深蓝学院
shenlanxueyuan.com

激光的前端配准算法

主讲人 曾书格

越凡创新技术负责人
597457483@qq.com

课程内容

帧间匹配算法

1、ICP匹配方法

2、PL-ICP匹配方法

3、基于优化的匹配方法

4、相关匹配方法及分支定界加速

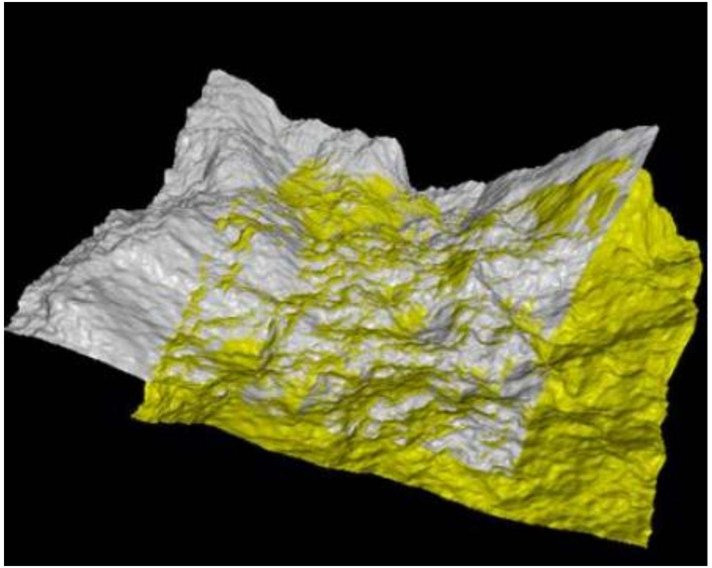
帧间匹配算法

帧间匹配算法

- 1、ICP匹配方法
- 2、PL-ICP匹配方法
- 3、基于优化的匹配方法
- 4、相关匹配方法及分支定界加速

ICP方法介绍

目的



ICP方法的目的

数学描述

- 给定两个点云集合:

$$X = \left\{x_1, x_2, \cdots, x_{N_x}\right\}$$
$$P = \left\{p_1, p_2, \cdots, p_{N_p}\right\}$$

- 求解R和t，使得下式最小:

$$E(R,t) = \frac{1}{N_p}\sum_{i=1}^{N_p}\|x_i - Rp_i - t\|^2$$

ICP方法介绍

已知对应点的求解方法

$$u_x = \frac{1}{N_p}\sum_{i=1}^{N_p} x_i \qquad u_p = \frac{1}{N_p}\sum_{i=1}^{N_p} p_i$$

$$X' = \{x_i - u_x\} = \{x_i'\}$$

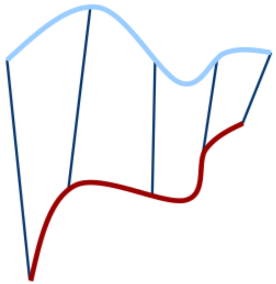
$$P' = \{p_i - u_p\} = \{p_i'\}$$

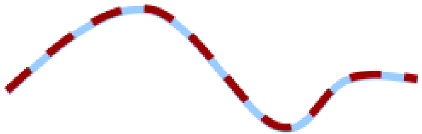
$$W = \sum_{i=1}^{N_p} x_i' p_i'^T = U \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & \sigma_3 \end{bmatrix} V^T$$

