End-to-End Rotation Averaging with Multi-Source Propagation

Luwei Yang[1] Heng Li[1] Jamal Ahmed Rahim[1] Zhaopeng Cui[2*] Ping Tan[1*]
[1] Simon Fraser University [2] State Key Lab of CAD & CG, Zhejiang University
{luweiy, lihengl, jrahim, pingtan}@sfu.ca, zhpcui@zju.edu.cn

Abstract

This paper presents an end-to-end neural network for multiple rotation averaging in SfM. Due to the manifold constraint of rotations, conventional methods usually take two separate steps involving spanning tree based initialization and iterative nonlinear optimization respectively. These methods can suffer from bad initializations due to the noisy spanning tree or outliers in input relative rotations. To handle these problems, we propose to integrate initialization and optimization together in an unified graph neural network via a novel differentiable multi-source propagation module. Specifically, our network utilizes the image context and geometric cues in feature correspondences to reduce the impact of outliers. Furthermore, unlike the methods that utilize the spanning tree to initialize orientations according to a single reference node in a top-down manner, our network initializes orientations according to multiple sources while utilizing information from all neighbors in a differentiable way. More importantly, our end-to-end formulation also enables iterative re-weighting of input relative orientations at test time to improve the accuracy of the final estimation by minimizing the impact of outliers. We demonstrate the effectiveness of our method on two real-world datasets, achieving state-of-the-art performance.

1. Introduction

Multiple Rotation Averaging (MRA) [20, 47, 31, 2] aims to estimate the absolute orientations $\mathbf{R}_1, \mathbf{R}_2, ..., \mathbf{R}_n \in SO(3)$ for a set of n cameras from the measurements of their relative orientations $\{\widetilde{\mathbf{R}}_{ij}\}$. It is a fundamental problem in 3D Vision and is important to many applications. For example, MRA is commonly used in Structure-From-Motion (SfM), especially in global SfM algorithms [8, 24] to determine camera orientations. MRA is also widely used in pose graph optimization in visual SLAM [10, 30, 43] or sensor networks [45, 46].

*Corresponding authors
The code is available at github.com/sfu-gruvi-3dv/msp_rot_avg.

The MRA problem is commonly formulated on a view-graph $G = \{V, E\}$, where each vertex represents the unknown absolute orientation and an edge connecting two vertices if their relative orientation is known. Typically, the optimal results $\{\mathbf{R}_i^*\}$ are computed by minimizing the discrepancy between the observed relative orientations $\{\widetilde{\mathbf{R}}_{ij}\}$ and the estimated relative orientations from $\{\mathbf{R}_{ij}^*\} = \{\mathbf{R}_j^* \mathbf{R}_i^{*-1}\}$, which can be written as:

$$\arg\min_{\{\mathbf{R}_i^*\}} \sum_{(i,j)\in E} \rho(d(\widetilde{\mathbf{R}}_{ij}, \mathbf{R}_{ij}^*)), \tag{1}$$

where the function d defines the distance between two rotation matrices and the ρ is a robust cost function.

There are several challenges in MRA. Firstly, the 3×3 rotation matrices form a 3D nonlinear manifold in a 9D linear space. Ignoring this manifold constraint leads to inferior results [27]. However, enforcing the manifold constraint in optimization is non-trivial, which often involves local linearization by Taylor expansion and iterative optimization [6]. Secondly, due to the non-linear optimization, many methods [6, 19, 32, 37] often rely on a Shortest Path Tree (SPT) or a Minimum Spanning Tree (MST) for initialization, which could be imprecise due to noisy input relative orientations, leading to poor initialization and inferior results. Thirdly, in many real applications, solving MRA needs to deal with incorrect relative orientations [33], *i.e.*, outliers, typically due to feature matching ambiguities. A robust MRA method, therefore, has to handle all these problems simultaneously.

A neural network approach was recently proposed in [32]. It utilizes a two-stage neural network architecture in which the first stage cleans outliers in the view-graph and builds a SPT to initialize the global orientations and the second stage further improves this initialization via a fine-tuning network. However, the whole network is not end-to-end trainable, and the separate stages leave a gap between the initialization and final prediction, which makes the second stage sensitive to the quality of initialization from the SPT. If the initialization is poor due to the failure of outliers removal, the second stage normally also fails to refine it and generate good estimation.

This work presents the first end-to-end trainable neural network for MRA. At the core of our method, a graph-convolution-based Multi-Source Propagator (MSP) computes initial absolute orientations for all nodes by applying graph convolution on a view-graph iteratively. Unlike SPT-based methods that determine the orientations of child nodes from only their parent nodes, at each iteration, our method calculates the absolute rotation of a node as a weighted sum over all neighbours. This allows neighbouring nodes with reliable edges (inliers) to be favoured in order to minimize the impact of outliers. Moreover, our graph-convolution-based method can propagate information from multiple reference nodes, which further improves our robustness to outliers to initialization.

Furthermore, we design a graph-based Appearance-Geometry Fusion (AGF) network that uses information from image context and corresponding corner points to evaluate the quality of each edge in the view-graph, which is used in the MSP for better initialization. Exploiting semantic information, the AGF can effectively learn to weight each edge to reflect the error in the measured relative orientation. In comparison, conventional methods usually use man-crafted features and/or loop consistency constraints for the same purpose and often have limited performance.

