OpenCV相机标定与畸变校正

> OpenCV单目相机标定，图像畸变校正

01 相机标定定义与原理

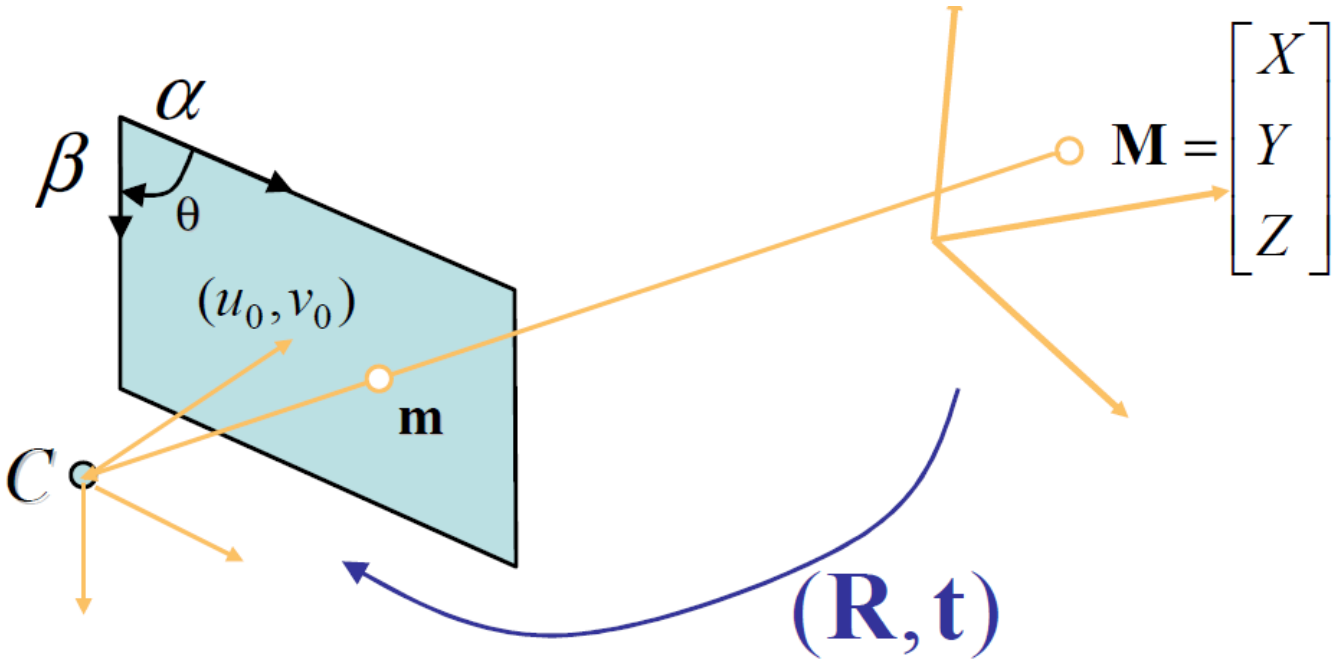
在图像测量过程以及机器视觉应用中，为确定空间物体表面某点的三维几何位置与其在图像中对应点之间的相互关系，必须建立相机成像的几何模型，这些几何模型参数就是相机参数。在大多数条件下这些参数必须通过实验与计算才能得到，这个求解参数的过程就称之为相机标定（或摄像机标定）。相机标定常见的分为：

> - 单目相机标定
> - 双目相机标定

相机标定是想从二维的图像中获取三维信息，实现图像的畸变校正、对象测量、三维重建等。由于光线投射导致实际对象物体跟投影到2D平面的图像不一致，幸运的是这种不一致性是稳定的，我们可以通过对相机标定，计算出畸变参数来实现对后续图像的畸变校正。根据标定技术不一样可以分为下面几类标定方法：

