ScaleLK: Registration of Point Clouds with Different Scales Using Deep Learning Methods

Shuai Xu and Yanlei Shang*
Institute of Network Technology, Beijing University of Posts and Telecommunications, Beijing, China

*Corresponding author e-mail: shangyl@bupt.edu.cn

Abstract. 3D point clouds are widely used in numerous research and applications, such as autonomous driving, industrial robots, and augmented reality, to represent the spatial structure of objects. The 3D point cloud registration aims to transform the source point cloud into the same coordinate system with the template point cloud, which is of great significance for the 3D reconstruction of the real world objects. ICP [1] is one of the most classic point cloud registration algorithms but it still has problems with efficiency and initialization. With the help of deep learning, PointNetLK [7] becomes a state-of-the-art point cloud registration method. Although PointNetLK is efficient and robust to some extent, it is not able to register point clouds with different scales. In this paper, we propose ScaleLK, an approach for registration of point clouds with different scales using deep learning methods. We have trained a feature extractor which supports scale feature and used this feature for registration of point clouds with different scales. We describe the architecture and compare its performance with other methods.

1. Introduction
Besl and McKay [1] proposed the Iterative Nearest Point Algorithm (ICP) in 1992. The algorithm iteratively selects the corresponding point pairs from source and template point clouds according to a certain correspondence and calculates the optimal rotation and translation parameters to minimize the Euclidean distance between the sets of the selected point pairs. In the ICP algorithm, the selection of the corresponding points from the source and the template point clouds relies on the nearest neighbor method, which is the most time-consuming step. Although the kd-tree [2, 3] greatly improved the execution efficiency of the ICP algorithm, the initialization problem remains to be solved.

In recent years, point cloud classification and point cloud registration algorithms based on deep learning have emerged. As for how to input the point cloud into the deep neural network, one common way is to input multi-view data and integrate the features extracted by the deep neural network [4]. Structured representation method such as voxel grids [5] is more common, but this representation consumes a lot of memory and is computationally intensive. Charles et al. [6] creatively proposed the PointNet network which directly used the 3D point cloud as the input of the deep neural network, and solved the disorder problem by adding the symmetric function. Inspired by PointNet, Yasuhiro et al. [7] proposed PointNetLK for point cloud registration, using feature extraction and IC-LK [8] to iteratively register point clouds. Although PointNetLK has some progress in terms of computational efficiency and robustness, it is not able to register point clouds with different scales.
Figure 1. PointNetLK is not able to register point clouds with different scales.

Registration of point clouds with different scales is very important for solving the coordinate transformation in a typical monocular Simultaneous Localization and Mapping (mono-SLAM) system, because the mono-SLAM algorithm cannot retrieve the absolute scale of objects, so the scales of point clouds in two independent augmented reality systems interacting together are different. By introducing the scale factor, our feature extractor can extract the scale features of the point clouds, which make ScaleLK support similarity transformation and possible to solve the relative scale of two point clouds.

2. ScaleLK

In Section 2.1 we introduce notation and formulation of ScaleLK. In Section 2.2 we provide the architecture of ScaleLK. In Section 2.3 we describe the training process and implementation details.

2.1. Notation and Formulation

ScaleLK contains two parts, one of which is the iteration part and the other is the feature extraction part.

2.1.1. The Iteration Part. We denote template \( P_T \) and source \( P_S \) point clouds after a given similarity transformation \( S \) as \( P_T(S) \) and \( P_S(S) \) respectively, and in particular, we use \( P_T(0) \) and \( P_S(0) \) to denote the initial state of template and source point clouds respectively.

For the given template and source point clouds with different scales, the registration problem can be described as finding transformation \( S \) on \( \text{Sim}(3) \) such that

\[
S = \arg \min_{S \in \text{Sim}(3)} \| P_S(S) - P_T(0) \|_2^2 \tag{1}
\]

It is a non-linear optimization problem to solve (1), so classic Lucas-Kanade algorithm can be used to solve \( S \) iteratively, each time we get the first-order Taylor approximate expansion

\[
\Delta S = \arg \min_{\Delta S \in \text{Sim}(3)} \| P_S(S_k) + \frac{\partial P_S(S_k)}{\partial S} \Delta S - P_T(0) \|_2^2 \tag{2}
\]

In fact, there is a high computational cost to recalculate the Jacobian at each iteration, which badly lower the efficiency of the algorithm. So we turn to IC-LK, where we do not recalculate \( \Delta S \) around the source cloud \( P_S(S_k) \) got from the \( k^{th} \) iteration, instead we recalculate \( \Delta S \) around the initial template point cloud \( P_T(0) \). By doing this, we get

\[
\Delta S = \arg \min_{\Delta S \in \text{Sim}(3)} \| P_S(S_k) - P_T(\Delta S) \|_2^2 \tag{3}
\]

and the accumulating process becomes

\[
S_{k+1} \leftarrow \Delta S^{-1} \cdot S_k \tag{4}
\]

The first-order Taylor approximate expansion of (3) is

\[
\Delta S = \arg \min_{\Delta S \in \text{Sim}(3)} \| P_S(S_k) - \frac{\partial P_T(0)}{\partial S} \Delta S - P_T(0) \|_2^2 \tag{5}
\]
The Jacobian \( J = \frac{\partial P_T(0)}{\partial S} \) only need to be calculated once, so that the efficiency is improved.