After obtaining the initial camera orientations using MSP, we refine them with the FineNet introduced in [32]. As it is impossible to distinguish outliers and inliers perfectly even with the well designed AGF, at test time, with our differentiable MSP module, we iteratively refine the edge weights, *i.e*., the confidence of input pairwise measurement, using the Adam optimizer by fixing the parameters of MSP and FineNet and minimizing the discrepancy in Equ. (1). This iterative edge re-weighting further improves results accuracy and the robustness against outliers.

Our contributions can be summarized as follows:

- An end-to-end neural network for multiple rotation averaging and optimization.
- A fully differentiable module, *i.e*., MSP, to initialize rotation averaging, which enables outlier refinement at test time.
- An Appearance-Geometry Fusion (AGF) network that fuses image context and corresponding keypoints to detect outliers in relative rotations.
- Outperforming both traditional and learning-based state-of-the-art methods on two real-world datasets.

2. Related Work

MRA has been extensively studied and two approaches have been explored, optimization based and deep learning based, with the vast majority of methods being optimization based.

Govindu first introduced MRA with his linear motion model [15], and then in [16] with lie-group based averaging. Robust optimization algorithms such as [6, 19, 48] have recently been proposed, and they could suppress the influence of outliers. Most of these algorithms are iterative methods, optimizing a robust cost function. Hartley *et al*. [19] use the Weiszfeld algorithm of L_1 averaging [5], and update the absolute orientations of each camera using the median of those computed from its neighbors in each iteration. Wang *et al*. [48] introduced Least Unsquared Deviation (LUD) as robust cost function and then solve the non-convex problem by using Semi-Definite Programming (SDP) with Convex Relaxation [39]. Chatterjee and Govindu [6] use an iterative re-weighted least squares (IRLS) minimization with a more robust loss function to fine-tune the initialization obtained by a spanning-tree. Recently, Shi and Lerman[37] use message-passing based MPLS for an alternative paradigm to IRLS and further improves the results. Arrigoni *et al*. [3] transfer the rotation averaging problem to a low-rank and sparse matrix decomposition task, the outliers and missing data are take into account when solving decomposition problem. Crandall *et al*. [7] first build a rough initialization with loopy belief propagation by ignoring the twist component of rotations to reduce solution space. They then refine the initialization by continuous Levenberg-Marquardt optimization. Fredriksson and Olsson [12] convert the original problem to a dual problem with Lagrangian Duality, and then solve it by SDP. This approach helps obtain globally minimum solutions. Multiple methods have been proposed and work on optimizing this pipeline [9, 11, 34, 48]. These methods, however, focus more on overcoming complexity of optimization, and dealing with outliers remains an open problem in such methods, as they either assume no noise or make mild assumptions on the noise distribution [9].

Rotation averaging has recently been studied with learning-based methods. [23, 14] take multiple 3D scans as input and learn to solve the rotation and translation synchronization for point cloud registration task. Purkait *et al*. [32] proposed a two-stage graphical neural networks based on the message passing neural network (MPNN) [13]. The first stage is a view-graph cleaning network to detect outlier edges. The second stage is a fine-tuning network to refine the absolute orientations. This method has the same limitation as the optimization-based methods that it is limited by the quality of the initialization, since its two stages are separated and the whole network is not end-to-end trainable. Furthermore, its first stage for outlier detection is limited to using only the noisy input relative rotations. In comparison, we further include image features to facilitate outlier detection.

Robust rotation averaging needs to identify and discard outlier relative rotations. Extensive works[17, 18, 24, 29, 36, 42, 51, 4] have been proposed for this purpose. How-

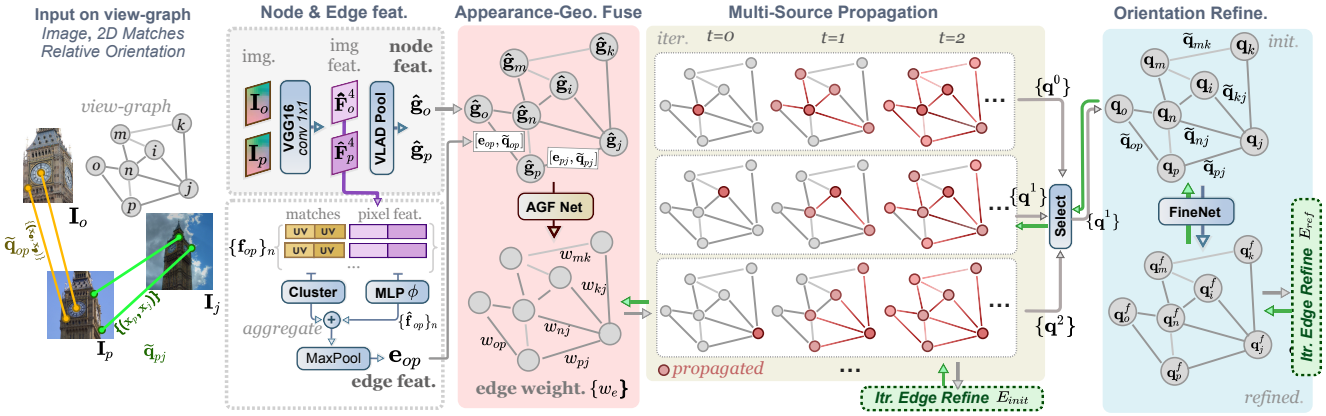


Figure 1. The pipeline of our method. For simplification, we demonstrate extracting node and edge features using example nodes o, p. Three different reference nodes are used to generate 3 initial candidates in the Multi-Source Propagator module. An initial candidate is then selected to be refined by the FineNet. Please refer to the main text for specific details.

ever, most of these methods rely on loop consistency to identify outliers and recover absolute camera poses. Yi *et al*. [28] proposed a CNN-based method utilizes semantic information and 2D correspondence distribution to score the quality of 2D correspondences on a pair of image. Our AGF module uses neural network to extract similar features from images and feature correspondences, however, with a different aim of cleaning and refining the view-graph. Moreover, thanks to the differential MSP module, we can interactively refine the outlier detection.