则ICP的解为：

$$R = VU^T$$

$$\mathrm{t} = u_x - Ru_p$$





已知对应点的求解方法—证明

$$E(R,t)=\frac{1}{N_p}\sum_{i=1}^{N_p}\|x_i-Rp_i-t\|^2$$

$$E(R,t)=\frac{1}{N_p}\sum_{i=1}^{N_p}\|x_i-Rp_i-t-u_x+Ru_p+u_x-Ru_p\|^2$$

$$=\frac{1}{N_p}\sum_{i=1}^{N_p}\|x_i-u_x-R(p_i-u_p)+(u_x-Ru_p-t)\|^2$$

$$\frac{1}{N_p}\sum_{i=1}^{N_p}\|x_i-u_x-R(p_i-u_p)\|^2+\|u_x-Ru_p-t)\|^2$$

$$+2\left(x_i-u_x-R\left(p_i-u_p\right)\right)^T(u_x-Ru_p-t))$$

$$=\frac{1}{N_p}\sum_{i=1}^{N_p}\|x_i-u_x-R(p_i-u_p)\|^2+\|u_x-Ru_p-t)\|^2$$

$\|x_i-u_x-R(p_i-u_p)\|^2$只跟R有关

当已知R时可以通过u_x-Ru_p-t求解得到t

转换为最小化函数：

$$E(R,t)=\frac{1}{N_p}\sum_{i=1}^{N_p}\|x_i-u_x-R(p_i-u_p)\|^2$$

ICP方法介绍

已知对应点的求解方法—证明

$$E(R,t)=\frac{1}{N_p}\sum_{i=1}^{N_p}\left\|x_i-u_x-R(p_i-u_p)\right\|^2$$

$$=\frac{1}{N_p}\sum_{i=1}^{N_p}\|x_i'-Rp_i'\|^2=\frac{1}{N_p}\sum_{i=1}^{N_p}x_i'^T x_i'+p_i'^T R^T R p_i'-2x_i'^T R p_i'$$

$$=\sum_{i=1}^{N_p}-2x_i'^T R p_i'$$

$$\sum_{i=1}^{N_p}x_i'^T R p_i'=\sum_{i=1}^{N_p}\mathrm{Trace}(Rx_i'p_i'^T)=Trace(RH)$$

