

激光的前端配准算法-二

主讲人 曾书格

越凡创新技术负责人
电子科技大学硕士



帧间匹配算法



1、爬山法(拟梯度法)



2、高斯牛顿优化方法



3、NDT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、爬山法(拟梯度法)



2、高斯牛顿优化方法



3、NDT方法



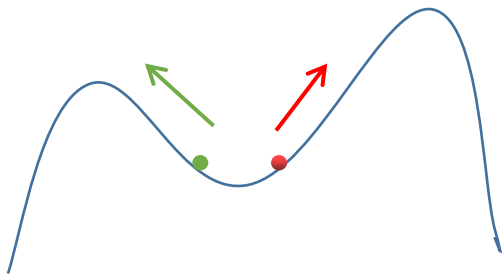
4、相关匹配方法及分支定界加速



爬山法(拟梯度法)



示意图

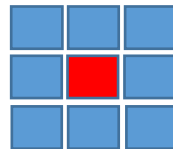


爬山法示意图

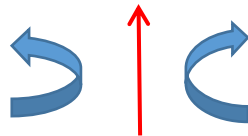


基本思想

- 搜索方向固定(只在有限的几个方向中选取, 平移8个, 旋转2个)



平移搜索



旋转搜索

- 搜索时, 保持得分单调递增; 不满足则把搜索步长减半
- 当搜索步长减半的次数大于一定的次数时, 则认为收敛



爬山法(拟梯度法)



得分函数定义

T 机器人的位姿

$S_i(T)$ 激光点 i 位姿变换 T 后的坐标

$M(x)$ 似然场地图中坐标 x 处的值

n 当前帧激光点的数量

$$Score(T) = \sum_{i=1}^n M(S_i(T))$$

- $Score(T)$ 函数表示位姿 T 在似然场地图中的得分
- 似然场地图由前一帧或前几帧激光数据生成



算法流程

- 搜索从一个初始位姿开始 $T(0)$
- 对于当前最优位姿搜索位姿 $T(i)$, 计算周围10个待搜索位姿的得分, 并得到最高得分对应的位姿 $T(*)$
- 如果最高得分比当前搜索位姿得分更高, 则更新当前搜索位姿 $T(i+1)=T(*)$
- 否则, 搜索步长减半, 继续从当前搜索位姿进行搜索 $T(i+1)=T(i)$
- 如果搜索步长减半的次数超过阈值, 则认为搜索结束, 返回当前的最优位姿



爬山法(拟梯度法)



伪代码

```
ComputePose(pose, dir, searchStep):  
    newPose = pose + dir * searchStep  
    return newPose
```

```
bestPose =  $T_0$   
bestScore = Score( $T_0$ )  
while(iterations < iteration_thres)  
    maxScore = bestScore  
    maxPose = bestPose  
    for move in all-search-direction  
        tmpPose = computePose(bestScore, move, searchStep)  
        tmpScore = Score(tmpPose)  
        if(tmpScore > maxScore)  
            maxScore = tmpScore  
            maxPose = tmpPose  
        endif  
    endfor  
    if(maxScore > bestScore)  
        bestScore = maxScore  
        bestPose = maxPose  
    else  
        searchStep = searchStep / 2  
        iterations ++  
    endif  
endwhile
```

爬山法伪代码



帧间匹配算法

帧间匹配算法



1、爬山法(拟梯度法)



