ICP点云匹配

张嘉浩（3190103683） 朱姜宇轩（3190103052） 蒋颜丞（3190102563）

1 解决思路

1.1 实验原理

主要使用点到点的ICP里程估计 $(Point-Point\ ICP)$ 实现点云匹配，具体算法如下：

Point-Point ICP (Iterative Closest Point)

估计P'集合点与P集合点的初始位姿关系
根据最近领域规则建立P'集合点与P集合点的关联
利用线性代数/非线性优化的方式估计旋转平移量　　# 可使用使用SVD分解，具体算法如下所示
对点集合P'的点进行旋转平移
如果旋转平移后重新关联的均方差小于阈值，则结束
否则迭代重复上述步骤

SVD分解

输入：点集合$P=\{p_1,...p_n\}$
　　点集合$P'=\{p'_1,...p'_n\}$
$p=\frac{1}{n}\sum_n^{i=1}p_i,\ p'=\frac{1}{n}\sum_n^{i=1}$　　# 定义两组集合的质心位置p, p'
$q_i=p_i-p,\ q'_i=p'_i-p'$　　# 计算每个点的去质心坐标
$W=\sum_{i=1}^{n}q'_iq_i^T=USV^T$　　# SVD分解求得$V,\ U$
$\Longrightarrow R=VU^T$
$\Longrightarrow t=p-Rp'$

1.2 实验改进

我们对算法做了如下改进：

（1）将欧氏距离搜索变为KD-Tree搜索，加快搜索速度：

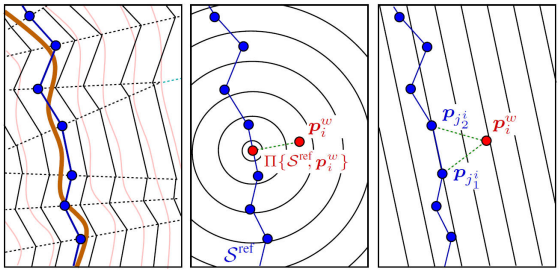
```
# KD-Tree 寻找最近点
kd_tree=KDTreeSearcher(p,'BucketSize',10);
[min_index, ~] = knnsearch(kd_tree, pp,'K',k);
```

（2）在对代码进行调试的过程中我们发现，对点云进行最近邻匹配后，仍有对应点之间的距离很大，这会给位姿的计算带来误差。文献[2]指出，剔除一些误差过大的点（离群点）有助于提高ICP的精度，相关指标主要有距离、曲率、法向量等，考虑到本例中的点云多为直线，因此采用两点之间的距离和法向量夹角作为指标来判断离群点，如若以上指标超过阈值，则会将对应的点剔除。

法向量计算

找到点p_i周围半径R范围内的所有点V_i
求解均值和协方差：$\mu_i=\frac{1}{|V_i|}\sum_{p_j\in V_i}p_i,\Sigma_i=\frac{1}{|V_i|}\sum_{p_j\in V_i}(p_i-\mu_i)^T(p_i-\mu_i)$
对协方差矩阵进行SVD分解，得到3个特征值$\lambda_1,\lambda_2,\lambda_3$：$\Sigma_i=R\begin{pmatrix}\lambda_1&0&0\\0&\lambda_2&0\\0&0&\lambda_3\end{pmatrix}R^T$
最小特征值λ_1对应的特征向量即为法向量

（3）PPICP存在一个明显的缺陷：由于机器人的移动，两帧激光点云数据中的点基本不可能表示的是空间中的相同位置，所以用点到点的距离作为误差方程势必会引入随机误差。文献[1]提出将ICP中的误差方程由点到点 $(Point-Point)$ 的距离改进为点到线 $(Point-Line)$ 的距离。



(a) Distance to curve and to polyline (b) Point-to-point metric (c) Point-to-line metric

Fig. 1. Near convergence, the point-to-line metric approximates the distance to the surface better than the point-to-point metric used in vanilla ICP.