3. Method

Fig. 1 shows an overview of our pipeline. The input is a view-graph, where each node represents a camera and two nodes are connected by an edge if their relative rotation is known. An image is attached to each node, while an edge contains pairwise relative rotations and the corresponding 2D corner points. The output of our system is the optimal absolute orientations of each node that minimizes the discrepancy defined in Equ. (1).

As shown in Fig. 1, our end-to-end pipeline consists of four modules including: **1)** a **Feature Extraction Module** that extracts features for the graph nodes and edges from the input images and matched correspondences; **2)** a graph-based **Appearance-Geometry Fusion (AGF) Module** that takes the node and edge features, along with the relative rotation measurements as input, and predicts a weight for each edge that indicates whether the edge is reliable or not; **3)** a **Multi-Source Propagation (MSP) Module** that takes the weighted view-graph and the relative rotations as input and generates initial camera orientations by iterative propagation; **4)** an **Orientation Refinement Module that produces refined orientations**. Thanks to the differential MSP module, we propose an **Iterative Edge Weight Refinement** strategy during test that re-weights the edge confidence iteratively by minimizing the discrepancy in Equ. (1) and enhances final results. We will explain each module in detail in the following sections.

3.1. Node and Edge Feature Extraction

We seek to extract features on the nodes and edges to facilitate the estimation of absolute orientations while minimizing the impact of noisy relative rotations and outliers. Conventional methods, *e.g*. [8, 25, 33, 41], often use the number of matched corner points to measure the quality of a relative rotation. However, such a man-crafted measurement ignores high level image semantics and may not best reflect the quality of relative motions. Therefore, we use a neural network to extract features on the nodes and edges.

The node feature captures global semantic information from the image. In particular, we use VGG16 [38] to extract feature map $\mathbf{F}_i^4$ from VGG16's Layer group 4 when constructing node features.

The edge feature encodes 2D corner points locations and their appearance information. Formally, given an image pair $\{\mathbf{I}_i, \mathbf{I}_j\}$ with n pairs of matched keypoints $\{(\mathbf{x}_i^a, \mathbf{x}_j^b)\}$ where $\mathbf{x}_i^a, \mathbf{x}_j^b$ are 2D coordinates of matched keypoints (a, b) defined in frames i and j respectively. We first reduce the channels of the feature maps $\mathbf{F}^4$ with a 1×1 convolution layer, resulting in $\hat{\mathbf{F}}^4$ with 64 channels. Subsequently, we gather the features for each pair of correspondence,

$$\{\mathbf{f}_{ij}\}_n = \{[\tilde{\mathbf{x}}_i^a, \tilde{\mathbf{x}}_j^b, \mathbf{f}_i^a, \mathbf{f}_j^b]\}_n, \tag{2}$$

where $\mathbf{f}_i^a = \hat{\mathbf{F}}_i^4(\mathbf{x}_i^a)$ and $\mathbf{f}_j^b = \hat{\mathbf{F}}_j^4(\mathbf{x}_j^b)$ are the features of points a, b in the feature map $\hat{\mathbf{F}}_i^4, \hat{\mathbf{F}}_j^4$ respectively, and $\tilde{\mathbf{x}}_i^a, \tilde{\mathbf{x}}_j^b$ are normalized 2D coordinates. Then, we use two layers of MLP $\phi(\cdot)$ to generate a vector encoding these features: $\{\hat{\mathbf{f}}_{ij}\}_n = \{\phi(\mathbf{f}_{ij})\}_n$.

There is a problem we need to address to apply these features in real data. Both the node and edge features should have a fixed dimension regardless of the input image size and the number of matched corner points between image

pairs. To deal with different image resolutions, we use the VLAD pooling[1] operation on $\mathbf{F}_i^4$ and get a fixed-size feature $\mathbf{g}_i$. The number of channels are then reduced for memory efficiency using 1×1 convolution to produce node feature $\hat{\mathbf{g}}_i$ of size 32×64.

Similarly, we employ a clustering method to deal with the different numbers of correspondences in image pairs. Specifically, we apply k-means clustering to the positions of keypoint $\{\mathbf{x}_i\}_n$ on frame i, resulting in K different groups. For each group m with features $\{\hat{\mathbf{f}}_{ij}\}_n^m$, we then use `Maxpool` to aggregate this set of features to a single feature vector and get the edge feature as: $\mathbf{e}_{ij} = \{\texttt{maxpool}(\{\hat{\mathbf{f}}_{ij}\}_n^m) | m < K\}$. This results in features with fixed $K \times 64$, where an edge feature $\mathbf{e}_{ij}$ encodes the coordinates and appearance of matched corner points.

At the end of this module, we obtain a set of node features $\hat{\mathbf{g}}_i$ and edge features $\mathbf{e}_{ij}$ for the input view graph G.

