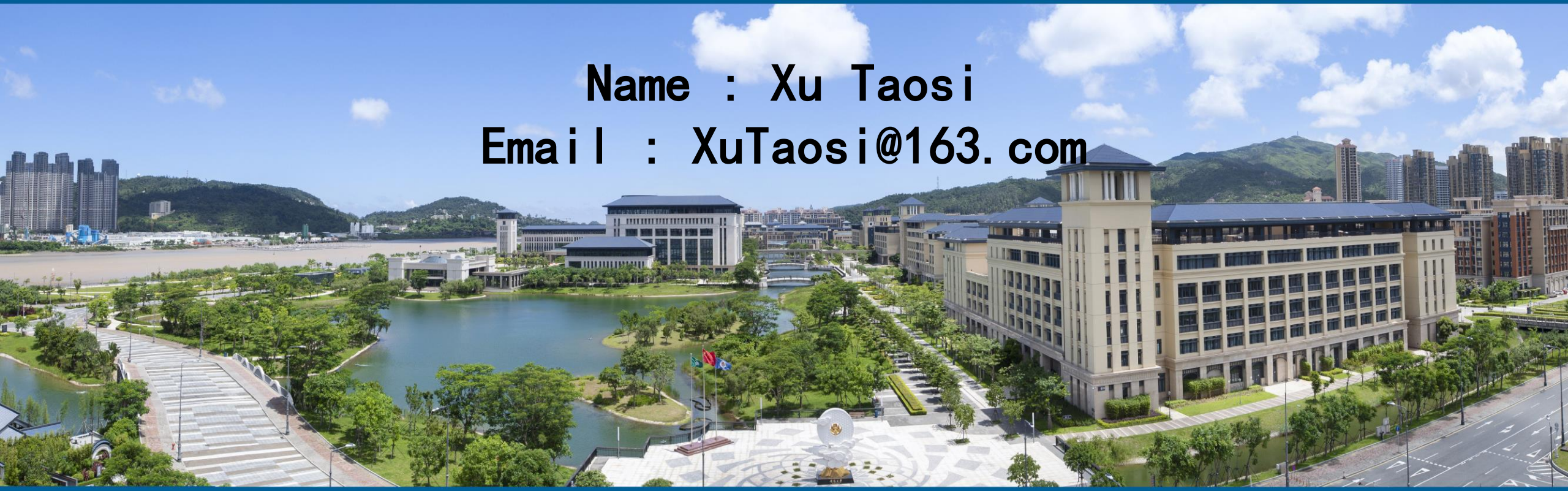
澳門大學
UNIVERSIDADE DE MACAU
UNIVERSITY OF MACAU

ARCS: Accurate Rotation and Correspondence Search

Name : Xu Taosi
Email : XuTaosi@163.com



ARCS： 精确 的旋转和对 应 搜索

彭良祖; 马诺利斯·萨基里斯（Manolis C.Tsakiris）；
勒内·维达尔

2022 IEEE/CVF 计算机视觉与模式识别会议 （CVPR）
年份： 2022 |会议论文|发布者： IEEE
被引|文献： 论文 （2）

ARCS: Accurate Rotation and Correspondence Search

- 提出问题
- 现阶段研究成果
- ARCS+方法介绍
- 实验
- 总结

Procrustes分析问题和Wahba问题

- Procrustes分析问题是一种数学问题，旨在通过对两个或多个数据集进行旋转、缩放和平移操作，将它们尽可能地对齐或匹配。

- Wahba问题是一种经典的数学问题，旨在寻找最佳旋转矩阵，将两个点集在三维空间中对齐。它的目标是找到一个旋转矩阵，最小化两个点集之间的差异。我们也可称这个问题为“同时旋转和对应搜索”。

问题一：同时旋转与对应搜索

Problem 1: Simultaneous Rotation and Correspondence Search Consider point sets $\mathcal{Q} = \{\boldsymbol{q}_1, \ldots, \boldsymbol{q}_m\} \subset \mathbb{R}^3$ and $\mathcal{P} = \{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n\} \subset \mathbb{R}^3$ with $m \geq n$. Let $\mathcal{C}^*$ be a subset of $[m] \times [n] := \{1, \ldots, m\} \times \{1, \ldots, n\}$ of size k^*, called the *inlier correspondence set*, such that all pairs (i_1, j_1) and (i_2, j_2) of $\mathcal{C}^*$ satisfy $i_1 \neq i_2$ and $j_1 \neq j_2$. Assume that

$$\boldsymbol{q}_i = \boldsymbol{R}^* \boldsymbol{p}_j + \boldsymbol{\epsilon}_{i,j}, \quad \text{if } (i,j) \in \mathcal{C}^* \tag{1}$$

where $\boldsymbol{\epsilon}_{i,j} \sim \mathcal{N}(0, \sigma^2 \boldsymbol{I}_3)$ is noise, $\boldsymbol{R}^*$ is an unknown 3D rotation, and $(\boldsymbol{q}_i, \boldsymbol{p}_j)$ is called an *inlier*. If $(i,j) \notin \mathcal{C}^*$ then $(\boldsymbol{q}_i, \boldsymbol{p}_j)$ is arbitrary and is called an *outlier*. The goal of the simultaneous rotation and correspondence search problem is to simultaneously estimate the 3D rotation $\boldsymbol{R}^*$ and the *inlier correspondence set* $\mathcal{C}^*$ from point sets $\mathcal{Q}$ and $\mathcal{P}$.

C^*:内点对集合，由m个点与n个点之间的索引对(i , j)组成的子集，每个索引对表示了一个初始的对关系。要求C^*中的每个索引都对应不同的q_i和p_i。

K^*:内点的层数，即我们只考虑了K^*个最相关的对应关系，为了限制搜索空间，减少计算的复杂度。

ϵ_{ij}:噪音，服从正态分布，其中均值为零表示噪声的期望值为零，即在平均情况下不引入任何偏差。协方差矩阵$\sigma^2 I_3$表示噪声在不同维度上的方差相等且无关，其中σ^2表示方差。

R^*:旋转矩阵，用于将点集P中的点与点集Q中的点相对应。

问题一的目标：

- 同时估计位置的旋转矩阵R^*和内点对集合C^*，以找到最佳的点集对应关系并消除离群值，从而获得点集Q和点集P之间的准确对应关系。

为解决问题一早期做的尝试：

- **经典的ICP算法**：是一种在点云配准（点云对齐）问题中常用的迭代优化算法

算法的具体步骤如下：

1. 初始化：选择一个初始的变换矩阵，将待配准的目标点云与参考点云进行初始对齐。
2. 最近点匹配：对于目标点云中的每个点，找到参考点云中距离最近的点，建立对应关系。
3. 配准变换估计：使用最近点匹配结果，估计目标点云到参考点云之间的刚体变换（如平移和旋转）。
4. 更新变换：将估计的变换应用到目标点云上，得到一个新的配准后的目标点云。
5. 终止条件判断：检查变换的改变是否满足终止条件，如果满足则结束算法，否则返回步骤2。
6. 迭代优化：如果终止条件不满足，继续迭代执行步骤2到步骤5，直到达到最优的配准结果。

为解决问题一早期做的尝试：

- **GO-ICP方法**：改进的ICP算法

算法的具体步骤如下：

1. 构建KD树（一种二叉树数据结构）：使用参考点云构建KD树，以加速最近点匹配的过程。

2. 最近点匹配：对于目标点云中的每个点，通过搜索KD树找到最近的参考点，并建立点对应关系。这种最近点匹配方法比传统的线性搜索更高效。

3. 配准变换估计：使用最近点匹配结果，通过最小二乘或最大似然估计方法计算目标点云到参考点云之间的刚体变换（如平移和旋转）。

4. 更新变换：将估计的变换应用到目标点云上，得到一个新的配准后的目标点云。

5. 终止条件判断：检查变换的改变是否满足终止条件，如果满足则结束算法，否则返回步骤2。

6. 迭代优化：如果终止条件不满足，继续迭代执行步骤2到步骤5，直到达到最优的配准结果。

准确性与可伸缩性的两难困境

准确性：指算法在估计旋转和对应关系时的精确程度。
可伸缩性：指算法在处理大规模数据时的效率和速度。

经典的ICP算法以交替的方式估计旋转和对应，实时运行，但需要高质量且通常不可用的初始化，以避免局部最小值和通常较差的最小值;同样的准确性限制也适用于它的后继者

而在GO-ICP中，通过枚举不同的初始化方式来提供给ICP算法的初始值，以在指数时间内达到全局最小值;同样的运行时间限制也适用于它的后继者

为提高准确性与可伸缩性做出的改善

1、通过手工制作或学习的特征描述符计算候选通信集$\overline{C}$:
选择该方法的原因：候选通信集C包含了可能的点对应关系通过计算特征描述符（(特征描述符是用于描述点云中的特征信息的向量或描述子（描述子是一种表示和描述数据特征的向量或特征向量）)，可以筛选出具有相似特征的点对，从而减少了匹配的搜索空间，提高了配准的效率和准确性。

2、估计由$\overline{C}$索引的点集的旋转矩阵：
选择该方法的目的是：根据候选通信集$\overline{C}$中点对应关系的索引，可以估计点集之间的旋转变换。通过计算点集之间的旋转，可以将目标点云相对于参考点云进行合适的旋转，以实现两个点云的配准

3、缺点
但通常会由于特征描述符质量不佳而导致$\overline{C}$中的内点数量少于2个，那么就无法通过这些内点来准确估计旋转变换。这就解释了为什么研究人员最近专注于为极端异常值率(例如100个异常值中≥90个)设计稳健的旋转搜索算法

在极端异常值率下的稳健旋转搜索算法

GORE算法：这是一种有保证的异常值去除算法，时间复杂度为$O(l^2 \log l)$，它大量利用了SO(3)的几何特性。SO(3)代表三维特殊正交群（Special Orthogonal Group in three dimensions），也被称为旋转群，它是所有三维旋转矩阵的集合。它的几何特性如下：