$$H=\sum_{i=1}^{N_p}x_i'p_i'^T$$

- 定理：
  假设矩阵A为正定对称矩阵，则对于任意的正交矩阵B，都有：

$$Track(A)\geq Track(BA)$$

$H=U\Lambda V^T$　　$X=VU^T$--正交矩阵

$XH=VU^TU\Lambda V^T=V\Lambda V^T$--正定对称

则：

$$Trace(XH)\geq Trace(BXH)$$

因此：$R=X=VU^T$

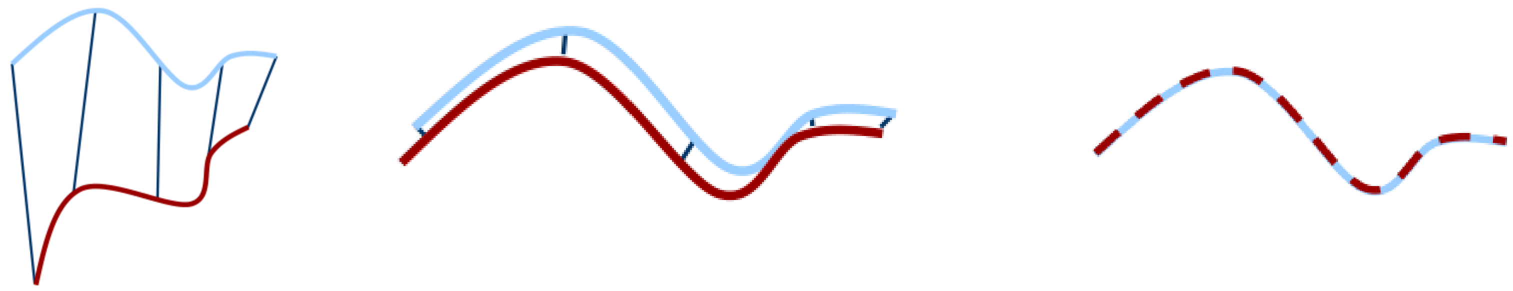
ICP方法介绍

未知对应点的求解方法

- 实际中，不知道对应点匹配
- 不能一步到位计算出R和t
- 进行迭代计算
- EM算法的一个特例

算法流程：

- 寻找对应点
- 根据对应点，计算R和t
- 对点云进行转换，计算误差
- 不断迭代，直至误差小于某一个值



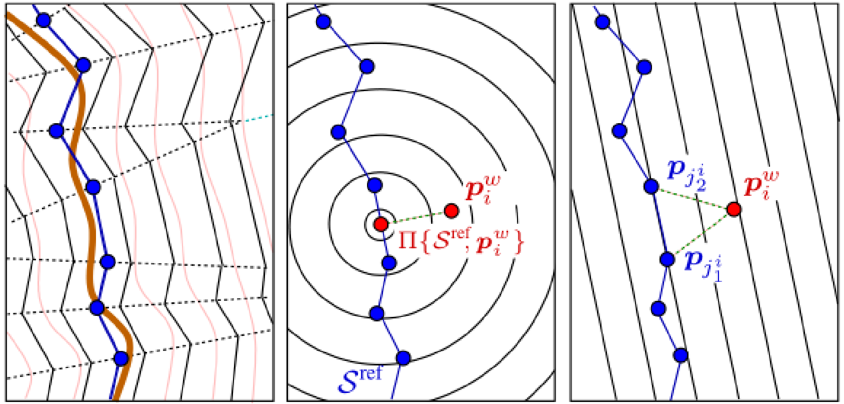
帧间匹配算法

帧间匹配算法

- 1、ICP匹配方法
- 2、PL-ICP匹配方法
- 3、基于优化的匹配方法
- 4、相关匹配方法及分支定界加速

PL-ICP方法介绍

示意图



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

PL-ICP方法示意图

数学描述

- 目标函数:

$$\min_{q_{k+1}} \sum_i \left(n_i^{\mathrm{T}} \left[p_i \oplus q_{k+1} - \Pi\{\mathcal{S}^{\mathrm{ref}}, p_i \oplus q_k\} \right] \right)^2$$

$$q = (t, \theta)$$

$$(p \oplus (t, \theta) \triangleq \mathbf{R}(\theta) p + t)$$

n_i为法向量

$\Pi\{\mathcal{S}^{\mathrm{ref}}, \cdot\}$ 表示在S^{ref}的投影

PL-ICP方法介绍

算法流程

- 1. 把当前帧的数据根据初始位姿投影到参考帧坐标系下。
- 2. 对于当前帧的点i，在参考帧中找到最近的两个点(j_1, j_2)。
- 3. 计算误差，并去除误差过大的点。
- 4. 最小化误差函数:

$$\sum_i \left(\boldsymbol{n}_i^{\mathrm{T}} \left[\mathbf{R}(\theta_{k+1}) \boldsymbol{p}_i + \boldsymbol{t}_{k+1} - \boldsymbol{p}_{j_1^i} \right] \right)^2$$

跟ICP的区别

- 1. 误差函数的形式不同，ICP对点对点的距离作为误差，PL-ICP为点到线的距离作为误差。
- 2. 收敛速度不同，ICP为一阶收敛，PL-ICP为二阶收敛。
- 3. PL-ICP的求解精度高于ICP
- 4. PL-ICP对初始值更敏感

