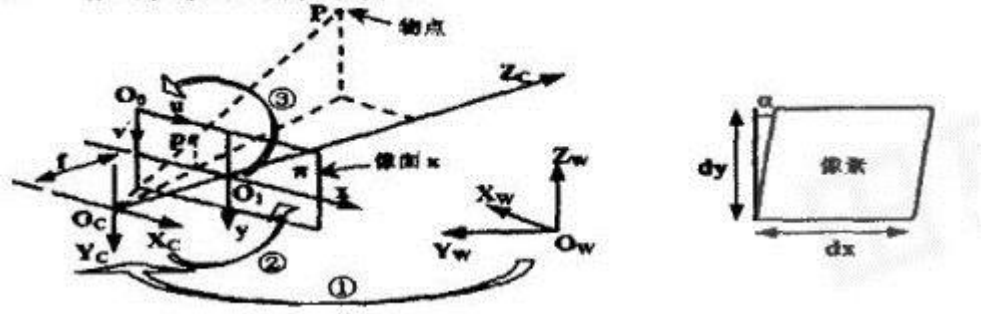
```
    while (key != 27) {//escape key stops
          CV_MAT_ELEM(*H, float, 2, 2) = Z;
          //    cvInvert(H,H_invt); //If you want to
invert the homography directly
          //
cvWarpPerspective(image,birds_image,H_invt,CV_INTER_LINEAR
+CV_WARP_FILL_OUTLIERS );
          //USE HOMOGRAPHY TO REMAP THE VIEW
          cvWarpPerspective(image, birds_image, H,
             CV_INTER_LINEAR + CV_WARP_INVERSE_MAP +
CV_WARP_FILL_OUTLIERS);
          cvShowImage("Birds_Eye", birds_image);
          key = cvWaitKey();
          if (key == 'u') Z += 0.5;
          if (key == 'd') Z -= 0.5;
       };
```

算法实现要点

1. **相机标定参数的确定和实际意义**

摄像机成像模型和四个坐标系

1 摄像机成像模型



摄像机模型采用经典的小孔模型，如图中 Oc（光心），像面 π 表示的是视野平面，其到光心的距离为 f（镜头焦距）。

四个坐标系分别为：世界坐标系（Ow），摄像机坐标系（Oc），图像物理坐标系（O1，单位 mm），图像像素坐标系（O，位于视野平面的左上角，单位 pix）。

空间某点 P 到其像点 p 的坐标转换过程主要是通过这四套坐标系的三次转换实现的，首先将世界坐标系进行平移和转换得到摄像机坐标系，然后根据三角几何变换得到图像物理坐标系，最后根据像素和公制单位的比率得到图像像素坐标系。（实际的应用过程是这个的逆过程，即由像素长度获知实际的长度）。转化的过程和公式参见

$$\begin{bmatrix} u_u \\ v_u \\ 1 \end{bmatrix} = \begin{bmatrix} 1/d_x & \mathrm{tg}a/d_x & u_0 \\ 0 & 1/d_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$= \frac{1}{Z_c} \begin{bmatrix} f_x & \gamma & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \lambda A[R, t] \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

cvCalibration2 通过对一定标板在不同方向多次（三次以上）完整拍照，不需要知道定标板的运动方式。直接获得相机的内参和畸变系数。该标定方法精度高于自定标法，且不需要高精度的定位仪器。

calibration 的算法包含两个模型：一.经典针孔模型，包含四个坐标系，二畸变模型

$$\delta_X(X,Y)=k_1X(X^2+Y^2)+(p_1(3X^2+Y^2)+2p_2XY)+s_1(X^2+Y^2)$$
$$\delta_Y(X,Y)=k_2X(X^2+Y^2)+(p_2(X^2+3Y^2)+2p_1XY)+s_2(X^2+Y^2)$$

