RAGO: Recurrent Graph Optimizer For Multiple Rotation Averaging Supplementary Material

Heng Li[1] Zhaopeng Cui[2] Shuaicheng Liu[4] Ping Tan[1,3]
[1]Simon Fraser University [2]State Key Lab of CAD&CG, Zhejiang University [3]Alibaba XR Lab
[4]University of Electronic Science and Technology of China
{lihengl,pingtan}@sfu.ca, zhpcui@zju.edu.cn, liushuaicheng@uestc.edu.cn

To make our submission self-contained, the supplementary material provides additional details about 1) Network definition and rotation representation, 2) Training and Testing, 3) Result details. The code is available at github.com/sfu-gruvi-3dv/RAGO

1. Network Details

1.1. The Number Of Feature Channels

As we mentioned in the main paper, we use Orth6D [9] as the default rotation representation in our RAGO. For the input rotation $\mathbf{R}_{3\times3} \in SO(3)$, we flatten it to $\mathbf{R}_{1\times9}$ as the input of the neural networks. Thus the input size of the nodes is $\{\mathbf{R}_u\} = (|\mathcal{V}, 9|)$, and the input of the edges is $\{\mathbf{R}_v\} = (|\mathcal{E}|, 9)$. The size of cost feature is $\{\mathbf{C}_u\} = \{|\mathcal{V}|, 48\}, \{\mathbf{C}_{uv}\} = (|\mathcal{E}|, 48)$. The feature and hidden state of the view-graph have the same size as the cost feature. The output $\Delta\mathbf{R}_{1\times6}$ in 6D space is then mapped to the $\Delta\mathbf{R}_{3\times3}$ on 3D rotation space $SO(3)$.

1.2. The Structure of MLP

We use the Multi-layered Linear Perception (MLP) in the Edge Convolution and the Graph Updater. For the MLP Φ_{node} and Φ_{edge} in Edge Convolution, we use MLP with 3 fully connected (FC) layers without dropout layers, while the inner feature channel is fixed to 48. We use the MLP with 6 FC layers in the graph updater.

1.3. The Structure of MPNN

In this section, we provide additional details about the definition of the neural networks used in RAGO. The architecture of Edge Conv as shown in Figure 1. As shown in Figure 2, we use the Message Passing Neural Network (MPNN) with 1 Edge Convolution layer to extract features from the input view-graph. The MPNN used for initializing the hidden state has the same structure with MPNN Θ_{feat} while following a tanh activation function to map the output to $(-1, 1)$. Notice that in the MPNN Θ_{cost}, the output

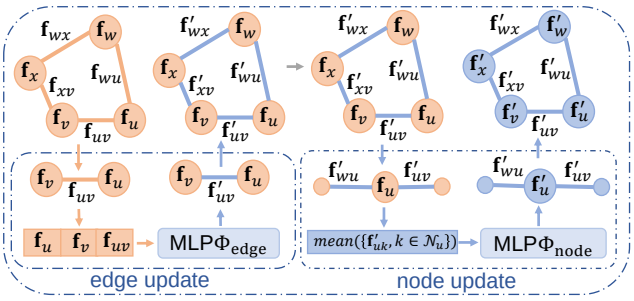

Figure 1. The structure of an Edge Convolution. An Edge Convolution update each edge feature with a weight shared MLP Φ_{edge}. Then, the node features are updated by passing the updated edge features from the neighbors to the node MLP Φ_{node}.

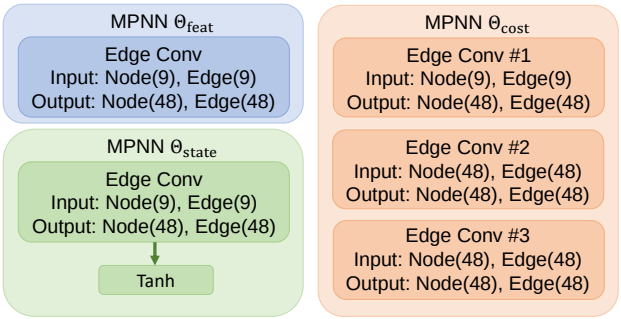

Figure 2. The structure of the MPNN we used in RAGO.

cost features have the information from its 3-order neighbors because it is updated by Edge Convolution 3 times.

2. Training Details

2.1. Edge Random Drop

To avoid overfitting on the real-world dataset, we randomly drop 20% edge from the view-graph. We first randomly generate a spanning tree from the view-graph, while the edge on the spanning tree will keep active to make sure the whole view-graph is connected. Then, we uniformly

drop 20% edges from the view-graph. During inference, all of the edges are active.