帧间匹配算法

帧间匹配算法

1、ICP匹配方法

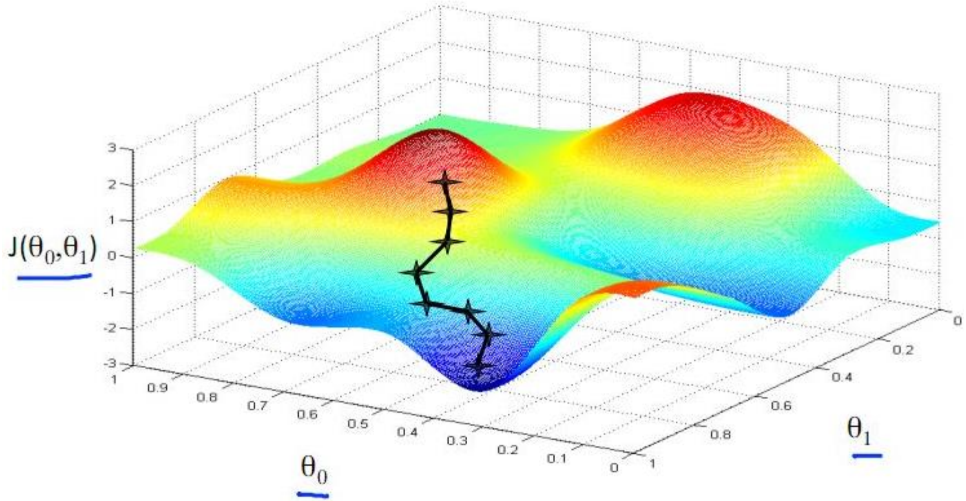
2、PL-ICP匹配方法

3、基于优化的匹配方法

4、相关匹配方法及分支定界加速

基于优化的方法(Optimization-based Method)

示意图



梯度下降示意图

数学描述

- 给定一个目标函数，把激光的帧间匹配问题转换为求解目标函数的极值问题：

$$E(T) = arg\min_{T}\sum[1 - M(S_i(T))]^2$$

$$T = (T_x, T_y, T_\theta)$$

$S_i(T)$表示把激光数据用位姿T进行转换

$M(x)$表示得到坐标x的地图占用概率

基于优化的方法(Optimization-based Method)

优化方法的求解

$$E(T)=arg\min_{T}\sum[1-M(S_i(T))]^2$$

$p_i=(p_{ix},p_{iy})$表示第i个激光点的坐标

$$S_i(T)=\begin{bmatrix}\cos T_\theta & -\sin T_\theta & T_x\\ \sin T_\theta & \cos T_\theta & T_y\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}p_{ix}\\ p_{iy}\\ 1\end{bmatrix}$$

$M(S_i(T))$为非线性函数，因此进行一阶泰勒展开，得：

$$E(T+\Delta T)=arg\min_{T}\sum\left[1-M\big(S_i(T)\big)-\nabla M(S_i(T))\frac{\partial S_i(T)}{\partial T}\Delta T\right]^2$$

对于线性系统，求其对ΔT的导数，并令其等于0：

$$\sum\left[\nabla M(S_i(T))\frac{\partial S_i(T)}{\partial T}\right]^T\left[1-M\big(S_i(T)\big)-\nabla M(S_i(T))\frac{\partial S_i(T)}{\partial T}\Delta T\right]=0$$

求解上式即可得到ΔT

令$T=T+\Delta T$，不断进行迭代即可

基于优化的方法(Optimization-based Method)

优化方法的求解

$$\sum\left[\nabla M(S_i(T))\frac{\partial S_i(T)}{\partial T}\right]^T\left[1-M\big(S_i(T)\big)-\nabla M(S_i(T))\frac{\partial S_i(T)}{\partial T}\Delta T\right]=0$$

展开得：

$$\sum\left[\nabla M(S_i(T))\frac{\partial S_i(T)}{\partial T}\right]^T\left[1-M\big(S_i(T)\big)\right]=\sum\left[\nabla M(S_i(T))\frac{\partial S_i(T)}{\partial T}\right]^T\left[\nabla M(S_i(T))\frac{\partial S_i(T)}{\partial T}\Delta T\right]$$

等式两边同时乘以H^{-1}：

$$\Delta T=H^{-1}\sum\left[\nabla M(S_i(T))\frac{\partial S_i(T)}{\partial T}\right]^T\left[1-M\big(S_i(T)\big)\right]$$

$$H=\sum\left[\nabla M(S_i(T))\frac{\partial S_i(T)}{\partial T}\right]^T\left[\nabla M(S_i(T))\frac{\partial S_i(T)}{\partial T}\Delta T\right]$$

基于优化的方法(Optimization-based Method)