3.2. Appearance-Geometry Fusion

With the node and edge features, we design a graph-based Appearance-Geometry Fusion (AGF) network to predict the reliability $w_{ij}, 0 < w_{ij} < 1$ for each edge (*i.e.* a relative rotation) of the view-graph G.

Our AGF consists of 6 layers of `Edge Convolution` from the Message Passing Neural Network (MPNN) [13] followed by a shared `MLP` for all edges. Consider an edge connecting the node i and j with edge feature $\mathbf{e}_{ij}$ and relative rotation $\widetilde{\mathbf{q}}_{ij}$. Our AGF takes the concatenated vector $[\mathbf{e}_{ij}, \widetilde{\mathbf{q}}_{ij}]$ as the edge feature for MPNN. The node feature for the MPNN is simply $\hat{\mathbf{g}}_i$.

At each layer of the MPNN, features are updated and passed to the next layer. The message through an edge (i, j) to the node i is computed from the features of nodes j and the feature of the edge (i, j). The output node feature is then computed by aggregating over all neighboring messages while the output edge feature is the message passed through the edge. Finally, we apply a one-layer MLP to the edge features from the last `Edge Convolution` layer to compute a normalized weight for each edge $\{w_{ij} | (i, j) \in \mathbf{E}, 0 < w_{ij} < 1\}$ with the `sigmoid` function. These weights measure the quality of the relative orientations, which will be used in the following modules. Please refer to our supplementary material for more details of the AGF.

3.3. Differentiable Multi-Source Propagation

Given the view-graph G and the edge weights w_{ij}, we now initialize the camera orientations according to the relative rotations. Conventional methods often use a SPT/MST to initialize camera orientations in a top-down manner. Specifically, these methods set the reference coordinate frame at the root node, and iteratively determine the orientation of remaining nodes according to their parent nodes in the SPT/MST tree. In comparison, we use a parameter-free graph convolution to initialize camera orientations, making the initialization process differentiable and hence, the full pipeline end-to-end trainable. Graph convolution is also advantageous since it considers information from all neighboring nodes instead of just the parent node in the SPT/MST.

Algorithm 1: Single-Source Propagation

Input : $\mathbf{G} = \{\mathbf{V}, \mathbf{E}\}, \{w_e, \widetilde{\mathbf{q}}_{ij} \text{ for } e_{ij} \in \mathbf{E}\}, \beta, t_{max}, \epsilon$
Output: Initial absolute rotation $\mathbf{q}_i$ for $i \in V$

Initialization:
$r \leftarrow \texttt{argmax}_{i \in \mathbf{V}}(\sum_{j \in \mathcal{N}(i)} w_{ji})$ ▷ select reference node
$l_r^0 \leftarrow 1; l_n^0 \leftarrow 0.1, \forall n \in \mathbf{V}, n \neq r$ ▷ set initial visit status
$\mathbf{q}_r^0 \leftarrow \mathbf{q}_\mathbf{I}$ ▷ set initial rotation to identity
Iterative Propagation:
$t \leftarrow 1$
while $(t < t_{max}$ *or* $|\mathbf{l}^{t-1} - \mathbf{l}^t| > \epsilon)$ **do**
  for $i \in V$ **do**
    $\mathbf{s}^t \leftarrow \{w_{ij} \cdot l_j^{t-1} | \forall j \in \mathcal{N}(i)\}$
    $\gamma^t \leftarrow \texttt{softmax}(\beta \cdot \mathbf{s}^t)$ ▷ compute kernel
    $\mathbf{q}_i^t \leftarrow \sum_{j \in \mathcal{N}(i)} \gamma_{ji}^t \cdot (\widetilde{\mathbf{q}}_{ji} \cdot \mathbf{q}_j^{t-1})$ ▷ weighted sum
    $\mathbf{q}_i^t \leftarrow \texttt{normalize}(\mathbf{q}_i^t)$ ▷ normalize to unit
    $l_i^t \leftarrow \sum_{j \in \mathcal{N}(i)} \gamma_{ji}^t \cdot l_j^{t-1}$ ▷ update node confidences
  end
  $t \leftarrow t + 1$
end

Iterative Graph Convolution: We apply the graph convolution multiple times. At each iteration t, we update the camera orientation at node i using its neighbors according to the following equation,

$$\mathbf{q}_i^{t+1} = \sum_{j \in \mathcal{N}(i)} \gamma_{ji}^t (\widetilde{\mathbf{q}}_{ji} \mathbf{q}_j^t). \tag{3}$$

Here, the superscript t is an iteration index, $\mathcal{N}(i)$ is the neighborhood of the node i. We follow NeuRoRA[32] and use quaternion to represent rotation, the term $\widetilde{\mathbf{q}}_{ji}\mathbf{q}_j^t$ computes the orientation of node i according to that of the neighboring node j and the relative rotation $\widetilde{\mathbf{q}}_{ji}$ from j to i. The result $\mathbf{q}_i^{t+1}$ is a weighted sum over all neighboring nodes according to the kernel weight γ_{ji}^t, and further normalized to a unit vector by $\mathbf{q}_i^{t+1} = \frac{\mathbf{q}_i^{t+1}}{||\mathbf{q}_i^{t+1}||}$.

