位姿变换中的离群点去除方法

First A. Author*. Second B. Author, Jr.**
Third C. Author***

**National Institute of Standards and Technology, Boulder, CO 80305 USA (Tel: 303-555-5555; e-mail: author@ boulder.nist.gov).*
***Colorado State University, Fort Collins, CO 80523 USA (e-mail: author@lamar. colostate.edu)*
**** Electrical Engineering Department, Seoul National University, Seoul, Korea, (e-mail: author@snu.ac.kr)}*

Abstract: xx.

Keywords: Registration, Scan-matching, Lidar, Transformation.

1. INTRODUCTION

results.

2. 问题描述

在 SLAM 算法中，通常将每个关键帧当作一个节点，使用传感器数据计算不同关键帧之间的位姿变换（旋转矩阵 R 和平移向量 t）；这些位姿变换相当于节点之间的边，那么就可以使用图的形式来表示各节点之间的关系；然而在这个图中，某些位姿变换可能存在离群点，使得对应的边与图中其他边不相容；

同样的情况也出现在传感器网络中；在传感器网络定位问题中，每个传感器代表一个节点，传感器可以测到得到自身与其他节点之间的位姿变换；那么我们可以使用图的形式来表示这个传感器网络；其中图的节点表示每个传感器，图的边表示传感器之间的位姿变换；因为噪声等因素的影响，直接测量得到的不同节点之间的位姿变换可能存在离群点，使得对应边与图中其他边不相容；对此，我们提出一种算法去除图中存在离群点的位姿变换，或者这些不相容的边；

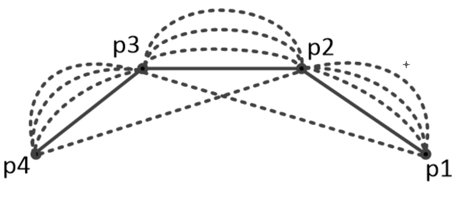

图 1. 节点位置和测量举例

图 1 给出了该问题的例子；该图中共有 4 个节点，任意两个节点之间的一条虚线表示一次测量得到的位姿变换；我们使用旋转平移矩阵$T \in R^{4\times4}$来表示位姿变换；节点 1 和节点 2 之间存在 3 次测量，得到位姿$T_{12}^1, T_{12}^2, T_{13}^3$；以此类推，节点 2 和节点 3 之间存在 3 次测量，节点 3 和节点 4 之间存在 3 次测量；然后不相邻的节点 1 和节点 3 之间存在 1 次测量，得到位姿变换T_{13}；不相邻的节点 2 和节点 4 之间存在 1 次测量，得到位姿变换T_{24}；然后我们假设节点 3 到节点 4 的测量中，某一次测量T_{34}^3是离群点；那么该算法的目的是识别出这个离群点并去除；

3. 算法流程

设p_1, p_2是图中的两个节点；p_1和p_2之间共进行了 3 次测量，得到 3 个位姿变换矩阵$T_{12}^1, T_{12}^2, T_{12}^3$；

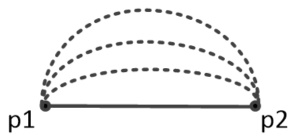

图 2.不借助中心节点测量

同时p_1, p_2经过另外一个节点（如p_3, p_4, p_5），也能构成连线；这样p_1和p_2之间的位姿变换可以借助中间节点表示如下；

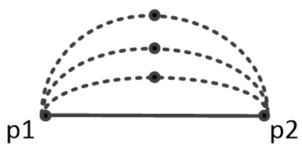

图 3.借助 1 个中间节点测量

$$T_{12}^4 = T_{13}T_{32}$$

$$T_{12}^5 = T_{14}T_{42}$$

$$T_{12}^6 = T_{15}T_{52}$$

同样的方式，p_1和p_2经过另外两个节点（如p_6与p_7、p_8与p_9、p_{10}与p_{11}）也能构成连线；这样p_1和p_2之间的位姿变换可以借助两个中间节点表示；

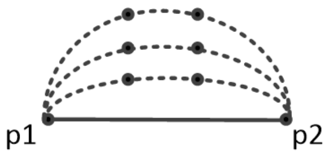

图 4.借助 2 个中间节点测量

$$T_{12}^{7} = T_{16}T_{67}T_{72}$$

$$T_{12}^{8} = T_{18}T_{89}T_{92}$$

$$T_{12}^{9} = T_{1,10}T_{10,11}T_{11,2}$$

以此类推，p_1和p_2之间的位姿变换可以借助多个中间节点来表示；这些中间节点的寻找，可以使用各种路径规划算法（如 A*算法）；假设通过该方式，共获得了 n 个p_1, p_2之间的位姿变换$T_{12}^{i}, i \in [1, n]$；