优化方法的求解

$$\Delta T = H^{-1}\sum\left[\nabla M(S_i(T))\frac{\partial S_i(T)}{\partial T}\right]^T\left[1 - M\big(S_i(T)\big)\right]$$

$$S_i(T) = \begin{bmatrix} \cos T_\theta & -\sin T_\theta & T_x \\ \sin T_\theta & \cos T_\theta & T_y \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} p_{ix} \\ p_{iy} \\ 1 \end{bmatrix}$$

$$S_i(T) = \begin{bmatrix} \cos T_\theta * p_{ix} - \sin T_\theta * p_{iy} + T_x \\ \sin T_\theta * p_{ix} + \cos T_\theta * p_{iy} + T_y \\ 1 \end{bmatrix}$$

$$\frac{\partial S_i(T)}{\partial T} = \begin{bmatrix} 1 & 0 & -\sin T_\theta * p_{ix} - \cos T_\theta * p_{iy} \\ 0 & 1 & \cos T_\theta * p_{ix} - \sin T_\theta * p_{iy} \end{bmatrix}$$

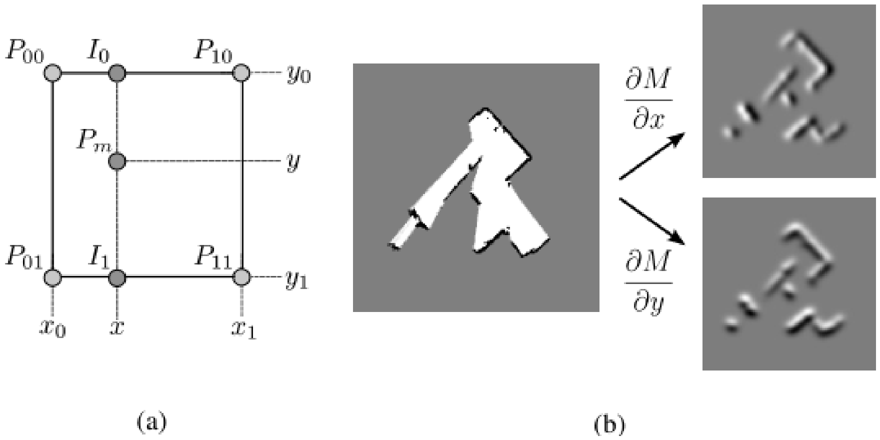
ΔT表达式中所有的量都已经知道，除了$\nabla M(S_i(T))$

$S_i(T)$表示地图坐标点，$\nabla M(S_i(T))$表示地图的导数

$\nabla M\big(S_i(T)\big)$的求解需要对地图进行插值

基于优化的方法(Optimization-based Method)

地图双线性插值



地图插值示意图

- 拉格朗日插值法
- X,Y两个方向进行插值
- 一维线性插值的推广

基于优化的方法(Optimization-based Method)

拉格朗日插值方法——一维线性插值

- 插值的定义

设函数$y = f(x)$在区间$[a, b]$上有定义，且存在已知点：

$$a \le x_0 < x_1 < \cdots < x_n \le b$$

处的函数值$y_i = f(x_i)$,若存在n次多项式：

$$L_n(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$$

使得值$L_n(x_i) = y_i$成立,则称$L_n(x)$为$f(x)$的插值多项式。

可以证明：$L_n(x)$存在且唯一

- 拉格朗日插值方法

实现上述插值的一种方法

主要特点为把插值多项式表示成基函数的线性组合：

$$L_n(x) = \sum_{i=0}^{n} l_i(x) y_i$$

基函数$l_i(x)$满足以下条件：

$$l_i(x_k) = \begin{cases} 1 & k = i \\ 0 & k \ne i \end{cases}$$

基于优化的方法(Optimization-based Method)

拉格朗日插值方法—基函数构造

基函数$l_i(x)$在除x_i以外的所有插值点都为0，即点$(x_0,\cdots,x_{i-1},x_{i+1},\cdots,x_n)$都是$l_i(x)$的解，因此可以构造函数:

$$l_i(x)=c(x-x_0)\cdots(x-x_{i-1})(x-x_{i+1})(x-x_n)$$

显然$l_i(x)$满足上述条件。同时$l_i(x_i)=1$，因此：

$$l_i(x_i)=c(x_i-x_0)\cdots(x_i-x_{i-1})(x_i-x_{i+1})(x_i-x_n)=1$$

因此：

$$c=\frac{1}{(x_i-x_0)\cdots(x_i-x_{i-1})(x_i-x_{i+1})(x_i-x_n)}$$

$$l_i(x)=\prod_{k=0,k\neq i}^{n}\frac{x-x_k}{(x_i-x_k)}$$

基于优化的方法(Optimization-based Method)

