基于双向 KMPE loss 实现的 BIK-ICP 算法

目标

将传统 icp 的 mse 损失函数更改为 kmpe 损失函数，使得 icp 算法获得更好的健壮性，抗噪性并解决模型部分重合时的配准问题，这里将使用 kmpe 作为损失函数的 icp 称为 bik-icp 算法。损失函数公式如下

1.mse 损失函数

$$\left(\mathbf{R}_k, \vec{t}_k\right) = \underset{\mathbf{R}^T\mathbf{R}=I_2, \det(\mathbf{R})=1, \vec{t}}{\arg\min} \left(\sum_{i=1}^{N_x} \left\| \mathbf{R}\vec{x}_i + \vec{t} - \vec{y}_{c_k(i)} \right\|_2^2 \right).$$

2.kmpe 损失函数

$$(\mathbf{R}^*, \vec{t}^*) = \underset{\mathbf{R}^T\mathbf{R}=\mathbf{I}_n, \det(\mathbf{R})=1, \vec{t}}{\arg\min} \sum_{i=1}^{N_x} \left(1\text{-}\kappa_\sigma(\mathbf{R}\vec{x}_i + \vec{t} - \vec{y}_{c(i)}) \right)^{p/2}$$

其中

$$\kappa_\sigma(X - Y) = \exp\left(-\|X - Y\|_2^2 / 2\sigma^2\right)$$

原理：

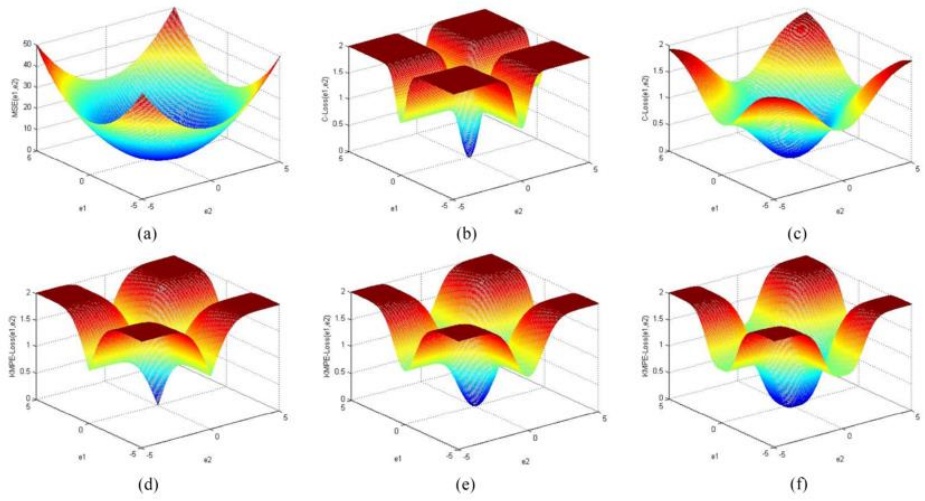

Fig. 1. Bidirectional loss function with different parameters: e_1 and e_2 denote the errors from two directions, respectively. (a) Bidirectional MSE: $e_1^2 + e_2^2$; (b) and (c) Bidirectional C-loss with Gauss kernel: $(1 - \kappa_\sigma(e_1)) + (1 - \kappa_\sigma(e_2))$, $\sigma = 0.5$ and 2; and (d)–(f) Bidirectional KMPE loss: $BiK(e_1, e_2)$, $\sigma = 1$ and $p = 1, 2, 3$.

和传统的基于二次幂累加的 mse 损失函数，基于高斯核函数累加的 kmpe 损失函数有明显优势。如上图所示，高斯核函数的特点是当对应点的距离越小时，其增长速率越大，且函数是有边界的；而二次幂函数随着对应点的距离越大，增长速率反而越快，且无边界。这和我们直觉是相反的，当对应点越接近时，权值应该越大，对应点距离越远，权值越小；其次幂函数无边界，那么异常值，噪声和非重合的点对损失函数的影响过大，将无法获得正确的旋转角和位移向量。