如果在计算p_1和p_2之间位姿变换时，涉及的所有节点之间的位姿变换只存在普通噪声（如方差较小的正态分布噪声），不存在离群点噪声，那么计算得到的$T_{12}^{i}, i \in [1, n]$应该比较接近；相反如果某些节点之间的位姿变换存在离群点噪声，会使得计算得到的$T_{12}^{i}, i \in [1, n]$中某些数值存在离群点；

我们将位姿变换矩阵T_{12}^{i}拆分为旋转矩阵R_{12}^{i}和平移向量t_{12}^{i}，分别旋转矩阵R_{12}^{i}和平移向量t_{12}^{i}去除离群点；如果某个位姿变换矩阵$T_{12}^{i}, i = i_1$对应的旋转矩阵R_{12}^{i1}或平移向量t_{12}^{i1}中，任意一个存在离群点，那么就认为该位姿变换矩阵T_{12}^{i1}属于离群点；在旋转矩阵$R_{12}^{i}, i \in [1, n]$和平移向量$t_{12}^{i}, i \in [1, n]$中去除离群点的具体步骤在后续章节中具体描述；

已知p_1和p_2点之间共获得了 n 个位姿变换，$T_{12}^{i}, \in [1, n]$；通过离群点去除算法，我们识别出某个位姿变换$T_{12}^{i}, i = i1$是离群点；如果T_{12}^{i1}只经过一条边（没有经过其他节点），那么T_{12}^{i1}对应的边属于离群点的概率 p=1；如果T_{12}^{i1}经过两条边（如$T_{12}^{4} = T_{13}T_{32}$），那么我们不能确定离群点具体位于$T_{13}$还是$T_{32}$，或者$T_{13}$和$T_{32}$都有离群点，那么可以设这每条属于离群点的概率 p=0.5；如果T_{12}^{i1}经过三条边（$T_{12}^{7} = T_{16}T_{67}T_{72}$），我们不能确定离群点具体位于$T_{16}$、$T_{67}$或$T_{72}$，或者都存在离群点；那么可以设每条边属于离群点的概率 p=1/3，以此类推；

通常输入的数据是图的形式$G = [V, E]$，图的节点 V 表示 SLAM 中的关键帧或者传感器网络中的节点，图的边 E 表示不同节点之间的位姿变换；采用上述方式，我们可以对图中任意两个相邻的节点p_k和p_{k+1}计算涉及的位姿变换$T_{k,k+1}^{i}, i \in [1, n]$；然后筛选出属于离群点的位姿变换$T_{k,k+1}^{i}, i \in S$，然后将相关的边属于离群点的概率 p 不断累加；

如果两个相邻节点p_k和p_{k+1}，通过路径规划算法（如 A*算法）寻找到的路径数量太少，使得可以使用的位姿变换$T_{k,k+1}^{i}, i \in [1, n_1]$数量太少（$n_1 < n_{thres}$），那么可以不进行离群点检测；因为数据点数量太少，会使得离群点检测的结果不可靠；

我们可以对图$G = [V, E]$中任意两个相邻节点$p_k, p_{k+1} \in V$，使用上述步骤寻找离群点，并累加每条边$e_i \in E$属于离群点的概率p_i；当图中任意两个相邻节点都判断后，可以根据每条边$e_i \in E$属于离群点的概率，删除对应的边；比如，如果边e_i对应的概率p_i大于阈值p_{thres}（可以取 1，或者更小数值），那么这条边可以判断为离群点；经过上述步骤，图$G = [V, E]$中剩余的位姿变换（图的边）会更加可靠；使用这些剩余的位姿变换，可以计算得到更加精确的节点坐标（使用 SLAM 中的算法或者传感器网络定位的算法）；该算法的伪代码如下；