拉格朗日插值方法—双线性插值

设平面中有四个点:

$$Z_1=f(x_0,y_0),Z_2=f(x_1,y_0)$$
$$Z_3=f(x_1,y_1),Z_4=f(x_0,y_1)$$

令:

$$\mathrm{u}=\frac{x-x_0}{x_1-x_0}\qquad \mathrm{v}=\frac{y-y_0}{y_1-y_0}$$

则对应的四个点的坐标变为:

$$(x_0,y_0)=(0,0)\quad (x_1,y_0)=(1,0)$$

$$(x_1,y_1)=(1,1)\quad (x_0,y_1)=(0,1)$$

构造基函数:

$$l_1(u,v)=(1-u)(1-v)$$

$$l_2(u,v)=u(1-v)$$

$$l_3(u,v)=uv$$

$$l_4(u,v)=(1-u)v$$

插值函数为:

$$L_4(u,v)=Z_1l_1(u,v)+Z_2l_2(u,v)$$
$$+Z_3l_3(u,v)+Z_4l_4(u,v)$$

基于优化的方法(Optimization-based Method)

地图插值

插值函数:

$$L_4(u,v) = Z_1 l_1(u,v) + Z_2 l_2(u,v) + Z_3 l_3(u,v) + Z_4 l_4(u,v)$$

把(u,v)替换回(x,y)可得:

$$L(x,y) = \frac{y-y_0}{y_1-y_0}\left(\frac{x-x_0}{x_1-x_0}M(P_{11}) + \frac{x_1-x}{x_1-x_0}M(P_{01})\right) + \frac{y_1-y}{y_1-y_0}\left(\frac{x-x_0}{x_1-x_0}M(P_{10}) + \frac{x_1-x}{x_1-x_0}M(P_{00})\right)$$

x的偏导数:

$$\frac{\partial L(x,y)}{\partial x} = \frac{y-y_0}{y_1-y_0}(M(P_{11})-M(P_{01})) + \frac{y_1-y}{y_1-y_0}(M(P_{10})-M(P_{00}))$$

y的偏导数:

$$\frac{\partial L(x,y)}{\partial y} = \frac{x-x_0}{x_1-x_0}(M(P_{11})-M(P_{10})) + \frac{x_1-x}{x_1-x_0}(M(P_{01})-M(P_{00}))$$

帧间匹配算法

帧间匹配算法

1、ICP匹配方法

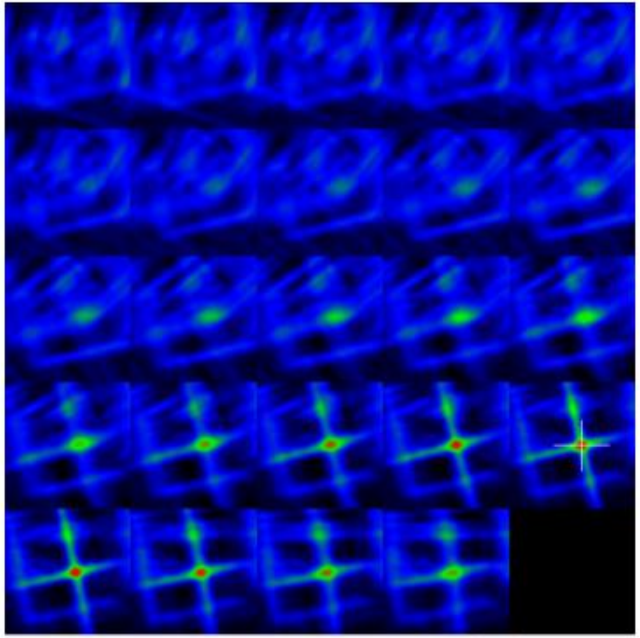
2、PL-ICP匹配方法

3、基于优化的匹配方法

4、相关匹配方法及分支定界加速

相关方法(Correlation-based Method)