1. 旋转性质：SO(3)中的每个元素都表示一个三维空间中的旋转操作。可以通过SO(3)的元素应用于三维向量来实现旋转操作。
2. 刚体变换：SO(3)可以用于描述刚体在三维空间中的旋转。通过SO(3)的乘法运算，可以将一个刚体的旋转表示为另一个刚体的旋转。
3. 无歧义性：SO(3)的元素表示唯一的旋转，不受坐标系的选择或表示方式的影响。任意两个SO(3)的元素表示不同的旋转，且它们之间不存在连续的变换。
4. 紧致性：SO(3)是紧致的，意味着它是有限且闭合的。SO(3)中的旋转操作不会导致向量的无限增长或趋近于无穷远。
5. 拓扑结构：SO(3)具有拓扑结构，可以被视为三维球面或三维实射影空间。

在极端异常值率下的稳健旋转搜索算法

GORE算法步骤：

1. 特征匹配：首先，通过特征提取和特征描述子计算，将两个视觉帧之间的特征点进行匹配。
2. 初始估计：使用匹配的特征点对计算一个初始的旋转估计。
3. 离群值检测：基于初始估计，计算每个特征点对的旋转残差。通过计算残差的标准差或中位数，识别出可能是异常值的特征点对。
4. 旋转搜索：对于可能是异常值的特征点对，采用旋转搜索的方法进行迭代优化，以找到更准确的旋转估计。在搜索过程中，利用几何约束和优化算法，尽可能地减小异常值对旋转估计的干扰。
5. 异常值移除：根据搜索得到的最终旋转估计，对于被确认为异常值的特征点对，将其移除，得到一个去除异常值的旋转结果。
6. 不断迭代

在极端异常值率下的稳健旋转搜索算法

半定松弛类星体算法：半定松弛类星体是指一类基于半定松弛的优化方法应用于星体姿态估计问题，它涉及对单位四元数的复杂操作

半定松弛（Semidefinite Relaxation）是一种将原始非凸优化问题转化为半定规划问题的方法。在半定松弛中，通过引入半定矩阵变量和相应的线性约束，将原始问题的约束条件进行松弛，从而得到一个更容易求解的凸优化问题。

"松弛"（Relaxation）指的是将一个复杂的问题转化为一个更简单但相关的问题的过程。通常，这个简化的问题更容易求解，但给出的解可能只是原始问题的近似解。

四元数是一种扩展的复数，由一个实部和三个虚部组成，通常表示为q = a + bi + cj + dk，其中a、b、c、d为实数，i、j、k为虚部。

在极端异常值率下的稳健旋转搜索算法

半定松弛类星体算法：半定松弛类星体是指一类基于半定松弛的优化方法应用于星体姿态估计问题，它涉及对单位四元数的复杂操作

1. 确定问题的形式：将原始的非凸优化问题转化为半定松弛问题。可以通过定义合适的目标函数和约束条件来形式化问题。
2. 半定松弛：将原始问题中的约束条件进行松弛，将其转化为半定松弛问题。这可以通过引入半定矩阵变量和相应的线性约束来实现。
3. 构建半定松弛问题：将原始问题的目标函数和约束条件表达为半定矩阵变量的线性组合。这通常涉及计算矩阵的迹、乘积和线性组合等操作。
4. 求解半定松弛问题：使用半定松弛优化算法对构建的半定松弛问题进行求解。这可以通过内点法、迭代法或凸优化求解器等方法来实现。
5. 解析原问题：将得到的半定松弛问题的解析解转化为原始问题的近似解。这可能涉及将半定矩阵变量映射回原始问题的变量空间，并根据需要进行进一步的处理。
6. 验证与调整：对求解得到的近似解进行验证和调整。这可以包括检查约束条件的满足程度、检查解的稳定性和鲁棒性等。
7. 迭代优化：根据需要，可以通过进一步的迭代和调整来改进近似解的质量。这可能涉及修改目标函数、调整松弛程度或增加额外的约束等。

在极端异常值率下的稳健旋转搜索算法

TEASER++算法：是一种用于点云配准的算法，它可以鲁棒地估计两个点云之间的刚体变换。该算法对离群值的鲁棒性主要来自于通过并行分支定界找到图的一个最大团，由于寻找最大团通常是NP-困难的(非确定性多项式时间（nondeterministic polynomial time）的缩写)，他们的算法在最坏的情况下需要指数级的时间。

分支定界（Branch and Bound）：是一种用于解决组合优化问题的算法。它是一种穷举搜索方法，通过不断分割问题空间并剪枝来寻找最优解，这种方法的优势在于，通过找到最大团，TEASER++可以在处理离群值时剔除不一致的点对，从而提高配准算法的鲁棒性。只有那些在最大团中的点对才会被用于计算刚体变换，而离群值点对则被排除在外

最大团表示一组最大的互相连接的节点，没有其他节点可以加入而保持团的性质。在TEASER++中，最大团中的点对表示具有高度一致性的内点

以上的方法都是准确性大于伸缩性的，下面介绍两种伸缩性大于准确性的算法

在极端异常值率下的稳健旋转搜索算法

FGR结合了渐变非凸性(GNC)和交替最小化：

FGR(Fast Global Registration)算法的主要思想是将点云配准问题转化为一个优化问题，通过最小化点云之间的对应误差来求解最佳的刚体变换。关键步骤：
1. 初始粗配准：通过一种快速的预配准方法，获取初始的粗略配准结果。
2. 特征提取：从配准后的点云中提取特征描述符，通常使用局部特征如FPFH（Fast Point Feature Histograms）或SHOT（Signature of Histograms of Orientations）。
3. 特征匹配：通过对特征描述符进行匹配，建立点云之间的对应关系。匹配过程可以使用一些快速的近似匹配方法，如最近邻搜索或哈希表。
4. 刚体变换估计：利用特征匹配的结果，通过最小化点云之间的对应误差，估计得到最佳的刚体变换矩阵（旋转和平移）。
5. 细化优化：对刚体变换进行细化优化，以进一步提高配准的准确性。这可以通过非线性优化方法，如Levenberg-Marquardt算法来实现。

渐变非凸性:是一种优化技术，用于处理非凸优化问题，通过使用梯度信息来寻找全局最优解
交替最小化:是一种迭代优化方法，通过交替更新变量来逐步求解最优解

在极端异常值率下的稳健旋转搜索算法

FGR结合了渐变非凸性(GNC)和交替最小化：

1. 初始化变换参数，如平移矩阵或旋转矩阵。

2. 在每次迭代中，首先固定一个变换参数，如平移或旋转，并优化其他参数。这可以通过最小化一个损失函数来实现（损失函数它用于衡量模型的预测输出与实际观测值之间的差异或误差程度）

3. 接下来，固定刚刚优化过的参数，并优化其他参数。这个过程继续进行，直到达到收敛条件。

4. 在每次迭代中，利用梯度信息来引入非凸性，以更好地捕捉图像间的非线性变换。

5. 为了加快收敛速度和降低计算复杂度，可以使用快速梯度法或其他高效的优化算法来求解子问题

在极端异常值率下的稳健旋转搜索算法

GNC-TLS：结合截断最小二乘、迭代重加权最小二乘和渐变非凸性(GNC)

1. 输入两幅待配准的图像，以及初始的变换参数。
2. 通过计算图像的梯度，即每个像素点处的梯度向量（表示了该像素点在图像中的灰度变化情况）。
3. 初始化迭代次数为0。
4. 在每次迭代中，执行以下步骤：

a. 将一幅图像根据当前的变换参数进行变换（如平移、旋转等），得到变换后的图像。

b. 计算变换后的图像与另一幅图像之间的渐变误差，即两幅图像在梯度方向上的差异。这可以通过计算图像梯度的差异来实现。

c. 利用全最小二乘法，将渐变误差和总体误差结合起来，建立一个优化问题。全最小二乘法考虑了数据中的噪声，并尽量减小整体误差。

d. 使用优化算法（如Levenberg-Marquardt算法）求解上述优化问题，得到调整后的变换参数。

e. 检查调整后的变换参数与上一次迭代的变换参数之间的差异。如果差异小于设定的阈值或达到最大迭代次数，则停止迭代；否则，返回步骤4继续迭代。

5. 输出最终的配准结果，即两幅图像对齐的变换参

在极端异常值率下的稳健旋转搜索算法

GNC-TLS：结合截断最小二乘、迭代重加权最小二乘和渐变非凸性(GNC）：

截断最小二乘：是一种回归方法，通过将误差函数中的大误差值进行截断来提高鲁棒性。

迭代重加权最小二乘：是一种迭代优化方法，通过重新分配权重来处理异常值。

总结：

这两种方法都具有鲁棒性，并能够适应异常值的存在。（鲁棒性是指算法对异常值或噪声的鲁棒性，即在存在异常值的情况下，仍能够产生准确的结果）且它们能够处理高达80%的异常值，并仍能保持良好的配准效果。

在极端异常值率下的稳健旋转搜索算法

GNC-TLS：结合截断最小二乘、迭代重加权最小二乘和渐变非凸性(GNC）：

截断最小二乘：是一种回归方法，通过将误差函数中的大误差值进行截断来提高鲁棒性。

迭代重加权最小二乘：是一种迭代优化方法，通过重新分配权重来处理异常值。

总结：

这两种方法都具有鲁棒性，并能够适应异常值的存在。（鲁棒性是指算法对异常值或噪声的鲁棒性，即在存在异常值的情况下，仍能够产生准确的结果）且它们能够处理高达80%的异常值，并仍能保持良好的配准效果。

思考：是否存在同时具有准确性和可扩展性的算法呢？

ARCS算法

定理1 (ARCS)：假设至少有两个内层， $k^* \geqslant 2$ ，并且问题1的点集Q和P是无噪声的“一般位置”。然后有一个算法在O(mlogm)时间和O(m)空间内解决问题1。

$k^* \geqslant 2$:是为了确保问题具有足够的约束和信息来进行准确的求解，内层是指数据集中的一组相对较接近的点，它们在数据集中形成了一种结构或模式

O(mlogm):线性对数时间复杂度
O(m):线性时间复杂度

这时间复杂度都代表了运行时间增长速率不快，相对而言这算法比较高效。

ARCS算法

备注1（一般位置假设）：在定理1中，通过“一般位置”，我们的意思是：
（1）对于任何离群值(q_i, p_j)，我们有$||q_i||_2 \neq ||p_j||_2$。
（2）存在一些内点对(q_{i1}, p_{j1})和(q_{i2}, p_{j2})使得q_{i1}和q_{i2}不平行。
如果点集Q和P随机从R^3中抽样，则这两个条件成立的概率为1。

