深蓝学院
shenlanxueyuan.com

激光SLAM第四次作业讲解

主讲人 郭子萱

第一题

深蓝学院 shenlanxueyuan.com

法向量和曲率的计算

- 找到点p_i周围半径R球形空间中的所有点v_i

$$\mu_i^s = \frac{1}{|\mathcal{V}_i|}\sum_{\mathbf{p}_j \in \mathcal{V}_i} \mathbf{p}_i$$

$$\mathbf{\Sigma}_i^s = \frac{1}{|\mathcal{V}_i|}\sum_{\mathbf{p}_j \in \mathcal{V}_i} (\mathbf{p}_i - \mu_i)^T (\mathbf{p}_i - \mu_i)$$

$$\Sigma_i^s = U \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} U^T \quad \lambda_2 > \lambda_1$$

- 曲率的定义：

$$\sigma_i = \frac{\lambda_1}{\lambda_1 + \lambda_2}$$

- 法向量的定义：
  最小特征值对应的特征向量

```
//TODO
//根据周围的激光点计算法向量，参考ppt中NICP计算法向量的方法
//1、计算点的重心
Eigen::Vector2d  gravityCentreVector;

double  gravityCentreX = 0.0;
double  gravityCentreY = 0.0;
for(int i = 0; i< nearPoints.size(); i++){
    gravityCentreX +=  nearPoints[i](0);
    gravityCentreY +=  nearPoints[i](1);
}

gravityCentreX = gravityCentreX / nearPoints.size();
gravityCentreY = gravityCentreY / nearPoints.size();

gravityCentreVector[0] = gravityCentreX;
gravityCentreVector[1] = gravityCentreY;
```

第一题

深蓝学院
shenlanxueyuan.com

```
//2、计算方差

Eigen::Matrix2d sigma = Eigen::Matrix2d::Zero();

for(int i = 0; i< nearPoints.size(); i++){

    sigma += (nearPoints[i] - gravityCentreVector) * (nearPoints[i] - gravityCentreVector).transpose();

}

sigma /= nearPoints.size();
```

第一题

深蓝学院
shenlanxueyuan.com

```
//3、对方差进行特征值求解，并求出法向量

Eigen::EigenSolver<Eigen::Matrix2d> eigenSolver(sigma);

Eigen::MatrixXd eigenValue = eigenSolver.eigenvalues().real();
Eigen::MatrixXd eigenVector = eigenSolver.eigenvectors().real();

Eigen::MatrixXd::Index evalsMin;

eigenValue.rowwise().sum().minCoeff(&evalsMin);

normal = eigenVector.col(evalsMin);
//end of TODO
```

第一题

深蓝学院
shenlanxueyuan.com

曲面重建

P_k:前n帧激光数据组成的子图;
n_k:点云集P_k中的点p_i的法向量;

$I^{P_k}(x)$: $\mathbb{R}^3$空间的点x到点云集合P_K隐藏曲面的距离。

$I^{P_k}(x)$定义如下:

$$I^{P_k}(x) = \frac{\sum_{p_i \in P_k} W_i(x)\left((x-p_i)\cdot\vec{n_i}\right)}{\sum_{p_j \in P_k} W_j(x)}$$

其中，权重$W_i(x)$被定义为:

$$W_i(x) = e^{-\|x-p_i\|^2/h^2}$$

```
//TODO
//根据函数进行投影. 计算height, 即ppt中的I(x)
//1、计算 每一个点的w(x)
std::vector<double> Wx;
double denominator = 0.0;

for(int i = 0; i < nearPoints.size();i++){
    double distq = (x - nearPoints[i]).norm() * (x - nearPoints[i]).norm();
    double Wxi = std::exp(-distq/(m_h*m_h));
    Wx.push_back(Wxi);

    denominator += Wxi;
}
//2、计算I(x)
double molecule = 0.0;
for(int i =0;i < nearPoints.size();i++){
    double tmp = ((x - nearPoints[i]).transpose()*nearNormals[i]);
    molecule += Wx[i] * tmp;
}

height = molecule / denominator;
//std::cout<<"height: "<<height<<std::endl;
//end of TODO
```