> - 基于3D对象参照标定
> - 基于2D平面标定
> - 基于1D线性标定
> - 自标定

最常见的相机成像方式是基于pinhole的模型、它的成像模型可以图示如下：



下面我们首先对这个相机成像模型做一番解释

下面我们首先对这个相机成像模型做一番解释：

一个二维点 m 表示为$\begin{bmatrix} u & v \end{bmatrix}^T$，一个三维点 M 表示为$\begin{bmatrix} X & Y & Z \end{bmatrix}^T$，其中图像上的 m 是来自 3D 对象中点 M 通过相机模型在像平面上的点，其中 C 表示的光学镜的中心，三点 C、m、M 在一条直线上。它们之间的关系可以通过下面公式表示：

$$s\tilde{m} = A[R \quad t]\tilde{M} \equiv P\tilde{M}$$

其中

$$A = \begin{bmatrix} \alpha & \gamma & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$P = A[R \quad t]$$

$$\tilde{m} = [u \quad v \quad 1]^T \quad \tilde{M} = [X \quad Y \quad Z \quad 1]^T$$

其中 s 表示任何放缩的尺度因子、（R,t）称为相机的外参数、A 成为相机的内参数。

$$\gamma = \alpha \cos\theta$$

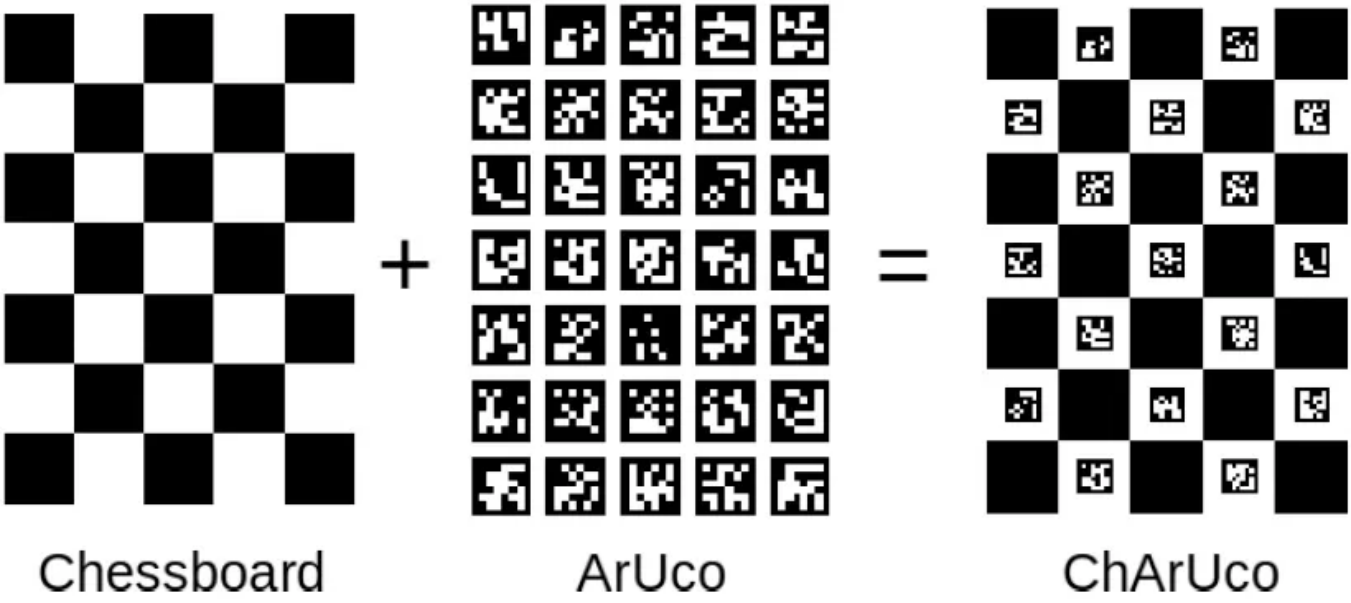
当像素表示为矩形时、$\theta = 90^0$、$\gamma = 0$。

通过标定算法同时求出相机内参与外参。最常用的算法是张正友标定算法。OpenCV/Matlab中均已经实现该算法。

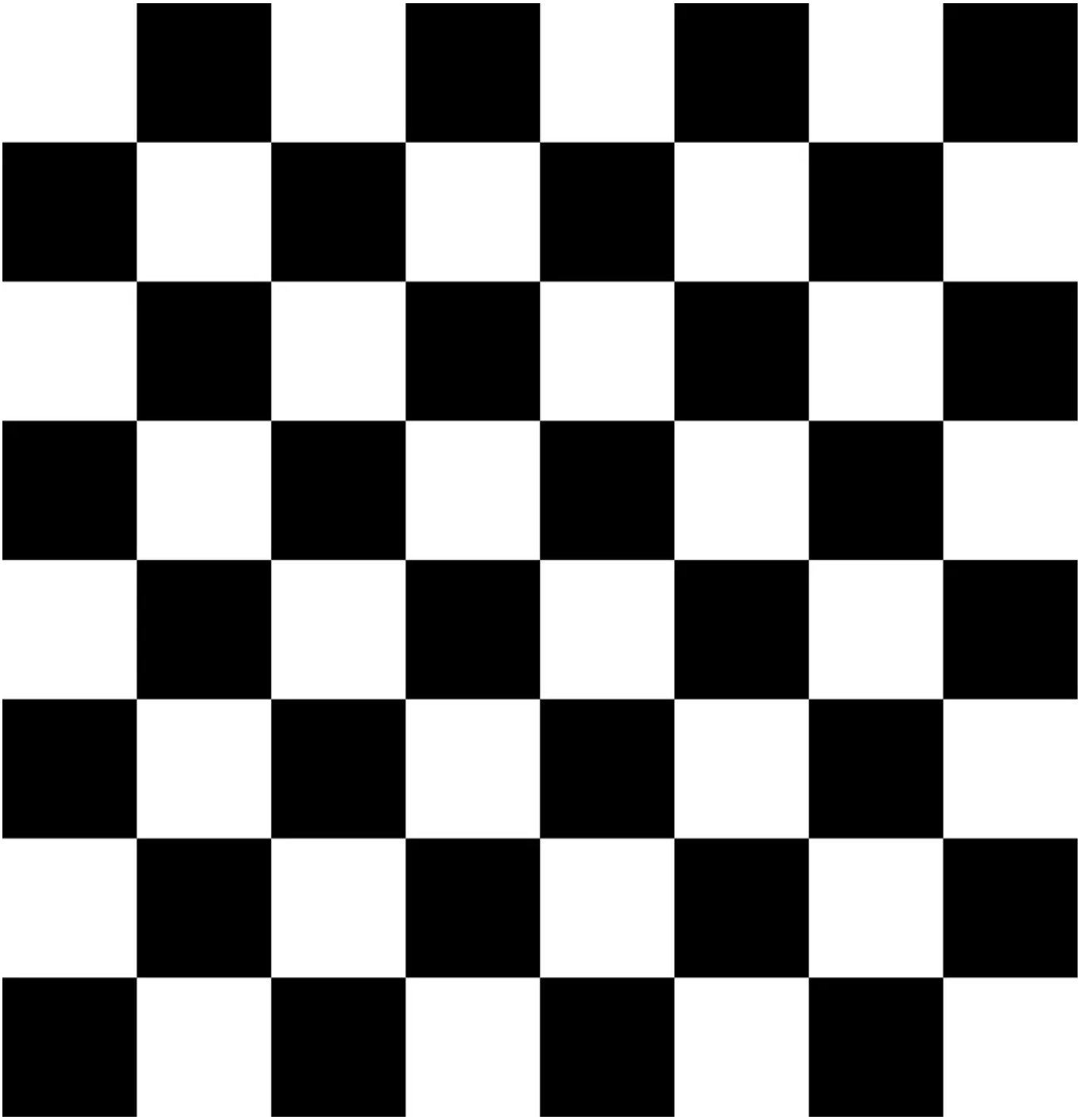
02 标定板介绍与制作

要想实现对相机的标定，我们首先需要给相机找到个参考对象，常见的就是标定版的类型有如下几种

> - Chessboard
> - Circel-grid
> - RandPattern
> - ArUco
> - ChArUc



OpenCV源码在其sample/data目录下面一个自带的棋盘图(chessboard.png)，显示如下:



在标定的时候，算法要求提供的棋盘格的**宽度**与**高度**，还有他们的**间隔距离**。需要特别注意是这里的宽高是指他们的内部交叉点的个数，以上图为例，它的大小为7x7而不是8x8。**间隔**是指棋盘格之间的距离，可以用像素距离表示，也可以用实际毫米为单位表示。

