相机的畸变

首先我想给一些参考资料，这些资料实在太棒了，真的感谢这些分享者：

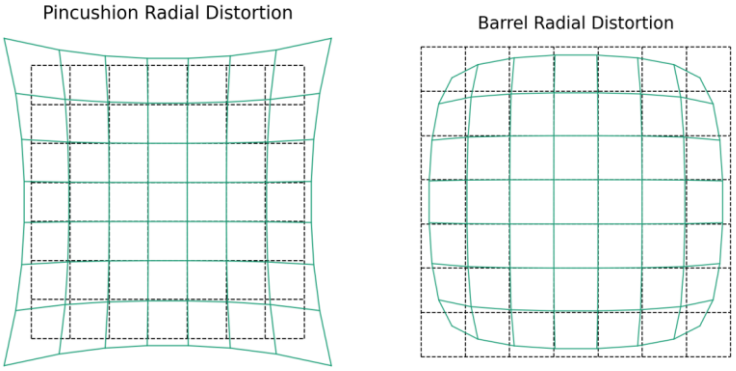
1. 对相机畸变特别好的解释，一共三篇，这是第一篇的链接：https://www.tangramvision.com/blog/camera-modeling-exploring-distortion-and-distortion-models-part-i
2. 这种精彩资料怎么能少得了 learnopencv 这个宝藏网站呢：https://learnopencv.com/understanding-lens-distortion/ 和 https://learnopencv.com/camera-calibration-using-opencv/
3. 一些中文的资料，毕竟是母语，可以快速了解：https://www.qinxing.xyz/posts/b7ea425d/ 和 https://zhuanlan.zhihu.com/p/94445955

一、 畸变的分类

之前找的很多资料都是直接给公式，云里雾里的，但是参考 1 就写的非常好，下面是个人总结。

1. Symmetric Radial Distortions：对称径向畸变

   最常讨论的畸变，很多资料也只写了这个，产生这个原因是因为镜头的工艺无法达到理想状态，这是不可避免的，具体解释没看，参考 3-1 有提到。一个枕形，一个桶形，直接看图：



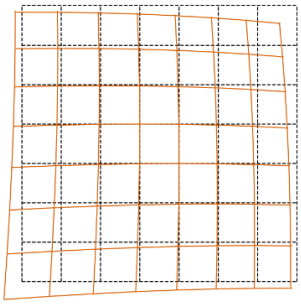
   能看出来，中心点很好，但是越边缘越差，这个畸变只和【到中心点的距离】有关系。

2. Asymmetric Radial Distortions：非对称径向畸变

   这个畸变不仅和【到中心点距离】有关，还和【物体离相机的距离】有关系。比如拍水下的两个物体，一个在远处，一个在近处，可能畸变程度就不同。这种畸变太难标定和测量了！所以通常不会考虑这种畸变，就当他不存在！参考 1 讲了一些该畸变，想细究可以去看。

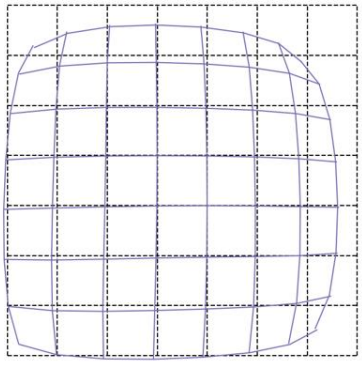
3. Tangential (De-centering) Distortions：切向畸变

   切向畸变通常是因为镜头和物体没有水平，可以看这个图：

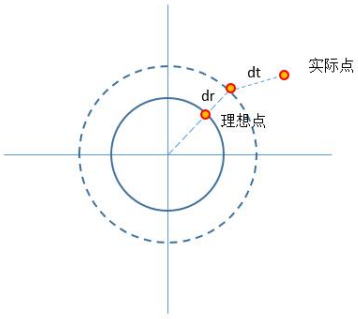
Tangential (Decentering) Distortion 	在用于机器人或自动车辆应用的大多数相机中，切向失真通常足够大，可以建模，但通常比【对称径向失真】小一个数量级。