第一题

深蓝学院
shenlanxueyuan.com

匹配求解

S_k:新一帧激光数据;

$I^{P_k}(x)$：当前帧S_k中的点x_i到曲面的距离;

$\overrightarrow{n_i}$： P_k中距离点x_i最近的点的法向量。

点x_i在曲面上的投影y_i为:

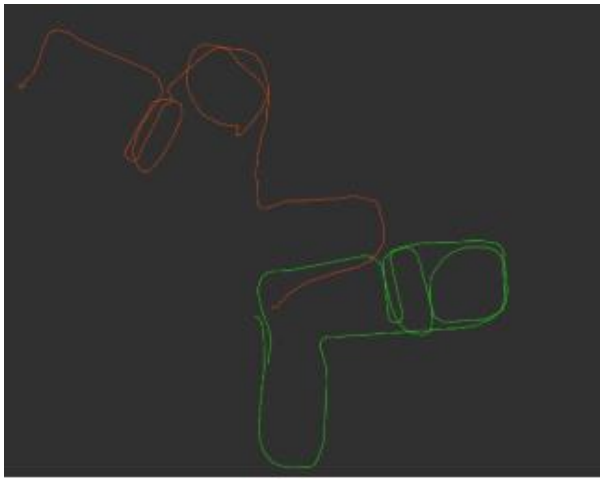
$$x_i^{'} = Rx_i + t \qquad\qquad y_i = x_i^{'} - I^{P_k}(x_i^{'})\overrightarrow{n_i}$$

对于新一帧激光数据S_k，通过最小化以下函数，求解姿态变换:

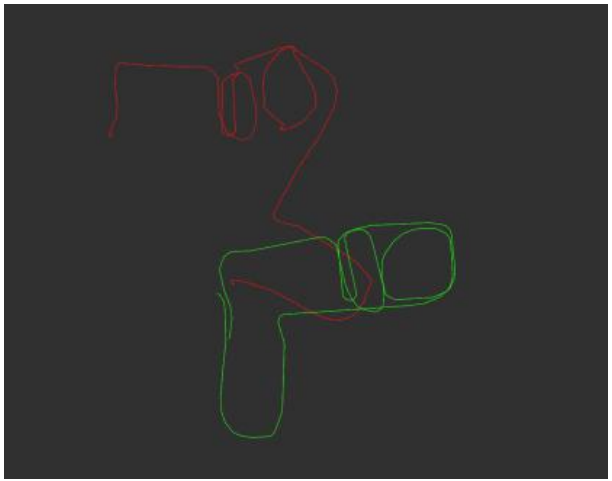
$$\sum_{x_i \in S_k} ((Rx_i + t - y_i) \cdot \vec{n}_i)^2$$

```
Eigen::Vector2d yi;
//TODO
//计算yi.

yi = xi - height * nearNormal;

//end of TODO
```

第二题



橙色为IMLS-ICP算法轨迹，绿色为里程计轨迹



橙色为PL-ICP算法轨迹，绿色为里程计轨迹

第二题

深蓝学院
shenlanxueyuan.com

csm改动代码

1、初始化PL-ICP参数

2、修改回调函数

3、将champion_nav_msgs::ChampionNavLaserScanConstPtr转为LDP格式

第三题

深蓝学院
shenlanxueyuan.com

总结ICP.PL-ICP.NICP.IMLS-ICP并比较其异同

参考课程和网上文档

第四题

题干：现在你已经了解了多种 ICP 算法，你是否也能提出一种改进的 ICP 算法，或能 提升 ICP 总体匹配精度或速度的技巧？请简述你的改进策略。

参考loam算法，计算点的曲率，根据曲率将点提取为角点和直线点，当前帧的角点和直线点，分别与上一帧最近的角点和直线点做ICP匹配，这样能够提高计算效率。

深蓝学院
shenlanxueyuan.com

感谢各位聆听！

Thanks for Listening