位姿变换中离群点去除算法伪代码

输入：图$G = [V, E], e_i \in E$表示位姿变换

输出：去除离群点后的图$G' = [V, E']$

For 任意相邻点$p_k, p_{k+1} \in V$

  使用路径规划算法，尝试寻找p_k到p_{k+1}的n_{expect}条路径，实际寻找到n条路径；

  If $n < n_{thres}$

    break

  对于p_k到p_{k+1}的n条路径，计算p_k到p_{k+1}的位姿变换，$T_{k,k+1}^{i}, i \in [1, n]$；

  使用离群点去除算法，得到$T_{k,k+1}^{i}, i \in [1, n]$中存在如下离群点，$T_{k,k+1}^{i}, i \in S$；

  For $i \in S$

    设$T_{k,k+1}^{i}$由m条边e_j对应的位姿变换相乘得到，$e_j, j \in [1, m]$；

    p_j是边e_j属于离群点的概率，$p_j = p_j + \frac{1}{m}, j \in [1, m]$；

For $e_i \in E$

  If $p_i > p_{thres}$

    e_i对应的位姿变换存在离群点，将e_i从E中删除；

得到去除离群点的图$G' = [V, E']$；

---

4. 离群点去除算法

4.1.加权 IQR 的离群点去除算法

离群点去除算法有很多，比如基于密度的离群点去除算法（DBSCAN）、基于聚类的离群点去除算法（K-means 算法）、基于分布的离群点去除算法（IQR）等；在当前问题中，我们假设数据点只有一个聚集的中心，并且为了减少计算量，所以选择 IQR 算法（四分位距）去除离群点；为了将 IQR 算法更好的应用于在当前问题，我们对 IQR 算法进行改进，提出一种加权的 IQR 算法；

我们首先简要介绍 IQR 算法的流程；设输入的数据为 $X=\{x_i\in R,i\in[1,n]\}$，其中包括 n 个数据点$x_i,i\in[1,n]$，并且数据点经过从小到大排序；然后需要计算第一四分位数（Q1）、第三四分位数（Q3）和四分位距 IQR；其中第一四分位数（Q1）计算如下；当数据点数量 n 为奇数时，Q1 是位置为$0.25\times(n+1)$的数据点（可以通过插值计算）；当数据点数量为偶数时，Q1 是第$0.25\times n$和$0.25\times(n+1)$两个位置之间的插值。Q3 可以通过类似的方式计算；对于奇数个数据点，Q3 是$0.75\times(n+1)$位置的值；对于偶数个数据点，Q3 是第$0.75\times n$和$0.75\times(n+1)$之间的插值。四分位距 IQR 是 Q3 与 Q1 的差值；

$$\mathrm{IQR}=\mathrm{Q3}-\mathrm{Q1}$$

根据 IQR 计算离群点的上下界。通常，使用$k=1.5$来定义离群点的判定范围；离群点的下界通过公式 $\mathrm{Q1}-1.5\times\mathrm{IQR}$ 计算。离群点的上界通过公式 $\mathrm{Q3}+1.5\times\mathrm{IQR}$ 计算。这两个界限定义了一个合理的数据范围，位于此范围之外的数据点被视为离群点。

根据上述步骤计算的上下界，将数据集中小于下界或大于上界的数据点标记为离群点。具体条件如下，如果 $x_i<\mathrm{Q1}-1.5\times\mathrm{IQR}$，则$x_i$被视为下侧的离群点。如果 $x_i>\mathrm{Q3}+1.5\times\mathrm{IQR}$，则$x_i$被视为上侧的离群点。可以通过以下方式过滤数据，只保留位于 $[\mathrm{Q1}-1.5\times\mathrm{IQR},\mathrm{Q3}+1.5\times\mathrm{IQR}]$ 之间的数据点。

IQR 算法的优势在于采用分位数计算，对离群点不敏感，并且计算量小；当把 IQR 算法用于当前问题时，数据点为$R_{12}^i,i\in[1,n]$；此时有些R_{12}^i是直接测量得到的，有些R_{12}^i是由多个位姿变换相乘得到的，不同的R_{12}^i的可靠性是不同的；因此我们提出一种加权的 IQR 算法；

设输入的数据为$X=\{x_i\in R,i\in[1,n]\}$，其中包括 n 个数据点$x_i,i\in[1,n]$，并且数据点经过从小到大排序；数据点 X 集合对应的权重集合为$W=\{w_i\in R,i\in[1,n]\}$；此处的权重w_i可以表示数据点x_i的可靠性或者非离群点的概率；这个通常可以根据测量得到x_i的传感器的方差决定，传感器的方差越小，数据x_i的可靠性越高；那么我们在计算主要数据分布区间（IQR 的区间）时，应该使用更多可靠性高的数据点；