综合上面的畸变起来，就是这个样子：

Compound Distortion



Dr：径向畸变　　Dt：切向畸变



二、 畸变的数学原理

有两种主要的失真模型已被广泛采用，以提供校正。虽然还有一些其他模型，但行业内在很大程度上已经将以下两种失真模型作为了标准化校正模型。下面规定一下符号：

1. (x,y)表示某个像素的坐标，中心点的坐标是(0,0)，有一些文献没说中心点坐标是(0,0)，是为了通用化，实际上没啥意思，就把中心点作为原点就行。参考 2 中给了这段话：

   > *c_x, c_y are the x and y coordinates of the optical center in the image plane. Using the center of the image is usually a good enough approximation.*

2. 径向畸变和切向畸变的变化，下标分别是 r 和 t。而且他们又各自有 x 和 y 方向的变化，所以是有四个变量。Delta_rx，delta_ry，delta_tx，delta_ty。

Brown-Conrady 标定模型

这个模型源于 Brown 和 Conrady 撰写的两篇文献。这些文件相当古老，可以追溯到一个世纪前，但今天仍然构成了许多关于表征和建模失真的思想的基础！

首先是径向畸变，公式如下。Kn 表示标定的系数，实际情况下通常只用 k1-k3，而有的时候镜头少的，k3 都可以不用。

$$r = \sqrt{x^2 + y^2}$$

$$\delta r = k_1 r^3 + k_2 r^5 + k_3 r^7 + \ldots + k_n r^{n+2}$$

这是整体的变化，又要分为 x 和 y 的分量，所以实际上为下图：

$$\delta x_r = \sin(\psi)\delta r = \frac{x}{r}(k_1 r^3 + k_2 r^5 + k_3 r^7)$$

$$\delta y_r = \cos(\psi)\delta r = \frac{y}{r}(k_1 r^3 + k_2 r^5 + k_3 r^7)$$

然后是切向畸变，公式如下，可以看到切向畸变并不完全靠【到中心距离】来决定的。一样的：p1 和 p2 是标定的系数。

$$\delta x_t = p_1(r^2 + 2x^2) + 2p_2 xy$$

$$\delta y_t = p_2(r^2 + 2y^2) + 2p_1 xy$$

Kannala-Brandt 标定模型

大约一个世纪后（2006 年），Juho Kannala 和 Sami Brandt 发表了他们自己的关于镜头失真的论文。本文的主要贡献是对广角、超广角和鱼眼镜头的镜头失真建模进行优化。Brown&Conrady 的模型在很大程度上建立在塞德尔像差的物理基础上，塞德尔像差最早是在 1867 年左右为当时的标准透镜物理制定的，其中不包括超宽透镜和鱼眼透镜。（偷懒，直接翻译的）

大多数人在使用该模型时会注意到的主要区别在于对称径向失真。Kannala Brandt 不是根据点离图像中心（半径）的距离来表征径向失真，而是将失真表征为穿过透镜的光的入射角的函数。这样做是因为当相对于该角度进行参数化时，失真函数更平滑。

后面就不细看了，如果不是超宽透镜和鱼眼透镜，上古的标定方法就够用了。参考 1 中 Part 2 讲了这个。

总结

最后的公式，即一个理想点变成正式点的公式，总结如下：

Fitting in with our previous camera model, we can formulate this in terms of the collinearity relationship (assuming Brown-Conrady distortions):

$$\begin{bmatrix} x_i \\ y_i \end{bmatrix} = \begin{bmatrix} f \cdot X_t / Z_t \\ f \cdot Y_t / Z_t \end{bmatrix} + \begin{bmatrix} a_1 x_i \\ 0 \end{bmatrix} + \begin{bmatrix} x_i(k_1 r^2 + k_2 r^4 + k_3 r^6) \\ y_i(k_1 r^2 + k_2 r^4 + k_3 r^6) \end{bmatrix} + \begin{bmatrix} p_1(r^2 + 2x_i^2) + 2p_2 x_i y_i \\ p_2(r^2 + 2y_i^2) + 2p_1 x_i y_i \end{bmatrix}$$

or, with Kannala Brandt distortions:

$$\begin{bmatrix} x_i \\ y_i \end{bmatrix} = \begin{bmatrix} f \cdot X_t / Z_t \\ f \cdot Y_t / Z_t \end{bmatrix} + \begin{bmatrix} a_1 x_i \\ 0 \end{bmatrix} + \begin{bmatrix} \frac{x_i}{r}(k_1 \theta + k_2 \theta^3 + k_3 \theta^5 + k_4 \theta^7) \\ \frac{y_i}{r}(k_1 \theta + k_2 \theta^3 + k_3 \theta^5 + k_4 \theta^7) \end{bmatrix}$$