假设（1）的主要原因是为了确保离群点能够在问题求解过程中被正确地区分和处理。如果P和Q中的离群点具有相似的欧氏距离，可能会导致算法在求解匹配或对应关系时将它们错误地匹配在一起。通过要求离群点的欧氏距离与其他离群点不相等，可以确保离群点之间的差异性，从而减少它们对其他点的干扰。

假设（2）的主要原因是如果所有内点对的向量都平行，它们所提供的几何信息将非常有限。通过要求存在一些内点对的向量不平行，可以提供更多的几何信息，这有助于算法更好地确定正确的对应关系

如果点集 Q 和 P 是从三维空间中随机采样得到的，那么满足一般位置假设的概率为1。这意味着在大多数情况下，我们可以假设点集 Q 和 P 满足一般位置的条件

ARCS算法步骤

Algorithm 1: ARCS

Input: $\mathcal{Q} = \{\boldsymbol{q}_i\}_{i=1}^{m}, \mathcal{P} = \{\boldsymbol{p}_j\}_{j=1}^{n}, c \geq 0$;
Sort $\mathcal{Q}$ so that (w.l.o.g.) $\|\boldsymbol{q}_1\|_2 \leq \cdots \leq \|\boldsymbol{q}_m\|_2$;
Sort $\mathcal{P}$ so that (w.l.o.g.) $\|\boldsymbol{p}_1\|_2 \leq \cdots \leq \|\boldsymbol{p}_n\|_2$;
$i = 1; j = 1; \overline{\mathcal{C}} = \varnothing$;
while $i \leq m$ *and* $j \leq n$ **do**
    $d_{i,j} \leftarrow \|\boldsymbol{q}_i\|_2 - \|\boldsymbol{p}_j\|_2$;
    if $d_{i,j} > c$ **then**
        $j \leftarrow j + 1$;
    end
    if $d_{i,j} < -c$ **then**
        $i \leftarrow i + 1$;
    end
    if $-c \leq d_{i,j} \leq c$ **then**
        $\overline{\mathcal{C}} \leftarrow \overline{\mathcal{C}} \cup (i, j)$; $(i, j) \leftarrow (i + 1, j + 1)$;
    end
end
return $\overline{\mathcal{C}}$;

输入：点集Q和点集P
输出：内点集合$\bar{C}$
计算Q和P的差分：
$d_{ij} = ||q_i||_2 - ||p_j||_2$

ARCS算法解问题1

同时估计位置的旋转矩阵R^*和内点对集合C^*，以找到最佳的点集对应关系并消除离群值，从而获得点集Q和点集P之间的准确对应关系

R^*可以用奇异值分解(SVD)从两个内线对中确定出来（奇异值分解是一种矩阵分解方法，用于将矩阵分解为三个矩阵的乘积。通过应用SVD，可以从这两个内点对中确定出R^*）。

1、构建矩阵A和B：A = $[[q_{1x}, q_{2x}], [q_{1y}, q_{2y}], [q_{1z}, q_{2z}]]$
2、计算A和B的中心化矩阵：A' = A - mean(A) B' = B - mean(B)
(矩阵的每一列减去每一列的均值，减完后的矩阵每一列均值为0，中心化后的矩阵记为 A' 和 B')
3、计算协方差矩阵：计算矩阵 A' 和 B' 的协方差矩阵 C。
C = (1/n)*A'* B'^T（n是A'和 B'的列数）
4、进行奇异值分解：对协方差矩阵 C 进行奇异值分解，得到左奇异向量矩阵 U、奇异值矩阵 S 和右奇异向量矩阵 V。 C = U * S * V^T
5、计算旋转矩阵 R*：通过计算旋转矩阵 R*，将点云 Q 旋转和平移到与点云 P 对齐的位置。
R* = V * U^T

ARCS算法的有效性：

m	10^4	10^5	10^6
n	8×10^3	8×10^4	8×10^5
G	5.9	15.0	212.8
Brute Force	73.8	8304	8380441.5
ARCS	1.51	8.4	121.1

表1：生成无噪声高斯点集(G)并通过ARCS求解问题1的时间(msec)(100次试验，k*= 2)。

蛮力算法：是一种简单而直接的问题求解方法，通过穷举所有可能的解决方案来解决问题。

在§1和§2中所提到的所有方法，如果直接适用，原则上都失效了。一个原因是它们不是为处理无噪音情况而设计的。

ARCS+：增强ARCS抗噪声能力

这里我们考虑带有噪声ϵ_{ij}的问题1。我们将通过假设$\epsilon_{ij} \sim N(0, \delta^2 I_3)$来说明我们的算法思想。ARCS+有三个步骤，我们将在接下来的三个小节中分别介绍它们。

步骤1： ARCS+N:在噪声下找到对应关系

一个简单的概率事实是：

$||q_i - R^* p_j||_2 \leqslant 5.54\sigma$，因此对于任意阶数$(q_i, p_j)$，$||d_{ij}|| \leqslant 5.54\sigma$，且概率至少为$1 - 10^{-6}$（参考[80]）。

Remark 20 (Probabilistic inlier noise). *If we assume the inliers follow the generative model (4) with* $\boldsymbol{\epsilon}_i \sim \mathcal{N}(\mathbf{0}_3, \sigma_i^2 \mathbf{I}_3)$ *and* $\boldsymbol{o}_i = \mathbf{0}$*, it holds:*

$$\frac{1}{\sigma_i^2}\|\boldsymbol{b}_i - \boldsymbol{R}\boldsymbol{a}_i\|^2 = \frac{1}{\sigma_i^2}\|\boldsymbol{\epsilon}_i\|^2 \sim \chi^2(3), \tag{29}$$

where $\chi^2(3)$ *is the Chi-squared distribution with three degrees of freedom. Therefore, with desired probability* p*, the weighted error* $\frac{1}{\sigma_i^2}\|\boldsymbol{\epsilon}_i\|^2$ *for the inliers satisfies:*

$$\mathbb{P}\left(\frac{\|\boldsymbol{\epsilon}_i\|^2}{\sigma_i^2} \leq \gamma^2\right) = p, \tag{30}$$

where γ^2 *is the quantile of the* χ^2 *distribution with three degrees of freedom and lower tail probability equal to* p (e.g., $\gamma = 3$ *for* $p = 0.97$)*. Therefore, one can simply set the noise bound* β_i *in Problem (6) to be* $\beta_i = \sigma_i$*, set the* $\bar{c} = \gamma$*. As mentioned in Section IV, from optimization standpoint, this is equivalent to setting* $\beta_i = \gamma\sigma_i$ *and setting* $\bar{c} = 1$*. The parameter* γ *monotonically increases with* p*; therefore, setting* p *close to 1 makes the formulation (6) more prone to accept*

ARCS+N:在噪声下找到对应关系

为了建立噪声下的对应关系，我们需要修改ARCS算法中的while循环，使其在O(I+mlogm)时间内返回所有大小为I的对应关系的集合C。

我们需要限制每个(i,j)∈C满足$|d_{ij}|\leqslant C$，此时C设为5.54σ，
为了存储输出对应，我们需要额外的O(I)时间，因为在存在噪声的情况下，I通常大于m。

我们将此修改版本称为ARCS+N，在这个算法中的内点集合C^*中包含了一组对应关系，这些对应关系具有至少$(1-10^{-6})^{k^*}$的概率成为真实对应关系的候选)。如果$k^* \leq 10^3$，则该概率大于99.9%，如果$k^* \leq 10^4$，则该概率大于99%。

ARCS+N:在噪声下找到对应关系

备注2(特征匹配vs全对全对应vs ARCS+N):

特征匹配:我们会提取参考点云A和目标点云B的特征描述子。然后，通过计算描述子之间的相似度，我们可以进行特征匹配，将参考点云A中的特征点与目标点云B中的特征点进行匹配。假设我们通过特征匹配算法得到了100个匹配对应关系。这些匹配对应关系是基于特征描述子相似度计算得到的，但并不能保证所有的匹配都是准确的。特征匹配方法通常受到噪声、遮挡或重复纹理等因素的影响，可能会产生一些误匹配或遗漏匹配的情况。

ARCS+N:在噪声下找到对应关系

备注2(特征匹配vs全对全对应vsARCS+N):

特征匹配:我们会提取参考点云A和目标点云B的特征描述子。然后，通过计算描述子之间的相似度，我们可以进行特征匹配，将参考点云A中的特征点与目标点云B中的特征点进行匹配。这些匹配对应关系是基于特征描述子相似度计算得到的，但并不能保证所有的匹配都是准确的。特征匹配方法通常受到噪声、遮挡或重复纹理等因素的影响，可能会产生一些误匹配或遗漏匹配的情况，即无法保留全部内点。

全对全对应：可以保留所有的内点，但是简单的计算需要O(mn)时间，并且会导致具有极端离群率的大规模问题。

ARCS+N：通过在O(l+mlogm)时间内提供一个大小为l的候选对应集$\bar{C}$来达到平衡，该候选对应集$\bar{C}$包含了所有内点，并且有很高的概率$l \ll mn$(可由图二证明)

ARCS+N:在噪声下找到对应关系

m	1000	5000	10000
n	800	4000	8000
k^*	200	1000	2000
ℓ	36622	931208	3762888
$\ell/(mn)$	4.58%	4.66%	4.70%

表2：在增加噪声的高斯点集上，ARCS+N产生的候选内点集的个数（I）。一次试验。这以是现阶段做的最好的情况，随后我们要去除异常值。

步骤2：ARCS+O 去除异常值

Problem 2. (*robust rotation search*) Consider ℓ pairs of 3D points $\{(\boldsymbol{y}_i, \boldsymbol{x}_i)\}_{i=1}^{\ell}$, with each pair satisfying

$$\boldsymbol{y}_i = \boldsymbol{R}^* \boldsymbol{x}_i + \boldsymbol{o}_i + \boldsymbol{\epsilon}_i. \tag{2}$$

Here $\boldsymbol{\epsilon}_i \sim \mathcal{N}(0, \sigma^2 \boldsymbol{I}_3)$ is noise, $\boldsymbol{o}_i = \boldsymbol{0}$ if $i \in \mathcal{I}^*$ where $\mathcal{I}^* \subset [\ell]$ is of size k^*, and if $i \notin \mathcal{I}^*$ then $\boldsymbol{o}_i$ is nonzero and arbitrary. The task is to find $\boldsymbol{R}^*$ and $\mathcal{I}^*$.