2、高斯牛顿优化方法



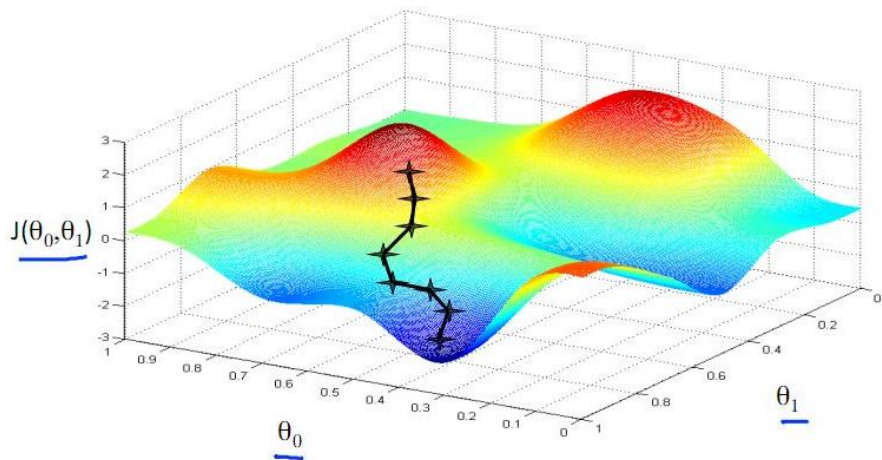
3、NDT方法



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：

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

其中,

$T = (T_x, T_y, T_\theta)$ 表示机器人的位姿

$p_i = (p_{ix}, p_{iy})$ 表示第*i*个激光点的坐标

$S_i(T)$ 表示第*i*个激光点位姿变换*T*后的坐标

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

$M(x)$ 表示似然场地图中坐标*x*处的值

基于优化的方法(Optimization-based Method)

优化方法的求解

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

$M(S_i(T))$ 为非线性函数, 进行一阶泰勒展开, 可得:

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

求 $E(T + \Delta T)$ 对 ΔT 的导数, 并令其等于0:

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



求得 ΔT



基于优化的方法(Optimization-based Method)



优化方法的求解

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

展开得：

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

等式两边同时乘以 H^{-1} ：

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

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

基于优化的方法(Optimization-based Method)

优化方法的求解

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

ΔT 表达式中似然场地图 $M(S_i(T))$ 已知, 未知量: $\nabla M(S_i(T))$ 、 $\frac{\partial S_i(T)}{\partial T}$

$$\frac{\partial S_i(T)}{\partial T}$$

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

$$\nabla M(S_i(T))$$

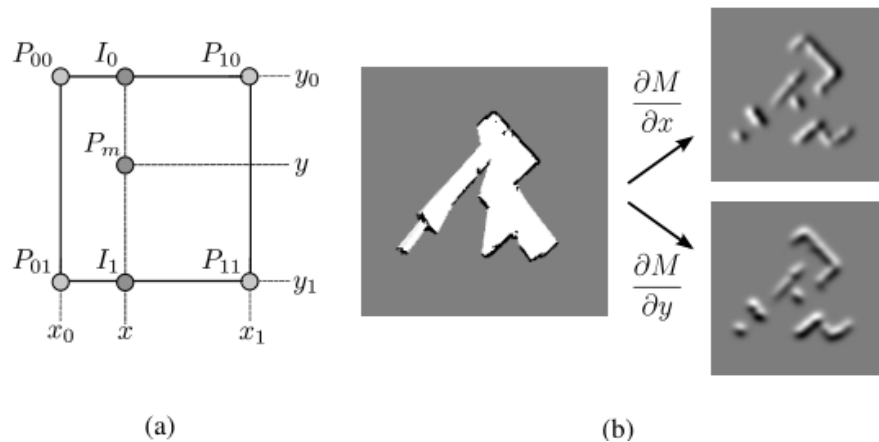
$M(S_i(T))$ 表示似然场地图中坐标 $S_i(T)$ 处的值; $\nabla M(S_i(T))$ 表示似然场对坐标位置的导数, 需要通过插值求解。



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——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:

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

处的函数值 $y_i = f(x_i)$, 若存在 n 次多项式:

$$L_n(x) = a_0 + a_1x + a_2x^2 + \cdots + a_nx^n$$

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:

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



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 $(x_0, \dots, x_{i-1}, x_{i+1}, \dots, x_n)$ 都是 $l_i(x)$ 的解, 因此可以构造函数:

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

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:

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

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

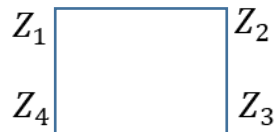


基于优化的方法(Optimization-based Method)



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:

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

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:

$$\begin{aligned} L_4(u, v) \\ = Z_1 l_1(u, v) + Z_2 l_2(u, v) + Z_3 l_3(u, v) + Z_4 l_4(u, v) \end{aligned}$$