2.2. Evaluation Cost on the Real-world Datasets

Due to the limitation of the training samples on the real-world datasets, RAGO may be overfitted on the training set. We could evaluate RAGO on the validation set on the synthetic dataset and keep the parameters with the best performance for testing. However, there is only 1 view-graph for testing on the real-world datasets due to the leave-one-out manner. Thus we should define an evaluation cost on the testing view-graph to save the model with the best performance to avoid overfitting. Inspired by the MRA objective function, We can define an evaluation cost based on the input relative orientations, the output rectified relative orientations and the estimated global camera orientations. The rectified relative orientations are used as the robust function. The parameters of the model minimize the following evaluation cost is saved for testing:

$$\sum_{(u,v)\in\mathcal{E}} \rho(d(\mathbf{R}^t_{uv}, \mathbf{R}_{uv}))^2 d(\mathbf{R}^t_u \mathbf{R}^{t\top}_v, \mathbf{R}_{uv}), \qquad (1)$$

where d is the angular distance of two rotations and the robust function is $\rho(x) = \text{relu}(20 - x)$.

2.3. Global Camera Initialization

Due to the limitation of the training samples on the real-world datasets (14 for *1DSfM*, 48 for *YFCC100*), we use CleanNet-SPT initialization [5] to make the training easier. The CleanNet-SPT initialization uses an MPNN with 3 Edge Convolution layers to predict a probability of an edge, which implies an outlier or not. Then the node with the most neighbors would be chosen as the root node to propagate an initialization through the minimum spanning tree. The strategy of root node selection is heuristic in NeuRoRA [5] because the error on edge will propagate through the spanning tree. Thus NeuRoRA empirically selects the node with the most neighbors as the root node to make the depth from the root to the leaf as small as possible. A random root node will produce an inferior result. Notice that RAGO does not suffer from the gauge freedom problem. Thus we randomly select a node as the root node.

To be noted that, since the scene `Trafalgar` in *1DSfM* is very complex and noisy, we need a better initialization than all the other dataset. As a result, for `Trafalgar`, we take a cascaded framework in which two RAGO networks are used: one for initialization and the other one for refinement. The first RAGO utilizes CleanNet-SPT for initialization, and then we use the discrepancy between the output global camera orientations from the first RAGO and the input relative orientations as the weight to generate a minimum spanning tree. The global camera orientations propagated through the spanning-tree are fed to the second RAGO to generate the final result.

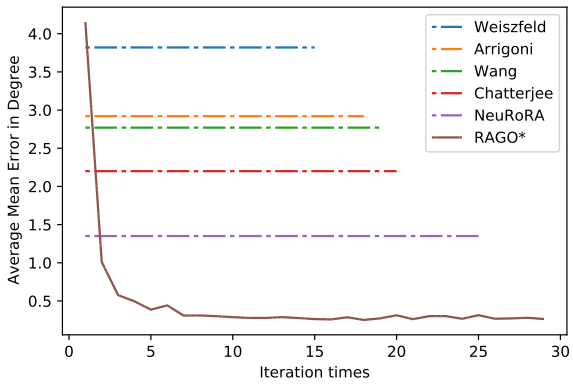

Figure 3. The average mean error of RAGO on the synthetic dataset compared with various MRA methods [1–3,5,6]. The optimized results of the previous approaches are visualized as the dash lines. The vertical axis represents the average mean angular error, while the horizontal axis shows the number of iterations.

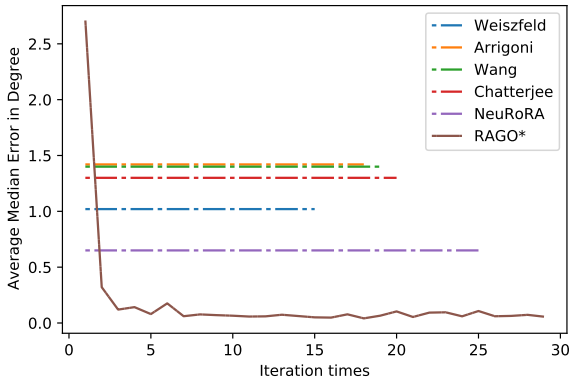

Figure 4. The average median error of RAGO on the synthetic dataset compared with various MRA methods [1–3,5,6]. The optimized results of the previous approaches are visualized as the dash lines. The vertical axis represents the average mean angular error, while the horizontal axis shows the number of iterations.