相比于普通的 IQR 算法，加权 IQR 算法需要在计算 Q1、Q2 的步骤做出如下修改；我们需要计算一个累加的权重集合$W_{add}=\{w_i^{add}\in R,i\in[1,n]\}$，其中元素$w_i^{add}$计算如下；

$$w_i^{add}=\sum_{j=1}^{i}w_i$$

总的权重和W_{sum}计算如下；

$$W_{sum}=\sum_{j=1}^{n}w_i$$

第一四分位数（Q1）计算如下；在累加的权重集合W_{add}中寻找$0.25\times W_{sum}$相等的 1 个权重w_{i1}^{add}（$w_{i1}^{add}=0.25\times W_{sum}$），那么$\mathrm{Q1}=x_{i1}$；或者在累加的权重集合$W_{add}$中寻找$0.25\times W_{sum}$最接近的 2 个权重$w_{i1}^{ad}d$和$w_{i1+1}^{add}$（$w_{i1}^{add}<0.25\times W_{add}<w_{i1+1}^{add}$），那么$\mathrm{Q1}=(x_{i1}+x_{i1+1})/2$；第三四分位数（Q3）可以类似的即使；在累加的权重集合W_{add}中寻找$0.75\times W_{sum}$相等的 1 个权重w_{i2}^{add}（$w_{i2}^{add}=0.75\times W_{sum}$），那么$\mathrm{Q1}=x_{i2}$；或者在累加的权重集合$W_{add}$中寻找$0.75\times W_{sum}$最接近的 2 个权重$w_{i2}^{add}$和$w_{i2+1}^{ad}d$（$w_{i2}^{add}<0.75\times W_{add}<w_{i2+1}^{add}$），那么 $\mathrm{Q1}=(x_{i2}+x_{i2+1})/2$；其他步骤中，加权 IQR 算法和普通的 IQR 的相同；那么加权 IQR 算法的伪代码如下；

加权 IQR 算法伪代码

输入：数据集合$X=\{x_i\in R,i\in[1,n]\}$，权重集合 $W=\{w_i\in R,i\in[1,n]\}$

输出：去除离群点的数据集合$X'=\{x_i'\in R,i\in[1,m]\}$

累加的权重集合$W_{add}=\{w_i^{add}\in R,i\in[1,n]\},w_i^{add}=\sum_{j=1}^{i}w_i$；

总的权重和$W_{sum}=\sum_{j=1}^{n}w_i$；

For $i\in[1,n]$

  If($w_i^{add}=0.25\times W_{sum}$)

    $\mathrm{Q1}=x_i$

  If($w_i^{add}<0.25\times W_{sum}$ and $w_{i+1}^{add}>0.25\times W_{sum}$)

    $\mathrm{Q1}=(x_i+x_{i+1})/2$

For $i\in[1,n]$

  If($w_i^{add}=0.75\times W_{sum}$)

    $\mathrm{Q3}=x_i$

  If($w_i^{add}<0.75\times W_{sum}$ and $w_{i+1}^{add}>0.75\times W_{sum}$)

    $\mathrm{Q3}=(x_i+x_{i+1})/2$

IQR=Q3-Q1；

保留$[\mathrm{Q1}-1.5\times\mathrm{IQR},\mathrm{Q3}+1.5\times\mathrm{IQR}]$之间的数据点；

4.2 权重计算

该算法中在p_1和p_2点之间获得了 n 个位姿变换，$T_{12}^i,\in[1,n]$；其中有些位姿变换T_{12}^i是直接测量得到的，有些位姿变换T_{12}^i是使用多个位姿变换矩阵相乘得到；那么不同的位姿变换T_{12}^i具有不同的可靠性；