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:

$$L(x, y) = \frac{y - y_0}{y_1 - y_0} \left(\frac{x - x_0}{x_1 - x_0} Z_3 + \frac{x_1 - x}{x_1 - x_0} Z_4 \right) + \frac{y_1 - y}{y_1 - y_0} \left(\frac{x - x_0}{x_1 - x_0} Z_2 + \frac{x_1 - x}{x_1 - x_0} Z_1 \right)$$

x 的偏导数:

$$\frac{\partial L(x, y)}{\partial x} = \frac{y - y_0}{y_1 - y_0} \left(\frac{Z_3 - Z_4}{x_1 - x_0} \right) + \frac{y_1 - y}{y_1 - y_0} \left(\frac{Z_2 - Z_1}{x_1 - x_0} \right)$$

y 的偏导数:

$$\frac{\partial L(x, y)}{\partial y} = \frac{1}{y_1 - y_0} \left(\frac{x - x_0}{x_1 - x_0} Z_3 + \frac{x_1 - x}{x_1 - x_0} Z_4 \right) - \frac{1}{y_1 - y_0} \left(\frac{x - x_0}{x_1 - x_0} Z_2 + \frac{x_1 - x}{x_1 - x_0} Z_1 \right)$$



帧间匹配算法

帧间匹配算法



1、爬山法(拟梯度法)



2、高斯牛顿优化方法



3、**NDT方法**



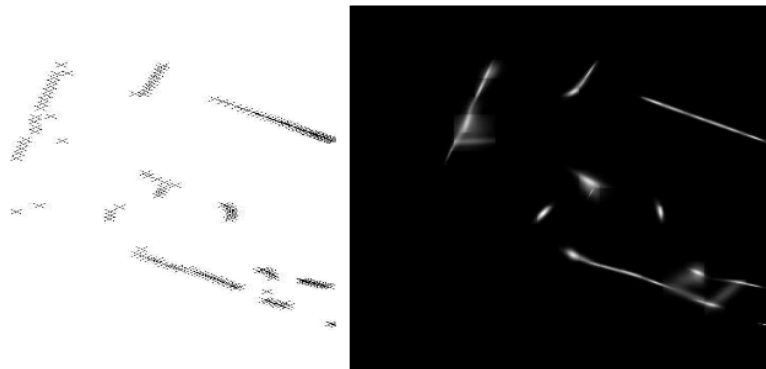
4、相关匹配方法及分支定界加速



NDT方法



示意图



NDT方法示意图



基本思想

- 把空间用cell进行划分;
- 用高斯分布去模拟似然场, 形成一个天然分段连续的似然场;
- 在得到连续的似然场之后, 直接用牛顿方法进行迭代即可;
- 似然场连续, 不受离散化带来的影响。



NDT方法



数学描述

$T = (T_x, T_y, T_\theta)$ 表示需要求解的姿态变换

$X_i = \begin{pmatrix} x_i \\ y_i \end{pmatrix}$ 表示第 i 个激光点的坐标

X'_i 表示第 i 个激光点经过姿态变换 T 后的坐标

$$X'_i = \begin{pmatrix} \cos T_\theta & -\sin T_\theta \\ \sin T_\theta & \cos T_\theta \end{pmatrix} \begin{pmatrix} x_i \\ y_i \end{pmatrix} + \begin{pmatrix} T_x \\ T_y \end{pmatrix}$$

q_i, Σ_i 表示点 X_i 对应的高斯分布的均值和方差

- 当前帧激光点 i 的得分为：

$$score_i = \exp\left(-\frac{(X'_i - q_i)^T \Sigma_i^{-1} (X'_i - q_i)}{2}\right)$$

- 因此匹配的目标函数：

$$\max \sum score_i \longrightarrow \min \sum -score_i$$

令 $q = X'_i - q_i$, 则

$$score_i = \exp\left(-\frac{q^T \Sigma_i^{-1} q}{2}\right)$$

- 目标函数并没有取误差的平方。
- 不对 $score()$ 函数进行线性化。
- 本式中可以直接求解Hessian矩阵。



NDT方法



牛顿方法

- 假设目标函数为：

$$\min f(x)$$

- 等价于：

$$g(x) = f'(x) = 0$$

- 进行泰勒展开，取一阶进行：

$$g(x + \Delta x) = g(x) + \frac{\partial g(x)}{\partial x} \Delta x = 0$$

$$\frac{\partial g(x)}{\partial x} \Delta x = -g(x)$$

$$x = x + \Delta x$$

- 等价于：

$$H\Delta x = -J$$

因此，用牛顿法进行迭代求解时，关键是要计算目标函数 $f(x)$ 的Hessian矩阵和Jacobian矩阵。

- 同时，如果：

$$f(x) = \sum f_i(x)$$

则：

$$J = \sum J_i$$

$$H = \sum H_i$$



NDT方法



NDT求解

显然:

$$f_i = -score_i = -\exp\left(-\frac{q^T \Sigma_i^{-1} q}{2}\right)$$

因此, 根据链式法则:

$$J_i = \frac{\partial f_i}{\partial T} = -\exp\left(-\frac{q^T \Sigma_i^{-1} q}{2}\right) (-q^T \Sigma_i^{-1}) \frac{\partial q}{\partial T}$$

$$H_i = \frac{\partial J_i}{\partial T} = -\exp\left(-\frac{q^T \Sigma_i^{-1} q}{2}\right) (-q^T \Sigma_i^{-1}) \frac{\partial q}{\partial T} (-q^T \Sigma_i^{-1}) \frac{\partial q}{\partial T} - \exp\left(-\frac{q^T \Sigma_i^{-1} q}{2}\right) \left(-\frac{\partial q^T}{\partial T} \Sigma_i^{-1} \frac{\partial q}{\partial T}\right) - \exp\left(-\frac{q^T \Sigma_i^{-1} q}{2}\right) (-q^T \Sigma_i^{-1}) \frac{\partial^2 q}{\partial T^2}$$

其中, $q = X_i' - q_i$ $\frac{\partial q}{\partial T} = \frac{\partial X_i'}{\partial T}$

$$X_i' = \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix} \begin{pmatrix} x_i \\ y_i \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix} \longrightarrow \frac{\partial X_i'}{\partial T} = \begin{pmatrix} 1 & 0 & -x_i \sin \theta - y_i \cos \theta \\ 0 & 1 & x_i \cos \theta - y_i \sin \theta \end{pmatrix}$$



NDT方法



算法流程

1. 将空间环境进行cell分割（若是二维空间，则离散为栅格，若是三维空间则离散划分为立方体），这样就可以将点云划分到不同cell中；
2. 用高斯分布刻画每个cell点云的分布，其均值和方差根据下方公式计算；

$$\begin{aligned}\mu_i^s &= \frac{1}{|\mathcal{V}_i|} \sum_{\mathbf{p}_j \in \mathcal{V}_i} \mathbf{p}_i \\ \Sigma_i^s &= \frac{1}{|\mathcal{V}_i|} \sum_{\mathbf{p}_j \in \mathcal{V}_i} (\mathbf{p}_i - \mu_i)^T (\mathbf{p}_i - \mu_i)\end{aligned}$$

3. 根据初始解把当前帧的激光点转换到参考帧，并确定在哪一个cell中；
4. 计算当前帧每个激光点在所属cell中的 $score_i$ ；
5. 计算目标函数 $\max \sum score_i$ ，通过牛顿方法进行迭代求解，得到姿态转换矩阵 T 。



帧间匹配算法

帧间匹配算法



1、爬山法(拟梯度法)



