RAGO: Recurrent Graph Optimizer For Multiple Rotation Averaging

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

Abstract

This paper proposes a deep recurrent Rotation Averaging Graph Optimizer (RAGO) for Multiple Rotation Averaging (MRA). Conventional optimization-based methods usually fail to produce accurate results due to corrupted and noisy relative measurements. Recent learning-based approaches regard MRA as a regression problem, while these methods are sensitive to initialization due to the gauge freedom problem. To handle these problems, we propose a learnable iterative graph optimizer minimizing a gauge-invariant cost function with an edge rectification strategy to mitigate the effect of inaccurate measurements. Our graph optimizer iteratively refines the global camera rotations by minimizing each node's single rotation objective function. Besides, our approach iteratively rectifies relative rotations to make them more consistent with the current camera orientations and observed relative rotations. Furthermore, we employ a gated recurrent unit to improve the result by tracing the temporal information of the cost graph. Our framework is a real-time learning-to-optimize rotation averaging graph optimizer with a tiny size deployed for real-world applications. RAGO outperforms previous traditional and deep methods on real-world and synthetic datasets. The code is available at github.com/sfu-gruvi-3dv/RAGO.

1. Introduction

Multiple Rotation Averaging (MRA) [4, 21, 28, 38] is a fundamental problem in 3D computer vision that aims to determine the global absolute orientations $\{\mathbf{R}_u, 1 \leq u \leq N\}$ of N cameras given their relative orientations $\mathbf{R}_{uv}$. It has been widely studied in 3D vision applications, *e.g.*, global Structure from Motion (SfM) [12, 24], pose graph optimization in Visual Simultaneous Localization and Mapping (SLAM) [14, 27, 35], and other sensor network problems [36, 37].

MRA is usually solved by minimizing the discrepancy between the observed relative orientations $\mathbf{R}_{uv}$ and the one calculated from the estimated global orientations, i.e. $\mathbf{R}_u\mathbf{R}_v^\top$. It is a difficult classical problem with several challenges [21]. Firstly, it is a highly nonlinear problem since the distance between two rotation matrices is a nonlinear function. Secondly, rotation matrices lie on the $SO(3)$ group, which requires careful parameterization and normalization during optimization. Thirdly, there are many outliers and various noises in input relative orientations $\mathbf{R}_{uv}$, which are often computed from noisy visual correspondences across images. These problems make the minimization objective function full of saddle points and local minimums in separate basins of attraction. Most of the time, the global optimum cannot be guaranteed. The MRA problem is typically solved by iterative optimization with careful initialization [6, 20, 40]. Robust cost functions with additional outlier filtering and iterative reweighting of the measurements are usually adopted, while these still do not tolerate severe corruptions and various noises.

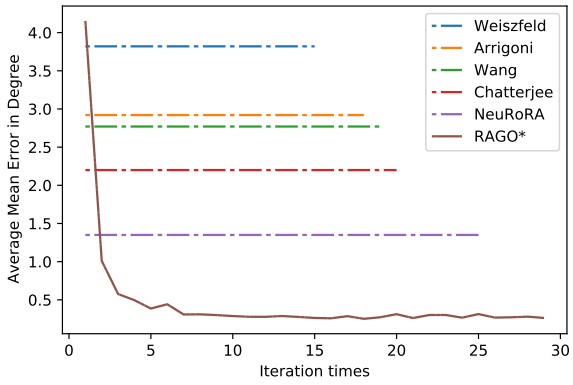

Figure 1. The average mean error of RAGO on the synthetic dataset compared with various MRA methods [5, 6, 20, 30, 39]. The optimized results of the previous approaches are visualized as the dash lines. The vertical axis represents the average mean angular error, while the horizontal axis shows the number of iterations.

Recently, some learning-based methods [30, 42] formulate MRA as a regression problem in order to benefit from data-driven prior knowledge. These methods rely on a good initialization, since they do not enforce any geometric constraint during inference, which may lead to an inferior re-

sult. Furthermore, there is a gauge freedom in MRA that prevents learning-based methods from direct end-to-end training, where $\{\mathbf{R}_u\}$ and $\{\mathbf{R}_u\mathbf{R}_0\}$ are essentially the same solution for an arbitrary rotation matrix $\mathbf{R}_0$. These learning-based methods [30,42] have to choose a root node as a reference to avoid a one-to-many mapping, which makes it hard to learn to solve the MRA problem.

This paper presents a novel learning-based method that has the advantages of both geometrical and learning-based methods. Specifically speaking, we decouple an MRA problem to multiple Single Rotation Averaging (SRA) problems as inspired by the traditional method in [20]. An SRA problem solves the rotation matrix $\mathbf{R}_u$ from all pairwise relative rotation $\{\mathbf{R}_{uv}, v \in \mathcal{N}_u\}$, where $\mathcal{N}_u$ is the neighborhood of u. We construct a cost graph by computing an SRA cost function for each node independently.We then apply a Massage Passing Neural Network (MPNN) to iteratively adjust the global camera rotations of all nodes by minimizing the SRA cost graph, resulting in an iterative optimization of the original MRA problem. Unlike traditional methods that only can consider one-hop neighbors, MPNN has large respective fields and achieves better results. Compared to previous learning-based methods, our framework focuses on solving SRA, a much simpler and smaller problem without the gauge ambiguity, making learning easier.

In addition, in order to handle noises and outliers, we also learn to rectify the relative measurements $\mathbf{R}_{uv}$. Our approach avoids time-consuming online refinement or edge reweighting, which makes training unstable. In order to make it more robust and efficient, we employ a Gated Recurrent Unit (GRU) module to utilize the historical information of the previous cost graph. This module helps our optimizer to converge to a better solution.

Experiments on real and synthetic datasets show that our method could converge to good result, even starting from random initialization, while previous methods usually required more careful initialization. As shown in Figure 1, we compare RAGO with various MRA methods on the synthetic datasets in terms of average mean angular error. RAGO outperforms these approaches after 2 iterations.

Our contributions can be summarized as follows:

- We present a novel end-to-end learning-to-optimize recurrent graph neural network for MRA.
- We decouple an MRA problem to multiple SRA problems, leading to better results and learning without gauge ambiguity.
- We propose to rectify the relative orientations during optimization to handle outliers and noises.
- Our method outperforms state-of-the-art methods on multiple real and synthetic datasets.