帧间匹配似然场



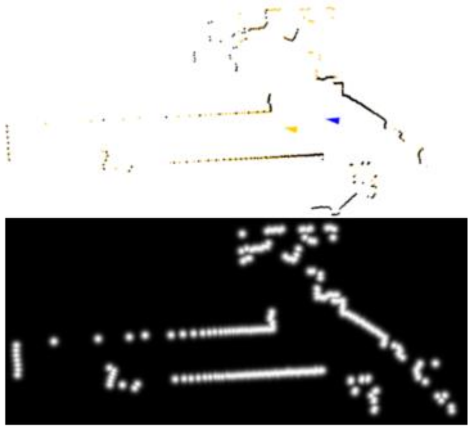
似然场示意图

- 高度非凸，存在很多的局部极值
- 对初值非常敏感
- 进行暴力匹配，排除初值影响
- 通过加速策略，降低计算量
- 计算位姿匹配方差

相关方法(Correlation-based Method)

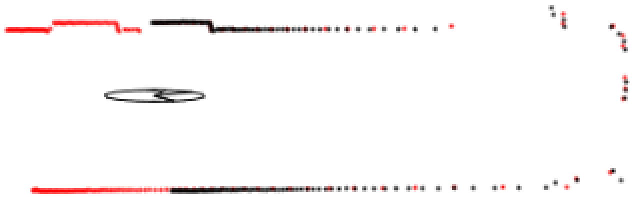
算法流程

- 1. 构造似然场



- 2. 在指定的搜索空间内，进行搜索，计算每一个位姿的得分

- 3. 根据步骤2中位姿的得分，计算本次位姿匹配的方差



相关方法(Correlation-based Method)

位姿搜索

- 1. 暴力搜索

三层for循环(x, y, θ)枚举每一个位姿，
分别计算每一个位姿的得分，计算量巨大。
因为激光雷达数据在每一个位姿都要重新投影，
投影需要计算sin和cos函数。

- 2. 预先投影搜索

把暴力搜索中的三层for循环(x, y, θ)交换一下顺序，
最外层对θ进行搜索，这样内层x, y的投影变成的加法，
需要计算sin和cos函数的位姿数从$n_x n_y n_\theta$降为n_θ 。
能极大的加速算法的运行速度

- 3. 多分辨率搜索

(1)构造粗分辨率$(25cm)$和
细分辨率$(2.5cm)$两个似然场

(2)首先在粗分辨率似然场上进行搜索，
获取最优位姿

(3)把粗分辨率最优位姿对应的栅格进行
细分辨率划分，然后再进行细分辨率
搜索，再次得到最优位姿。

(4)粗分辨率地图的栅格的似然值为对应
的细分辨率地图对应空间的所有栅格的
最大值

相关方法(Correlation-based Method)

分枝定界算法

- 常用的树形搜索剪枝算法
- 求解整数规划问题
- 解的数量为有限个
- 把最优解求解问题转换为树形搜索问题，根节点表示整个解空间，叶子节点表示最优解，中间的节点表示解空间的某一部分子空间。

- 分枝：即根节点表示整个解空间空间，深度为1的节点表示解空间的子空间，深度为2的节点表示深度1空间的子空间，这样层层划分，直到划分到真实解，也就是叶子节点为止。
- 定界：对于搜索树种的每一个节点，确定以该节点为根节点的子树的界。对于最小值问题，确定下界；对于最大值问题，确定上界。(SLAM中为上界)

相关方法(Correlation-based Method)

分枝定界算法

Algorithm 2 Generic branch and bound

$best\_score \leftarrow -\infty$
$\mathcal{C} \leftarrow \mathcal{C}_0$
while $\mathcal{C} \neq \emptyset$ **do**
  Select a node $c \in \mathcal{C}$ and remove it from the set.
  if c is a leaf node **then**
    if $score(c) > best\_score$ **then**
      $solution \leftarrow n$
      $best\_score \leftarrow score(c)$
    end if
  else
    if $score(c) > best\_score$ **then**
      Branch: Split c into nodes $\mathcal{C}_c$.
      $\mathcal{C} \leftarrow \mathcal{C} \cup \mathcal{C}_c$
    else
      Bound.
    end if
  end if
end while
return $best\_score$ and $solution$ when set.