In order to eliminate the constraint in the transformation matrix, we introduce 7-dimensional Lie algebra \( \zeta = (\zeta_1, \zeta_2, \zeta_3, \zeta_4, \zeta_5, \zeta_6, \zeta_7)^T \) to replace \( \text{Sim}(3) \), where the first three dimensions represent translation and the next three represent rotation. Particularly, the last dimension \( \zeta_7 \) is the scale factor so that \( e^{\zeta_7} \) is the relative scale between template and source point clouds. Then we can represent \( S \) with an exponential map as follows:

\[
S = \exp\left(\sum_{i=1}^{7} \zeta_i G_i\right), \quad R = \exp\left(\sum_{i=4}^{6} \zeta_i G_i\right), \quad \rho = (\zeta_1, \zeta_2, \zeta_3)^T
\]  

(6)

where \( R \) is element on \( \text{SO}(3) \), \( G_i \) are the generators of \( \text{Sim}(3) \) group, \( \rho \) is translation vector and \( F \) is a coefficient matrix.

\[
F = \frac{e^{\zeta_7-1} I + \frac{\zeta_7e^{\zeta_7 \sin \theta}}{\zeta_7^2+\theta^2} a^a + (\frac{e^{\zeta_7-1} - (e^{\zeta_7 \cos \theta - 1}) \zeta_7 + (e^{\zeta_7 \sin \theta}) \zeta_7 \zeta_7^2+\theta^2) a^a}{\zeta_7^2+\theta^2}
\]

(7)

where \( a^a \) is the normalization of the corresponding anti-symmetric matrix \( A = \sum_{i=4}^{6} \zeta_i G_i \) with respect to \( \theta \).

2.1.2. Feature Extraction. In the feature extraction part, we use a similar structure as the feature extraction part of the PointNet. The original T-Net is removed because that our target is let the feature extractor discriminate the spatial transformation feature of point clouds rather than eliminate its influence on classification.

As we introduce the feature extraction function \( \varphi(P) \), the registration problem can then be described as finding \( S \) such that

\[
\varphi(P_S(S)) = \varphi(P_T(0))
\]

(8)

According to (5) and (8), we get

\[
\Delta \zeta = J^+ [\varphi(P_S(S_k)) - \varphi(P_T(0))]
\]

(9)

where \( J^+ \) is the pseudoinverse matrix of \( J = \frac{\partial \varphi(P_T(0))}{\partial \zeta} \), and it need to be calculated only once and so does \( \varphi(P_T(0)) \). Then we can get \( \Delta S \) through exponential map according to (6):

\[
\Delta S = \exp(\sum_{i=1}^{7} \Delta \zeta_i G_i)
\]

(10)

The iteration stops when \( \Delta S \) reaches a certain threshold and the final registration result is the accumulation as follows:

\[
S = \Delta S_k^{-1} \cdot ... \cdot \Delta S_1^{-1} \cdot \Delta S_0^{-1}
\]

(11)

2.2. ScaleLK Architecture

The architecture of our feature extractor is shown in figure 2, each hidden layer contains convolution, batch normalization and ReLU activation. And a max-pooling function is used at the end of the extractor to solve the disorder problem and the mismatch of number of points between two point clouds.

\[
\text{Figure 2. The architecture of the feature extractor.}
\]
The entire architecture of our algorithm is shown in figure 3. At each iteration, the feature extractor extracts the k-dimensional feature vector from the source point cloud $P_S(S_k)$ and then $\Delta S$ is calculated and then accumulated to $S$, and this iteration stops when the accuracy reaches the given threshold.

![Figure 3. The entire architecture of ScaleLK.](image)

2.3. Training Details

The dimension of our feature extractor is the same as used in PointNet (64, 64, 128, 128, K=1024) and this setting shows satisfying result in our experiments.

We apply data augmentation to the training set of the ModelNet40 dataset by randomly generating vectors which are normally distributed in $\mathbb{S}_3$, a 7-dimensional Lie algebra. And these vectors are transformed into similarity transformation matrices through exponential map as the ground-truth (GT). The source point clouds are generated by applying transformation matrices and some noise to the original template point clouds in the training set.

