激光雷达与摄像头融合

笔者：纸壳小宝

1. 摄像头和单线激光雷达的联合标定原理

理论上激光雷达点要先从世界坐标系经过旋转、平移、缩放等变换转换到相机坐标，再乘以相机内参矩阵转换到图像坐标。但其实二维激光雷达坐标到图像坐标的变换可以更简洁地看成是一个射影变换，二维激光的激光点就相当于俯视图，摄像头拍到的图像可以想象为主视图，而俯视图通过乘以一个单应矩阵变换到主视图的过程就是射影变换。转换的一般形式如下：

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} * \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{1.1}$$

其中[x y 1]是激光雷达的齐次坐标，如图 2.1 左，[u v 1]是激光转换到图像坐标系后的图像齐次坐标，我们提出 h 矩阵中的 h_9 并放到等式左边作为缩放因子 λ 后，可将公式写成如下形式：

$$\lambda * \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} n_1 & n_2 & n_3 \\ n_4 & n_5 & n_6 \\ n_7 & n_8 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{1.2}$$

上式的 n 矩阵就是我们需要求出来的变换矩阵参数，怎么求，这里需要消去缩放因子$\lambda=n_7 * x + n_8 * y + 1$. 便可得到：

$$\begin{cases} (n_7x + n_8y + 1)u = n_1x + n_2y + n_3 \\ (n_7x + n_8y + 1)v = n_4x + n_5y + n_6 \end{cases} \Rightarrow \begin{cases} u = n_1x + n_2y + n_3 - n_7ux - n_8uy \\ v = n_4x + n_5y + n_6 - n_7vx - n_8vy \end{cases} \tag{1.3}$$

假设收集到了 k 组对应点，可将上述方程写成矩阵的形式

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -u_1x_1 & -u_1y_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -v_1x_1 & -v_1y_1 \\ & & & & & \cdots & & \\ & & & & & \cdots & & \\ x_k & y_k & 1 & 0 & 0 & 0 & -u_kx_k & -u_ky_k \\ 0 & 0 & 0 & x_k & y_k & 1 & -v_kx_k & -v_ky_k \end{bmatrix} * \begin{bmatrix} n_1 \\ n_2 \\ n_3 \\ n_4 \\ n_5 \\ n_6 \\ n_7 \\ n_8 \end{bmatrix} = \begin{bmatrix} u_1 \\ v_1 \\ \cdots \\ \cdots \\ u_k \\ v_k \end{bmatrix} \tag{1.4}$$

2. 数据采集方法

从(1.4)式中可以看出，一组数据对应两个 n 参数，那么要求出 n 向量至少需要 4 组数据，组成超定方程求最优解至少需要 5 组数据，超定方程的求解就是最小二乘了，稍后我会给出 matlab 代码，这里先说一下我的数据采集方法。

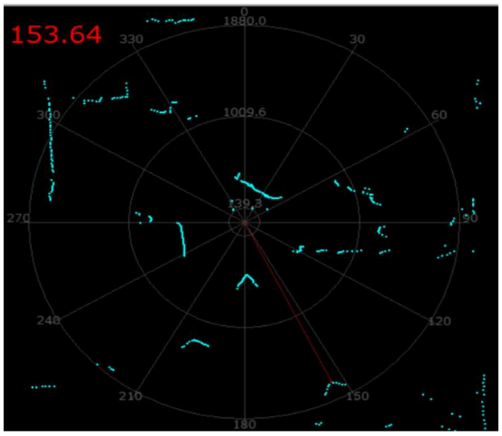

图 2.1：数据采集方法

如上图所示，我把相同的几张白纸对折，并在与激光雷达等高的位置画上了黑点，调用 opencv 程序可以很快得到黑点的图像坐标（即 u v 的值），程序在 calibration 文件夹中（在 linux 或者 windows 的 vs 中运行 take_photo，按 q 可以截取摄像头画面，运行 mouse 读取之前截取的画面，它可以显示鼠标左击所在坐标）。再看左边激光雷达的图，图中下半部分那三个三角是不是很明显，这就是我对折的原因，对折顶点位置就是激光雷达点的坐标（即 x y 的值），我使用的激光雷达是 rplidarA1，图 2.1 左是 rplidar 的 frame_grabber 软件（rplidar sdk 文件夹中，记得装驱动 cp210），它会显示鼠标所在激光点的距离和角度，通过距离和角度就可以算出 x 和 y。我用这样的方式总共收集了 6 组点来求解向量 n，收集时尽量让点的位置交错开，不要三点共线。

3. Matlab代码求向量n

最后在 Matlab 中求解向量 n，假设采集了 6 组，代码如下

```
x=[x_1 x_2 … x_6];                          % 雷达坐标 x 轴
y=[y_1 y_2 … y_6];                          % 雷达坐标 y 轴
u=[u_1 u_2 … u_6];                          % 图像坐标 u 轴
v=[v_1 v_2 … v_6];                          % 图像坐标 v 轴
z=[u_1 v_1 u_2 v_2 … u_6 v_6]';             % 所有 u v 依次排下去，即（1.4）式等号右边的向量
% X 为公式（1.4）中的左边第一个矩阵
X=[x(1) y(1) 1 0 0 0 -u(1).*x(1) -u(1).*y(1) ;
      0 0 0 x(1) y(1) 1 -v(1).*x(1) -v(1).*y(1) ;
      x(2) y(2) 1 0 0 0 -u(2).*x(2) -u(2).*y(2) ;
      0 0 0 x(2) y(2) 1 -v(2).*x(2) -v(2).*y(2) ;
      x(3) y(3) 1 0 0 0 -u(3).*x(3) -u(3).*y(3) ;
      0 0 0 x(3) y(3) 1 -v(3).*x(3) -v(3).*y(3) ;
      x(4) y(4) 1 0 0 0 -u(4).*x(4) -u(4).*y(4) ;
```

```
    0 0 0 x(4) y(4) 1 -v(4).*x(4) -v(4).*y(4) ;
    x(5) y(5) 1 0 0 0 -u(5).*x(5) -u(5).*y(5) ;
    0 0 0 x(5) y(5) 1 -v(5).*x(5) -v(5).*y(5) ;
    x(6) y(6) 1 0 0 0 -u(6).*x(6) -u(6).*y(6) ;
    0 0 0 x(6) y(6) 1 -v(6).*x(6) -v(6).*y(6) ];
n=X\z;                  % 向量 n = X^(-1) * z   即 X 矩阵的逆乘以 z 向量
```

4. 坐标转换的程序

求得向量 n 后，就可以在自己的程序里对所有点进行转换了，代码在 lidar_camera_fusion 包中的 opencv_lidar.cpp，将求出的向量 n 放入数组 a[]中，要先启动你的激光雷达 ros 节点发布/scan 话题，再运行 rosrun lidar_camera_fusion fusion，需要一些 ros 知识，效果图如下所示：