单目相机的坐标变换与 PnP 解算

Lisii

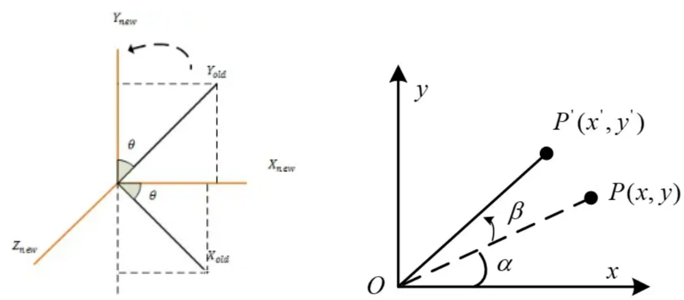
December 27, 2023

1 坐标变换

1.1 世界坐标系到相机坐标系

相机在世界中的坐标可以通过用一定的旋转矩阵 R 和平移矩阵 T 表示成世界坐标。如果将一个相机放在世界坐标系内，相机坐标相对于世界坐标系的原点的平移可以记为向量 t，旋转的角度记为 R。

对于平移向量 t，我们需要用它来表示平移，任意一个平移包括三个方向，因此可以用一个三维向量 (x, y, z) 表示；对于旋转矩阵 R，分别按照 x, y, z 三个不同的轴进行旋转:



(a) 沿着 z 轴旋转　　(b) 二维旋转

Figure 1: 旋转矩阵

如果沿着 z 轴旋转，那么三维坐标点的 z 值不变，唯一改变的是 x 和 y 方向的值，相当于二维平面坐标下的旋转

由于 $|OP| = |OP'|$

$x' = |OP'| \cdot cos(\alpha + \beta) = |OP|cos\alpha \cdot cos\beta - |OP|sin\alpha \cdot sin\beta = x \cdot cos\beta - y \cdot sin\beta$

$y' = |OP'| \cdot sin(\alpha + \beta) = |OP|cos\alpha \cdot sin\beta - |OP|sin\alpha \cdot cos\beta = x \cdot sin\beta + y \cdot cos\beta$

用矩阵形式重新表示为:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} cos\beta & -sin\beta \\ sin\beta & cos\beta \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix}$$

加上 z:

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} cos\beta & -sin\beta & 0 \\ sin\beta & cos\beta & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

同理可得，沿着 x 轴和沿着 y 轴旋转的变化分别如下：

$$\begin{pmatrix} x_{cam} \\ y_{cam} \\ z_{cam} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos\beta & sin\beta \\ 0 & -sin\beta & cos\beta \end{pmatrix} \begin{pmatrix} x_{word} \\ y_{word} \\ z_{word} \end{pmatrix}$$

$$\begin{pmatrix} x_{cam} \\ y_{cam} \\ z_{cam} \end{pmatrix} = \begin{pmatrix} 0 & cos\gamma & -sin\gamma \\ 0 & 1 & 0 \\ 0 & sin\gamma & cos\gamma \end{pmatrix} \begin{pmatrix} x_{word} \\ y_{word} \\ z_{word} \end{pmatrix}$$

于是，新的相机坐标，即同时沿着 x, y, z 轴三个方向旋转的矩阵 R 如下:

$$R = \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos\beta & sin\beta \\ 0 & -sin\beta & cos\beta \end{pmatrix} \begin{pmatrix} 0 & cos\gamma & -sin\gamma \\ 0 & 1 & 0 \\ 0 & sin\gamma & cos\gamma \end{pmatrix} \begin{pmatrix} cos\alpha & sin\alpha & 0 \\ -sin\alpha & cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix}$$
$$= \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix}$$

$$T = \begin{pmatrix} T_x \\ T_y \\ T_z \end{pmatrix}$$

于是，世界坐标系到相机坐标系的转化可以写成： $\begin{pmatrix} x_c \\ y_c \\ z_c \end{pmatrix} = R \cdot \begin{pmatrix} x_w \\ y_w \\ z_w \end{pmatrix} + T$