具体的算法如下：

Point-Line ICP (Iterative Closest Point)

给定一个初始的转换矩阵q_0，将当前激光帧的数据转换到参考帧坐标系下：$p_i^\omega \equiv p_i \oplus q_k = R(\theta_k)p_i + t_k$

为当前激光帧中的每一个点，找到其最近的两个点j_1和j_2

去除误差过大的点构建最小化误差方程：$J(q_{k+1}, C+k) = \sum_i (n_i^T[R(\theta_{k+1})p_i + t_{k+1} - p_{j_1^i}])^2$

求解出位姿转换矩阵q_{k+1} ,然后将其用于下次迭代计算

但是这一算法中的误差函数最优化问题较难求解，文献所提供的数学求解方法有些晦涩难懂，我们没能编写出相应的程序。但是我们采用了一个非直接的方法：在每次最近邻匹配时，选择最近两点（a点和b点）所确定的直线与待匹配点（c点）的距离正交点（d点）作为匹配点，并以之为目标点调用已有的PPICP算法。这样的做法虽然尚未在数学上严格证明，但从实践结果来看，似乎是可以达到渐进点线匹配的，最终输出的结果也还不错。

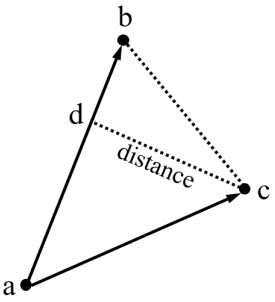


Fig.2 正交点示意图

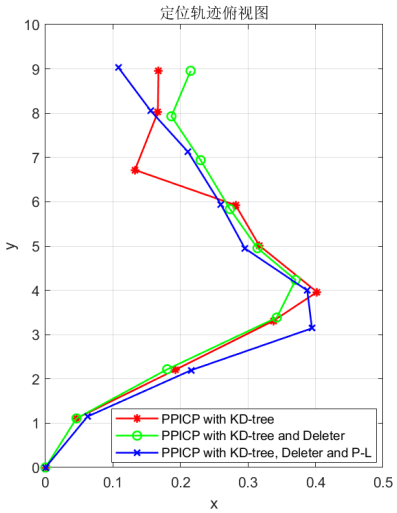


Fig.3 实验得到的定位轨迹

2 实验结果与分析

使用上述三种改进方法，得到点云地图与定位轨迹图如下：

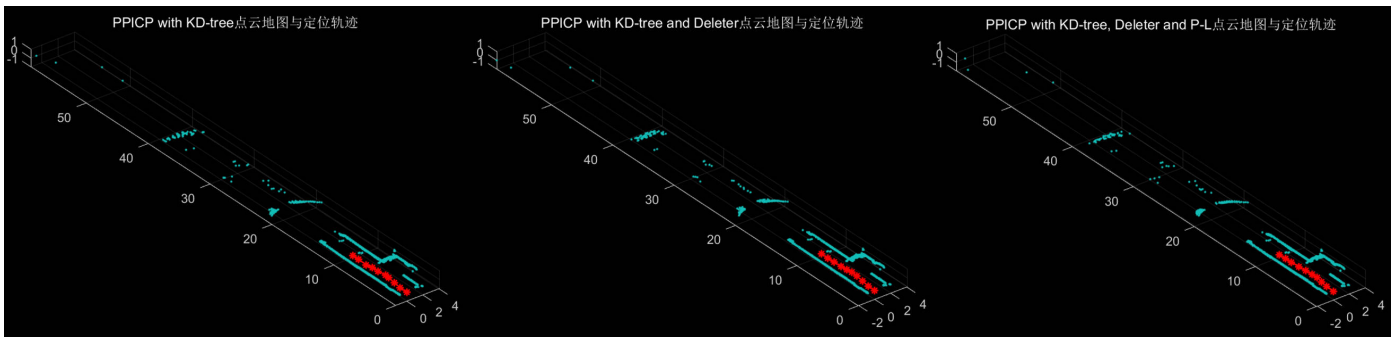


Fig.4 点云地图

从$Fig.3$可以看到，机器人的行动轨迹总体是以“之”字型移动，在PPICP with KD-tree与PPICP with KD-tree and Deleter的轨迹中，其末端出现了一定的偏折，而在PPICP with KD-tree, Deleter and P-L中，轨迹曲线较为平缓，没有出现太大的震荡。最终的数值结果为(0,0,0,0°), (0.062255,1.1508,0,-1.5453°), (0.2156,2.1952,0,-5.0795°), (0.39473,3.1431,0,-7.5923°), (0.38785,3.9964,0,-1.4402°), (0.29529,4.9469,0,3.4261°), (0.25975,5.9379,0,3.3718°), (0.21132,7.1225,0,2.194°), (0.15591,8.0523,0,-2.7989°), (0.1078,9.0348,0,-3.2547°)