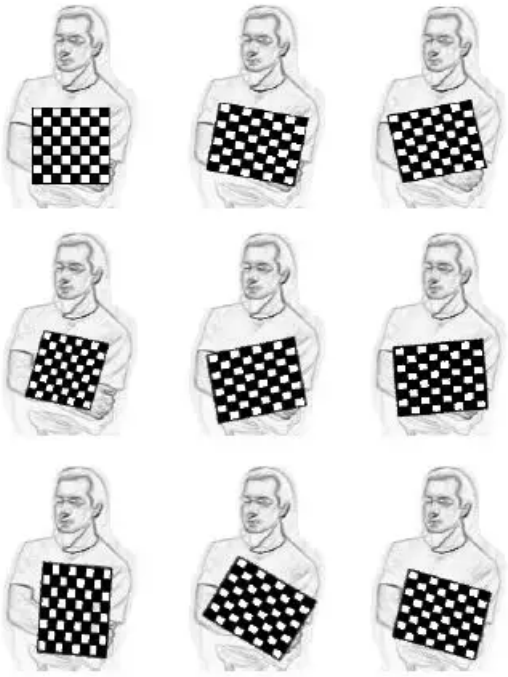
03 制作标定版与图像生成

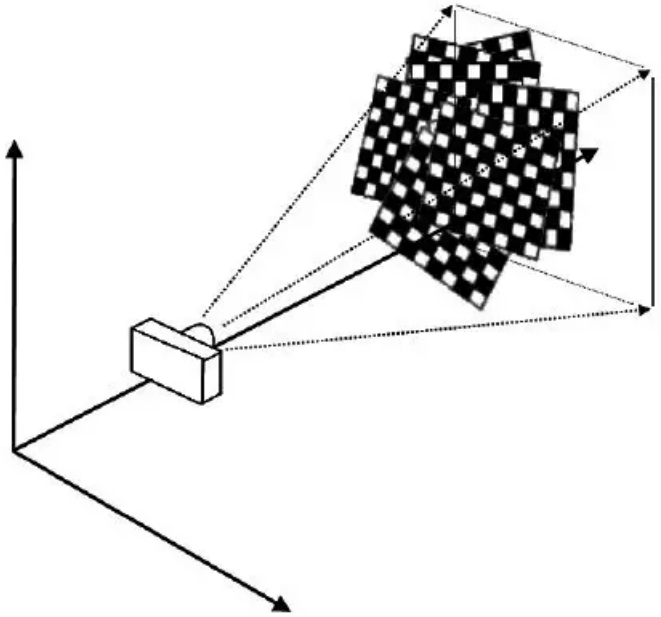
最简单的办法就是把上述图像直接打印出来，贴到一个塑料底板上就好啦。如果是土豪可以直接购买各种玻璃底板的标定板。另外还有一个更恶搞的方法，连打印都省啦，直接把chessboard.png这张图在另外一台电脑的显示器上显示，然后把摄像头对着它各种牌即可完成图像数据采集。这个是我手写的采集程序代码，每次想保存图像的时候请安Q字母键即可，代码如下：

```
void create_images() {
    Mat frame;
    VideoCapture capture(0);
```

```
    int index = 1;
    while (true) {
        bool ret = capture.read(frame);
        flip(frame, frame, 1);
        if (!ret) break;
        imshow("frame", frame);
        char c = waitKey(50);
        printf("%d \n", c);
        if (c == 113) { // Q
            imwrite(format("D:/images/zsxq/%d.png", index), frame);
            index += 1;
        }
        if (c == 27) {
            break; // ESC
        }
    }
    capture.release();
}
```

记得拿着棋盘格图，在镜头面前各种摆POSE，这个是属于你的表演时间，不要客气！具体参考下图：





04 相机标定程序实现

大家好，现在我们开始程序实现环节，OpenCV中在camera模块中已经实现了张正友标定算法。我们只需要正确调用，就可以计算出相机的内参与外参，完成相机的标定。具体的代码实现步骤如下：