我们设图 G=[V,E]中每条边$e_i\in E$的权重为w_i^e；这个权重表示这条边对应的位姿变换T_{ei}属于非离群点的先验概率，或者表示位姿变换T_{ei}的可靠性；权重w_i^e可以根据测量的传感器的可靠性或者方差来预先设定；假设p_1和p_2点借助p_3点构成连线，那么p_1,p_2点之间的位姿变换如下；

$$T_{12}=T_{13}T_{32}$$

设T_{13}对应的权重为w_{13}（非离群点的概率），T_{32}对应的权重为w_{32}；那么T_{12}对应的权重w_{12}如下；

$$w_{12}=w_{13}w_{32}$$

通常情况下，要求T_{13}是非离群点并且T_{32}也是非离群点，这样计算出来的T_{12}才是非离群点；因此T_{12}属于非离群点的概率是w_{13}和w_{32}的乘积；然而也存在这样的概率，T_{13}是离群点并且T_{32}是离群点，计算出来的T_{12}因为巧合不是离群点；但这样的概率通常比较小，并且为了简化计算，我们忽略这种情况的概率；此处我们讨论了第一种情况，当位姿变换T_{12}^i对应的 m 条边$e_j, j \in S$，并且每条边e_j对应的权重为w_j^e；那么位姿变换T_{12}^i对应的权重w_{12}^i如下；

$$w_{12}^i = \prod_{j \in S} w_j^e$$

使用上述步骤，我们可以计算得到 n 个位姿变换$T_{12}^i, i \in [1,n]$对应的权重，$w_{12}^i, i \in [1,n]$；

4.3 平移向量中离群点去除

我们将位姿变换T_{12}^i中的旋转矩阵R_{12}^i和平移向量t_{12}^i分开，依次去除离群点；下面首先描述平移向量$t_{12}^i, i \in [1,n]$中去除离群点的方法；如果整个系统在平面上运行，那么平移向量$t_{12}^i, i \in [1,n]$如下；

$$t_{12}^i = \left[tx_{12}^i, ty_{12}^i\right]^T \in R^2, i \in [1,n]$$

对$t_{12}^i, i \in [1,n]$中每个分量（tx_{12}^i和ty_{12}^i），依次使用加权 IQR 算法去除离群点；只要某个分量被判断为离群点，那么整个数据点t_{12}^i就被当作离群点去除；

如果整个系统在三维空间上运行，那么平移向量$t_{12}^i, i \in [1,n]$如下；

$$t_{12}^i = \left[tx_{12}^i, ty_{12}^i, tz_{12}^i\right]^T \in R^3, i \in [1,n]$$

对$t_{12}^i, i \in [1,n]$中每个分量（tx_{12}^i、ty_{12}^i和tz_{12}^i），依次使用加权 IQR 算法去除离群点；只要某个分量被判断为离群点，那么整个数据点t_{12}^i就被当作离群点去除；三维空间中平移向量的离群点去除方式伪代码如下；

平移向量中离群点去除算法伪代码

输入：平移集合$t_{12}^i = \left[tx_{12}^i, ty_{12}^i, tz_{12}^i\right]^T \in R^3, i \in [1,n]$，权重集合$w_i \in [1,n]$

输出：去除离群点的集合，$t_{12}^i \in S$

对分量tx_{12}^i使用加权IQR算法去除离群点，得到$tx_{12}^i, i \in S_1$；

对分量ty_{12}^i使用加权IQR算法去除离群点，得到$ty_{12}^i, i \in S_2$；

对分量tz_{12}^i使用加权IQR算法去除离群点，得到$tz_{12}^i, i \in S_3$；

去除离群点的平移集合，$t_{12}^i, i \in S, S = S_1 \cap S_2 \cap S_3$；

4.4 二维旋转矩阵中离群点去除

如果整个系统在平面上运行，那么位姿变换矩阵 T 如下；

$$T = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \in R^{3\times3}$$

$$R = \begin{bmatrix} r_{11} & r_{12} \\ r_{21} & r_{22} \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix} \in R^{2\times2}$$

我们可以将旋转矩阵 R 转为单个旋转角度θ；

$$\theta = \arcsin(r_{12}) \in [0,2\pi]$$

按照这个方式，n 个位姿变换矩阵$T_{12}^i, i \in [1,n]$可以得到如下旋转角度集合$\theta_i \in [0,2\pi], i \in [1,n]$；但是因为旋转角度具有周期性，如果直接使用离群点去除算法对θ_i集合处理，会遇到如下问题；如果多个θ_i数值分布在 0 附近，那么有些角度是略大于 0 的数值，有些角度是略小于2π的数值；因为角度值跨越分界线，会使得原本相差很小的角度在数值上差异很大；