- yi 是观测到的三维点；
- xi 是真实的三维点；
- R* 是旋转矩阵；
- oi 是偏移向量，表示两个坐标系之间的平移关系；
- ϵi 是噪声

具体来说，内点集合 I* 是一个大小为 k* 的索引集合，其中 k* 是内点的数量。如果某个索引 i 属于 I*，则对应的 oi 为零，表示这个点是内点。如果某个索引 i 不属于 I*，则对应的 oi 可以是任意非零值，表示这个点是异常点。

目标是找到最佳的旋转矩阵R*和对应的索引集合I*

区间刺穿：寻找在ω处重叠的极大子集

Consider a collection of subsets of $\mathbb{R}$, $\{\mathcal{J}_i\}_{i=1}^{L}$, where each $\mathcal{J}_i$ is an interval of the form $[a, b]$. In the interval stabbing problem, one needs to determine a point $\omega \in \mathbb{R}$ and a subset $\mathcal{I}$ of $\{\mathcal{J}_i\}_{i=1}^{L}$, so that $\mathcal{I}$ is a maximal subset whose intervals overlap at ω. Formally, we need to solve

$$\max_{\mathcal{I}\subset[L],\omega\in\mathbb{R}} \quad |\mathcal{I}| \qquad (3)$$
$$\text{s.t.} \quad \omega \in \mathcal{J}_i, \ \forall i \in \mathcal{I}$$

- $\{J_i\}_{I=1}^{L}$：是R中的子集，形式为[a, b]的区间
- I：是$\{J_i\}_{I=1}^{L}$中最大的重叠子集
- ||I||：表示子集I元素的数量。
- ω：表示一个实数，是我们要优化的变量。
- Ji：表示与元素 i 相关联的一组可能的取值，这些取值限制了ω的可选范围

目标是需要确定点ω∈R和$\{J_i\}_{I=1}^{L}$的子集I，使得I是区间在ω处重叠的极大子集。

区间刺穿

Interval Stabbing（区间刺穿）是一个在计算几何和算法中常见的问题。它涉及到处理一组区间和一组点，任务是确定每个点被多少个区间所"刺穿"。

具体来说，给定一组闭合的区间和一组离散的点，我们想知道每个点被多少个区间所包含。这个问题可以表示为：对于每个点，计算它被多少个区间所“刺穿”，即在每个点处重叠了多少集合。

离群点去除算法

我们现在考虑以下共识最大化:共识最大化是指通过协调个体之间的选择或决策，以达到在一个群体或网络中最大化整体共识的目标。

$$\max_{\mathcal{I}\subset[\ell],\boldsymbol{R}\in\mathrm{SO}(3)} \quad |\mathcal{I}| \qquad (4)$$
$$\text{s.t.} \quad \left\|\boldsymbol{y}_i - \boldsymbol{R}\boldsymbol{x}_i\right\|_2 \leq c, \ \ \forall i \in \mathcal{I}.$$

- I：表示一个子集。
- R：表示一个旋转矩阵，它是我们要优化的变量
- R∈SO(3):是指三维特殊正交群（Special Orthogonal Group in three dimensions）即所有旋转矩阵（3x3的矩阵）的集合。
- c：表示一个阈值，它限制了向量$y_i - Rx_i$的欧氏距离。

这个公式的含义是，在满足约束条件的前提下，找到一个子集 I 和一个旋转矩阵 R，使得子集 I 的基数最大化

降维：从SO(3)到S^2

文献[73]表明，对于非常相关的鲁棒拟合问题［对于给定的数据集，寻找一个拟合模型（例如回归模型），能够在存在异常值或噪声的情况下，尽可能准确地描述数据的整体趋势]，这种共识最大化通常是NP-hard。因此，将我们的计算目标从精确求解(4)转换为近似求解。从SO(3)到s2。为了实现这一目标，我们首先将注意力转移到s2，其中R^*的旋转轴b^*存在。

降维：从SO(3)到S^2

Proposition 1. *Let* $\boldsymbol{v}_i := \boldsymbol{y}_i - \boldsymbol{x}_i$. *Recall* $\boldsymbol{\epsilon}_i \sim \mathcal{N}(0, \sigma^2 \boldsymbol{I}_3)$. *If* $(\boldsymbol{y}_i, \boldsymbol{x}_i)$ *is an inlier pair, then* $\boldsymbol{v}_i^\top \boldsymbol{b}^* \sim \mathcal{N}(0, \sigma^2)$, *and so* $|\boldsymbol{v}_i^\top \boldsymbol{b}^*| \leq 4.9\sigma$ *with probability at least* $1 - 10^{-6}$.

C. Proof of Proposition 1

Since $\boldsymbol{b}^*$ is the rotation axis of $\boldsymbol{R}^*$, we have $(\boldsymbol{R}^*)^\top \boldsymbol{b}^* = \boldsymbol{b}^*$. Recall $\boldsymbol{v}_i = \boldsymbol{y}_i - \boldsymbol{x}_i$ for every $i \in \mathcal{I}$. If $i \in \mathcal{I}^*$ then

$$\boldsymbol{v}_i^\top \boldsymbol{b}^* = (\boldsymbol{y}_i - \boldsymbol{x}_i)^\top \boldsymbol{b}^* = (\boldsymbol{y}_i - \boldsymbol{R}^* \boldsymbol{x}_i)^\top \boldsymbol{b}^*, \tag{34}$$

and further more if $(\boldsymbol{y}_i, \boldsymbol{x}_i)$ is an inlier pair we get that

$$|\boldsymbol{v}_i^\top \boldsymbol{b}^*| = \boldsymbol{\epsilon}_i^\top \boldsymbol{b}. \tag{35}$$

Clearly $\boldsymbol{\epsilon}_i^\top \boldsymbol{b}$ is a Gaussian random variable with zero mean and variance σ^2. The rest of the proof follows from a standard probability calculation.

降维：从SO(3)到S^2

$$\max_{\mathcal{I}\subset[\ell],\boldsymbol{b}\in\mathbb{S}^2} \quad |\mathcal{I}|$$
$$\text{s.t.} \quad |\boldsymbol{v}_i^\top \boldsymbol{b}| \le \bar{c}, \ \forall i \in \mathcal{I} \qquad (5)$$
$$b_2 \ge 0.$$

- I：表示一个子集。
- b：是旋转轴，它是我们要优化的变量。
- S^2：表示二维球面上的单位向量集合。
- $b_2 \geqslant$：是为了消除对称性。这是因为在球面 S2 上，对于一个给定的向量 b，-b 也是一个合法的向量。
- 命题1建议设$\bar{c}$=4.9σ（这个阈值的选择可以保证在高概率下，内点对与b^*的投影之间的误差较小。

这个公式的含义是，在满足约束条件的前提下，找到一个子集I和一个旋转轴b^*，使得子集I的基数最大化。

降维：从S^2到[0, π]

$$\begin{aligned} &\max_{\mathcal{I}\subset[\ell],\theta\in[0,\pi]} \quad |\mathcal{I}| \\ \text{s.t.} \quad &|\boldsymbol{v}_i^\top \boldsymbol{b}| \le \bar{c}, \ \ \forall i \in \mathcal{I} \\ &\boldsymbol{b} = [\sin(\theta)\cos(\phi),\ \sin(\theta)\sin(\phi),\ \cos(\theta)]^\top. \end{aligned} \tag{6}$$

- b 的第一个分量 sin(θ) cos(ϕ) 对应于 b 向量在 x 轴上的投影。
- b 的第二个分量 sin(θ) sin(ϕ) 对应于 b 向量在 y 轴上的投影。
- b 的第三个分量 cos(θ) 对应于 b 向量在 z 轴上的投影。

其中 θ 表示与 z 轴的夹角，ϕ 表示在 xy 平面上的投影与 x 轴的夹角

降维：从S^2到[0，π]

显然，只要固定了参数ϕ，就变成了一个自由度，所以要解决(5)，只需最小化函数f:[0，π]→R，该函数将任意的$\phi_0 \in [0, \pi]$映射到(6)的目标值，其中$\phi=\phi_0$。