2. Related Work

Conventional MRA: Govindu first introduced MRA with his linear motion model [18] and lie-group-based averaging [19]. More recent iterative optimization-based approaches [6, 11, 20, 33, 40, 40] introduce robust optimization strategies to reduce the influence of outliers. The vast majority of these algorithms were iterative and aimed to optimize a robust cost function. Hartley *et al*. [20] optimized each camera's absolute orientation using the median orientation calculated from its neighbors in each iteration using the Weiszfeld averaging algorithm. Chatterjee *et al*. [6] fine-tuned the initialization provided by a spanning tree using iterative reweighted least-squares (IRLS) minimization with an L1 loss function. Fredriksson and Olsson [15] turn the original problem into a dual problem utilizing Lagrangian Duality and then solve it using SDP to arrive at an optimized solution. Numerous approaches [5, 13, 26, 29] are based on this pipeline for improved performance, as this approach benefits in achieving the global minimum. Dellaert *et al*. [13] solves the MRA locally on $SO(3)$ and then increases the manifold dimension to start the optimization again. Moreira *et al*. [26] present a primal-dual method to solve MRA, inspired by in optimization algorithms with orthogonality constraints. These approaches primarily aim to decrease the complexity of non-convex optimization. Dealing with outliers remains an open issue, as they either assume no noise or assume a specific kind of noise model.

Learning-based MRA: Recently, several neural network based methods [17, 23, 30, 42, 43] have been proposed. NeuRoRA [30] employs a two-stage neural network architecture based on MPNN [16]. The first network filters outliers and rectifies relative orientations to improve the SPT-based initialization. The second stage fine-tunes the camera's orientation for a better result. MSP [42], based on NeuRoRA [30], takes appearance information as input and introduces a differentiable SPT method to achieve a robust initialization result. The initialization is further improved through non-learnable iterative edge reweighting. However, these approaches have to choose a node as root to enforce a unique solution, making their results sensitive to initialization. In contrast, we regard the MRA as an optimization problem and iteratively update the variables with a message passing neural network combined with gated recurrent units to exploit temporal information.

Learning to Optimize: Numerous recent publications attempt to combine the strength of neural networks with classic optimization-based methods. There are primarily two dominant directions in optimization learning. The first one [2, 3, 9, 34] substitutes a neural network for the non-differentiable component of a traditional optimizer during end-to-end training. Other approaches [1, 7, 10, 31] use machine learning to update optimization variables based on the input data directly. However, all approaches need ex-

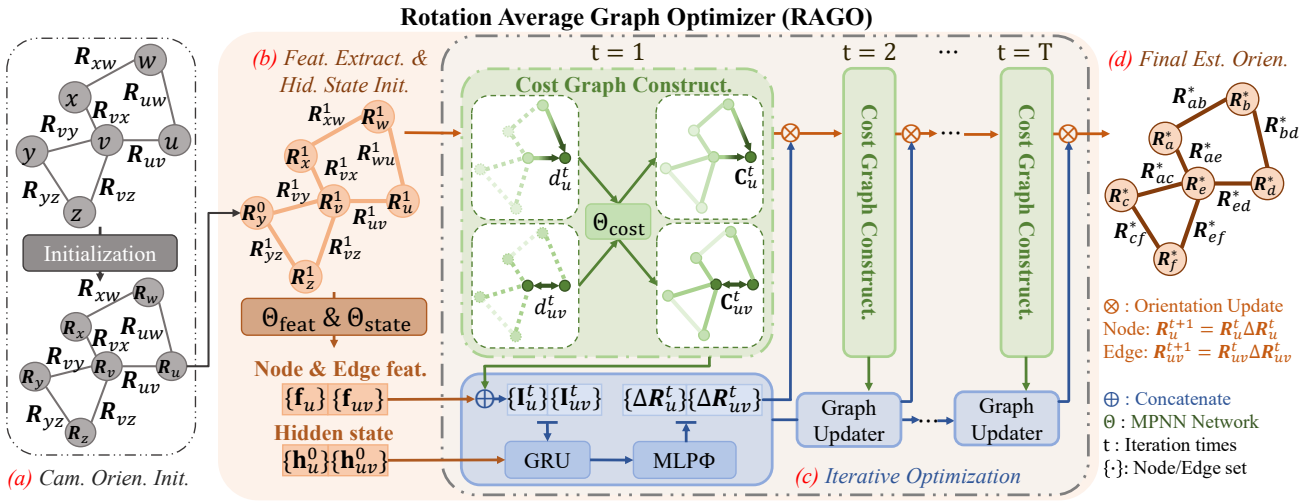

Figure 2. The pipeline of our Rotation Averaging Graph Optimizer (RAGO). (a) We assign a random rotation to each camera. In (b), Two Message Passing Neural Network, Θ_{feat} and Θ_{state}, extract the feature and initialize the hidden state. (c) During iterative optimization, we first build a cost graph based on the SRA objective function. Then, we employ a GRU to update the camera orientations to minimize the cost graph. (d) The camera orientations eventually converge to an optimized solution. Please refer to the main text for more details.

plicit formulation of the solver and are restricted to problems with easily defined objective functions. Additionally, the approaches [10,34] must evaluate the gradient of the objective functions, which is complicated with many issues, particularly optimization on a manifold. Unlike these previous works, our method decouples MRA to multiple SRA problems, which is easier to learn, avoiding gradient computations.

3. Deep Graph Optimizer for MRA

Consider a view-graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ where each vertex $v \in \mathcal{V}$ corresponds to an unknown absolute camera orientation and each edge $e \in \mathcal{E}$ is an observed relative orientation. The MRA problem aims to estimate a set of camera orientations $\{\mathbf{R}_u^*\} = \{\mathbf{R}_1^*, \dots, \mathbf{R}_N^*\}$ that minimizes the discrepancy between the estimated and observed relative orientations, which can be formulated as:

$$\{\mathbf{R}_u^*\} = \arg\min_{\{\mathbf{R}_u\}} \sum_{(u,v)\in\mathcal{E}} \rho(d(\mathbf{R}_{uv}, \mathbf{R}_u\mathbf{R}_v^\top)), \tag{1}$$

where $\{\mathbf{R}_u^*\}$ is the set of optimized global camera orientations, $\rho(\cdot)$ is a robust cost function and $d(\cdot,\cdot)$ is the distance between two rotation matrices.

In Single Rotation Averaging (SRA) [21], a single rotation is averaged over several observed relative rotations, which can be formulated as:

$$\mathbf{R}_u^* = \arg\min_{\mathbf{R}_u} \sum_{v\in\mathcal{N}_u} \rho(d(\mathbf{R}_u, \mathbf{R}_{uv}\mathbf{R}_v)), \tag{2}$$

where $\mathcal{N}_u$ is the set of neighboring nodes of u. The MRA problem can be solved by iteratively solving multiple SRA problems to adjust each camera's rotation based on the orientations of its direct neighbors [20]. The overall cost decreases at each step of this procedure and therefore converges to a local minimum.