2、高斯牛顿优化方法



3、NDT方法



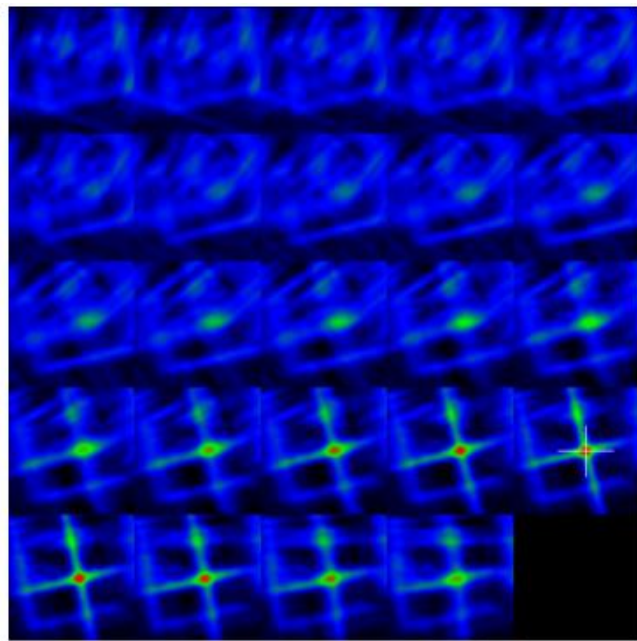
4、相关匹配方法及分支定界加速



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

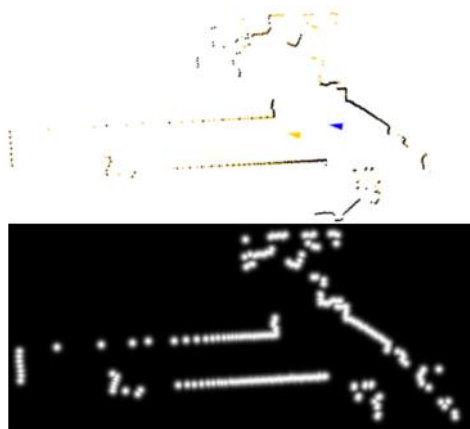


相关方法(Correlation-based Method)



算法流程

- 1. 构造似然场(方法见第3节课);
- 2. 在指定的位姿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, 得分最高的位姿即为要求解的位姿, 同时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。

- 计算量巨大，投影矩阵生成需要计算 \sin 和 \cos ，
计算的次数为: $n_x n_y n_\theta$
- 点的投影涉及到矩阵乘法，需要计算的次数: $n_x n_y n_\theta n$

```

bestPose = (x_0, y_0, \theta_0)
bestScore = Score(V2T(bestPose), Z_t)
for dx \in (-\Delta x, \Delta x)
  for dy \in (-\Delta y, \Delta y)
    for d\theta \in (-\Delta \theta, \Delta \theta)
      T = V2T(x_0 + \Delta x, y_0 + \Delta y, \theta_0 + \Delta \theta)
      if (Score(T, Z_t) > bestScore)
        bestScore = Score(T, Z_t)
        bestPose = (x_0 + dx, y_0 + dy, \theta_0 + d\theta)
      endif
    
```

```

Score(T, Z):
  tmpScore = 0
  for i = 1:n
    z'_i = Tz_i
    tmpScore += M(z'_i)
  endfor
  return tmpScore
  
```



相关方法(Correlation-based Method)



位姿搜索

2. 预先投影搜索

交换暴力搜索中的循环顺序，
最外层对 θ 进行搜索，生成投影矩阵时，
 \sin 和 \cos 的计算从 $n_x n_y n_\theta$ 降低到 n_θ

Score 函数中的点投影运算乘法运算消除，
只剩下加法运算

```

bestPose = (x_0, y_0, \theta_0)
bestScore = Score(V2T(bestPose), Z_i)
for d\theta \in (-\Delta\theta, \Delta\theta)
    R = R(\theta + d\theta)
    Z'_i = RZ_i
    for dx \in (-\Delta x, \Delta x)
        for dy \in (-\Delta y, \Delta y)
            t = [x_0 + \Delta x \quad y_0 + \Delta y]^T
            T = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix}
            if (Score(T, Z'_i) > bestScore)
                bestScore = Score(T, Z'_i)
                bestPose = (x_0 + dx, y_0 + dy, \theta_0 + d\theta)
            endif

```

```

Score(T, Z):
    tmpScore = 0
    for i = 1:n
        z'_i = z + t_i
        tmpScore += M(z'_i)
    endfor
    return tmpScore

```



相关方法(Correlation-based Method)



位姿搜索

3. 多分辨率搜索

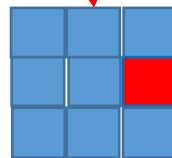
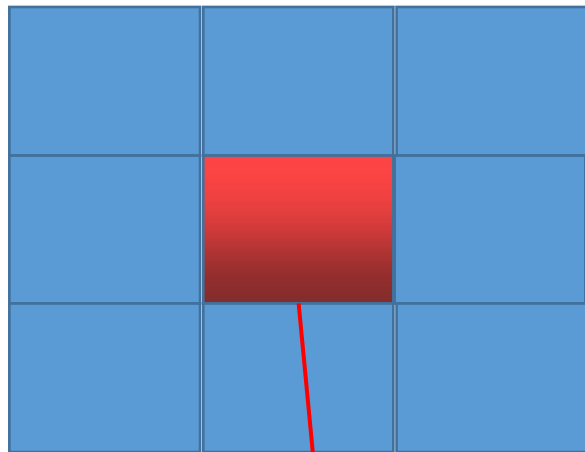
(1)构造粗分辨率($25cm$)和
细分辨率($2.5cm$)两个似然场

(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

不能保证得到最优
解!!!