高斯核函数还有个优点就是可以调节幂 p 和σ参数，其中 p 越小，函数的底部就越尖锐，σ越大，函数底部就越宽，使其可配准各种不同初始状态下的目标模型。

步骤

step1　通过 kdtree 获得双向对应点集 U，V

a. 定义和推导

我们通过 kdtree 获得对应点，即当两个点之间的距离最小则它们互为对应点。其中原模型的点集为 X，目标模型的点集为 Y。为了解决不适定问题，这里我们取双向对应点。公式如下:

$$\begin{cases} c_k(i) = \arg\min_{j\in\{1,2,\ldots,N_y\}} \left\| \left(\mathbf{W}_{k-1}\vec{x}_i + \vec{t}_{k-1}\right) - \vec{y}_j \right\|_2^2 \\ d_k(j) = \arg\min_{i\in\{1,2,\ldots,N_x\}} \left\| \left(\mathbf{W}_{k-1}\vec{x}_i + \vec{t}_{k-1}\right) - \vec{y}_j \right\|_2^2. \end{cases} \quad (8)$$

$c_k(i)$为了得到$\vec{y}_i$向量到$\vec{x}_i$向量的最短距离，对应点$\vec{y}_i$向量和$\vec{x}_i$向量互为对应点。$d_k(j)$为了得到$\vec{x}_i$向量到$\vec{y}_i$向量的最短距离，这样对应点$\vec{x}_i$向量和$\vec{y}_i$向量互为对应点。

b.将双向匹配的点集融合成新的点集 U 和 V

将(8)中获得的对应点，融合成两个具有对应关系的新的点集 U 和 V （N_x是 X 点集点的个数，N_y是 Y 点集的点的个数），如下：

$$\vec{u}_i = \begin{cases} \vec{x}_i, & 1 \le i \le N_x \\ \vec{x}_{d_k(i-N_x)}, & N_x + 1 \le i \le N \end{cases}$$

$$\vec{v}_i = \begin{cases} \vec{y}_{c_k(i)}, & 1 \le i \le N_x \\ \vec{y}_{i-N_x}, & N_x + 1 \le i \le N. \end{cases}$$

因此我们获得新的点集 U，V，此时损失函数可表示为：

$$\left(\mathbf{W}_k, \vec{t}_k\right) = \arg\min_{W,\vec{\mathbf{t}}} \frac{1}{N} \sum_{i-1}^{N} \left(1 - \kappa_\sigma\left(\mathbf{W}\vec{u}_i + \vec{t} - \vec{v}_i\right)\right)^{p/2}. \quad (9)$$

step2　求得σ

a. 分析

这里的σ是决定了高斯核的宽度，配准刚开始的时候，σ应尽量大些，这样可以更充分的利用点的信息，而不是将他们误归类为异常点。

b. 计算

The kernel width σ is an important parameter in PCA-KMPE. In general, one can employ the Silvermans rule [38], to adjust the kernel width

$$\sigma^2 = 1.06 \times \min\left\{\sigma_E, \frac{R}{1.354}\right\} \times (n)^{-1/5} \quad (30)$$

where σ_E is the standard deviation of $\|\mathbf{e}_i\|_2^2$ and R is the interquartile range.

step3　求得 t_k

a. 推导公式：

1. 根据(9)式对 t 求导可得：

$$\vec{t}_k = \sum_{i=1}^{N} \varphi(e_i)(\vec{v}_i - s\mathbf{R}\vec{u}_i) \Big/ \sum_{i=1}^{N} \varphi(e_i) \qquad (10)$$

where $\varphi(e_i) \triangleq (1-\kappa_\sigma(e_i))^{[(p-2)/2]}\kappa_\sigma(e_i)$ and $e_i = s\mathbf{R}\vec{u}_i + \vec{t} - \vec{v}_i$.

注意：(9)式中的 W 是一个仿射矩阵，可定义为 W=s*R，s 是缩放系数，R 是旋转矩阵。

b. 求$\varphi(e_i)$

由于 e_i 中含有$\vec{t}$向量，为了计算方便，用$\overrightarrow{t_{k-1}}$近似代替，可近似为 $e_i = sR\overrightarrow{u_i} + \vec{t}_{i-1} - \overrightarrow{v_i}$

c. 求得 t_k

这里的 R 也用R_{k-1}近似代替，直接由（10）就可以计算得到（？？？是否可以先求出R_k后再代入（10）式）

step4 求点集 P 和 Q

a. 公式推导

将（10）式中的$\vec{t}$代入(9)式得，并定义

$$\begin{cases} \vec{p}_i \triangleq \vec{u}_i - \sum_{i=1}^{N} \varphi(e_i)\vec{u}_i / \sum_{i=1}^{N} \varphi(e_i) \\ \vec{q}_i \triangleq \vec{v}_i - \sum_{i=1}^{N} \varphi(e_i)\vec{v}_i / \sum_{i=1}^{N} \varphi(e_i) \end{cases}$$