We now explain the computation of the kernel $\gamma_i^t = \{\gamma_{ji}^t, j \in \mathcal{N}(i)\}$. For this purpose, we introduce a confidence score, $l_i^t, 0 < l_i^t < 1$, for each node i, representing our 'confidence' of the orientation at that node. For each node i, we compute,

$$\gamma_i^t = \texttt{softmax}(\{\beta(l_j^t \cdot w_{ji}) | \forall j \in \mathcal{N}(i)\}). \tag{4}$$

Intuitively, a kernel weight γ_{ji}^t is large only when both w_{ji} and l_j^t are large, *i.e.* the edge between i, j is an inlier and the orientation of node j is reliable. The kernel is further normalized by `softmax` with a fixed temperature factor

$\beta = 30$, which makes the distribution peak near the reliable neighbors [22]. Similarly, the confidence of node i is updated with the kernel γ_{ji}^t:

$$l_i^{t+1} = \sum_{j \in \mathcal{N}(i)} \gamma_{ji}^t l_j^t \tag{5}$$

Stopping Criteria: The iteration stops either a maximum iteration count t_{max} is reached or there are no changes in the confidence of all nodes for a threshold: $|l_i^{t+1} - l_i^t| < \epsilon$ where $\epsilon = 1.0 \times 10^{-4}$.

Graph Convolution Initialization: To begin this iterative graph convolution, we first choose a reference node r and set the world coordinate frame aligned with its camera frame, *i.e.* the node r has identity quaternion $\mathbf{q}_r^0 = \mathbf{q}_\mathbf{I}$. Similar to [32], the reference node r is chosen as the node with the highest sum of adjacent edge weights, which represents the 'mass center' of a view-graph and other nodes can be easily reached with fewer steps [20]. Accordingly, the confidence of the reference node r is set to $l_r^0 = 1.0$ while all the remaining nodes is set to 0.1. The quaternion at all other nodes are set to random values. The described graph convolution will propagate the camera orientation from the reference node to all other nodes. All steps are summarized in Algorithm 1.

Multi-Source Propagation: To make our result more robust to the selection of the reference node r, we design a multi-source propagation by using multiple reference nodes at once and performing the propagation in batch-style. Specifically, we select the top-M nodes with the highest sums of adjacent edge weights as reference points. In this way, at the end of this MSP, we get multiple initial orientations $\{\mathbf{q}_i^m | m < M\}$ for each node i in the view-graph G.

3.4. Orientation Refinement

We further use the FineNet in [32] to improve the orientations obtained by the MSP module. Since MSP generate M results, we randomly select one of them to make the network less sensitive to the reference node selection.

FineNet is based on the MPNN architecture, with 4 layers of Edge Convolution. We set the node features as the camera orienations $\mathbf{q}_i$, and edge features as the residual of the estimated and measured relative rotations:

$$f_{ij} = \mathbf{q}_j^{-1} \widetilde{\mathbf{q}}_{ij} \mathbf{q}_i.$$

Here f_{ij} is the feature for edge (i, j). The output node features are the refined camera orientations. For more details, we refer the reader to the supplementary material and [32].

3.5. Iterative Edge Weight Refinement

Given the MSP and FineNet modules, our network can compute the absolute orientation at each camera from the weighted view-graph and the input relative rotations. We can regard this two network modules as the following function,

$$\{\mathbf{q}_i^f\} = \mathcal{F}\left(\mathbf{w}; G, \{\widetilde{\mathbf{q}}_{ij}\}, \theta\right), \tag{6}$$

where $\mathbf{w}$ is a matrix formed by all the edge weights w_{ij}, θ is the network parameters of the FineNet, and the superscript f indicates the result generated by the FineNet.

However, the weights $\mathbf{w}$ generated by the AGF might be imprecise. Similar to the edge re-weighting in [6], we therefore apply an iterative weight refinement scheme to improve the results. In principal, we can start with the the AGF module computed weights $\mathbf{w}^0$ and optimize them iteratively by minimizing the discrepancy error in Equ. (1). Note this refinement of $\mathbf{w}$ is possible because both the MSP and FineNet modules are differentiable. Once the optimal weights $\mathbf{w}^*$ are obtained, we compute the final result by applying the MSP and FineNet one more time as in Equ. (6).

In practice, we minimize the following energy term to refine $\mathbf{w}$ for better performance,

$$E = E_{ini} + E_{ref}, \tag{7}$$

where the two energy terms are defined for the orientations generated by the MSP module and the FineNet module respecitvely. Specifically, they are,

$$E_{ini} = \frac{1}{|M||E|} \sum_{m \in M} \sum_{(i,j) \in E} w_{ij} \rho(d_q(\widetilde{\mathbf{q}}_{ij}, \mathbf{q}_{ij}^m)), \tag{8}$$

and

$$E_{ref} = \frac{1}{|E|} \sum_{(i,j) \in E} w_{ij} \rho(d_q(\widetilde{\mathbf{q}}_{ij}, \mathbf{q}_{ij}^f)). \tag{9}$$

The function $d_q(\cdot)$ is the distance of two quaternions and ρ denotes smooth L_1 cost. The quaternions $\mathbf{q}_{ij}^m$ and $\mathbf{q}_{ij}^f$ are the relative orientations computed from the outputs of MSP & FineNet respectively.

For this optimization of $\mathbf{w}$, we fix the parameters in FineNet (note the MSP module is parameter-free) and apply the Adam optimizer[26] with learning rate $lr = 1.0$. For more details, please refer to supplementary document.

3.6. Training Loss Function

We train our network in a supervised manner with known ground truth camera orientations to measure the error in the results by the MSP module $\{\mathbf{q}^m\}$ and the FineNet $\{\mathbf{q}^f\}$. We also define an additional loss function to facilitate the extraction of edge features.