3. Dataset Detail

3.1. Synthetic Dataset

We use the same configurations and script [5] to generate the synthetic dataset. Because the author does not report the random seed they used to create the dataset, we use the current time as the random seed. We test RAGO and NeuRoRA [5] on the same synthetic dataset and cite the results of other methods from [5]. The visualization of the average of mean and median angular error for comparison during optimization as shown on Figure 3 and Figure 4. We fix rectified relative orientations after 10 iterations.

3.2. Real-World Dataset

1DSfM: We cite the result of other methods on *1DSfM* [7] from MSP [8]. Notice the MSP uses additional input. Thus we underline the second performance for comparison. We fix rectified relative orientations after 10 iterations.
YFCC100: We cite the result of other methods on *YFCC100* [4] from MSP [8]. We use the reconstructed camera pose provided by the author as the ground truth. The relative orientations are provided by [8] computed from COLMAP. We fix rectified relative orientations after 10 iterations.

3.3. Robust Check

We generate several different synthetic datasets for robustness check. We use the following configuration as the default setting: the number of the nodes $|\mathcal{V}| \in [550, 600]$, the percentage of the edge $|\mathcal{E}| \in [25\%, 35\%]$, the error of the edge $\sigma \in [10°, 20°]$ and the percentage of the outliers $o \in [10\%, 20\%]$. This configuration is labeled as $(600, 30\%, 15°, 15\%)$. To check the generalization of RAGO on the view-graph with different number of nodes, we generate two synthetic datasets where $|\mathcal{V}| \in [250, 350]$ labeled as 300, and $|\mathcal{V}| \in [1400, 1600]$ labeled as 1500. For the evaluation of robustness with different $|\mathcal{E}|$, we change $|\mathcal{E}|$ to $[2\%, 4\%]$ and $[55\%, 65\%]$, labeled as 3% and 60%, respectively. The σ is changed to $[4°, 6°]$ and $[50°, 60°]$, marked as 5° and 55° to evaluate the performance of RAGO with different levels of noise. Finally, we generate two synthetic datasets with different percentages of outliers, $[2\%, 4\%]$ and $[55\%, 65\%]$, marked as 3% and 60% to check the influence of the outliers.

4. Limitation

There are some limitations for RAGO. Firstly, RAGO is a learning-to-optimize optimizer, which needs a lot of view-graphs for training. The real-world datasets only have 16 view-graphs on *1DSfM* and 72 view-graphs on *YFCC100*, leading to deploy a robust initialization for a better performance. Notice that the error will accumulate through the edge during camera initialization. The previous methods usually require a heuristic strategy to select a root node to make the depth of the spanning tree as small as possible. However, RAGO only requires a good spanning tree. This property makes RAGO has low sensitivity on initialization. Secondly, RAGO is trained under a supervised manner, which needs the view-graph with ground-truth for training. However, the ground-truth camera orientations of the real-world view-graph are hard to obtain.

References

[1] F. Arrigoni, B. Rossi, P. Fragneto, and A. Fusiello. Robust synchronization in so(3) and se(3) via low-rank and sparse matrix decomposition. *Comput. Vis. and Image Underst.*, 174:95–113, 2018. 2

[2] Avishek Chatterjee and Venu Madhav Govindu. Efficient and robust large-scale rotation averaging. *Int. Conf. Comput. Vis.*, 2013. 2

[3] Richard Hartley, Khurrum Aftab, and Jochen Trumpf. L1 rotation averaging using the weiszfeld algorithm. *IEEE Conf. Comput. Vis. Pattern Recog.*, 2011. 2

[4] Jared Heinly, Johannes Lutz Schönberger, Enrique Dunn, and Jan-Michael Frahm. Reconstructing the World* in Six Days *(As Captured by the Yahoo 100 Million Image Dataset). *IEEE Conf. Comput. Vis. Pattern Recog.*, 2015. 3

[5] Pulak Purkait, Tat-Jun Chin, and Ian Reid. Neurora: Neural robust rotation averaging. *Eur. Conf. Comput. Vis.*, 2020. 2

[6] Lanhui Wang and Amit Singer. Exact and stable recovery of rotations for robust synchronization. *Information and Inference: A Journal of the IMA*, 2(2):145–193, 2013. 2

[7] Kyle Wilson and Noah Snavely. Robust global translations with 1dsfm. *Eur. Conf. Comput. Vis.*, 2014. 3

[8] Luwei Yang, Heng Li, Jamal Ahmed Rahim, Zhaopeng Cui, and Ping Tan. End-to-end rotation averaging with multi-source propagation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11774–11783, June 2021. 3

[9] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks. *IEEE Conf. Comput. Vis. Pattern Recog.*, 2019. 1