相关方法(Correlation-based Method)

分枝定界在相关方法的加速作用

- 搜索树中的节点表示一个正方形的索索范围:$(c_x, c_y, c_\theta, c_h)$

$$\overline{\overline{\mathcal{W}}}_c = \left(\left\{ (j_x, j_y) \in \mathbb{Z}^2 : \begin{array}{l} c_x \leq j_x < c_x + 2^{c_h} \\ c_y \leq j_y < c_y + 2^{c_h} \end{array} \right\} \times \{c_\theta\} \right)$$

- 分枝：对于节点$(c_x, c_y, c_\theta, c_h)$，分枝为4个子节点，四个子节点为：

$$\mathcal{C}_c = \left(\left(\{c_x, c_x + 2^{c_h - 1}\} \times \{c_y, c_y + 2^{c_h - 1}\} \times c_\theta \right) \cap \overline{\mathcal{W}} \right) \times \{c_h - 1\}.$$

- 定界：构造得分函数，使得得分函数对于节点l的打分，是以节点l为根节点的子树的上界：

$$\begin{aligned} score(c) &= \sum_{k=1}^{K} \max_{j \in \overline{\overline{\mathcal{W}}}_c} M_{\text{nearest}}(T_{\xi_j} h_k) \\ &\geq \sum_{k=1}^{K} \max_{j \in \overline{\mathcal{W}}_c} M_{\text{nearest}}(T_{\xi_j} h_k) \\ &\geq \max_{j \in \overline{\mathcal{W}}_c} \sum_{k=1}^{K} M_{\text{nearest}}(T_{\xi_j} h_k). \end{aligned}$$

$$\overline{\mathcal{W}}_c = \overline{\overline{\mathcal{W}}}_c \cap \overline{\mathcal{W}}.$$

相关方法(Correlation-based Method)

分枝定界在相关方法的加速作用

Algorithm 2 Generic branch and bound

$best\_score \leftarrow -\infty$
$\mathcal{C} \leftarrow \mathcal{C}_0$
while $\mathcal{C} \neq \emptyset$ **do**
  Select a node $c \in \mathcal{C}$ and remove it from the set.
  if c is a leaf node **then**
    if $score(c) > best\_score$ **then**
      $solution \leftarrow n$
      $best\_score \leftarrow score(c)$
    end if
  else
    if $score(c) > best\_score$ **then**
      Branch: Split c into nodes $\mathcal{C}_c$.
      $\mathcal{C} \leftarrow \mathcal{C} \cup \mathcal{C}_c$
    else
      Bound.
    end if
  end if
end while
return $best\_score$ and $solution$ when set.

Algorithm 3 DFS branch and bound scan matcher for (BBS)

$best\_score \leftarrow score\_threshold$
Compute and memorize a score for each element in $\mathcal{C}_0$.
Initialize a stack $\mathcal{C}$ with $\mathcal{C}_0$ sorted by score, the maximum score at the top.
while $\mathcal{C}$ is not empty **do**
  Pop c from the stack $\mathcal{C}$.
  if $score(c) > best\_score$ **then**
    if c is a leaf node **then**
      $match \leftarrow \xi_c$
      $best\_score \leftarrow score(c)$
    else
      Branch: Split c into nodes $\mathcal{C}_c$.
      Compute and memorize a score for each element in $\mathcal{C}_c$.
      Push $\mathcal{C}_c$ onto the stack $\mathcal{C}$, sorted by score, the maximum score last.
    end if
  end if
end while
return $best\_score$ and $match$ when set.

作业

比较四种不同算法帧间匹配的效果(选做)

- 目前介绍的四种方法网上都有开源的实现
- ICP在PCL中有实现
- PL-ICP可以直接二进制安装CSM
- 优化方法在HectorSLAM和Cartographer中有实现
- 相关方法在Cartographer中有实现

结语

感谢各位聆听!

Thanks for Listening