对此我们提出如下方式；已知角度集合为$\theta_i \in [0,2\pi], i \in [1,n]$，将这些角度转换为单位圆上的坐标$P = [\cos(\theta_i), \sin(\theta_i)], i \in [1,n]$；因为我们假设这些数据只有一个聚集的中心，那么在二维平面上，这些单位圆上的坐标也应该只有一个聚集中心；下面对集合 P 中的两个分量（$\cos(\theta_i)$和$\sin(\theta_i)$）依次使用加权 IQR 算法去除离群点；只要某个分量被判断为离群点，那么对应的θ_i就被当作离群点去除；该算法的伪代码如下；

二维旋转矩阵中离群点去除算法伪代码

输入：旋转矩阵集合$R_{12}^i, i \in [1,n]$，权重集合$w_{12}^i, i \in [1,n]$

输出：去除离群点的集合$R_{12}^i, i \in S$

转换为角度集合，$\theta_i \in [0,2\pi], i \in [1,n], \theta_i = \arcsin(r_{12}^i)$；

转换为单位圆坐标，$[\cos(\theta_i), \sin(\theta_i)], i \in [1,n]$；

对分量$\cos(\theta_i)$使用加权IQR算法去除离群点，得到$\cos(\theta_i), i \in S_1$；

对分量$\sin(\theta_i)$使用加权IQR算法去除离群点，得到$\sin(\theta_i), i \in S_2$；

去除离群点的旋转矩阵集合为，$R_{12}^i, i \in S, S = S_1 \cap S_2$；

4.5 三维旋转矩阵中离群点去除

如果整个系统在三维空间中运行，那么位姿变换矩阵 T 如下；

$$T = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \in R^{4\times4}$$

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \in R^{3\times3}$$

我们可以将旋转矩阵 R 转为旋转向量 v；

$$v = \theta n \in R^3$$

其中$n \in R^3$是单位向量，表示旋转轴；$\theta \in [0,2\pi]$是旋转角度；按照这个方式，n 个位姿变换矩阵$T_{12}^i, i \in [1,n]$可以得到如下旋转向量集合$v_i = \theta_i n_i \in R^3, i \in [1,n], \theta_i \in [0,2\pi]$；但是因为旋转角度具有周期性，如果直接使用离群点去除算法对$v_i, i \in [1,n]$集合处理，会遇到如下问题；如果多个v_i对应的θ_i数值分布在 0 附近，那么有些角度是

略大于 0 的数值，有些角度是略小于2π的数值；因为角度值跨越分界线，会使得原本相差很小的角度在数值上差异很大；那么会使得这些旋转向量v_i在三维空间上的位置相差很大；

对此我们提出如下方式；首先需要寻找旋转向量$v_i, i \in [1,n]$集合中旋转角度$\theta_i, i \in [1,n]$的聚集范围；设$\theta_i, i \in [1,n]$集合中大部分角度值分布在θ_{ave}附近，那么我们可以将θ_i的表示范围从$[0,2\pi]$调整为$[\theta_{ave} - \pi, \theta_{ave} + \pi]$；然后对调整后的旋转向量$v_i, i \in [1,n]$集合的每个分量使用加权 IQR 算法去除离群点；

具体处理步骤如下；已知角度集合为$\theta_i \in [0,2\pi], i \in [1,n]$，将这些角度转换为单位圆上的坐标$P = [\cos(\theta_i), \sin(\theta_i)], i \in [1,n]$；因为我们假设这些数据只有一个聚集的中心，那么在二维平面上，这些单位圆上的坐标也应该只有一个聚集中心；下面对集合 P 中的两个分量（$\cos(\theta_i)$和$\sin(\theta_i)$）依次使用加权 IQR 算法去除离群点；只要某个分量被判断为离群点，那么对应的θ_i就被当作离群点去除；假设剩余的角度集合为，$[\cos(\theta_i), \sin(\theta_i)], i \in S$；那么角度分布的平均位置如下；

$$\theta_{ave} = \frac{1}{|S|}\sum_{i \in S}\theta_i$$

角度集合中每个元素调整为如下形式，$\theta_i \in [\theta_{ave} - \pi, \theta_{ave} + \pi], i \in [1,n]$；然后旋转向量调整为如下；

$$v_i = \theta_i n_i \in R^3, i \in [1,n]$$

$$\theta_i \in [\theta_{ave} - \pi, \theta_{ave} + \pi], i \in [1,n]$$

对$v_i, i \in [1,n]$中每个分量（v_i^1、v_i^2和v_i^3），依次使用加权 IQR 算法去除离群点；只要某个分量被判断为离群点，那么整个数据点R_{12}^i就被当作离群点去除；三维空间中旋转矩阵的离群点去除方式伪代码如下；

