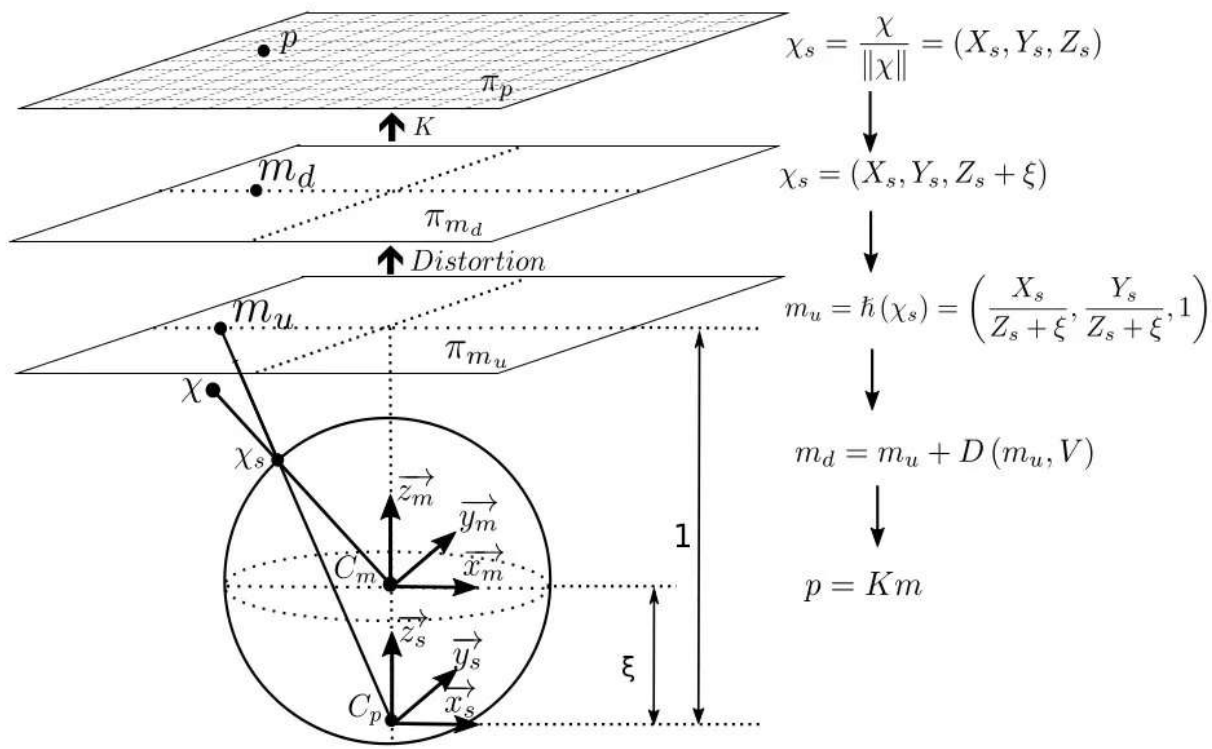


代码解读 | VINS_Mono中的鱼眼相机模型

前一篇文章《代码解读 | VINS 视觉前端》介绍了VINS前端。本文来说说鱼眼相机模型。
VINS_Mono代码支持的相机包括针孔模型和鱼眼模型相机，针孔模型大家都比较熟悉了，今天向大家介绍一种鱼眼相机模型——MEI模型。

相机模型

相比针孔模型可以将3d点直接投影到归一化平面，鱼眼相机则多了一个中间过程：先将3d点投影到单位球面，再将单位球面上的点投影到归一化平面上。废话不多说，请看鱼眼相机投影模型示意图：



1. 将以 C_m 为原点的相机坐标系下3D点 \mathcal{X} 转换到单位球面上：

$$(\mathcal{X})_{C_m} \rightarrow (\mathcal{X}_s)_{C_m} = \frac{\mathcal{X}}{\|\mathcal{X}\|} = (X_s, Y_s, Z_s)^T \quad (1)$$

2. 然后，将球面上 \mathcal{X}_s 变换到以 $C_p = (0, 0, \xi)$ 为中心的坐标系，

$$(\mathcal{X}_s)_{C_m} \rightarrow (\mathcal{X}_s)_{C_p} = (X_s, Y_s, Z_s + \xi)^T \quad (2)$$