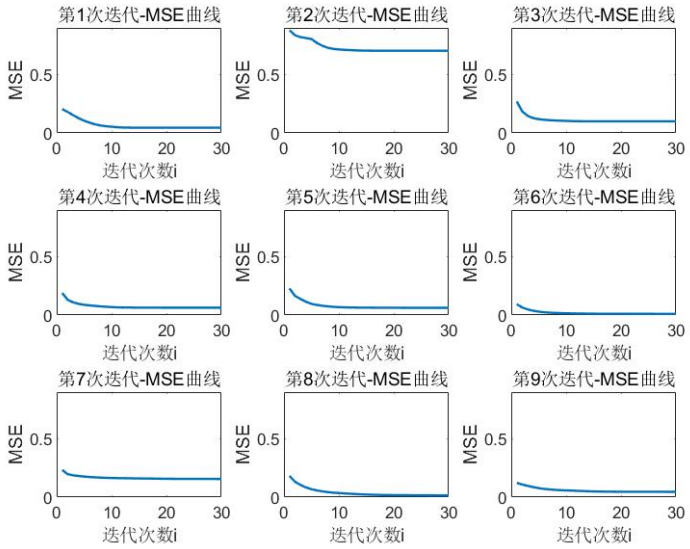
为了更好的评价我们的算法，将各算法及matlab自带函数计算得到的最终位姿结果（累计误差最大）和运行时间列表如下。其中，相对误差计算以matlab自带函数的计算结果为参考值（并不为真实值，仅供参考）

算法	最终x	x_{error}	最终y	y_{error}	最终θ(°)	θ_{error}	运行时间(s)
matlab自带函数(参考值)	0.1687	/	8.8705	/	-3.494	/	1.495
PPICP	0.1673	0.85%	8.9522	0.92%	-3.534	1.13%	3.334
PPICP with KD-tree	0.1673	0.85%	8.9522	0.92%	-3.534	1.13%	2.018
PPICP with KD-tree and Deleter	0.2150	27.42%	8.9557	0.96%	-1.912	45.28%	3.532
PPICP with KD-tree, Deleter and P-L	0.1078	36.11%	9.0348	1.85%	-3.255	6.86%	37.128

由于我们在matlab自带函数中选择了策略为Point-to-Point，所以我们的PPICP算法误差都比较小，而增加了Deleter和P-L的算法由于未在matlab中找到对应的自带函数，对于其误差不予评价，以真实值为准。

3 待改进之处

在现有的迭代算法中，我们将上一个点云配准得到的变换矩阵作为粗配准矩阵，这并不十分精确。由于ICP算法严重依赖于初值，在初值不恰当的情况下，有很大可能陷入局部最优，使得迭代不能收敛到正确的配准结果。例如，下图所示的第二次迭代表现并不好，MSE高达0.7。



在实际使用中，常使用轮式里程估计、惯性里程估计等作为粗配准值，在算法层面，也可以利用SAC-IA、PCL-NDT等算法得到的结果作为ICP的粗配准值。

4 分工

蒋颜丞，张嘉浩，朱姜宇轩共同完成编程，查找资料，撰写报告。

参考文献

[1] Censi A. An ICP variant using a point-to-line metric[C]//2008 IEEE International Conference on Robotics and Automation. Ieee, 2008: 19-25.
[2] Serafin J, Grisetti G. NICP: Dense normal based point cloud registration[C]//2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2015: 742-749.