只看第一个公式，这是 Brown-Conrady 标定：

1. 第一项表示理想的坐标，里面的 f 啥的不要管了，反正他是现实三维转到图像二维的方式。
2. 第二项有点意思，是指轴之间的倾斜，来自参考 2 中的第二篇文章，引用：a1 is the skew between the axes. It is usually 0.
3. 第三项就是径向畸变，第四项就是切向畸变。

可以去看 OpenCV 的文章：https://docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html

三、 畸变的标定（Opencv）

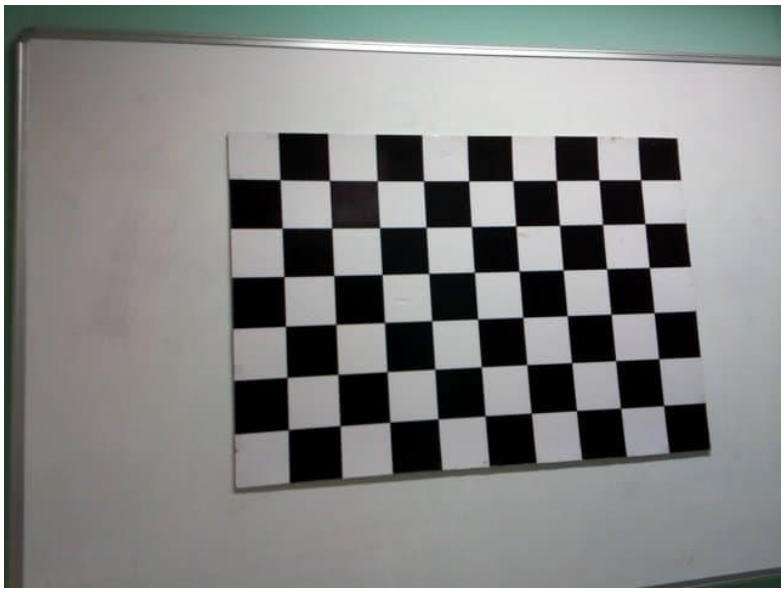
有了这样的公式，下面就可以标定参数了，也就是计算出 k1-k3、p1-p2 的值。实际上标定有两种方式：前向标定和后向标定。先来讲一下前向标定，Opencv 中使用的标定就是这种标定方式。

前向标定：对现实世界的三维坐标进行建模，让模型的输出和我们拍摄图片结果很接近。即：畸变=f(理想)。

下面的步骤多数**来自参考 2 的第二篇文章，里面也有代码，非常好**：

1. 世界坐标系的建立，不同角度拍图

通常用棋盘图进行标定（因为他们的拐角太容易辨别了）。原点可以选择板子的任意角，而 Z 轴是垂直板子的方向，所以棋盘上的点肯定 Z=0。此时每个点都有一个三维坐标，不管什么角度拍，每个点都是固定的，只是相机的三维坐标不同而已。



2. 计算 2D 图片中各个拐角的坐标：找出棋盘图的拐角、Refine 棋盘图拐角。注意这里的二维坐标，对于某个特定点，他在不同图片下的二维坐标是不一样的。

3. 根据二维坐标和三维坐标，标定相机。原理是基于一篇论文，很明显是数学上的，就不折腾了。作者是中国人张正友，非常牛逼的一个人，21 年是腾讯最高的级别研究员。他提出的方法叫做张氏标定法，想细看可以去网上搜索，很多资料，影响力相当卓著，截止到 23 年 12 月，引用量为恐怖的 19475!!