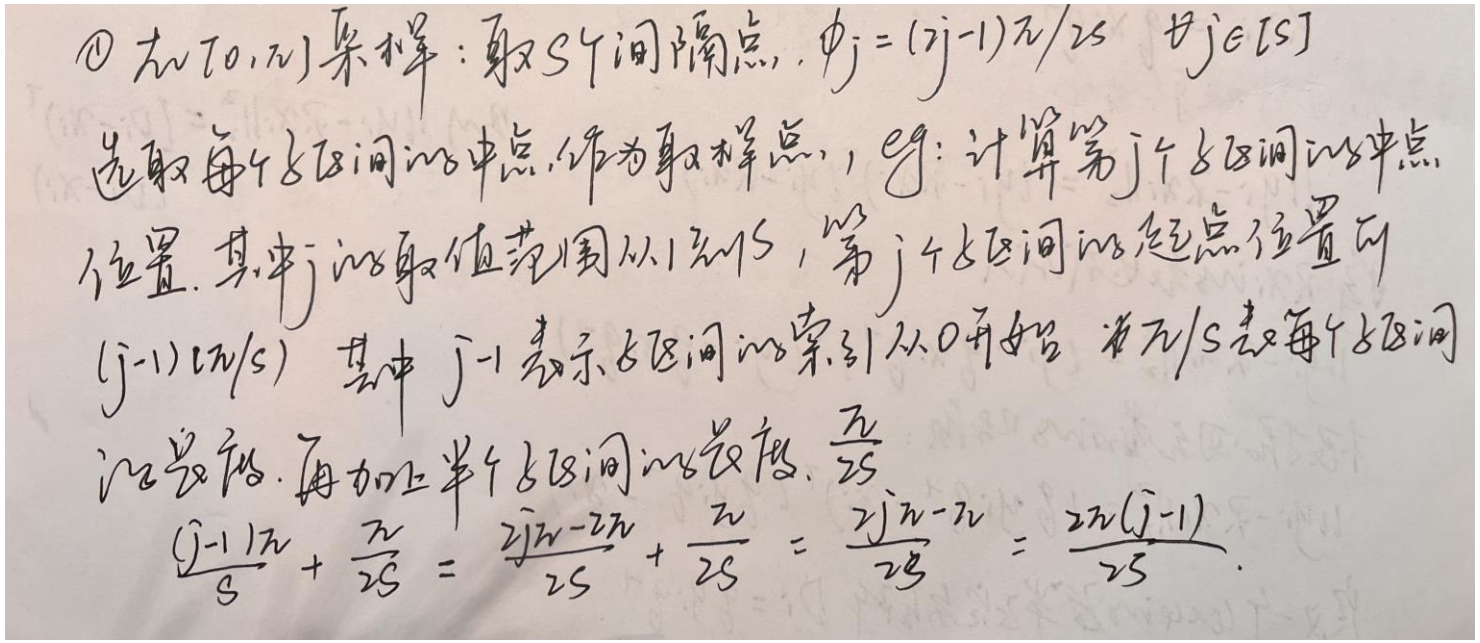
f的定义域是[0，π]，值域是R，函数f将每个ϕ_0映射到一个实数，表示在（6）中当$\phi=\phi_0$时，满足约束条件的解所对应的目标函数的取值。

通过最小化f可确定使得问题（6）达到最优的ϕ值是因为在更小的搜索空间内进行优化，从而提高求解效率。由（6）的解逼近（5）的解再逼近（4）的解）

利用区间刺穿解决问题

第一步：从[0，π]中采样

取s个等间隔点，$\emptyset_j = (2j-1)\pi/(2s)$



利用区间刺穿解决问题

第二步： 在S^2中区间刺穿（为获得候选对应点集合和旋转轴）

对于每个j∈[s]，用ϕ=ϕ_j求解(6)，得到s个候选共识集I_j和s个角度θ_j。从每个ϕ_j和θ_j我们得到一个候选旋转轴b_j。

利用区间刺穿解决问题

第三步： 在SO(3)中区间刺穿（为确定旋转角度）

因为现在我们有旋转轴的估计，bj，还剩下一个自由度，旋转角ω。对此我们考虑

$$\max_{\mathcal{I}\subset[\ell],\omega\in[0,2\pi]} \quad |\mathcal{I}|$$

$$\text{s.t.} \quad \left\|\boldsymbol{y}_i - \boldsymbol{R}\boldsymbol{x}_i\right\|_2 \leq c, \ \ \forall i \in \mathcal{I} \tag{7}$$

$$\boldsymbol{R} = \boldsymbol{b}\boldsymbol{b}^\top + [\,\boldsymbol{b}\,]_\times \sin(\omega) + (\boldsymbol{I}_3 - \boldsymbol{b}\boldsymbol{b}^\top)\cos(\omega)$$

这个等式是旋转矩阵的罗德里格斯公式（Rodrigues' formula），用于计算绕任意轴进行旋转的旋转矩阵。

利用区间刺穿解决问题

$$\max_{\mathcal{I}\subset[\ell],\omega\in[0,2\pi]} \quad |\mathcal{I}|$$

$$\text{s.t.} \quad \left\|\boldsymbol{y}_i - \boldsymbol{R}\boldsymbol{x}_i\right\|_2 \leq c, \ \ \forall i \in \mathcal{I} \tag{7}$$

$$\boldsymbol{R} = \boldsymbol{b}\boldsymbol{b}^\top + [\,\boldsymbol{b}\,]_\times \sin(\omega) + (\boldsymbol{I}_3 - \boldsymbol{b}\boldsymbol{b}^\top)\cos(\omega)$$

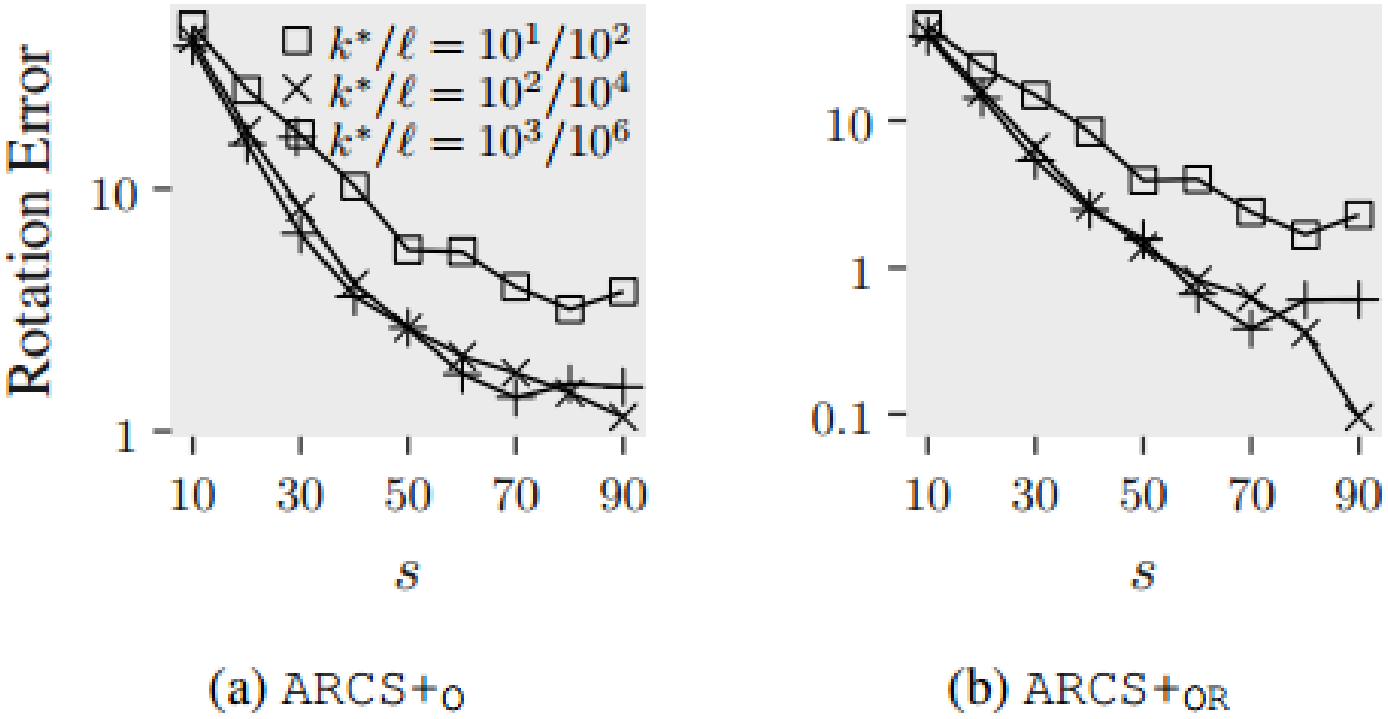
- R 是一个 3x3 的旋转矩阵。
- b 是一个单位向量，表示旋转轴的方向。
- bb^T是一个 3x3 的矩阵，表示外积。在这里是一个投影矩阵，表示旋转轴上的平行投影。
- $[b]_x$是一个反对称矩阵，表示向量 b 的叉乘。在这里表示围绕旋转轴的旋转。
- ω 是旋转的角度。
- I_3 是一个 3x3 的单位矩阵。
- sin(ω) 和 cos(ω) 分别表示角度 ω 的正弦值和余弦值。
- $(I_3 - bb^T)$cos(ω) 是一个对称矩阵，表示沿着与旋转轴垂直的方向的旋转

利用区间刺穿解决问题

$$\begin{aligned} &\max_{\mathcal{I}\subset[\ell],\omega\in[0,2\pi]} \quad |\mathcal{I}| \\ \text{s.t.} \quad &\left\|\boldsymbol{y}_i - \boldsymbol{R}\boldsymbol{x}_i\right\|_2 \leq c, \ \ \forall i \in \mathcal{I} \\ &\boldsymbol{R} = \boldsymbol{b}\boldsymbol{b}^\top + [\,\boldsymbol{b}\,]_\times \sin(\omega) + (\boldsymbol{I}_3 - \boldsymbol{b}\boldsymbol{b}^\top)\cos(\omega) \end{aligned} \tag{7}$$

对每个j∈[s]用b = bj求解(7)，得到s个候选内点集I_1…… I_S，我们选择基数最大的一个作为(4)的近似解。

利用区间刺穿解决问题



(a) ARCS+$_O$ (b) ARCS+$_{OR}$

图1：对于s（s=90）变化的稳健旋转搜索方法(500次试验，σ = 0.01)，步骤2和步骤3的旋转误差(以度为单位)。

步骤3：ARCS+R 旋转估计

ARCS+R是对ARCS+O的输出对应 I进行鲁棒旋转搜索的细化过程，由于ARCS+O已经去除了异常值。如表2和表3所示：

m	1000	5000	10000
n	800	4000	8000
k^*	200	1000	2000
ℓ	36622	931208	3762888
$\ell/(mn)$	4.58%	4.66%	4.70%

表2：在合成噪声高斯点集上，ARCS+N产生的候选对应的个数。一次试验。

Input Inlier Ratio	$\frac{200}{36622}$	$\frac{1000}{931208}$	$\frac{2000}{3762888}$
Output Inlier Ratio	$\frac{199}{213}$	$\frac{993}{1314}$	$\frac{1951}{3184}$

表3：ARCS+O的输出，输入来自表2。

步骤3：ARCS+R 旋转估计

所以我们假设它具有很少的异常值(假设≤50%)。那么，公式如下

$$\min_{\boldsymbol{R}\in \mathrm{SO}(3)} \sum_{i=1}^{\ell} \left\| \boldsymbol{y}_i - \boldsymbol{R}\boldsymbol{x}_i \right\|_2. \tag{8}$$