To extract useful edge features for AGF, we provide supervision with binary cross-entropy loss. The edge features are passed through a one-layer MLP $\psi(\cdot)$ and `sigmoid` to predict the probability of the edge being an inlier (relative rotation error $\leq 20°$). Therefore, an loss $L_\mathbf{e}$ can be defined on edges:

$$L_\mathbf{e} = \frac{1}{|E|} \sum_{(i,j) \in E} \mu_{ij} L_{\mathbf{bce}}(\psi(\mathbf{e}_{ij}), \hat{y}_{ij}), \tag{10}$$

Datasets			MPLS[37]		Chatterjee[6]		NeuRoRA[32]		NeuRoRA[32] + Refine.		Ours	
#image	#edges	Names	mn	md	mn	md	mn	md	mn	md	mn	md
1881	4.2%	colosseum_exterior	failed		7.21	4.16	25.09	2.48	22.81	3.06	**2.7**	**1.66**
228	28.2%	piazza_san_marco	failed		2.31	**1.2**	8.08	4.01	3.53	2.28	**2.02**	1.55
163	66.6%	big_ben_2	14.54	3.20	14.56	2.96	7.56	2.36	**5.58**	**1.38**	5.77	1.43
182	41.7%	palazzo_pubblico	6.78	2.39	5.69	1.91	3.58	1.58	3.49	**1.21**	**3.22**	1.4
624	12.4%	louvre	failed		7.69	2.69	8.55	4.82	6.48	1.47	**5.04**	**0.9**
188	59.5%	big_ben_1	13.90	3.82	12.57	2.59	5.22	2.60	9.01	2.60	**3.42**	**1**
104	63.2%	petra_jordan	8.32	1.56	8.68	1.76	5.15	3.19	4.19	0.75	**2.85**	**0.5**
100	53.7%	statue_of_liberty_2	13.05	3.85	10.06	4.17	5.80	2.23	4.90	1.99	**2.93**	**1.2**
269	23.1%	st_peters_basilica_interior_2	failed		7.43	2.72	6.24	2.44	4.91	**1.08**	**4.63**	1.43
90	66.0%	statue_of_liberty_1	7.05	2.35	6.79	2.45	5.71	2.34	4.43	1.99	**3.22**	**1.35**
103	55.9%	florence_cathedral_side	9.30	2.31	8.56	3.46	2.87	1.19	2.91	0.78	**1.55**	**0.62**
136	43.6%	palace_of_versailles_chapel	failed		13	2.76	**2.98**	0.96	5.01	1.13	3.12	**0.64**
496	14.6%	notre_dame_rosary_window	7.14	1.3	7.41	1.94	7.06	3.83	4.41	1.67	**2.79**	**0.96**
745	10.8%	lincoln_memorial_statue	7.89	1.16	8.08	1.54	2.87	1.48	3.74	1.19	**1.95**	**0.96**

Table 1. 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.

where for each edge (i, j), $\hat{y}_{ij}$ is the ground truth label and the weight μ_{ij} is set to 0.75 when (i, j) is an outlier edge and 0.1 otherwise. This weighting helps to deal with the unbalanced distribution of inlier and outlier edges, where outlier edges are rare.

Similar to NeuRoRA [32], the loss for the initial orientations by the MSP module is calculated on all M initialization candidates $\{\{\mathbf{q}_i\}^m | i \in V, m < M\}$ as:

$$L_{\textbf{init}} = \frac{1}{|M||E|} \sum_{m \in M} \sum_{(i,j) \in E} \rho(d_q(\hat{\mathbf{q}}_{ij}, \mathbf{q}_{ij}^m)) + \omega_{abs} \frac{1}{|M||V|} \sum_{m \in M} \sum_{i \in V} \rho(d_q(\hat{\mathbf{q}}_i \hat{\mathbf{q}}_{r_m}^{-1}, \mathbf{q}_i^m)). \quad (11)$$

Here, the first term enforces consistency of relative orientations, where $\hat{\mathbf{q}}_{ij}$ is the ground-truth relative rotation computed as $\hat{\mathbf{q}}_{ij} = \hat{\mathbf{q}}_j \hat{\mathbf{q}}_i^{-1}$ and $\mathbf{q}_{ij}^m$ denotes the relative pose computed from the results of the MSP module. The second term measures the absolute rotation error for all nodes. To fix the gauge freedom, the ground truth orientations are aligned to the reference frame attached to the node r_m by multiplying with $\hat{\mathbf{q}}_{r_m}^{-1}$. The weight $\omega_{abs} = 0.5$ balances the two terms.

For the refined orientation $\{\mathbf{q}_i^f | i \in V\}$, we can define a similar loss function as $L_{\textbf{init}}$. Specifically, suppose the reference node is r, this loss is defined as:

$$L_{\textbf{opt}} = \frac{1}{|E|} \sum_{(i,j) \in E} \rho(d_q(\hat{\mathbf{q}}_{ij}, \mathbf{q}_{ij}^f)) + \omega_{abs} \frac{1}{|V|} \sum_{i \in V} \rho(d_q(\hat{\mathbf{q}}_i \hat{\mathbf{q}}_r^{-1}, \mathbf{q}_i^f)). \quad (12)$$

Finally, our training loss is

$$L = L_{\mathbf{e}} + L_{\textbf{init}} + L_{\textbf{opt}}. \quad (13)$$

More training details are in the supplementary document.