1.2 图像坐标系到像素坐标系

普通相机的成像模型采用小孔成像，初中的物理知识告诉我们，物体经小孔后，在成像平面成倒立的像。为了更好的进行理论阐述，一般默认采用虚拟成像平面进行分析。

将小孔成像模型简化成几何表达的形式:

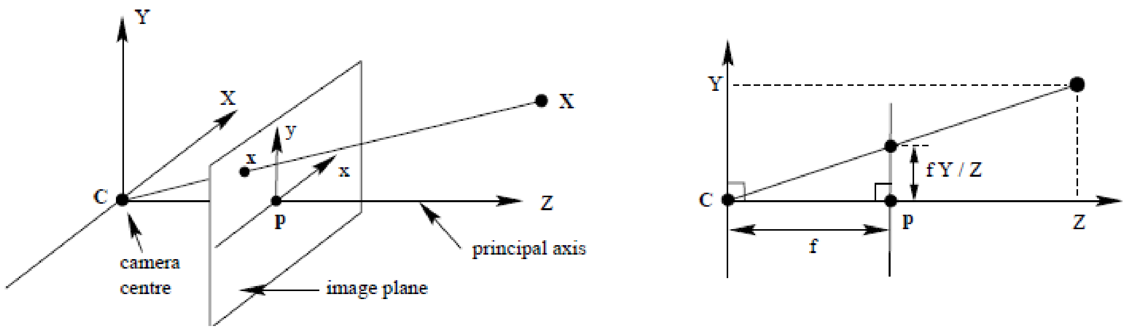


Figure 2: 小孔成像模型

根据简单的相似三角形几何知识，可以推出 3D 目标点在相机坐标系下的坐标与图像像素坐标之间的关系:

$\frac{X}{u} = \frac{Z_c}{P_c} \Rightarrow u = P_c \cdot \frac{X}{Z_c}$

$\frac{Y}{v} = \frac{Z_c}{P_c} \Rightarrow v = P_c \cdot \frac{Y}{Z_c}$

Z_c 是 ZC,P_c 是焦距 $f = PC$，(X, Y, Z) 是 3D 点在相机坐标系下坐标，(u, v) 是图像坐标

即：$\begin{cases} u = f \cdot \frac{X}{Z_c} \\ v = f \cdot \frac{Y}{Z_c} \end{cases}$

写成齐次坐标为：$Z_c \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$

(u_0, v_0) 为从图像坐标系中心点到像素坐标系中心点的一个偏移量。属于相机内参的一部分。另外，如果我们已知图像坐标系的一个点, 我们还应该知道，横坐标的每毫米对应像素是多少。

即有如下公式：$p(u, v) = p(\frac{x}{dx+u_0}, \frac{y}{dy+v_0})$

原先的式子就可以写成：

$$Z_c \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{1}{dx} & 0 & u_0 \\ 0 & \frac{1}{dy} & v_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{f}{dx} & 0 & u_0 & 0 \\ 0 & \frac{f}{dy} & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

$$= \begin{pmatrix} f_x & 0 & u_0 & 0 \\ 0 & f_y & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

1.3 世界坐标系到像素坐标系

这样我们就完成了从世界坐标系到相机像素坐标系的一个转换，像素坐标系和世界坐标系可以之间的关系可以表示为如下的式子：$Z_c \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} f_x & 0 & u_0 & 0 \\ 0 & f_y & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} R & t \\ 0^T & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$

其中 $K_1 = \begin{pmatrix} f_x & 0 & u_0 & 0 \\ 0 & f_y & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$ 为相机的内参，$K_2 = \begin{pmatrix} R & t \\ 0^T & 1 \end{pmatrix}$ 为相机的外参

即：$Z_c \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = K_1 \cdot K_2 \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$

1.4 像素坐标系到世界坐标系

即世界坐标系到像素坐标系的逆变换：$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = Z_c \cdot K_2^{-1} \cdot K_1^{-1} \cdot \begin{pmatrix} u \\ v \\ 1 \end{pmatrix}$

2 PnP 解算

我们已经了解了相机的坐标变换的公式，可以先通过相机标定求出相机的内参矩阵 K_1，Matlab 和 OpenCV 都提供了相关功能进行相机标定。若要根据已知条件求解相机坐标系相对于世界坐标系的位姿，就需要用到 PnP 解算：