这个公式是一个优化问题的目标函数，用于描述一个旋转估计的问题。该问题的目标是找到一个旋转矩阵R∈SO(3)（特殊正交群）来最小化给定的点集的残差。

解决问题（8）

步骤3：ARCS+R 旋转估计

Proposition 4. *We have* $\boldsymbol{w}^\top \boldsymbol{D}_i \boldsymbol{w} = \left\| \boldsymbol{y}_i - \boldsymbol{R}\boldsymbol{x}_i \right\|_2^2$, *where* $\boldsymbol{w} \in \mathbb{S}^3$ *is a quaternion representation of* $\boldsymbol{R}$ *of* (8), *and* $\boldsymbol{D}_i \in \mathbb{R}^{4\times 4}$ *is a positive semi-definite matrix whose entries depend on* $\boldsymbol{x}_i$, $\boldsymbol{y}_i$. *So Problem* (8) *is equivalent to*

$$\min_{\boldsymbol{w}\in\mathbb{S}^3} h(\boldsymbol{w}), \quad h(\boldsymbol{w}) = \sum_{i=1}^{\ell} \sqrt{\boldsymbol{w}^\top \boldsymbol{D}_i \boldsymbol{w}}. \tag{9}$$

w$\in S^3$是旋转矩阵R的四元数表示，D_i∈R时一个4×4是一个正半定矩阵，其项依赖于x_i，$\mathrm{y_i}$，于是问题（8）等价于问题（9）。注：（9）的目标函数时凸函数。

命题4的证明过程

A. Proof of Proposition 4

We consider a stronger version of Proposition 4:

Proposition 1. *We have* $\boldsymbol{w}^\top \boldsymbol{D}_i \boldsymbol{w} = \left\|\boldsymbol{y}_i - \boldsymbol{R}\boldsymbol{x}_i\right\|_2^2$, *where* $\boldsymbol{w}$ *is a quaternion representation of* $\boldsymbol{R}$ *of* (8), *and* $\boldsymbol{D}_i \in \mathbb{R}^{4\times 4}$ *is a positive semi-definite matrix whose entries depend on* $\boldsymbol{x}_i, \boldsymbol{y}_i$. *So Problem* (8) *is equivalent to*

$$\min_{\boldsymbol{w}\in\mathbb{S}^3} h(\boldsymbol{w}), \quad h(\boldsymbol{w}) = \sum_{i=1}^{\ell} \sqrt{\boldsymbol{w}^\top \boldsymbol{D}_i \boldsymbol{w}}. \tag{1}$$

Moreover, $\boldsymbol{D}_i$ *has eigenvalues* $4, 4, 0, 0$ *if* $\boldsymbol{x}_i$ *and* $\boldsymbol{y}_i$ *are normalized (that is* $\left\|\boldsymbol{x}_i\right\|_2 = \left\|\boldsymbol{y}_i\right\|_2 = 1$*).*

We first recall some basics about unit quaternions, an algebraic construction invented by Hamilton in the 1840's, when the notion of *vector* does not exist; see the beautiful account of [1]. In our current notation, each element $\boldsymbol{w}$ of $\mathbb{S}^3$ is called a unit quaternion. The most crucial fact is that $\mathrm{SO}(3)$ is isomorphic to the 3-sphere $\mathbb{S}^3$ *up to sign*, that is $\mathrm{SO}(3) \equiv \mathbb{S}^3/\{\pm 1\}$. This implies a two-to-one correspondence between unit quaternions and 3D rotations. Algebraically, any $\boldsymbol{R} \in \mathrm{SO}(3)$ can be written as a 3×3 matrix

$$\begin{array}{lll} w_1^2+w_2^2-w_3^2-w_4^2 & 2(w_2w_3-w_1w_4) & 2(w_2w_4+w_1w_3) \\ 2(w_2w_3+w_1w_4) & w_1^2+w_3^2-w_2^2-w_4^2 & 2(w_3w_4-w_1w_2) \\ 2(w_2w_4-w_1w_3) & 2(w_3w_4+w_1w_2) & w_1^2+w_4^2-w_2^2-w_3^2 \end{array}$$

where $\boldsymbol{w} = [w_1, w_2, w_3, w_4]^\top \in \mathbb{S}^3$. We can now write the three entries of $\boldsymbol{R}\boldsymbol{x}_i$ as quadratic forms $\boldsymbol{w}^\top \boldsymbol{X}_{i,1}\boldsymbol{w}$, $\boldsymbol{w}^\top \boldsymbol{X}_{i,2}\boldsymbol{w}$, and $\boldsymbol{w}^\top \boldsymbol{X}_{i,3}\boldsymbol{w}$, respectively. Here $\boldsymbol{X}_{i,1}$, $\boldsymbol{X}_{i,2}$, and $\boldsymbol{X}_{i,3}$ are 4×4 symmetric matrices, defined as

$$\boldsymbol{X}_{i,1} = \begin{bmatrix} [\boldsymbol{x}_i]_1 & 0 & [\boldsymbol{x}_i]_3 & -[\boldsymbol{x}_i]_2 \\ 0 & [\boldsymbol{x}_i]_1 & [\boldsymbol{x}_i]_2 & [\boldsymbol{x}_i]_3 \\ [\boldsymbol{x}_i]_3 & [\boldsymbol{x}_i]_2 & -[\boldsymbol{x}_i]_1 & 0 \\ -[\boldsymbol{x}_i]_2 & [\boldsymbol{x}_i]_3 & 0 & -[\boldsymbol{x}_i]_1 \end{bmatrix} \tag{2}$$

$$\boldsymbol{X}_{i,2} = \begin{bmatrix} [\boldsymbol{x}_i]_2 & -[\boldsymbol{x}_i]_3 & 0 & [\boldsymbol{x}_i]_1 \\ -[\boldsymbol{x}_i]_3 & -[\boldsymbol{x}_i]_2 & [\boldsymbol{x}_i]_1 & 0 \\ 0 & [\boldsymbol{x}_i]_1 & [\boldsymbol{x}_i]_2 & [\boldsymbol{x}_i]_3 \\ [\boldsymbol{x}_i]_1 & 0 & [\boldsymbol{x}_i]_3 & -[\boldsymbol{x}_i]_2 \end{bmatrix} \tag{3}$$

$$\boldsymbol{X}_{i,3} = \begin{bmatrix} [\boldsymbol{x}_i]_3 & [\boldsymbol{x}_i]_2 & -[\boldsymbol{x}_i]_1 & 0 \\ [\boldsymbol{x}_i]_2 & -[\boldsymbol{x}_i]_3 & 0 & [\boldsymbol{x}_i]_1 \\ -[\boldsymbol{x}_i]_1 & 0 & -[\boldsymbol{x}_i]_3 & [\boldsymbol{x}_i]_2 \\ 0 & [\boldsymbol{x}_i]_1 & [\boldsymbol{x}_i]_2 & [\boldsymbol{x}_i]_3 \end{bmatrix} \tag{4}$$

Defining $\boldsymbol{C}_i := [\boldsymbol{y}_i]_1 \boldsymbol{X}_{i,1} + [\boldsymbol{y}_i]_2 \boldsymbol{X}_{i,2} + [\boldsymbol{y}_i]_3 \boldsymbol{X}_{i,3}$, we get that $\boldsymbol{y}_i^\top \boldsymbol{R}\boldsymbol{x}_i = \boldsymbol{w}^\top \boldsymbol{C}_i \boldsymbol{w}$. And defining

$$\boldsymbol{D}_i = (\left\|\boldsymbol{y}_i\right\|_2^2 + \left\|\boldsymbol{x}_i\right\|_2^2)\boldsymbol{I}_4 - 2\boldsymbol{C}_i \tag{5}$$

with $\boldsymbol{I}_4$ the 4×4 identity matrix, we obtain the equality

$$\begin{aligned} \left\|\boldsymbol{y}_i - \boldsymbol{R}\boldsymbol{x}_i\right\|_2^2 &= \left\|\boldsymbol{y}_i\right\|_2^2 + \left\|\boldsymbol{x}_i\right\|_2^2 - 2\boldsymbol{y}_i^\top \boldsymbol{R}\boldsymbol{x}_i & (6) \\ &= \boldsymbol{w}^\top \boldsymbol{D}_i \boldsymbol{w}. & (7) \end{aligned}$$

Since $\boldsymbol{D}_i$ is symmetric and $\boldsymbol{w}^\top \boldsymbol{D}_i \boldsymbol{w} \geq 0$ for any $\boldsymbol{w} \in \mathbb{S}^3$, we know that $\boldsymbol{D}_i \in \mathbb{R}^{4\times 4}$ is positive semi-definite.

Suppose $\left\|\boldsymbol{y}_i\right\|_2 = \left\|\boldsymbol{x}_i\right\|_2 = 1$. Then there is at least two different 3D rotations $\boldsymbol{R}_1$ and $\boldsymbol{R}_2$ satisfying $\boldsymbol{y}_i = \boldsymbol{R}_1\boldsymbol{x}_i = \boldsymbol{R}_2\boldsymbol{x}_i$. Thus, with the factorization $\boldsymbol{D}_i = \boldsymbol{Z}_i\boldsymbol{Z}_i^\top$, there are at least two quaternions $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$ with $\boldsymbol{w}_1 \neq \pm\boldsymbol{w}_2$ satisfying that $\boldsymbol{Z}_i^\top \boldsymbol{w}_1 = \boldsymbol{Z}_i^\top \boldsymbol{w}_2 = 0$. So $\mathrm{rank}(\boldsymbol{D}_i) = \mathrm{rank}(\boldsymbol{Z}_i) \leq 2$. Recalling $\boldsymbol{D}_i = 2\boldsymbol{I}_4 - 2\boldsymbol{C}_i$, we see that 1 is an eigenvalue of $\boldsymbol{C}_i$ that has multiplicity at least 2. Similarly, we can derive that $\left\|\boldsymbol{y}_i + \boldsymbol{R}\boldsymbol{x}_i\right\|_2^2 = \boldsymbol{w}^\top \boldsymbol{D}_i' \boldsymbol{w}$ where $\boldsymbol{D}_i' = (\left\|\boldsymbol{y}_i\right\|_2^2 + \left\|\boldsymbol{x}_i\right\|_2^2)\boldsymbol{I}_4 + 2\boldsymbol{C}_i = 2\boldsymbol{I}_4 + 2\boldsymbol{C}_i$ is positive semi-definite of rank at most 2. That is, -1 is an eigenvalue of $\boldsymbol{C}_i$ of multiplicity at least 2. Concluding, $\boldsymbol{C}_i$ has eigenvalues $1, 1, -1, -1$ and $\boldsymbol{D}_i$ has eigenvalues $4, 4, 0, 0$.