4. 消除畸变。得到一些矩阵，就是标定的结果。所以我们得到前向的转换方式，就是三维坐标乘以一些矩阵，最后得到了实际的成像点（畸变后的点）。但是我们现在有实际成像点，想要转为真实的成像点呀。这也是数学上的计算呗，参考 2 的第一篇文章写了如何用 Opencv 调用 API 计算。（直接看最后面部分）

四、 畸变的前向标定和后向标定

在第三节中提到标定有两种方式：前向标定和后向标定。

前向标定：对现实世界的三维坐标进行建模，让模型的输出和我们拍摄图片结果很接近。即：畸变=f(理想)。Opencv 就是这种方式，**不好地方在于我得到转换矩阵后，计算起来还是挺麻烦的。**即下图中我算出了 k 和 p，我的已知量是畸变后的坐标 xi 和 yi，我需要真实的坐标 Xc 和 Yc，计算起来是挺麻烦的。（上一节中我们用 Opencv 调用的，省事是因为 Opencv 帮你做了计算）

We'll use this term to model the effects of radial and tangential distortion on our light rays, and solve for where those distorted rays intersect the image plane:

$$\begin{bmatrix} x_i \\ y_i \end{bmatrix} = f \cdot \begin{bmatrix} X_c/Z_c \\ Y_c/Z_c \end{bmatrix} + \begin{bmatrix} c_x \\ c_y \end{bmatrix} + \begin{bmatrix} X_c \left(k_1 r_f^2 + k_2 r_f^4 + k_3 r_f^6 \right) \\ Y_c \left(k_1 r_f^2 + k_2 r_f^4 + k_3 r_f^6 \right) \end{bmatrix} + \begin{bmatrix} p_1 \left(r_f^2 + 2X_c^2 \right) + 2p_2 X_c Y_c \\ p_2 \left(r_f^2 + 2Y_c^2 \right) + 2p_1 X_c Y_c \end{bmatrix}$$

后向标定：对畸变的二维坐标进行建模，模型转换后得到真实的二维坐标。他的好处很显然就是前向标定的坏处嘛，方便。只要标定好系数之后，我有 xi 和 yi，可以立马得出 Xc 和 Yc 了。

$$r_i = \sqrt{x_i^2 + y_i^2}$$

Using this r_i term, we can now model the ***correction*** of our distorted points.

$$\begin{bmatrix} x_i \\ y_i \end{bmatrix} = f \cdot \begin{bmatrix} X_c/Z_c \\ Y_c/Z_c \end{bmatrix} + \begin{bmatrix} x_i \left(k_1 r_i^2 + k_2 r_i^4 + k_3 r_i^6 \right) \\ y_i \left(k_1 r_i^2 + k_2 r_i^4 + k_3 r_i^6 \right) \end{bmatrix} + \begin{bmatrix} p_1 \left(r_i^2 + 2x_i^2 \right) + 2p_2 x_i y_i \\ p_2 \left(r_i^2 + 2y_i^2 \right) + 2p_1 x_i y_i \end{bmatrix}$$

The inverse model therefore places our image-space coordinates on *both* sides of the equation!

Optimizing across a non-parametric equation like the inverse model is more difficult than optimizing a forward model; some of the cost functions are more difficult to implement. The benefit to the inverse approach is that, once calibrated, one could formulate the problem as follows:

$$\begin{bmatrix} x_i \\ y_i \end{bmatrix} - \begin{bmatrix} c_x \\ c_y \end{bmatrix} - \begin{bmatrix} u_i \left(k_1 r_i^2 + k_2 r_i^4 + k_3 r_i^6 \right) \\ y_i \left(k_1 r_i^2 + k_2 r_i^4 + k_3 r_i^6 \right) \end{bmatrix} - \begin{bmatrix} p_1 \left(r_i^2 + 2u_i^2 \right) + 2p_2 u_i v_i \\ p_2 \left(r_i^2 + 2v_i^2 \right) + 2p_1 u_i v_i \end{bmatrix} = f \cdot \begin{bmatrix} X_c/Z_c \\ Y_c/Z_c \end{bmatrix}$$