3.1. Overview

The overall pipeline of our framework is depicted in the Figure 2. For camera orientation initialization in Figure 2 (a), we assign a random rotation to each camera, because our method is designed to work with random initialization. We can also use more sophisticated initialization [30, 42] to enhance the robustness of our method further. In particular, we choose CleanNet-SPT [30] initialization for all real-world datasets.

In Figure 2 (b), two neural networks based on Message Passing Neural Network (MPNN) [16], Θ_{feat} and Θ_{state}, extract local features $\{\mathbf{f}_u, \mathbf{f}_{uv}\}$ and initialize the hidden state $\{\mathbf{h}_u^0, \mathbf{h}_{uv}^0\}$ of each node and edge in the view-graph, which is introduced in Section 3.2.1 and Section 3.2.2.

Figure 2 (c) is the iterative optimization explained in Section 3.2.3, where we compute an SRA cost $\{d_u^t\}$ at each node and $\{d_{uv}^t\}$ at each edge from the current result at the t-th iteration. Then, an MPNN Θ_{cost} extracts the cost feature $\{\mathbf{C}_u^t, \mathbf{C}_{uv}^t\}$ from the cost graph $\{d_u^t, d_{uv}^t\}$. The graph updater receives the cost features $\{\mathbf{C}_u^t, \mathbf{C}_{uv}^t\}$, the graph features $\{\mathbf{f}_u, \mathbf{f}_{uv}\}$, and the previous hidden states $\{\mathbf{h}_u^{t-1}, \mathbf{h}_{uv}^{t-1}\}$ to generate an incremental update $\{\Delta\mathbf{R}^t\}$ to minimize the cost graph.

After several iterations, the global camera orientation converges to an optimized solution as Figure 2 (d). The Algorithm 1 demonstrate the pipeline of RAGO without alternative optimization mentioned in Section 3.

3.2. Rotation Averaging Graph Optimizer(RAGO)

3.2.1 Graph Feature Extraction

We use a Message Passing Neural Network [16] (MPNN) Θ_{feat} with one Edge Convolution layer as the backbone to extract the node feature $\{\mathbf{f}_u\}$ and edge feature $\{\mathbf{f}_{uv}\}$ of the input view-graph. We replace camera orientations on nodes with zero vectors and only extract features from observed relative orientations on edges. Consider an edge e_{uv} with a feature $\mathbf{f}_{uv}$ connecting nodes u and v, where the node feature is denoted by $\mathbf{f}_u$ and $\mathbf{f}_v$. At each Edge Convolution layer, the node and edge features are updated by aggregating their neighbors' information and then passed to the next layer. To update the edge feature, Edge Convolution concatenates the node feature and edge feature as $[\mathbf{f}_u, \mathbf{f}_v, \mathbf{f}_{uv}]$. Then the concatenated feature is passed through a 3-layer Multi-layered Linear Perception (MLP) Φ_{edge} to generate the updated edge feature $\mathbf{f}'_{uv}$. A node MLP Φ_{node} then updates the node feature $\mathbf{f}'_u$ by aggregating the adjacent updated edge features. The structure inside an Edge Convolution is as follows:

$$\begin{aligned} \mathbf{f}^{'}_{uv} &= \Phi_{\text{edge}}([\mathbf{f}_u, \mathbf{f}_v, \mathbf{f}_{uv}]), \\ \mathbf{f}^{'}_{u} &= \Phi_{\text{node}}(\text{mean}(\{\mathbf{f}^{'}_{uv}, v \in \mathcal{N}_u\})), \end{aligned} \tag{3}$$

where $\mathcal{N}_u$ is the neighbor node set of the node u. Finally, we apply a 3-layer MLP to nodes and edges from the final Edge Convolution layer to get a feature of specified dimension on both nodes as $\{\mathbf{f}_u\}$ and edges as $\{\mathbf{f}_{uv}\}$. Please refer to the supplementary material for more details.

3.2.2 Hidden state Initialization

The Gated Recurrent Unit (GRU) module in our graph updater introduced in Section 3.2.3 requires a hidden state for each node and edge in the view-graph to utilize temporal information during iteration. We generate the initial hidden states $\{\mathbf{h}^0_u\}$ on nodes and $\{\mathbf{h}^0_{uv}\}$ on edges by passing the view-graph to another MPNN Θ_{state}, which has the same structure and input as Θ_{feat}. Finally, The hidden states $\{\mathbf{h}^0_u, \mathbf{h}^0_{uv}\}$ is mapped to $(-1, 1)$ by the tanh function. Similar to the graph feature extraction, the orientations on nodes are replaced with zeros vectors.

3.2.3 Iterative optimization

SRA Cost Graph Construction: At the heart of our proposed framework is the construction of the cost graph. Here we define a cost on each node and each edge.

Conventional optimization-based methods for MRA usually use Equation 1 as the objective function. However, it is difficult to enforce the minimization of this objective function in learning-based methods. Compared with solving MRA directly, SRA is a simpler problem and easier to learn for a neural network. Thus, we decouple the MRA problem into multiple SRA problems, and compute an SRA cost for each node as,

$$d^t_u = \frac{1}{|\mathcal{N}_u|} \sum_{v \in \mathcal{N}_u} ||\mathbf{R}^t_u - \mathbf{R}_{uv}\mathbf{R}^t_v||_1, \tag{4}$$

where $||.||_1$ is the L1 norm, $\mathcal{N}_u$ is the neighbor node set of node u.

Outlier rejection during optimization is non-trivial too. Previously, many methods reweight the edge during optimization. We find that reweighting the edge makes training unstable. In contrast, we introduce a relaxing parameter $\mathbf{R}^t_{uv}$ on each edge to mitigate the influence of outliers and noisy orientations during optimization. In particular, for each input relative orientation $\mathbf{R}_{uv}$, we compute the estimated relative orientation from global camera orientations as $\mathbf{R}^t_u\mathbf{R}^{t\top}_v$. We then estimate a rectified relative orientation $\mathbf{R}^t_{uv}$ that is close to both $\mathbf{R}_{uv}$ and $\mathbf{R}^t_u\mathbf{R}^{t\top}_v$. If a measurement $\mathbf{R}_{uv}$ is an outlier, the rectified rotation $\mathbf{R}^t_{uv}$ will be far from the input rotation $\mathbf{R}_{uv}$. The SRA cost function on the edge is then defined as:

$$d^t_{uv} = ||\mathbf{R}^t_{uv} - \mathbf{R}^t_u\mathbf{R}^{t\top}_v||_1 + ||\mathbf{R}^t_{uv} - \mathbf{R}_{uv}||_1. \tag{5}$$

Finally, an MPNN Θ_{cost} with three Edge Convolution layers extracts cost features $\{\mathbf{C}^t\}$ from the cost graph $\{d^t\}$. The cost feature at a node or edge has information on all of its 3-order neighbors, because it is updated three times by the Edge Convolutions, which leads to better convergence in our iterative optimization. Notice that the choice of distance $d(\cdot,\cdot)$ and robust functions $\rho(\cdot)$ is trivial in RAGO since the cost eventually maps to a feature space.