通过化简，可得新公式

$$(\mathbf{R}_k, s_k) = \underset{\mathbf{R}^T\mathbf{R}=\mathbf{I}_2, \det(\mathbf{R})=1, s}{\arg\min} \frac{1}{N} \sum_{i=1}^{N} (1-\kappa_\sigma(s\mathbf{R}\vec{p}_i - \vec{q}_i))^{p/2}. \qquad (11)$$

因此可将原损失函数转化为点集 P 和 Q 相关的函数，其中 P，Q 是向量$\overrightarrow{p_i}$ 和对应向量$\overrightarrow{q_i}$的集合。

b. 求解

$\varphi(e_i)$和集合 U，V 都是已知的，在 step3 中已经求出，只要按照上述定义计算即可获得 P，Q

step5 求得 s

a. 公式推导

公式（11）直接对 s 求导得：

$$s_k = \sum_{i=1}^{N} \vec{q}_i^T \mathbf{R}_k \varphi(e_i) \vec{p}_i \Big/ \sum_{i=1}^{N} \vec{p}_i^T \varphi(e_i) \vec{p}_i. \qquad (12)$$

b. 求解

$\varphi(e_i)$和集合 P，Q 都是已知的，已经在 step4 和 step5 求出，使用R_{k-1}近似代替 R，就可以根据公式（12）轻松求出 s_k

step6 求得 R

a. 公式推导

论文中并没有直接给出，可参考 Robust rigid registration algorithm based on pointwise correspondence and correntropy

$$\mathbf{H} = \frac{1}{N}\sum_{i=1}^{N} s_k \vec{p}_i \varphi(e_i) \vec{q}_i^T$$

and then decompose $\mathbf{H}$ by SVD as follows: $\mathbf{H} = \mathbf{S}\Lambda\mathbf{D}^T$. Accordingly, $\mathbf{R}_k$ can be estimated as

$$\mathbf{R}_k = \mathbf{S}\tilde{\mathbf{I}}\mathbf{D}^T \tag{13}$$

where

$$\tilde{\mathbf{I}} = \begin{cases} \mathbf{I}_2 & \det(\mathbf{H}) > 0 \\ \mathrm{diag}(1,-1) & \det(\mathbf{H}) < 0. \end{cases}$$

？？？？这里的求解有疑问，不同的论文给出了不同的公式，且该公式在实验中无法获得正确的R_k，我使用的是$R_k = D\tilde{I}S^T$，但不具备论文实验数据的健壮性，只能偏离度数 10 度以内才可以正确配准

伪代码

TABLE I
BIK-ICP ALGORITHM

Algorithm 1: BiK-ICP

Require: Two shape point sets $\mathbf{X} \triangleq \{\vec{x}_i\}_{i=1}^{N_x}$ and $\mathbf{Y} \triangleq \{\vec{y}_i\}_{i=1}^{N_y}$; Initial parameters: s_0, $\mathbf{R_0}$ (or $\mathbf{A}_0$) and $\vec{t}_0$, σ_0 and p.

1: **For** $k = 1, ..., K$.
2: Set up correspondences $c_k(i)$ and $d_k(j)$ via (8).
3: Compute $\varphi(e_i)$ by s_{k-1}, $\mathbf{R}_{k-1}$, and t_{k-1}.
4: Reconstruct matrices $\mathbf{P}$ and $\mathbf{Q}$.
5: Solve s_k and $\mathbf{R}_k$ via (12) and (13) or $\mathbf{A}_k$ via (18).
6: Solve the translation vector $\vec{t}_k$ via (10) or (14).
7: Update σ.
8: Compute the bidirectional MSE e_k.
9: **If** $|e_k - e_{k-1}| \leq \varepsilon$.
10: break;
11: **end if**.
12: **end for**.

Return: the transformation parameters s, $\mathbf{R}$ or $\mathbf{A}$ and $\vec{t}$.

论文实验中初始化如下：

s_0=1， R_0=I_2， t_0=[0,0]， σ_0由 step2 计算得到， p=0.2， K=200

注意：我们也可以通过粗配准得到初始的 s_0， R_0 和 t_0；p 的取值范围在[0.2 ，8]，通过贪婪算法[0.2, 0.4, 0.6, … ,8]观察最佳效果确定 p（???目前还无靠谱的策略得到 p）