4. Experiments

Datasets: We evaluate our method and compare it with some SOTA algorithms on the *YFCC100* [21] and *1DSfM* [50] datasets. The *YFCC100* dataset consists of internet images at 72 city-scale scenes. We use the author's reconstructed camera poses as the ground-truth and run the COLMAP [35] to obtain missing view-graphs and relative poses. The *1DSfM* contains 14 outdoor scenes with 'ground-truth' camera poses obtained using Bundler [40].

Comparison: We compare our approach with various optimized-based methods including Chatterjee[6], MPLS [37], Arrigoni[3] and a recent deep learning based method NeuRoRA [32] using a publicly available evaluation script[1]. In our experiments, we use the latest implementation with the default parameters and cost function of the shared scripts[1][2][3] on [6, 32, 37]. As the pre-trained model is not released, we train the NeuRoRA [32] by feeding the full view-graphs of the *YFCC100* scenes with default parameters and cite their results on 1DSfM directly.

Implementation Details: We use a GTX 1080Ti with 11G memory to train and test our model. The MSP module is implemented using the DGL library[49]. For FineNet, we use the author's shared implementation directly.

To train our model on *YFCC100*, we randomly split all the 72 models into two sets, one for training (58 models) while the other for testing (14 models). Due to the GPU memory limitation, we first train our full pipeline on sub-graphs with 30 nodes, which are sampled from the original view-graph. We then fix the node and edge feature extractor and train subsequent modules over sampled sub-graphs

[1] http://www.ee.iisc.ac.in/labs/cvl/research/rotaveraging/
[2] https://github.com/pulak09/NeuRoRA
[3] https://github.com/yunpeng-shi/MPLS

Datasets			Chatterjee[6]		Arrigoni[3]		NeuRoRA[32]		MPLS[37]		Ours	
#image	#edge	Names	mn	md	mn	md	mn	md	mn	md	mn	md
577	59.5%	Alamo	4.2	**1.1**	6.2	1.6	4.94	1.16	3.44	1.16	**2.89**	**1.07**
227	66.8%	Ellis Island	2.8	**0.5**	3.9	1.2	2.59	0.64	2.61	0.88	**1.88**	0.83
677	17.5%	Gendarmenmarkt	37.6	7.7	41.6	13.3	**4.51**	**2.94**	44.9	8.0	6.29	3.69
341	30.7%	Madrid Metropolis	6.9	1.2	6.0	1.7	**2.55**	1.13	4.65	1.26	2.96	**1.09**
450	46.8%	Montreal Notre Dame	1.5	**0.5**	4.8	0.9	1.2	0.6	1.04	0.51	**0.91**	**0.5**
338	39.5%	Piazza del Popolo	4	0.8	10.8	1.2	3.05	0.79	3.73	1.93	**2.68**	**0.76**
1084	10.9%	Roman Forum	3.1	1.5	13.2	8.2	2.39	1.31	2.62	1.37	**2.04**	**1.19**
472	18.5%	Tower of London	3.9	2.4	4.6	1.8	2.63	1.46	3.16	2.2	**2.55**	**1.25**
5058	4.6%	Trafalgar	3.5	2	48.6	13.2	5.33	2.25	-		Out of Memory	
789	5.9%	Union Square	9.3	3.9	9.2	4.4	5.98	2.01	6.54	3.48	**4.37**	**1.85**
836	24.6%	Vienna Cathedral	8.2	1.2	19.3	2.39	**3.9**	1.5	7.21	2.83	3.91	**1.1**
437	26.5%	Yorkminster	3.5	1.6	4.5	1.6	2.52	0.99	2.47	1.45	**2.27**	**0.91**
2152	10.2%	Piccadilly	6.9	2.9	22.0	9.7	4.75	1.91	3.93	1.81	**3.63**	**1.8**
332	29.3%	NYC Library	3	1.3	3.9	1.5	1.9	1.18	2.63	1.24	**1.75**	**1.12**

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 are bolded.

with 80 nodes. In the end, we train our model on sub-graphs with 180 nodes. For this sub-graph sampling, we randomly sample 10–200 sub-graphs from each view-graph, depending on its total number of nodes. Typically, this generates 3,000–4,000 sub-graphs for training.

With *1DSfM*, limited by the number of scenes, we train and test our network in a leave-one-out manner which is also applied in NeuRoRA[32]. We employ the same sub-graph sampling approach used with *YFCC100*, and the network parameters are initialized with those trained on *YFCC100*. During training, the number of initial candidates is set to $M = 15$ in MSP.

During testing, GPU memory prevents us from loading all the nodes and edges of the view-graph at once. Therefore, we also sample sub-graphs with 80 nodes, until every edge is covered. We then feed these sub-graphs into the feature extractor and AGF module one by one, caching the output edge weights $\mathbf{w}$. At last, we build a complete weighted view-graph where the weight for an edge is the average of its weights in all the sub-graphs containing it. This complete weighted view-graph is subsequently fed into our MSP module with $M = 15$, which is then followed by the FineNet. More details about the implementation and training & testing parameters are in the supplementary material.

4.1. Results on Internet Dataset

YFCC100: We evaluate the performance in terms of mean (mn) and median (md) angular error of estimated camera rotations. The comparison results are listed in Table 1. Our approach (Ours) outperforms Chatterjee [6], MLPS [37] and NeuRoRA [32] in most scenes. NeuRoRA generates large error in the `colosseum_exterior` example, because of poor initialization generated by a SPT while MPLS[37] also suffers from bad initialization and produces large error on some examples as marked as *failed* in the table. On the other hand, our end-to-end trainable approach (Ours) avoids such issues and shows great performance, especially on `colosseum_exterior`. Lastly, our method's performance is only slightly inferior to NeuRoRA on `palace_of_versailles_chapel`.