Recurrent Graph Updater: Our graph updater includes two GRUs [8] to update the global camera orientations on the nodes and the rectified relative orientations on edges, respectively. The GRUs can efficiently utilize the information in the previous iteration steps for better optimization. We concatenate the current cost feature on the node (edge) $\mathbf{C}^t$, current orientations $\mathbf{R}^t$, and the graph local feature $\mathbf{f}$ to create an input $\mathbf{I}^t = [\mathbf{C}^t, \mathbf{R}^t, \mathbf{f}]$ for each iteration. GRUs receive previous hidden states $\{\mathbf{h}^{t-1}\}$ and the current input $\{\mathbf{I}^t\}$, then outputs the current hidden states $\{\mathbf{h}^t\}$. Then, the incremental update rotation $\Delta\mathbf{R}^t$ is predicted from the hidden state $\{\mathbf{h}^t\}$ by an MLP:

$$\begin{aligned} \mathbf{h}^t &= \text{GRU}(\mathbf{h}^{t-1}, \mathbf{I}^t), \\ \Delta\mathbf{R}^t &= \Phi(\mathbf{h}^t). \end{aligned} \tag{6}$$

The estimated and rectified orientations on nodes and edges are then updated as:

$$\mathbf{R}^{t+1}_u = \mathbf{R}^t_u\Delta\mathbf{R}^t_u, \quad \mathbf{R}^{t+1}_{uv} = \mathbf{R}^t_{uv}\Delta\mathbf{R}^t_{uv}. \tag{7}$$

With this rotation averaging graph optimizer, starting from an initial guess, the orientations on the view-graph are

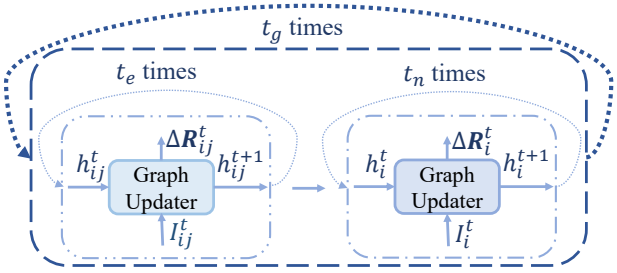

Figure 3. Alternative Optimization. RAGO first refines the relative orientations for t_e times, and then optimizes the estimated camera orientations for t_n times. The orientations on edges and nodes will iterative $t_e \times t_g$ and $t_n \times t_g$ times in total, respectively.

refined by the optimization iterations and eventually converge to optimized camera orientations and relative orientations, $\mathbf{R}_u^* \leftarrow \mathbf{R}_u^t, \quad \mathbf{R}_{uv}^* \leftarrow \mathbf{R}_{uv}^t$.

3.2.4 Alternative Optimization

Although we can optimize the variables on nodes and edges simultaneously, we find that the rectified relative rotation converges faster than the estimated camera orientation. Thus, as shown in the Figure 3, we iteratively optimize edge and node in turns. For all of our experiments, we fix the number of iterations for graph optimization t_g, edge optimization t_e, and node optimization t_n during training.

3.3. Training Loss

We train our graph optimizer in a supervised manner with ground-truth camera orientations. Different from the previous learning-based methods [30, 42], we do not define a loss on the absolute camera rotation to enforce a unique result. We only use ground-truth relative orientations to supervise our graph optimizer:

$$L_{\text{opt}} = \frac{1}{|\mathcal{E}|}\sum_{i=1}^{T_n}\sum_{(u,v)\in\mathcal{E}}\gamma^{T_n-i}||\mathbf{R}_u^i\mathbf{R}_v^{i\top} - \bar{\mathbf{R}}_{uv}||_1 + \frac{1}{|\mathcal{E}|}\sum_{i=1}^{T_e}\sum_{(u,v)\in\mathcal{E}}\gamma^{T_e-i}||\mathbf{R}_{uv}^i - \bar{\mathbf{R}}_{uv}||_1, \tag{8}$$

where $\bar{\mathbf{R}}_{uv}$ is the ground truth relative orientation, γ is a discounting factor and T_n and T_e are the total number of optimization iterations for nodes and edges, where $T_e = t_g \times t_e$, and $T_n = t_g \times t_n$.

4. Experiments

Synthetic dataset: We evaluate on the public synthetic dataset [30]. This dataset is generated randomly with carefully designed noise and outlier distributions resembling real-world data. Generally speaking, a synthetic view-graph is generated by the following steps: 1) The number of nodes is sampled uniformly between 250 and 1000, and the orientation on each node is generated randomly on a horizontal plane. 2) Edges indicating relative rotations are randomly generated by the Erdős–Rényi model. The number of edges is set to $[10\% - 30\%]$ of all possible pairs. 3) The relative orientations are corrupted by a Gaussian noise with a standard deviation σ uniformly sampled in the range $[5° - 30°]$. 4) Finally, $[0\% - 30\%]$ of edges in the view-graph are replaced by random orientations as outliers. Similar to NeuRoRA [30], we generate 1,000 view-graphs for training, 100 for validation, and 100 for testing. The parameters that yield the minimum validation loss are kept for testing.

Algorithm 1: RAGO Inference

Input : $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}, \{\mathbf{R}_{uv}\}|(u,v) \in \mathcal{E}$
Output: Global absolute rotations $\mathbf{R}_u$ for $u \in \mathcal{V}$

Initialization:
$R_u^1 \leftarrow \text{rand}(SO(3)), \forall u \in \mathcal{V}$ ▷ Set initial to a random rotation
$R_{uv}^1 \leftarrow \mathbf{R}_1, \forall uv \in \mathcal{E}$ ▷ Set initial to identity matrix
$\{\mathbf{f}\} \leftarrow \Theta_{\text{feat}}(\mathbf{R}_{uv})$ ▷ Extract feature from relative rotations
$\{\mathbf{h}^0\} \leftarrow \Theta_{\text{state}}(\mathbf{R}_{uv})$ ▷ Initialize hidden state for GRU
Iterative Optimization:
$t \leftarrow 1$
while $(t \le T)$ **do**
  Building Cost Graph:
  $\{\mathbf{d}_u^t\} \leftarrow$ Equation 4 ▷ Computing SRA cost
  $\{\mathbf{d}_{uv}^t\} \leftarrow$ Equation 5 ▷ Computing edge cost
  $\{\mathbf{C}^t\} \leftarrow \Theta_{\text{cost}}(\{\mathbf{d}^t\})$ ▷ Extracting cost feature
  Updating Rotation:
  $\mathbf{I}^t = [\mathbf{C}^t, \mathbf{R}^t, \mathbf{f}]$ ▷ Creating input feature
  $\{\Delta\mathbf{R}^t\}, \{\mathbf{h}^t\} \leftarrow$ Equation 6 ▷ Computing updating rotation
  $\mathbf{R}_u^{t+1} = \mathbf{R}_u^t \Delta\mathbf{R}_u^t$ ▷ Updating camera rotation
  $\mathbf{R}_{uv}^{t+1} = \mathbf{R}_{uv}^t \Delta\mathbf{R}_{uv}^t$ ▷ Updating rectification matrices
  $t \leftarrow t + 1$
end