三维旋转矩阵中离群点去除算法伪代码

输入：旋转矩阵集合$R_{12}^i, i \in [1,n]$，权重集合$w_{12}^i, i \in [1,n]$

输出：去除离群点的集合$R_{12}^i, i \in S$

转换为旋转向量，$v_i = \theta_i n_i \in R^3, \theta_i \in [0,2\pi]$；

旋转角度集合为，$\theta_i \in [0,2\pi], i \in [1,n]$；

旋转角度转为为单位圆坐标，$[\cos(\theta_i), \sin(\theta_i)], i \in [1,n]$；

对单位圆坐标分量$\cos(\theta_i)$使用加权IQR算法去除离群点，得到$\cos(\theta_i), i \in S_1$；

对单位圆坐标分量$\sin(\theta_i)$使用加权IQR算法去除离群点，得到$\sin(\theta_i), i \in S_2$；

计算角度分布均值，$\theta_{ave} = \frac{1}{|S|}\sum_{i \in S}\theta_i, i \in S_1 \cap S_2$；

角度集合转换为，$\theta_i \in [\theta_{ave} - \pi, \theta_{ave} + \pi], i \in [1,n]$；

旋转向量集合转换为，$v_i = [v_i^1, v_i^2, v_i^3]^T = \theta_i n_i \in R^3, \theta_i \in [\theta_{ave} - \pi, \theta_{ave} + \pi], i \in [1,n]$；

对旋转向量中分量v_i^1使用加权IQR算法去除离群点，得到$v_i^1, i \in Q_1$；

对旋转向量中分量v_i^2使用加权IQR算法去除离群点，得到$v_i^2, i \in Q_2$；

对旋转向量中分量v_i^3使用加权IQR算法去除离群点，得到$v_i^3, i \in Q_3$；

去除离群点的旋转矩阵集合为，$R_{12}^i, i \in Q_1 \cap Q_2 \cap Q_3$；

---

5. 路径规划算法

5.1 Dijkstra 算法

该算法需要在p_1和p_2点之间寻找多个位姿变换，$T_{12}^i, \in [1,n]$；该问题相当于寻找从p_1到p_2的多条路径，可以使用 Dijkstra 算法；Dijkstra 算法在寻找路径时没有启发式函数提供的方向；如果我们以p_1点为起点，Dijkstra 算法从p_1点开始往周围扩散，计算每个点对应的最短路径代价，直到遇到p_2点结束；但是该算法中，我们通常只要求p_1和p_2点之间最多经过一个或者两个其他点构成连线；如果p_1和p_2点之间经过太多点构成连线，那么累加得到p_1, p_2的位姿变换会因为噪声而变得不再准确；所以即使 Dijksra 算法在寻找路径时没有启发式函数提供的方向，在寻找的路径很短时，增加的计算量并没有很多；

我们设图$G = [V, E]$中每条边e_i的权重为w_i^e；这个权重表示这条边对应的位姿变换T_{ei}属于非离群点的概率，或者表示位姿变换T_{ei}的可靠性；权重w_i^e可以根据测量的传感器的可靠性或者方差来预先设定；

假设p_1和p_2点借助p_3点构成连线，那么p_1, p_2点之间的位姿变换如下；

$$T_{12} = T_{13}T_{32}$$

设T_{13}对应的权重为w_{13}（非离群点的概率），T_{32}对应的权重为w_{32}；那么T_{12}属于非离群点的概率w_{12}如下；

$$w_{12} = w_{13}w_{32}$$

那么我们需要寻找一条从p_1到p_2的路径，使得累计的权重（非离群点的概率）最大化；然后上述形式的误差很难用于路径规划算法，比如 Dijkstra 算法要求路径的误差是累加形式；因为我们对上述误差取对数再乘-1，得到如下；

$$-\log(w_{12}) = -\log(w_{13}w_{32})$$

$$-\log(w_{12}) = -\log(w_{13}) - \log(w_{32})$$

我们使用每条边的权重w_i^e的对数乘-1，作为这条边在路径规划算法中的权重，$-\log(w_i^e)$；首先上述权重形式符合路径代价是累加的形式；然后在没有取负数之前，我们需要寻找路径代价w_{12}的最大值；在取负数后，我们需要寻找$-\log(w_{12})$的最小值，这个符合路径规划算法中寻找最短路径的要求；然后边的权重w_i^e是[0,1]的数值，取