3. 接着，将 $(\mathcal{X}_s)_{C_p}$ 投影到归一化平面：

$$\mathbf{m}_u = \left(\frac{X_s}{Z_s + \xi}, \frac{Y_s}{Z_s + \xi}, 1 \right)^T = h(\mathcal{X}_s) \quad (3)$$

4. 考虑到投影过程中存在的径向畸变和切向畸变：

$$\begin{aligned} x_{distorted} &= x + k_1 \rho^2 + k_2 \rho^4 + k_3 \rho^6 + 2p_1 xy + p_2 (\rho^2 + 2x^2) \\ y_{distorted} &= y + k_1 \rho^2 + k_2 \rho^4 + k_3 \rho^6 + p_1 (\rho^2 + 2y^2) + 2p_2 xy \end{aligned} \quad (4)$$

加上畸变：

$$\mathbf{m}_d = \mathbf{m}_u + D(\mathbf{m}_u, V) \quad (5)$$

其中 $\rho = \sqrt{x^2 + y^2}$, $V = (k_1, k_2, k_3, p_1, p_2)$ 。


5. 将包含畸变后的点利用内参矩阵 K 将 \mathbf{m} 变幻到像素平面：

$$\mathbf{p} = K\mathbf{m}_d = \begin{bmatrix} f_x & \alpha & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{m}_d \quad (5)$$

其中， f_x, f_y 为相机在 x, y 轴方向上的焦距，一般大致相等， u_0, v_0 为主点坐标（相对于成像平面）， α 为坐标轴倾斜参数，理想情况下为0。

由以上分析，通过使用投影模型，可以通过以下等式将归一化相机平面上的点投影到到单位球面：

$$h^{-1}(\mathbf{m}_u) = \begin{bmatrix} \frac{\xi + \sqrt{1 + (1 - \xi^2)(x^2 + y^2)}}{x^2 + y^2 + 1} x \\ \frac{\xi + \sqrt{1 + (1 - \xi^2)(x^2 + y^2)}}{x^2 + y^2 + 1} y \\ \frac{\xi + \sqrt{1 + (1 - \xi^2)(x^2 + y^2)}}{x^2 + y^2 + 1} - \xi \end{bmatrix} \quad (7)$$

 计算机视觉life

代码解读

VINS_Mono中相机模型对应代码在

/VINS-Mono/camera_model/src/camera_models/CataCamera.cc

文件**liftSphere()**函数中，该函数是将2d投影到3d点（单位球面上），首先对2d畸变，然后再投影到单位球面上。

去畸变过程代码如下：

//去畸变过程

```
int n = 6;
Eigen::Vector2d d_u;
distortion(Eigen::Vector2d(mx_d, my_d), d_u); //得到畸变量
// Approximate value
mx_u = mx_d - d_u(0);
my_u = my_d - d_u(1);

for (int i = 1; i < n; ++i) //循环去畸变多次, 使结果更接近真值
{
    distortion(Eigen::Vector2d(mx_u, my_u), d_u);
    mx_u = mx_d - d_u(0);
    my_u = my_d - d_u(1);
}
```

将去畸变后的 2d 点投影到归一化球面代码如下:

```
double xi = mParameters.xi();
if (xi == 1.0)
{
    lambda = 2.0 / (mx_u * mx_u + my_u * my_u + 1.0);
    P << lambda * mx_u, lambda * my_u, lambda - 1.0;
}
else
{
    lambda = (xi + sqrt(1.0 + (1.0 - xi * xi) * (mx_u * mx_u + my_u * my_u))) / (1.0 + xi);
    P << lambda * mx_u, lambda * my_u, lambda - xi;
}
```

其中xi对应公式 (7) 中的 ξ 。

reference

1. Mei, C. and P. Rives. Single view point omnidirectional camera calibration from planar grids. in Robotics and Automation, 2007 IEEE International Conference on. 2007. IEEE.
2. Jamaluddin A Z , Mazhar O , Morel O , et al. Design and calibration of an omni-RGB+D camera. International Conference on Ubiquitous Robots & Ambient Intelligence. IEEE, 2016.
3. Camera Calibration