Real-world datasets: We also evaluate on the real-world datasets *1DSfM* [41] and *YFCC100* [22]. The *1DSfM* contains 14 outdoor scenes with ground-truth camera poses and relative orientations computed by Bundler [32]. Only the cameras with ground-truth orientations are used for training and testing. The *YFCC100* dataset consists of internet images at 72 city-scale scenes. We use the author's reconstructed camera poses as ground-truth and use relative orientations by COLMAP [32] provided in MSP [42] for training and testing. We train and test our method on the *1DSfM* dataset in a leave-one-out manner. The *YFCC100* dataset is split into two sets, one for training (58 scenes) while the other for testing (14 scenes). To avoid overfitting, we randomly drop 20% edges of the view-graph during training. Due to the limitation of the training sample, we use CleanNet-SPT [30] for global camera orientation initialization during training and testing on *1DSfM* and *YFCC100*.

Method	$t=1$		$t=5$		opt.		Converge
	mn	md	mn	md	mn	md	Y/N
Ours	**4.03**	**2.41**	**0.66**	**0.20**	**0.24**	**0.04**	Y
NeuRoRA [30]	-	-	-	-	1.35	0.65	Y
Chatterjee [6]	-	-	-	-	2.20	1.30	Y
Shonan [13]	-	-	-	-	2.43	1.58	Y
MAKS [26]	-	-	-	-	2.64	1.40	Y
Wang [39]	-	-	-	-	2.77	1.40	Y
Arrigoni [5]	-	-	-	-	2.92	1.42	Y
Weiszfeld [20]	-	-	-	-	3.35	1.02	Y
with GRU	4.03	2.41	**0.66**	**0.20**	**0.24**	**0.04**	Y
w/o GRU	**3.31**	**2.18**	0.75	0.26	0.46	0.17	N
SRA Cost	**4.03**	**2.41**	0.66	0.20	**0.24**	**0.04**	Y
Deg. Met.	4.38	2.99	**0.48**	**0.13**	0.28	0.06	Y
Null Vec.	4.40	3.71	1.09	0.46	0.65	0.32	Y
MRA Cost	6.65	4.23	3.95	2.31	3.80	2.22	N
3 Edge Conv	**4.03**	**2.41**	**0.66**	**0.20**	**0.24**	**0.04**	Y
2 Edge Conv	4.37	2.62	0.68	0.21	0.41	0.11	Y
1 Edge Conv	7.15	4.05	1.39	0.58	0.71	0.25	Y
Random Init.	4.03	2.41	0.66	0.20	0.24	**0.04**	Y
Rand. SPT	3.24	1.84	0.44	0.14	0.27	0.06	Y
Clean. SPT	**2.74**	**1.44**	**0.34**	**0.12**	**0.23**	**0.04**	Y

Table 1. The results of comparison and ablation study on the synthetic dataset. We mark the final results of various MRA methods [5, 6, 13, 20, 26, 30, 39] as opt. The average mean(mn) and median(md) angular errors on the view-graphs in the test set are reported. The entries with the best performance are **bolded**. The settings used in the proposed model are underlined.

Comparison: We compare our method with conventional optimized-based methods, including Chatterjee *et al*. [6], MPLS [33], Arrigoni *et al*. [5], Wang *et al*. [39], Weiszfeld [20], Shonan [13], MAKS [26] and state-of-art deep learning based methods, including NeuRoRA [30], MSP [42]. We use a publicly available evaluation script [6] to compare predicted absolute camera orientations and ground-truth camera orientations in terms of mean(mn) and median(md) angular errors. Notice that RAGO does not resolve the gauge ambiguity. The output camera rotations need to align with the ground truth to evaluate the accuracy.

implementation details: Our approach is implemented in Pytorch with an Nvidia V100 GPU. The model is trained with a adamW [25] optimizer ($\beta_1 = 0.9, \beta_1 = 0.999$). The training runs for $20,000$ epochs started with a learning rate 1×10^{-3}. After 100 epochs, the learning rate decay exponentially by 0.999 for each epoch. The γ defined in Section 3.3 is set as 0.8 for all experiments. During Training, we set t_g to 3, t_e to 1, and t_n to 4. We empirically terminate it when $t_g = 5$ during testing. The channel number of feature and hidden state is 48. We use the Orth6D rotation representation proposed in [44]. Orth6D is a continuous 6D space for 3D rotation matrices, while the quaternion, rotation matrix, and Euler angles are not contiguous in Euclidean space. In comparison, Orth6D enables RAGO to use the rotation matrix as the neural network's direct input and output. Please refer to supplementary material for more details.

4.1. Synthetic dataset

Table 1 shows the results of our method on the synthetic dataset. We show the average mean and median angular errors for all view-graphs in the test set. We report the results of our method at iteration 1 and 5. The result of iteration 20 is marked as opt. We compare our method with the learning-based approach NeuRoRA [30] and the conventional method Chatterjee [6], Arrigoni [5], Wang [39] and Weiszfeld [20], Shonan [13] and MAKS [26]. The final results of these approaches are marked bolded. Although our results have relatively large error at the 1st iteration, they outperform the other methods after the 5-th iteration and finally converge to 0.24 and 0.04 in mean and median angular error. Figure 1 shows the mean error during iterative optimization, indicating fast convergence and significantly improved accuracy compared with conventional and learning-based methods.

4.2. Ablation study

To understand the different components of our method, we conduct an ablation study on the synthetic dataset, with results summarized in Table 1.

GRU Module: To see the effectiveness of utilizing the history information during optimization, we replace the GRU module with a 6-layers MLP. The errors also reduce rapidly in the first several iterations, but the results eventually diverge, yielding a poorer accuracy.