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(深度为 n)表示整个解空间，深度 $n-1$ 子节点表示解空间的子空间，深度 $n-2$ 的节点表示深度 $n-1$ 节点的子空间，这样层层划分，直到划分到真实解，也就是叶子节点(深度为0)为止。
- 定界：对于搜索树中的每一个节点，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)
- SLAM中一个多棵树同时搜索，每棵树表示一个固定的角度



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  $\text{score}(c) > \text{best\_score}$  then  
      solution  $\leftarrow c$   
      best_score  $\leftarrow \text{score}(c)$   
    end if  
  else  
    if  $\text{score}(c) > \text{best\_score}$  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搜索范围: $(c_x, c_y, c_\theta, c_h)$

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

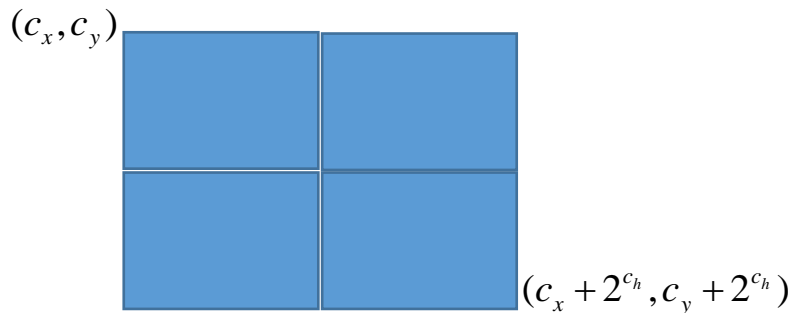
(c_x, c_y) 表示搜索空间的起点

c_h 表示树的深度

c_θ 表示该节点对应的角度

- 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:

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





相关方法(Correlation-based Method)



分枝定界在相关方法的加速作用

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

$$\begin{aligned}
 score(c) &= \sum_{k=1}^K \max_{j \in \overline{\mathcal{W}}_c} M_{\text{nearest}}(T_{\xi_j} h_k) \quad \longrightarrow \quad \text{对于每一个激光点，遍历该节点的每一个搜索位姿，选择得分最高的位姿的得分，作为该搜索空间的得分} \\
 &\geq \sum_{k=1}^K \max_{j \in \overline{\mathcal{W}}_c} M_{\text{nearest}}(T_{\xi_j} h_k) \\
 &\geq \max_{j \in \overline{\mathcal{W}}_c} \sum_{k=1}^K M_{\text{nearest}}(T_{\xi_j} h_k). \quad \longrightarrow \quad \text{以节点C为根节点的子树的最高得分}
 \end{aligned}$$

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

$\overline{\mathcal{W}}$ 表示地图的可行区域

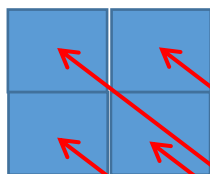
$M_{\text{nearest}}(X)$ 表示地图中点X的似然值



相关方法(Correlation-based Method)



多分辨率地图



激光运动范围



位姿搜索范围



遍历位姿等价于遍历激光点击中范围

得分函数可以通过多分辨率地图的方式解决

机器人位姿不变，不同层节点构造不同分辨率地图



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.

```



参考资料

- [1] A Flexible and Scalable SLAM System with Full 3D Motion Estimation
- [2] The Normal Distributions Transform: A New Approach to Laser Scan Matching
- [3] Real-Time Correlative Scan Matching
- [4] Real-Time Loop Closure in 2D LIDAR SLAM



作业



详细见作业说明



结语

感谢聆听！
Thanks for Listening