The loss of the network includes two parts, which are the MSE between the feature vectors of source and template point clouds after registration and the MSE between the result transformation matrix and the ground-truth. We use AdaGrad as our gradient method and batch size is set to 32. In fact, the training aims to make our feature extractor extract features more suitable for the iteration part. We finally choose max pooling as our symmetric function.

3. Experiments

We use the validation part of ModelNet40 dataset and apply randomly generated similarity transformation to it to get the source point clouds as we did in the training process. We test our ScaleLK algorithm and compare the performance with PointNetLK and Scale-ICP.

3.1. Scale estimation.

Experiment shows that our ScaleLK can successfully register point clouds with different scales and here we randomly choose two registration results to demonstrate. As shown in figure 4, the template and source point clouds are colored red and blue respectively. Using our ScaleLK, the registration of airplane point clouds completes after 13 iterations and the registration of guitar point clouds completes after 11 iterations. As a comparison shown in figure 1, the registration using PointNetLK failed, because the PointNetLK algorithm does not support different scales.

Moreover, we notice that in the early state of the iteration, the scales of the source and template point clouds become quite close, much faster than the estimation of rotation and translation, which means our feature extractor can better discriminate the scale factor.

![Figure 4. Registration results of airplane and guitar point clouds using ScaleLK](image)
3.2. The performance.

In terms of performance, we mainly compare our ScaleLK with Scale-ICP because algorithms like PointNetLK is not able to handle point clouds with different scales. The specification of the server used in our experiment is shown in Table 1.

Table 1. The specification of the server.

| Device | Model or Size  | Frequency   |
|--------|---------------|-------------|
| CPU    | Intel Xeon E5-2650 | 2.2GHz(48T) |
| Memory | 128GB         | 2400MHz     |
| GPU    | NVIDIA TITAN Xp | 1582 MHz    |

In fact the source and template point clouds we use in our experiment are not identical in term of structure because they are produced by two random sampling process to simulate the real demand. The experiment shows that ScaleLK is more robust than Scale-ICP in terms of this structure difference between source and template point clouds. Our ScaleLK reaches higher success ratio and lower average MSE with the help of the trained feature extractor. Moreover, we achieve a lower cost in average registration time, comparing with Scale-ICP.

Table 2. Performance comparison.

| Algorithm                      | Scale-ICP | ScaleLK |
|--------------------------------|-----------|---------|
| Average MSE Loss of Transform Matrix | 0.07      | 0.0216  |
| Average Registration Time      | 2.71 s    | 0.9 s   |
| Success Ratio                  | 69.6%     | 80.5%   |

4. Conclusion

We have proposed ScaleLK, a registration algorithm based on deep learning that support point clouds with different scales. By introducing scale factor, we extend the rigid body transformation in the original PointNetLK to similarity transformation. We have trained a feature extractor which is able to describe the scale feature of the point clouds and this feature vector can then be used to register source and template point clouds with different scales. Experiments have shown that our approach obtains ability in terms of scale and impressive computational efficiency.

5. Acknowledgments

The work presented in this study is supported by the National Natural Science Foundation of China “Research on Web-based Dynamic Service Delivery Mechanism and Method in Information Center Network” (61671081) and the Natural Science Foundation of Beijing “Research on Dynamic Service Provision Mechanism of Information Center Network Based on Edge Computing” (4172042).

References

[1] Besl, McKay, Method for registration of 3-D shapes, Sensor fusion IV: control paradigms and data structures, Vol. 1611, 1992.
[2] Greenspan, Michael, and Mike Yurick, Approximate kd tree search for efficient ICP, 3DIM Proceedings (2003): 442-448.
[3] Nuchter, Andreas, Kai Lingemann, and Joachim Hertzberg, Cached kd tree search for ICP algorithms, 3DIM Proceedings (2007).
[4] Su Hang, et al, Multi-view convolutional neural networks for 3d shape recognition, Proceedings of the IEEE international conference on computer vision (2015).
[5] Maturana, Daniel, and Sebastian Scherer, Voxnet: A 3d convolutional neural network for real-time object recognition, IEEE/RSJ International Conference on Intelligent Robots and Systems (2015).
[6] Qi, Charles R., et al, Pointnet: Deep learning on point sets for 3d classification and segmentation, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2017).
[7] Yasuhiro, et al, PointNetLK: Robust & efficient point cloud registration using PointNet, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2019).
[8] Baker, Simon, and Iain Matthews, Lucas-Kanade 20 years on: A unifying framework, International journal of computer vision 56.3 (2004): 221-255.