步骤3：ARCS+R 旋转估计

给出一些概念：

单位四元数（unit quaternion）：时四元数是一种数学结构，可以用来表示三维空间中的旋转。单位四元数是指其模长（长度）等于1的四元数。单位四元数常用于旋转表示，具有较好的数学性质和几何意义。单位四元数是R4的单位向量，单位四元数的空间是s3。

R^4：R^4表示四维实数空间，由四个实数组成的元组构成，单位四元数可以被看作是R^4中的一个向量。

S^3：S^3表示三维单位球面的空间。单位四元数的空间是指由所有单位四元数组成的集合，这个集合形状上等价于三维单位球面。这是因为单位四元数的模长为1，所以可以将其视为三维球面上的点。

求解(9)的ARCS+R算法

ARCS+R属于一般的黎曼次梯度下降框架。
黎曼次梯度算法如下：
1. 初始化：选择初始点x_0在黎曼流形上。(黎曼流形是一种广义的数学结构，它在局部上类似于欧几里德空间，但在全局上具有弯曲性质)
2. 迭代更新：对于每个迭代步骤 t = 0, 1, 2, ...，执行以下步骤：
a. 计算在当前点x_t处的梯度的次梯度。次梯度是一个表示在非光滑点处的梯度的概念，它通常是一个集合，代表在该点的梯度可能的方向。
b. 在次梯度集合中选择一个方向 g ∈ ∂f(x_t)，其中 ∂f(x_t) 表示函数 f 在 x_t处的次梯度集合。
c. 根据学习率参数选择步长(learning rate)：η_t。
d. 更新当前点 x_{t+1} = Exp[x_t](- η_t g)，其中Exp x_t 表示在x_t处的流形上的指数映射操作。
3. 终止条件：根据预先设定的停止准则判断是否终止迭代。常见的终止条件包括达到最大迭代次数、函数值变化不大或梯度范数小于某个阈值等。
4. 输出结果：返回最终的优化结果 x*，即最优解的估计。

求解(9)的ARCS+R算法

先初始化为某个单位四元数w(0)$\in S_3$，并按以下算法迭代：

$$\boldsymbol{w}^{(t+1)} \leftarrow \mathrm{Proj}_{\mathbb{S}^3}\left(\boldsymbol{w}^{(t)} - \gamma^{(t)}\,\tilde{\nabla}_{\mathrm{s}}\, h(\boldsymbol{w}^{(t)})\right), \qquad (10)$$

1. $Proj_{S^3}$：这是一个将向量投影到S^3(单位四元数集合)上的操作。在算法中，需要将更新后的w_{k+1}向量投影回S^3，以确保它仍然是一个单位四元数。
2. $\gamma^{(t)}$：这是步长（学习率），用于控制每次迭代中更新的幅度。γ(t)可以是一个固定的常数或者根据迭代次数t进行自适应调整。
3. ~∇s h($w^{(t)}$)：这是目标函数h在当前点w(t)处的黎曼子梯度。黎曼子梯度是一种推广的概念，用于在黎曼流形上定义的优化问题中进行梯度计算。

求解(9)的ARCS+R算法

理论:现在我们可以从理论的角度比较(8)和(9)。如文献[14]所证明的，对于任何固定的离群比(异常点的比例)且$k^*>0$，黎曼次梯度下降在有限时间内收敛于R^*，只要

(1)R足够大（样本值够大）
(2)所有的点y_i和x_i都均匀分布在S^2上
(3)没有噪声。

但在[14]中没有给出收敛速率。建立收敛率的一个主要挑战是，在SO(3)上的投影不具有某种非扩张性（在SO(3)上的投影不具有某种非扩张性意味着在进行投影操作时，不会改变向量的长度或模。换句话说，投影操作保持向量的范数不变）

求解(9)的ARCS+R算法

另一方面，（9)在s3上的投影满足不具有某种非扩张性这个性质。因此，我们能够为ARCS+R提供收敛速度保证。

例如，从[53]的定理2可以直接得出，即使任意初始化，ARCS+R算法(10)在$O(\varepsilon^{-4})$次迭代中收敛到ε−平稳点。

$O(\varepsilon^{-4})$次迭代：意味着在问题规模或精度趋近于无穷大时，该算法或方法的迭代次数与ε^{-4}成正比。换句话说，迭代次数随着精度的提高而快速减少。

收敛到ε−平稳点：表示在一定精度范围内，目标函数的变化量小于或等于 ε，即目标函数在该点的变化足够小。

求解(9)的ARCS+R算法

然后给出ARCS+R线性收敛于表示R^*的真值单位四元数（真值单位四元数是指模长为1且实部为正的四元数）$\pm w^*$的条件:

设单位四元数w与$\pm w^*$之间的距离为:

$$\mathrm{dist}(\boldsymbol{w}, \pm\boldsymbol{w}^*) := \min\left\{\left\|\boldsymbol{w}-\boldsymbol{w}^*\right\|_2, \left\|\boldsymbol{w}+\boldsymbol{w}^*\right\|_2\right\}.$$

若$\mathrm{dist}(w, \pm w^*) < \rho$且$\rho > 0$，则称w为ρ-接近$\pm w^*$。

锐度

Definition 1 (*sharpness* [15, 44, 49, 53]). We say that $\pm\boldsymbol{w}^*$ is an α-*sharp minimum* of (9) if $\alpha > 0$ and if there exists a number $\rho_\alpha > 0$ such that any unit quaternion $\boldsymbol{w} \in \mathbb{S}^3$ that is ρ_α-*close* to $\pm\boldsymbol{w}^*$ satisfies the inequality

$$h(\boldsymbol{w}) - h(\boldsymbol{w}^*) \geq \alpha \operatorname{dist}(\boldsymbol{w}, \pm\boldsymbol{w}^*). \tag{11}$$

$\pm w^*$是(9)的α-锐极小值。

锐度

我们给出$\pm w^*$为α*尖锐的一个条件（附录B. 1中得到证明）：

Proposition 5. *If* $\alpha^* := k^*\eta_{\min}/\sqrt{2} - (\ell - k^*)\eta_{\max} > 0$ *and if* $\epsilon_i = \mathbf{0}$ *in Problem 2, then Problem (9) admits* $\pm w^*$ *as an* α^**-sharp minimum. Here* $\eta_{\min}$, $\eta_{\max}$ *are respectively*

$$\eta_{\min} := \frac{1}{k^*} \min_{\boldsymbol{w} \in \mathcal{S}^* \cap \mathbb{S}^3} \sum_{i \in \mathcal{I}^*} \sqrt{\boldsymbol{w}^\top \boldsymbol{D}_i \boldsymbol{w}}, \quad \textit{and} \tag{12}$$

$$\eta_{\max} := \frac{1}{\ell - k^*} \max_{\boldsymbol{w} \in \mathbb{S}^3} \sum_{i \in [\ell] \setminus \mathcal{I}^*} \sqrt{\boldsymbol{w}^\top \boldsymbol{D}_i \boldsymbol{w}}, \tag{13}$$

where $\mathcal{S}^*$ *is the hyperplane of* $\mathbb{R}^4$ *perpendicular to* $\pm \boldsymbol{w}^*$.

其中S^*是垂直于$\pm w^*$的R^4的超平面（一个超平面是一个维数比空间本身低一的子空间，在这里也就是一个三维的子空间）

收敛到基真单位四元素

利用[53]的定理4和命题5，我们得到在ARCS+R算法(10)中，如果初始化适当且具有适当的步长，只要±w*是α*-尖锐的，线性收敛于基真单位四元数±w*。一个正式的声明是:

Theorem 2. *Suppose* $\alpha^* := k^*\eta_{\min}/\sqrt{2} - (\ell - k^*)\eta_{\max} > 0$. *Let* L_h *be a Lipschitz constant of* h. *Run Riemannian subgradient descent* ARCS+$_R$ (10) *with initialization* $\boldsymbol{w}^{(0)}$ *satisfying* $\operatorname{dist}(\boldsymbol{w}^{(0)}, \pm\boldsymbol{w}^*) \le \min\{\alpha^*/L_h, \rho_{\alpha^*}\}$ *and with geometrically diminishing stepsizes* $\gamma^{(t)} = \beta^t\gamma^{(0)}$, *where*

$$\gamma^{(0)} < \min\left\{\frac{2e_0(\alpha^* - L_h e_0)}{L_h^2}, \frac{e_0}{2(\alpha^* - L_h e_0)}\right\},$$

$$\beta^2 \in \left[1 + 2\left(L_h - \frac{\alpha^*}{e_0}\right)\gamma^{(0)} + \frac{L_h^2(\gamma^{(0)})^2}{e_0^2},\ 1\right),$$

$$e_0 = \min\left\{\max\left\{\operatorname{dist}(\boldsymbol{w}^{(0)}, \pm\boldsymbol{w}^*), \frac{\alpha^*}{2L_h}\right\}, \rho_{\alpha^*}\right\}.$$

In the noiseless case ($\boldsymbol{\epsilon}_i = 0$) *we have each* $\boldsymbol{w}^{(t)}$ *satisfying*

$$\operatorname{dist}(\boldsymbol{w}^{(t)}, \pm\boldsymbol{w}^*) \le \beta^t e_0. \tag{14}$$

1、对于一个函数f（x），如果存在一个常数L，使得对于该函数上的任意两个点x和y，都有以下不等式成立：
|f（x） - f（y）|≤ L * |x - y|
那么这个常数L就被称为函数f的李普希茨常数。这个常数L表示了函数变化的最大速率，也可以理解为函数的斜率的上界。