$-\log(w_i^e)$的范围在[0,+inf]，符合 Dijkstra 算法对边权重是非负数的要求；

在该算法中，我们需要寻找p_1到p_2的 n 条路径，要求这些路径的长度尽可能短（或者路径代价尽量小），并且要求这些路径尽量不使用重复边；不使用重复边的原因是，如果两条路径中大量边是重复的，那么这两条路径对应的p_1, p_2之间的位姿变换的独立性很低，不能代表两次独立的p_1, p_2之间位姿变换的测量；

对此我们可以采用如下方式获得 n 条路径；首先使用 Dijkstra 算法寻找图$G = [V, E]$中p_1到p_2的最短路径；然后对于这条路径上的所有边的权重，都设置为一个非常大的数值Δ（例如10^5），其他边的权重不变；然后再次使用 Dijkstra 算法寻找从p_1到p_2的最短路径；此时路径规划算法会尽量避免使用上次路径中已经使用过的边（因为代价为Δ），除非必须使用某条边，否则无法从p_1到达p_2；依次类推，每次获得一条新的从p_1到p_2的路径中，就把已经使用的边的权重设为Δ；该算法的停止条件可以是寻找到 n 条路径，或者寻找到一条新的路径，该路径的所有边都是由重复边组成（路径权重为$k \times \Delta$，k 是路径的边数量）；该算法的伪代码如下；

使用 Dijkstra 算法寻找 n 条路径伪代码

输入：图$G = [V, E]$，边的权重$w_i^e, i \in [1, |E|]$，路径数量n

输出：从p_1到p_2的k条路径，$k \le n$

计算负对数权重，$\omega_i^e = -\log(w_i^e), i \in [1, |E|]$；

For j in 1 to n

  使用ω_i^e作为边权重，使用Dijkstra算法寻找p_1到p_2的最短路径，路径代价为C[j]，路径为Path[j]；

  将该路径对应的所有边权重设置为Δ（10^5）；

  If(Path[j]的边数*Δ=C[j])

    删除Path[j], break

输出p_1到p_2的k条路径，$k \le n$；

4.2 A*算法

该算法需要在p_1和p_2点之间寻找多个位姿变换，$T_{12}^i, \in [1, n]$；该问题相当于寻找从p_1到p_2的多条路径；除了使用 Dijkstra 算法，也可以采用 A*算法；A*算法采用如下目标函数来评估路径上的每个节点；

$$f(n) = g(n) + h(n)$$

其中$f(n)$表示从起点p_1到终点p_2的总估计代价；$g(n)$表示从起点p_1到节点 n 的实际代价（已经走过的所有边权重和）；$h(n)$是启发式函数，用于估计从节点 n 到终点p_2的估计代价；该函数$h(n)$给算法提供了终点的方向，使得算法可以更快的到达终点p_2；如果不使用函数$h(n)$，那么 A*算法会退化为 Dijkstra 算法；

A*算法中的启发式函数$h(n)$，我们可以采用欧几里得距离形式；

$$h(n) = \lambda \sqrt{\left(x_n - x_{p2}\right)^2 + \left(y_n - y_{p2}\right)^2}$$

该启发式函数要求预先知道图$G = [V, E]$中每个节点的三维坐标；这个可以使用没有去除离群点的图$G = [V, E]$先计算一组粗略的每个节点的坐标，用于该启发式函数；A*算法中启发式函数的可接受性要求$h(n)$不能高于真实的代价$h^*(n)$，否则寻找的路径可能是次优路径；然后我们使用没有去除离群点的图$G = [V, E]$计算得到的节点坐标的粗略的，所以如果直接使用欧式距离，可能使得$h(n)$大于真实的代价$h^*(n)$；因为我们在欧式距离上乘一个系数$\lambda \in [0,1]$；使用 A*算法在寻找p_1到p_2的路径中，其他步骤和使用 Dijkstra 算法相同；

REFERENCES

Brown, F., Harris, M.G., and Other, A.N. (1998). Name of paper. In Name(s) of editor(s) (ed.), *Name of book in italics*, page numbers. Publisher, Place of publication.

Smith, S.E. (2004). *Name of book in italics*, page or chapter numbers if relevant. Publisher, Place of publication.

Smith, S.E. and Jones, L.Q. (2008). Name of paper. *Name of journal in italics*, volume (number), page numbers.

【待补充，调整格式后加入】