Different Metrics on Cost Graph: To evaluate the effectiveness of different cost functions in the cost graph, we experiment the cost defined in Section 3.2.3 with angular degree distance, MRA cost, and null vector. For angular degree distance cost, we substitute angular degree error for L1 norm. For MRA cost, we maintain the cost constant on edge but substitute the cost on node with the cost function of MRA defined as Equation 1. Finally, to assess the efficacy of the cost graph, we substitute a zero vector as a null vector for all costs on the graph. As shown on Table 1, the model with the angular degree metrics has results as the proposed model. Although the cost graph with the null vector has inferior results than the proposed model, it still outperforms the baselines due to iterative optimization and temporal information. The result drops dramatically, if we use MRA cost because it is evaluated on the whole view-graph, making it hard for the neural network to learn. This comparison demonstrates the advantages of solving MRA through solving multiple SRA problems.

Number of Edge Conv: As introduced in Section 3.2.1, At each Edge Convolution layer, the node and edge features are updated by aggregating the neighbors' information and then passed to the next layer. Thus, the number of layers of Edge Convolution layer will affect the receptive field of each entity on the view graph. To demonstrate the effectiveness of the size of the receptive field, We train 3 models with dif-

Datasets			Chatterjee [6]		Weiszfeld [20]		NeuRoRA [30]		MPLS [33]		MSP [42]		Ours	
#image	#edge	Names	mn	md	mn	md	mn	md	mn	md	mn	md	mn	md
577	59.5%	Alamo	4.2	1.1	4.9	1.4	4.94	1.16	3.44	1.16	2.89	1.07	**2.82**	**0.88**
227	66.8%	Ellis Island	2.8	0.5	4.4	1.0	2.59	0.64	2.61	0.88	1.88	0.83	**1.74**	**0.46**
677	17.5%	Gendarmenmarkt	37.6	7.7	29.4	9.6	**4.51**	2.94	44.9	8.0	6.29	3.69	5.24	**2.68**
341	30.7%	Madrid Metropolis	6.9	1.2	7.5	2.7	**2.55**	1.13	4.65	1.26	2.96	1.09	3.05	**1.03**
450	46.8%	Montreal Notre Dame	1.5	0.5	2.1	0.7	1.2	0.6	1.04	0.51	0.91	0.5	**0.86**	**0.46**
338	39.5%	Piazza del Popolo	4	0.8	4.8	1.3	3.05	0.79	3.73	1.93	2.68	0.76	**1.91**	**0.63**
1084	10.9%	Roman Forum	3.1	1.5	4.8	1.8	2.39	1.31	2.62	1.37	**2.04**	1.19	2.55	**1.10**
472	18.5%	Tower of London	3.9	2.4	4.7	2.9	2.63	1.46	3.16	2.2	2.55	1.25	**2.51**	**1.20**
5058	4.6%	Trafalgar	3.5	2	15.6	11.3	5.33	2.25	-		Out of Memory		**2.23**	**1.53**
789	5.9%	Union Square	9.3	3.9	40.9	10.3	5.98	2.01	6.54	3.48	**4.37**	**1.85**	4.68	1.92
836	24.6%	Vienna Cathedral	8.2	1.2	11.7	1.9	**3.9**	1.5	7.21	2.83	3.91	1.1	6.05	**0.89**
437	26.5%	Yorkminster	3.5	1.6	5.7	2.0	2.52	0.99	2.47	1.45	2.27	**0.91**	**2.18**	0.92
2152	10.2%	Piccadilly	6.9	2.9	26.4	7.5	4.75	1.91	3.93	1.81	3.63	1.8	**2.44**	**0.58**
332	29.3%	NYC Library	3	1.3	3.8	2.1	1.9	1.18	2.63	1.24	**1.75**	1.12	2.02	**0.71**

Table 2. Comparison of results on the ***1DSfM*** dataset. We compare our method with various SOTA MRA methods. Mean(mn) and median(md) angular errors on the estimated absolute rotations are compared. The entries with the best performance is **bolded**, the second is underlined. Notice that the result of MSP is computed by using additional input, such as image and correspondence.

ferent numbers of Edge Convolution layers in MPNN Θ_{cost}. The models trained with 2 and 3 Edge Convolution layers have comparable results, while the performance will drop significantly on the setting with only 1 Edge Convolution layer. This experiment shows the information from farther neighbors would be helpful to achieve a better convergence.

Camera Orientation Initialization: To study the effectiveness of different initialization, we train 3 models respectively with: random initialization (Random Init.), random spanning tree initialization (Rand. SPT), CleanNet-SPT initialization (Clean. SPT). For the random spanning tree initialization, we randomly generate a spanning tree of the view-graph, then uniformly select a root node to compute propagate an initialization through the spanning tree. CleanNet-SPT uses an MPNN with 3 Edge Convolution layers to predict each edge is 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 ablation result shows that all three initialization methods can converge to an optimized solution, while the models with random SPT and CleanNet-SPT initialization could converge faster. Notice that RAGO has not been associated with gauge ambiguity. The strategy of how to select a root node becomes trivial. Although our method has similar optimized results with the different approaches on the synthetic dataset, an appropriate initialization is still needed for more complicated real-world datasets due to the limitation of the training samples.

4.3. Results on Real World Dataset

1DSfM: The comparison on the *1DSfM* dataset are listed in Table 2. Notice that MSP [42] uses additional information as input, *e.g.* correspondences, while others only use observed relative orientations as input. Our method outperforms other methods in median angular error in most scenes. In terms of mean of angular error, our approach has the best performance on half of the scenes and yields comparable results on the remaining ones. Our graph optimizer only performs slightly inferior on `Gendarmenmarkt`, `Vienna Cathedral` and `Madrid Metropolis` compared with NeuRoRA [30] with the same input.

YFCC100: The results on *YFCC100* dataset are listed on Table 3. We cite the result from MSP [42] directly. We compare our graph optimizer with several MRA methods. Our method outperforms previous methods in most scenes in terms of mean of angular error except `Florence_cathedral_side` because *YFCC100* contains more view-graphs for training compared with *1DSfM*. NeuRoRA produces large mean and median error on `colosseum_exterior`, while ours could still optimize to a strong result. MSP [42] has a slightly better result on `statue_of_liberty_1` compared with ours due to additional input and robust global camera initialization.