公式三项依次表示，径向畸变，切线畸变，薄棱镜畸变。

2. 畸变消除和俯瞰视角变换

畸变消除实质上是得到了畸变参数后执行的一个逆过程。透视变换(Perspective Transformation)是将图片投影到一个新的视平面(Viewing Plane)，也称作投影映射(Projective Mapping)。通用的变换公式为：

$$[x',y',w']=[u,v,w]\begin{bmatrix}a_{11} & a_{12} & a_{13}\\ a_{21} & a_{22} & a_{23}\\ a_{31} & a_{32} & a_{33}\end{bmatrix}$$

u,v 是原始图片左边，对应得到变换后的图片坐标 x,y,其中 $x=x'/w', y=y'/w'$。

变换矩阵$\begin{bmatrix}a_{11} & a_{12} & a_{13}\\ a_{21} & a_{22} & a_{23}\\ a_{31} & a_{32} & a_{33}\end{bmatrix}$可以拆成 4 部分，$\begin{bmatrix}a_{11} & a_{12}\\ a_{21} & a_{22}\end{bmatrix}$表示线性变换，比如 scaling，shearing 和 ratotion。$\begin{bmatrix}a_{31} & a_{32}\end{bmatrix}$用于平移，$\begin{bmatrix}a_{13} & a_{23}\end{bmatrix}^T$产生透视变换。所以可以理解成仿射等是透视变换的特殊形式。经过透视变换之后的图片通常不是平行四边形（除非映射视平面和原来平面平行的情况）。

重写之前的变换公式可以得到：

$$x = \frac{x'}{w'} = \frac{a_{11}u + a_{21}v + a_{31}}{a_{13}u + a_{23}v + a_{33}}$$

$$y = \frac{y'}{w'} = \frac{a_{12}u + a_{22}v + a_{32}}{a_{13}u + a_{23}v + a_{33}}$$

在 OpenCV 中也实现了透视变换的公式求解和变换函数。

求解变换公式的函数：

Mat getPerspectiveTransform(const Point2f src[], const Point2f dst[])

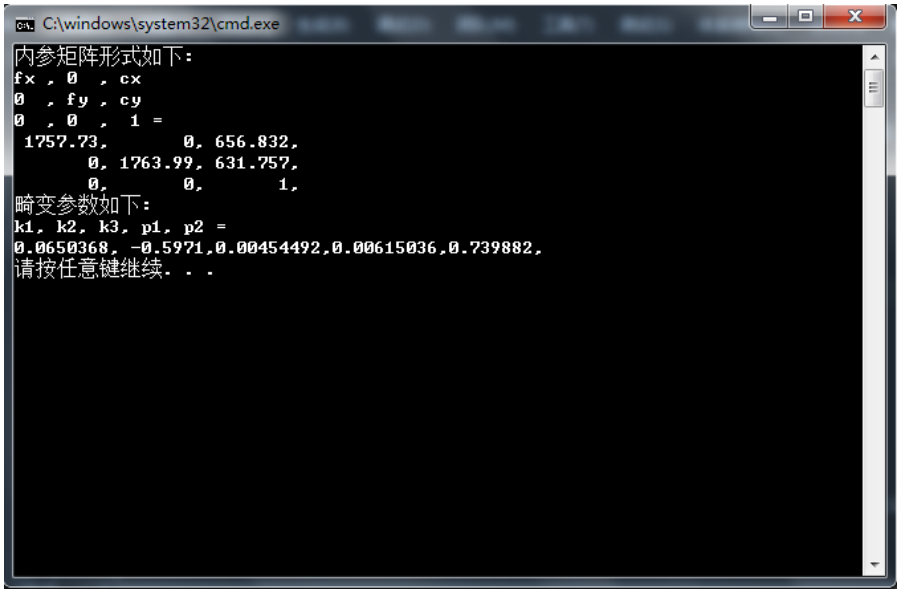
输入原始图像和变换之后的图像的对应 4 个点，便可以得到变换矩阵。之后用求解得到的矩阵输入 perspectiveTransform 便可以对一组点进行变换：

void perspectiveTransform(InputArray src, OutputArray dst, InputArray m)