$PnP(Perspective-n-Point)$ 是求解 3D 到 2D 点对运动的方法，目的是求解相机坐标系相对世界坐标系的位姿。它描述了已知 n 个 3D 点的坐标 (相对世界坐标系) 以及这些点的像素坐标时，如何估计相机的位姿 (即求解世界坐标系到相机坐标系的旋转矩阵和平移向量)。

2.1 OpenCV-PnPSolver

OpenCV 提供了 PnP 问题的解算函数，且包含有多种解法。有以下两个函数:

- **solvePnP**

```
bool solvePnP( InputArray objectPoints , InputArray imagePoints,
          InputArray cameraMatrix, InputArray distCoeffs ,
          OutputArray rvec, OutputArray tvec,
          bool useExtrinsicGuess = false , int flags = SOLVEPNP_ITERATIVE );
```

$objectPoints$：世界坐标系（O_w, X_w, Y_w, Z_w）下的 3D 点坐标数组

$imagePoints$：图像（o, u, v）中对应 3D 点的成像点坐标数组

$cameraMatrix$：相机内参矩阵，3×3

$distCoeffs$：相机畸变系数数组，可以为 $NULL$，此时视为无畸变。

$rvec$ 和 $tvec$：计算结果输出，$rvec$ 为旋转向量，$tvec$ 为平移向量，两者合并表达的是物体整体（即世界坐标系）在相机坐标系中的位姿

$useExtrinsicGuess$ 参数为可选: 这个参数仅用于当 $flags = SOLVEPNP_ITERATIVE$，此值如果为 true，需要 $rvec$ 和 $tvec$ 有输入值，以便函数把输入值作为旋转和平移的估计初始值.

- **solvePnPRansac**

```
bool solvePnPRansac( InputArray objectPoints , InputArray imagePoints,
          InputArray cameraMatrix, InputArray distCoeffs ,
          OutputArray rvec, OutputArray tvec,
          bool useExtrinsicGuess = false , int iterationsCount = 100,
          float reprojectionError = 8.0, double confidence = 0.99,
          OutputArray inliers = noArray(), int flags = SOLVEPNP_ITERATIVE );
```

与 solvePnP 功能相同，但这个函数使用 RANSAC 算法剔除异常样本。

使用 solvePnP 前，需要已具备如下参数：

```
vector<Point3f>objPts; //3D点数组，世界坐标系物体点坐标，至少4个点
vector<Point2f>imgPts; //2D点数组，与以上物体点一一对应的图像点坐标
Mat cameraMatrix;      //相机内参矩阵，3x3矩阵
Mat distCoeff;  //相机畸变系数矩阵，一般用1x5矩阵，如果相机没有畸变，可以把所有元素置为0
```

然后调用

```
Mat rvec, tvec; //声明用于接收运算结果的两个矢量
solvePnP(objPts, imgPts, cameraMatrix, distCoeff , rvec, tvec);
```

得到解算结果后，$rvec$ 为旋转矢量形式，后续计算不便，所以一般会用 Rodrigues 公式转为旋转矩阵，以下直接将 $rvec$ 和 $tvec$ 一起转为位姿矩阵

```
Mat wldToCam = Mat::zeros(4, 4, CV_64FC1);
Rodrigues(rvec, wldToCam(Rect(0, 0, 3, 3)));
tvec.copyTo(wldToCam(Rect(0, 3, 1, 3)));
```

以上得到的 $wldToCam$ 即为世界坐标系在相机坐标系中的位姿，如果需要求相机在世界坐标系中的位姿，可取逆：

```
Mat camToWld = wldToCam.inv();
```

3 深度信息的获取

由于从世界坐标至相机画面的变换过程中，丢失了深度信息，若想通过相机画面坐标还原物体的世界三维坐标，除了通过相机标定获取相机内参矩阵 K_1。PnP 解算出相机外参矩阵 K_2 外，还需要提供物体距离相机的深度信息 Z_c。我们可以通过多种方式获取深度信息，例如双目相机、激光雷达等。