4.4. Robust Check

This experiment shows the generalization capacity of the RAGO. To generate the synthetic datasets, we use the configuration of $(|\mathcal{V}|, |\mathcal{E}|, \sigma, o) = (600, 30\%, 15°, 15\%)$ as the default setting. To check the individual effects of different sensor settings, we generate some synthetic datasets varying: 1) the number of the cameras $|\mathcal{V}|$, 2) the percentage of the edges $|\mathcal{E}|$, 3) std of the edge error σ and 4) the percentage of outlier edge o. RAGO is then trained on one of such datasets and evaluated on the others. Each dataset consists of $1,000$ view-graphs for training, 100 for testing.The results of the robustness check as shown in Table 4. We report the average mean and median angular error on the

Datasets			Chatterjee [6]		NeuRoRA [30]		NeuRoRA [30] + MSP Refine. [42]		MSP [42]		Ours	
#image	#edges	Names	mn	md	mn	md	mn	md	mn	md	mn	md
1881	4.2%	colosseum_exterior	7.21	4.16	25.09	2.48	22.81	3.06	2.7	1.66	**1.97**	**1.35**
228	28.2%	piazza_san_marco	2.31	1.2	8.08	4.01	3.53	2.28	**2.02**	1.55	2.24	**1.12**
163	66.6%	big_ben_2	14.56	2.96	7.56	2.36	5.58	1.38	5.77	1.43	**3.42**	**1.21**
182	41.7%	palazzo_pubblico	5.69	1.91	3.58	1.58	3.49	1.21	3.22	1.4	**2.16**	**0.91**
624	12.4%	louvre	7.69	2.69	8.55	4.82	6.48	1.47	5.04	**0.9**	**3.47**	**0.90**
188	59.5%	big_ben_1	12.57	2.59	5.22	2.60	9.01	2.60	3.42	1	**2.97**	**0.94**
104	63.2%	petra_jordan	8.68	1.76	5.15	3.19	4.19	0.75	2.85	**0.5**	**2.76**	0.81
100	53.7%	statue_of_liberty_2	10.06	4.17	5.80	2.23	4.90	1.99	2.93	1.2	**2.54**	**1.02**
269	23.1%	st_peters_basilica_interior_2	7.43	2.72	6.24	2.44	4.91	1.08	4.63	1.43	**3.33**	**0.92**
90	66.0%	statue_of_liberty_1	6.79	2.45	5.71	2.34	4.43	1.99	**3.22**	**1.35**	3.44	1.55
103	55.9%	florence_cathedral_side	8.56	3.46	2.87	1.19	2.91	0.78	**1.55**	**0.62**	1.75	1.57
136	43.6%	palace_of_versailles_chapel	13	2.76	2.98	0.96	5.01	1.13	3.12	0.64	**2.74**	**0.61**
496	14.6%	notre_dame_rosary_window	7.41	1.94	7.06	3.83	4.41	1.67	2.79	0.96	**2.08**	**0.80**
745	10.8%	lincoln_memorial_statue	8.08	1.54	2.87	1.48	3.74	1.19	1.95	**0.96**	**1.87**	1.21

Table 3. Comparison of results on the ***YFCC100*** dataset. We compare our method with various SOTA MRA methods, mean(mn) and median(md) angular errors on the estimated absolute rotations are compared. The entries with the best performance are bolded. The second is underlined. Notice that the result of MSP is computed by using additional input, such as image and correspondence.

Robust.	Eval.	Train.	mn	md	Train.	mn	md	Time
$\|\mathcal{V}\|$	300	600	0.27	0.04	300	0.24	0.04	0.005
	600		0.15	0.03	600	-	-	0.007
	1500		0.38	0.16	1500	0.24	0.09	0.011
$\|\mathcal{E}\|$	3%	30%	2.98	0.47	3%	1.75	0.32	0.006
	30%		0.15	0.03	30%	-	-	0.007
	60%		0.35	0.18	60%	0.14	0.04	0.015
σ	5°	15°	0.13	0.03	5°	0.17	0.05	0.008
	15°		0.15	0.03	15°	-	-	0.007
	55°		0.57	0.33	55°	0.35	0.17	0.008
o	3%	15%	0.06	0.04	3%	0.07	0.04	0.011
	15%		0.15	0.03	15%	-	-	0.007
	60%		0.82	0.13	60%	0.53	0.12	0.011

Table 4. The results of robustness check on the different synthetic datasets. We compare the results of our models trained on the synthetic datasets with the different settings. We report the result in terms of average mean and median angular error. We show the runtime of our graph optimizer with different kinds of view-graph in terms of second per iteration.

testing set. For the column from 3 to 5, We train RAGO on the default synthetic dataset and evaluate it under different configurations. The RAGO training and testing results under the synthetic dataset of the same settings are shown from column 6 to 8. The experiments demonstrate that RAGO generalizes well across dataset changes except when the model is trained on the sparse view-graph.

4.5. Time-Space Complexity and Model Size

RAGO only contains appropriately 0.396M parameters and can be easily deployed for real-world applications. We deploy our method on a Nvidia V100 GPU and evaluate it using view-graphs with different of nodes and edges. The average running time for each iteration during the optimization is shown on Table 4. For the view-graph with 600 nodes and 30% edges, RAGO only takes 0.007 seconds for each iteration. It uses 0.015 seconds per iteration on the view-graph with 600 nodes and 60% edges. The time and space complexity of our graph optimizer is $O(|\mathcal{E}| + |\mathcal{V}|)$. The running time and memory consumption of our method increases linearly as the size of the input view-graph increases.

5. Conclusion

We propose a learning-to-optimize graph-based optimizer (RAGO) for the Multiple Rotation Averaging (MRA) problem. RAGO solves the original MRA by building a cost graph based on the Single Rotation Averaging (SRA) objective function to update camera orientations iteratively. During optimization, the relative orientations are rectified to handle the outliers and the noises. The Gated Recurrent Unit (GRU) is employed to exploit temporal information during iterations. RAGO outperforms previous methods on synthetic and real-world datasets and is also highly efficient in running time and memory.

Limitation: RAGO belongs to learning-based methods, which suffer from cross-domain generalization problems, e.g., RAGO trained on indoor scenes might work poorly on outdoor scenes. Furthermore, RAGO is trained in a supervised manner, while precise ground truth of real data is hard to obtain. We leave unsupervised training for future work.

Acknowledgement. This research is supported in part by the Canada NSERC Discovery project 611664 and the National Natural Science Foundation of China (NSFC), under grants No. 61872067, No. 61720106004 and No. 62102356.