2、步长几何递减是一种常见的优化算法中的步长调整策略。在这种策略中，算法的步长或学习率按照几何级数递减，通常形式为γ（t） = γ（0） * β^t

5. 实验

在本节中，有两个实验。我们通过问题1（同步旋转和对应搜索）与问题2（鲁棒旋转搜索）的真实实验来评估ARCS+O方法和ARCS+R方法，还有他们的组合方法ARCS+OR。

对于这两个问题，我们比较了以下最先进的方法(参见§2):FGR [85]， GORE [16]， RANSAC, GNC-TLS[76]和TEASER++[80]

5. 实验

RANSAC（Random Sample Consensus）：通常被称为“随机抽样一致性算法”是一种经典的迭代方法。算法步骤如下：

1、选择一个样本数量为n的随机子集，这些样本被称为内点集合。
2、使用选定的内点集合来拟合一个模型，根据问题的具体情况选择适当的模型类型。
3、对于剩余的数据点，计算它们与拟合模型之间的误差，并将其与预定义的阈值进行比较。
4、根据阈值判断数据点是内点还是离群点。一般而言，如果数据点的误差小于阈值，则将其划分为内点；否则，将其划分为离群点。
5、统计内点的数量。
6、重复步骤1到步骤5一定次数（迭代次数）。
7、在迭代过程中选择具有最大内点数量的模型作为最终的估计模型。
8、使用所有被划分为内点的数据点重新拟合最终的模型。

5. 实验

5.1. 合成点云的实验

实验设置：作者指定了实验中使用的参数，比如标准差（σ=0.01）、一个常数值（c=5.54 σ ）、样本数量（n=0。8m）以及时间（s=90）。
实验使用MATLAB实现，没有使用并行化或特殊的速度优化。

5. 实验

5.1. 合成点云的实验

鲁棒性旋转搜索：作者从N(0, I_3)（三维随机向量）中随机采样点对{(y_i, x_i)} $_{i=1}^{l}$ 其中由k^*个内点和噪声$\in_i$~N(0, σ^2 I_3). 从S^2随机采样的旋转轴和位于[0, 2π]的角度可生成旋转矩阵R^*的基真值（它表示关于数据或问题的真实、准确的标签、值或答案）。外点对可以在约束条件：-c $\leqslant$|| y_i ||-|| x_i ||$\leqslant$ c下检测和删除。

然后作者将ARCS+O、ARCS+R以及它们的组合ARCS+OR与其他现有方法进行了比较。

5. 实验

Inlier Ratio $\frac{k^*}{\ell}$	$\frac{10^3}{10^5} = 1\%$	$\frac{10^3}{10^6} = 0.1\%$	$\frac{3\times10^3}{5\times10^6} = 0.06\%$	$\frac{3\times10^3}{10^7} = 0.03\%$	$\frac{10^3}{10^7} = 0.01\%$
TEASER++ [80]	out-of-memory				
RANSAC	0.39 | 0.20 | 29.1	$\geq$ 8.4 hours			
GORE [16, 63]	3.43 | 2.10 | 1698	$\geq$ 12 hours			
FGR [85]	52.2 | 68.5 | 3.64	95.0 | 60.9 | 37.7	84.9 | 59.4 | 145	86.5 | 56.9 | 311	97.3 | 61.3 | 314
GNC-TLS [76]	3.86 | 9.51 | 0.13	63.4 | 50.5 | 2.26	49.9 | 31.1 | 15.9	90.2 | 45.6 | 40.1	120 | 34.3 | 36.3
ARCS+$_{\mathrm{R}}$	9.92 | 13.1 | 0.12	65.2 | 48.9 | 0.96	55.6 | 38.3 | 5.58	88.4 | 36.2 | 12.6	98.2 | 36.0 | 12.2
ARCS+$_{\mathrm{O}}$	0.86 | 0.29 | 1.71	0.99 | 0.37 | 23.2	0.91 | 0.30 | 125	0.98 | 0.42 | 287	55.6 | 60.9 | 281
ARCS+$_{\mathrm{OR}}$	0.03 | 0.03 | 1.72	0.09 | 0.07 | 23.2	0.11 | 0.07 | 125	0.22 | 0.15 | 287	55.4 | 60.1 | 281

表4：各种算法在合成数据上的平均误差度|标准差|运行时间秒(20次试验)

他们讨论了准确性和可扩展性之间的权衡。例如，RANSAC在k*/ℓ = 103/105时具有较好的准确性，误差仅为0.39，但随着内点比例的减少，运行时间显著增加。而像GNC-TLS和FGR这样的方法注重可扩展性，但在存在大量离群点时效果不佳。ARCS+O在准确性和可扩展性之间取得了平衡，只要内点比例超过3 × 103/107 = 0.03%，误差就能小于1度。组合方法ARCS+OR进一步提高了准确性。在可扩展性方面，ARCS+OR始终比FGR快，并且对于k*/ℓ = 103/106 = 0.1%，至少比GORE快1800倍。然而，与GORE和RANSAC在更大点集上的速度比较并未提供。作者提到ARCS+OR在k*/ℓ = 103/107 = 0.01%时失败。

5. 实验

5.1. 合成点云的实验

同时旋转和对应搜索：作者进行了旋转估计和对应搜索的实验，从N(0, I_3)采取了随机点集Q和P他们生成了的内点对和噪声。 每个离群点也是随机独立地从N(0, I_3)中绘制的。

5. 实验

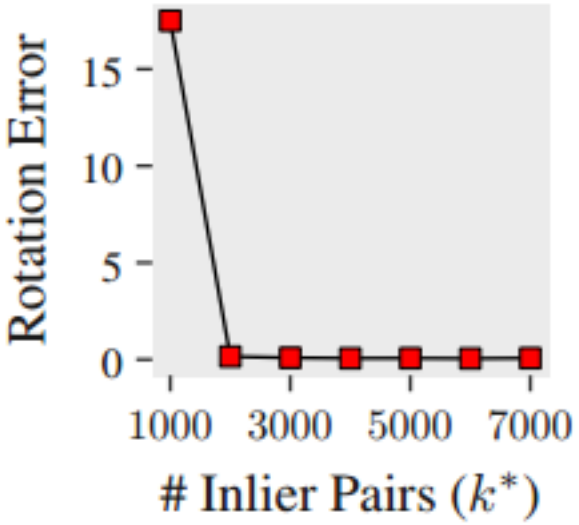


图2：ARCS+在合成高斯点云上的旋转误差。20次试验，m = 104, n =0.8m, σ = 0.01。

作者展示了ARCS+在k* ≥ 2000时能够准确估计旋转（在90秒内），在k* = 1000时效果不佳。作者提到没有将ARCS+与TEASER++、GORE或RANSAC等方法进行比较，因为根据之前的表格（表2和表4），这些方法的运行时间或准确性都不理想。他们还指出，像FPFH这样的特征匹配方法在随机合成数据上表现不佳。

Inlier Ratio $\frac{k^*}{\ell}$	$\frac{10^3}{10^5} = 1\%$	$\frac{10^3}{10^6} = 0.1\%$	$\frac{3\times10^3}{5\times10^6} = 0.06\%$	$\frac{3\times10^3}{10^7} = 0.03\%$	$\frac{10^3}{10^7} = 0.01\%$
TEASER++ [80]	out-of-memory				
RANSAC	0.39 | 0.20 | 29.1	$\geq$ 8.4 hours			
GORE [16, 63]	3.43 | 2.10 | 1698	$\geq$ 12 hours			
FGR [85]	52.2 | 68.5 | 3.64	95.0 | 60.9 | 37.7	84.9 | 59.4 | 145	86.5 | 56.9 | 311	97.3 | 61.3 | 314
GNC-TLS [76]	3.86 | 9.51 | 0.13	63.4 | 50.5 | 2.26	49.9 | 31.1 | 15.9	90.2 | 45.6 | 40.1	120 | 34.3 | 36.3
ARCS+$_R$	9.92 | 13.1 | 0.12	65.2 | 48.9 | 0.96	55.6 | 38.3 | 5.58	88.4 | 36.2 | 12.6	98.2 | 36.0 | 12.2
ARCS+$_O$	0.86 | 0.29 | 1.71	0.99 | 0.37 | 23.2	0.91 | 0.30 | 125	0.98 | 0.42 | 287	55.6 | 60.9 | 281
ARCS+$_{OR}$	0.03 | 0.03 | 1.72	0.09 | 0.07 | 23.2	0.11 | 0.07 | 125	0.22 | 0.15 | 287	55.4 | 60.1 | 281

5. 实验

5.2. 3d匹配实验

3DMatch数据集包含1000多个用于测试的点云，代表8个不同的场景，而每个场景的点云数量从77到506不等。每个点云中有超过10^5个点，我们使用3DSmoothNet的预训练模型7[30]从这些关键点提取描述符，并使用Matlab函数pcmatchfeatures进行匹配，其参数MatchThreshold设置为最大值1（他们设置了参数MatchThreshold的值为最大值1，这意味着只有那些非常相似的描述符才会被匹配成功）。

5. 实验

Scene Type # Scene Pairs	Kitchen 506	Home 1 156	Home 2 208	Hotel 1 226	Hotel 2 104	Hotel 3 54	Study Room 292	MIT Lab 77	Overall 1623
TEASER++	**99.0%**	**98.1%**	94.7%	98.7%	**99.0%**	98.1%	97.0%	94.8%	97.72%
ARCS++$_{OR}$	98.4%	97.4%	**95.7%**	98.7%	98.1%	**100%**	**97.3%**	**96.1%**	97.72%

表5：方法在3DMatch数据集的场景对上运行的成功率，提供了地真旋转和平移(旋转误差小于10度意味着成功

我们发现TEASER++和ARCS+OR性能不相上下。我们在这里没有比较其他方法，因为TEASER++目前在3DMatch上的性能最好

6.总结

- ARCS+算法进行了讨论并指出了一些限制。
- 对于小型数据集，可以考虑其他方法，如MAGSAC++、VSAC、TEASER++和GORE。对于更大的数据集，作者推荐使用GNC-TLS和RANSAC算法。
- ARCS+算法在去除异常值方面表现良好，但没有优化保证。他们认为在优化保证、准确性和可扩展性三者之间存在权衡，而同时满足这三个属性需要高效地解决大规模的问题

Thank You!