定义相机标定的相关常量设置与变量

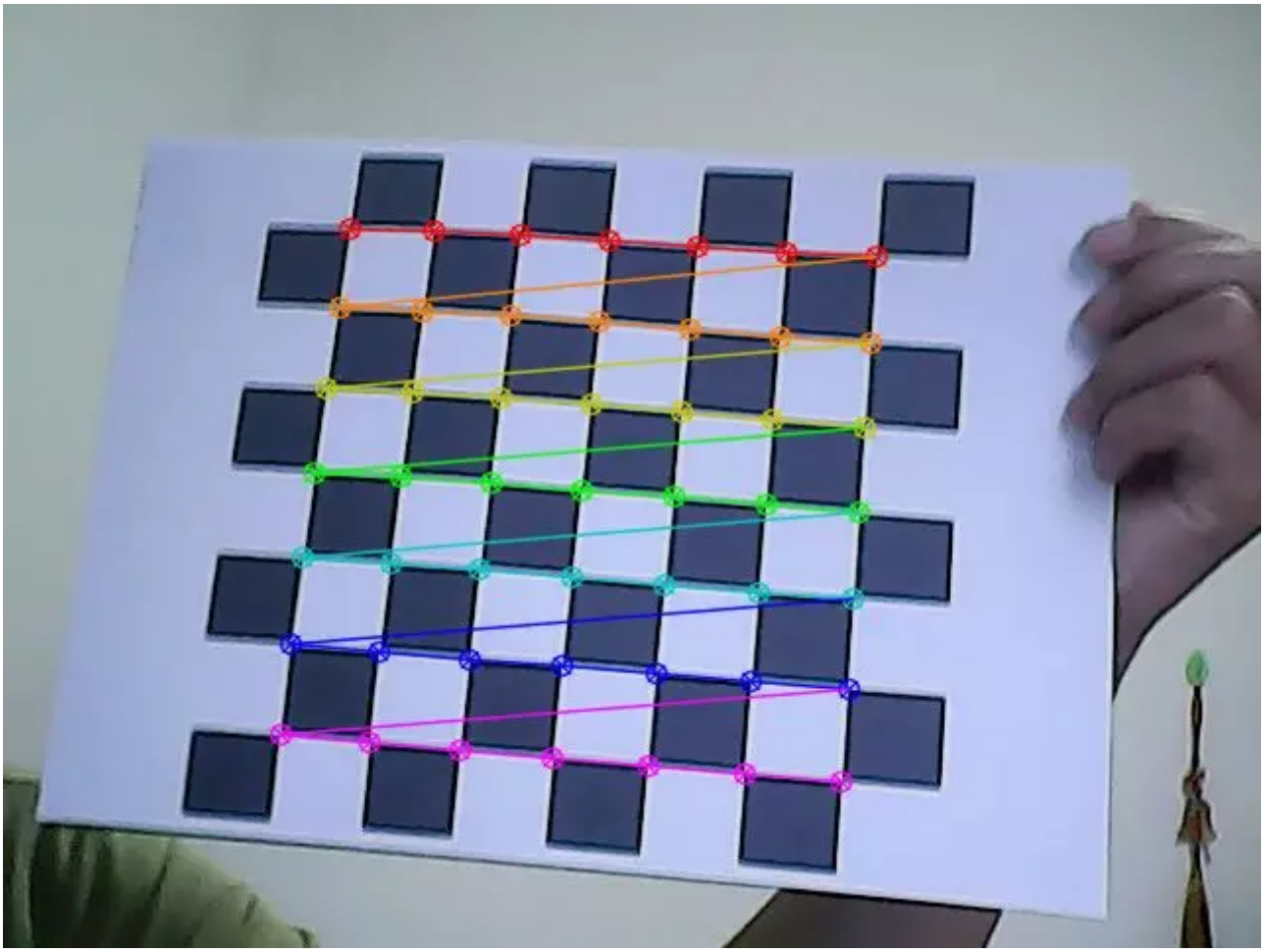
```
// load image files
vector<string> files;
glob("D:/images/camera2d", files);

// 定义变量
vector<vector<Point2f>> imagePoints;
```

```
vector<vector<Point3f>> objectPoints;
TermCriteria criteria = TermCriteria(TermCriteria::EPS + TermCriteria::MAX_ITER,
int numCornersHor = 7;
int numCornersVer = 7;
int numSquares = 50;
vector<Point3f> obj;
for (int i = 0; i < numCornersHor; i++)
    for (int j = 0; j < numCornersVer; j++)
        obj.push_back(Point3f((float)j * numSquares, (float)i * numSquares, 0));
```