References

[1] Jonas Adler and Ozan Öktem. Solving ill-posed inverse problems using iterative deep neural networks. *Inverse Problems*, 33(12):124007, 2017. 2

[2] Akshay Agrawal, Brandon Amos, Shane Barratt, Stephen Boyd, Steven Diamond, and Zico Kolter. Differentiable convex optimization layers. *arXiv preprint arXiv:1910.12430*, 2019. 2

[3] Brandon Amos and J Zico Kolter. Optnet: Differentiable optimization as a layer in neural networks. In *International Conference on Machine Learning*, pages 136–145. PMLR, 2017. 2

[4] Federica Arrigoni and Andrea Fusiello. Synchronization problems in computer vision with closed-form solutions. *Int. J. Comput. Vis.*, 128, 01 2020. 1

[5] 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. 1, 2, 6

[6] Avishek Chatterjee and Venu Madhav Govindu. Efficient and robust large-scale rotation averaging. *Int. Conf. Comput. Vis.*, 2013. 1, 2, 6, 7, 8

[7] Yutian Chen, Matthew W Hoffman, Sergio Gómez Colmenarejo, Misha Denil, Timothy P Lillicrap, Matt Botvinick, and Nando Freitas. Learning to learn without gradient descent by gradient descent. In *International Conference on Machine Learning*, pages 748–756. PMLR, 2017. 2

[8] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014. 4

[9] Ronald Clark, Michael Bloesch, Jan Czarnowski, Stefan Leutenegger, and Andrew J Davison. Learning to solve nonlinear least squares for monocular stereo. In *Eur. Conf. Comput. Vis.*, pages 284–299, 2018. 2

[10] Ronald Clark, Michael Bloesch, Jan Czarnowski, Stefan Leutenegger, and Andrew J Davison. Ls-net: Learning to solve nonlinear least squares for monocular stereo. *arXiv preprint arXiv:1809.02966*, 2018. 2, 3

[11] D. Crandall, A. Owens, N. Snavely, and D. Huttenlocher. Discrete-continuous optimization for large-scale structure from motion. *IEEE Conf. Comput. Vis. Pattern Recog.*, 2011. 2

[12] Zhaopeng Cui and Ping Tan. Global structure-from-motion by similarity averaging. *Int. Conf. Comput. Vis.*, 2015. 1

[13] Frank Dellaert, David M Rosen, Jing Wu, Robert Mahony, and Luca Carlone. Shonan rotation averaging: Global optimality by surfing $so(p)^n$. In *Eur. Conf. Comput. Vis.* Springer, 2020. 2, 6

[14] Jakob Engel, Vladlen Koltun, and Daniel Cremers. Direct sparse odometry. *IEEE Trans. Pattern Anal. Mach. Intell.*, 40(3), 2017. 1

[15] Johan Fredriksson and Carl Olsson. Simultaneous multiple rotation averaging using lagrangian duality. *Asian Conf. on Comput. Vis.*, 2012. 2

[16] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *Int. Conf. on Machine Learning*, pages 1263–1272. PMLR, 2017. 2, 3, 4

[17] Zan Gojcic, Caifa Zhou, Jan D Wegner, Leonidas J Guibas, and Tolga Birdal. Learning multiview 3d point cloud registration. *IEEE Conf. Comput. Vis. Pattern Recog.*, 2020. 2

[18] Venu Madhav Govindu. Combining two-view constraints for motion estimation. *IEEE Conf. Comput. Vis. Pattern Recog.*, 2, 2001. 2

[19] Venu Madhav Govindu. Lie-algebraic averaging for globally consistent motion estimation. *IEEE Conf. Comput. Vis. Pattern Recog.*, 1, 2004. 2

[20] Richard Hartley, Khurrum Aftab, and Jochen Trumpf. L1 rotation averaging using the weiszfeld algorithm. *IEEE Conf. Comput. Vis. Pattern Recog.*, 2011. 1, 2, 3, 6, 7

[21] Richard Hartley, Jochen Trumpf, Yuchao Dai, and Hongdong Li. Rotation averaging. *Int. J. Comput. Vis.*, 103(3):267–305, 2013. 1, 3

[22] 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. 5

[23] Xiangru Huang, Zhenxiao Liang, Xiaowei Zhou, Yao Xie, Leonidas J Guibas, and Qixing Huang. Learning transformation synchronization. *IEEE Conf. Comput. Vis. Pattern Recog.*, 2019. 2

[24] Nianjuan Jiang, Zhaopeng Cui, and Ping Tan. A global linear method for camera pose registration. *Int. Conf. Comput. Vis.*, 2013. 1

[25] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 6

[26] Gabriel Moreira, Manuel Marques, and João Paulo Costeira. Rotation averaging in a split second: A primal-dual method and a closed-form for cycle graphs. In *Int. Conf. Comput. Vis.*, pages 5452–5460, 2021. 2, 6

[27] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE Trans. Robotics*, 31(5):1147–1163, 2015. 1

[28] Onur Özyesil, V. Voroninski, R. Basri, and A. Singer. A survey of structure from motion * . *Acta Numerica*, 26:305 – 364, 2017. 1

[29] Alvaro Parra, Shin-Fang Chng, Tat-Jun Chin, Anders Eriksson, and Ian Reid. Rotation coordinate descent for fast globally optimal rotation averaging. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 4298–4307, 2021. 2

[30] Pulak Purkait, Tat-Jun Chin, and Ian Reid. Neurora: Neural robust rotation averaging. *Eur. Conf. Comput. Vis.*, 2020. 1, 2, 3, 5, 6, 7, 8

[31] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *IEEE Conf. Comput. Vis. Pattern Recog.*, 2017. 2

[32] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. *IEEE Conf. Comput. Vis. Pattern Recog.*, 2016. 5

[33] Yunpeng Shi and G. Lerman. Message passing least squares framework and its application to rotation synchronization. *Int. Conf. on Machine Learning*, 2020. 2, 6, 7

[34] Chengzhou Tang and Ping Tan. Ba-net: Dense bundle adjustment network. *arXiv preprint arXiv:1806.04807*, 2018. 2, 3

[35] Chengzhou Tang, Oliver Wang, and Ping Tan. Gslam: Initialization-robust monocular visual slam via global structure-from-motion. *Int. Conf. on 3D Vision*, 2017. 1

[36] Roberto Tron and René Vidal. Distributed image-based 3-d localization of camera sensor networks. *Int. Conf. on Decision and Control*, 2009. 1

[37] Roberto Tron, René Vidal, and Andreas Terzis. Distributed pose averaging in camera networks via consensus on se (3). *Second ACM/IEEE International Conference on Distributed Smart Cameras*, 2008. 1

[38] Roberto Tron, Xiaowei Zhou, and Kostas Daniilidis. A survey on rotation optimization in structure from motion. *IEEE Conf. on Comput. Vis. and Pattern Recog. Workshops*, 2016. 1

[39] 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. 1, 6

[40] 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. 1, 2

[41] Kyle Wilson and Noah Snavely. Robust global translations with 1dsfm. *Eur. Conf. Comput. Vis.*, 2014. 5

[42] Luwei Yang, Heng Li, Jamal Ahmed Rahim, Zhaopeng Cui, and Ping Tan. End-to-end rotation averaging with multi-source propagation. *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 11774–11783, June 2021. 1, 2, 3, 5, 6, 7, 8

[43] Zi Jian Yew and Gim Hee Lee. Learning iterative robust transformation synchronization. In *Int. Conf. on 3D Vision*, pages 1206–1215. IEEE, 2021. 2

[44] 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. 6