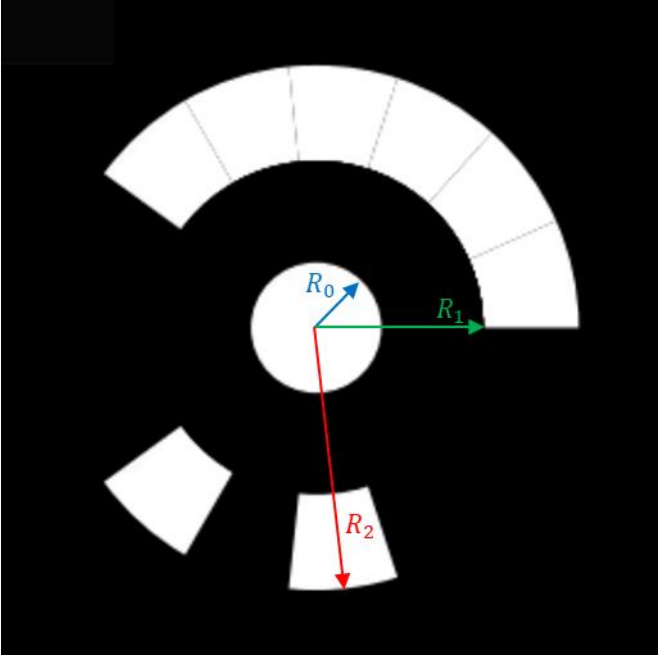


一、点云拼接基本原理

目前算法支持使用编码圆标志物做点云拼接。
基本原理是在 **2D** 图上找到相同 **id** 的编码圆的圆心作为相同的特征点，再去点云上获取两组对应的 **3d** 点，然后算得旋转平移矩阵，就可以实现点云拼接。

二、编码圆形状

中间是白色的圆，四周环带上带有编码标记。
环带等分为 N 份，每份作为一个二进制位，可以取 **0**（对应黑色）或者 **1**（对应白色）。
编码的规则是：将对应的 **0** 和 **1** 记录成 N 位的二进制，对 N 位二进制进行循环右移操作，取其对应十进制数最小的数作为编码的唯一码值。
注意：这里为了方便说明，所以环带上带有黑色的分隔线，实际环带不应该带有这些黑色的分割线。



编码圆有不同的规格型号， R_0 、 R_1 、 R_2 、 N 可以是不同的值，因此编码圆检测需要输入该型号下 $\frac{R_1}{R_0}$ ， $\frac{R_2}{R_0}$ ， N 三个已知变量才可得到正确的编码值。

- 为了提高识别精度和准确率，应该做到：
- 1、编码圆最里面的小圆半径至少有 4 个像素；
 - 2、打印的编码圆边界印刷要清晰，圆弧标准无锯齿现象；
 - 3、编码圆应完整，无破损和折痕；
 - 4、编码圆应尽量整个贴平，没有弯曲；
 - 5、拍摄的编码圆应避免过曝、过暗或倾斜角度过大；
 - 6、拍摄时相机和编码圆应保持相对静置，不应有运动模糊。

编码圆图案可以自行打印，见：[Examples/Python/Utils/GenerateCodedCircle.py](#)。
如果自行打印编码圆质量较差，也可以网上购买，一般专业打印店打印的圆边缘清晰锐利，质量会远好过自行打印的编码圆，从而识别精度更高。

三、拼接流程

以拼接三帧点云为例进行说明，更多帧点云的拼接流程参照三帧点云的拼接流程。

- 1、
粘贴好编码圆，保证相邻两帧之间能共同看到 3 个以上的编码圆图案（编码圆越多，精度和稳定性越高），所有编码圆的圆心不能落在同一条直线上，编码圆分布越广，效果越好；
- 2、
拍照获得 PointMap1 和 Image1；
- 3、
移动相机拍照获得 PointMap2 和 Image2；
- 4、
移动相机拍照获得 PointMap3 和 Image3；
- 5、
根据基准坐标系的不同，可以选择不同的拼接方法：

5.1
如果以第一帧的点云坐标系为基准，那么就先将 PointMap1、Image1、PointMap2 和 Image2 依次传入 **GetTwoCameraTransformByCodedCircleMarker()**函数作为前四个参数，得到第二帧点云坐标系关于第一帧点云坐标系的旋转矩阵 **R1** 和平移向量 **t1**，再将 **R1**、**t1** 和 **PointMap2** 传入 **TransformPointCloud()**将第二帧点云转移到第一帧点云坐标系下。
然后将 **PointMap2**、**Image2**、**PointMap3** 和 **Image3** 依次传入 **GetTwoCameraTransformByCodedCircleMarker()**函数作为前四个参数，得到第三帧点云坐标系关于第二帧点云坐标系的旋转矩阵 **R2** 和平移向量 **t2**，再将 **R2**、**t2** 和 **PointMap3** 传入 **TransformPointCloud()**将第三帧点云转移到第二帧点云坐标系下，由于此前已经将第二帧点云转移到第一帧点云坐标系，所以第二帧点云坐标系等同于第一帧点云坐标系，即三帧点云都在第一帧点云坐标系下。

5.2

如果以最后一帧也就是第三帧点云坐标系为基准，那么可以参照 5.1 的方法，区别仅仅在于传入 `GetTwoCameraTransformByCodedCircleMarker()`函数时，第 1、2 个参数需要和第 3、4 个参数对调一下。
6、
拼接完成后，可以查看拼接后的所有点云，观察是否拼接完整。

流程中需要特别注意，拍照获得的 2D 图要能清晰看到编码圆，中间圆的半径至少要有 4 个像素，圆心附近的点云不能有缺失。

示例程序：`PointCloudStitchingExample.cpp` / `PointCloudStitchingExample.py`。