注意这里 src 和 dst 的输入并不是图像，而是图像对应的坐标。

实验结果展示

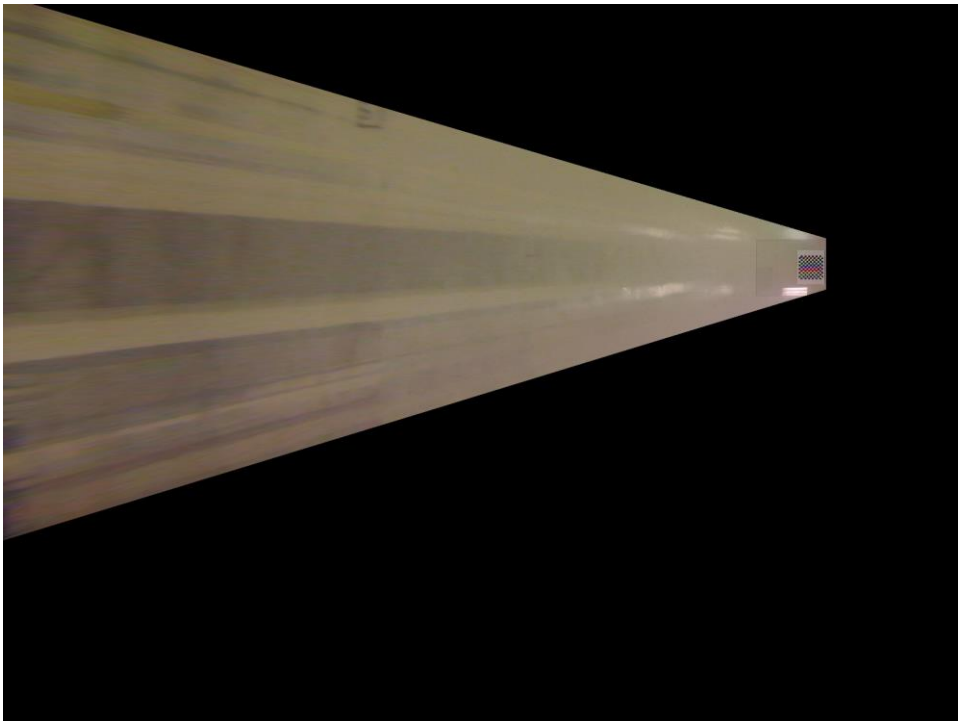
1. 输出畸变参数



2. 对图像进行畸变参数的消除

由于原图大小为 1600×1200，屏幕分辨率无法全部显示，这里是显示角点检测和绘制情况，中间过程是用 cvShowImage 函数输出的，所以只显示图像的部分。

3. bird_eye 透视投影视图



编程体会

1. **善于阅读和理解他人的代码**

   这次实验能够成功，很大部分原因要归功于 LearnOpenCV 这本书的指导，在看过这本书以后结合上课所学的知识，达到了一个从系统理论的角度到实践角度的转变，同时书中给出的代码也是很好的实例，让我得以减少工作量，并且学习到了最标准的代码书写规范。

2. **在学习借鉴的基础上有所反思**

   这一次遇到的最大的问题，出在一个很小的细节上，在编写相机标定部分的时候我借鉴了书中给出的 ch11_ex11_1_fromdisk.cpp 的代码，结果运行起来，以第一组图片（IMG_0191-IMG0203）作为实验的测试图片是始终会出现错误并终端，经过不断查找我发现错误来源于 cvtColor 函数，检查该函数的参数，我推测可能是 gray_Image 图像产生了错误，后来发现，标程在转化的时候每一次并没有 release(gray_image)图像，如果 image 标定失败，那么在第二次初始化转化灰度图像的时候，gray_image 的 size 和里面的初始值都将不匹配而产生问题。在改正这个错误以后，就一帆风顺了。