发现与绘制棋盘格位置

```
// 发现棋盘格与绘制
Size s;
for (int i = 0; i < files.size(); i++) {
    printf("image file : %s \n", files[i].c_str());
    Mat image = imread(files[i]);
    s = image.size();
    Mat gray;
    cvtColor(image, gray, COLOR_BGR2GRAY);
    vector<Point2f> corners;
    bool ret = findChessboardCorners(gray, Size(7, 7), corners, CALIB_CB_ADAPTIV
    if (ret) {
        cornerSubPix(gray, corners, Size(11, 11), Size(-1, -1), criteria);
        drawChessboardCorners(image, Size(7, 7), corners, ret);
        imagePoints.push_back(corners);
        objectPoints.push_back(obj);
        imshow("calibration-demo", image);
        waitKey(500);
    }
}
```

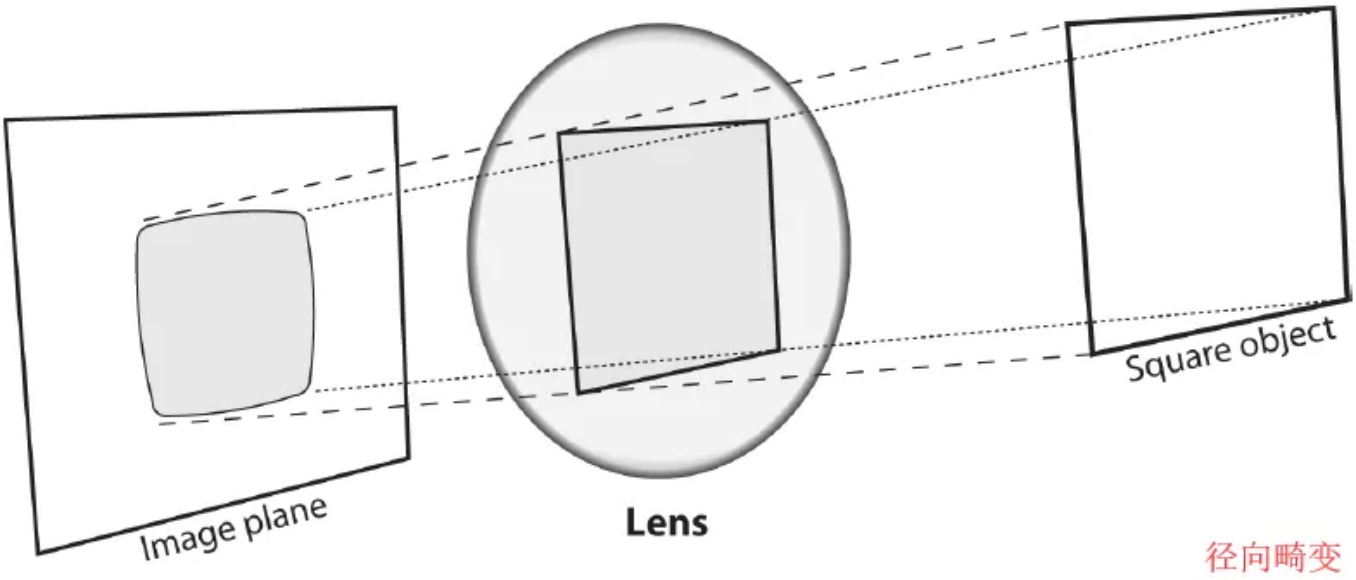
发现棋盘格显示如下（我是直接打印OpenCV自带那张图的）



相机校正-计算内参数

```
// 相机校正
Mat intrinsic = Mat(3, 3, CV_32FC1);
Mat distCoeffs;
vector<Mat> rvecs;
vector<Mat> tvecs;
intrinsic.ptr<float>(0)[0] = 1;
intrinsic.ptr<float>(1)[1] = 1;
calibrateCamera(objectPoints, imagePoints, s, intrinsic, distCoeffs, rvecs, tvec
```

05 畸变图像校正



关于畸变类型，常见的图像畸变类型有**径向与切向**畸变、OpenCV中的相机标定方法只能对径向畸变有效，使用内参对畸变图像实现校正。相关的代码如下：

```
// 畸变校正
for (int i = 0; i < files.size(); i++) {
    Mat dst;
    Mat image = imread(files[i]);
    undistort(image, dst, intrinsic, distCoeffs);
    imshow("image", image);
    imshow("undistort image", dst);
    waitKey(1000);
}
```