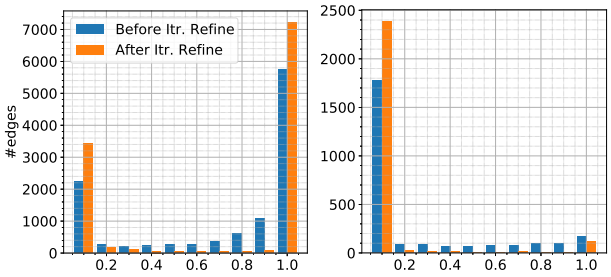
Methods	avg. mn	avg. md	**Methods**	avg. mn	avg. md
M = 1	4.65	1.19	SPTW	9.37	5.75
M = 5	4.66	1.24	Ours (NEFO)	4.18	1.34
M = 10	4.1	1.22	Ours (RMO)	3.98	1.47
Ours (M=15)	**3.23**	**1.11**			

Table 3. The averaged mean and median angular error over all test sets of *YFCC100* on different baseline methods. Note that both Ours (NEFO) and Ours (RMO) are trained with $M = 15$. Please refer to Section 4.2 for more details.

1DSfM: Comparison results on the *1DSfM* dataset are listed in the Table 2 where the numbers are cited from the reference papers. In most scenes, our method outperforms [3, 6, 32, 37] in terms of the mean angular error and median angular error. Our method is only slightly inferior to NeuRoRA[32] on `Madrid Metropolis`, and performs comparably on `Vienna Cathedra`. Our method performs inferior than NeuRoRA on `Gendarmenmarkt`, but better than the other two methods. Note that `Trafalgar` has a large number of nodes (5k) and edges (600k) and requires GPU memory that exceeds our current configuration (GTX 1080Ti, 11GB) even with only $M = 1$ in MSP.

4.2. Ablation Study

Effectiveness of Node and Edge Feature: To verify the effectiveness w.r.t different types of input features, we modified the AGF module with two different types of inputs as baseline methods: Node/Edge Features Only (NEFO) and Relative Measurements Only (RMO) respectively. For RMO, we replace each node feature by a 1×4 zero vector while the input edge only takes relative measured rota-



(a) Relative Err: $\leq 10°$ (b) Relative Err: $\geq 45°$

Figure 2. Histograms of the edge weight w w.r.t relative orientation error ($\leq 10°$ and $\geq 45°$ respectively) on `Madrid`. The distribution of w before and after iterative refinement are compared. For each figure, the horizontal axis represents the value of w while the vertical axis represents the number of edges.

tion as input. As for the NEFO baseline, we only remove the relative measurement from the input edge features. We compare those two baselines with our full-version model (Ours) and the results are listed in Table 3. The two baseline methods, Ours(NEFO) and Ours(RMO), yield comparable results while our full-version performs notably better. The results indicate that both Node&Edge features and relative rotations are necessary in our pipeline and subsequently, produce better absolute orientations.

Effectiveness of Multi-Source Propagation: We evaluate the Multi-Source Propagation module by training and testing with various $M = \{1, 5, 10, 15\}$. Furthermore, we compare with a SPT-based baseline (SPTW) by replacing the MSP module in our pipeline with a SPT, which is built with the inverse edge weight $1 - w_{ij}$ as distance [44]. We train a separate FineNet with initialization generated by the SPT accordingly. Table 3 reports the averaged mean(mn) and median(md) angular error on all 14 test sets of *YFCC100*. The model with most source nodes $M = 15$ in MSP achieves the best performance among all settings, while $M = 10$ yields slightly better results than $M = 1$ and $M = 5$. The model trained with only $M = 1$ and $M = 5$ are comparable. The SPT-based baseline method (SPTW) is dramatically inferior to our complete pipeline regardless of different M in MSP module. This verifies the effectiveness that our MSP module which is benefit from both using multiple reference nodes and the graph convolution that using multiple neighbors instead of one.

Effectiveness of Iterative Edge Refinement: The iterative refinement of edge weight $\mathbf{w}$ (Section 3.5) is important for our system performance. We seek to implement this scheme to the method NeuRoRA [32] to further investigate the origin of this performance improvement.

Specifically, the FineNet in NeuRoRA [32] can be regarded as the following function,

$$\{\mathbf{q}_i^{f\,\prime}\} = \mathcal{F}'(\mathbf{q}_{init}; G, \theta') \,. \tag{14}$$

Here, $\mathbf{q}_{init}$ is the initial camera orientations generated by its SPT-based method, G is the view-graph, and θ' is the network parameters of the FineNet.

We design a similar energy function as Equ. (9) to iteratively refine $\mathbf{q}_{init}$ as

$$E_{\textbf{NeuRoRA}} = \frac{1}{|E|} \sum_{(i,j)\in E} w_{ij}\rho(d_q(\widetilde{\mathbf{q}}_{ij}, \mathbf{q}_{ij}^{f\,\prime})) \tag{15}$$

where the weight w_{ij} is edge weight produced by CleanNet in [32], $\mathbf{q}_{ij}^{f\,\prime}$ is the relative orientation computed from the FineNet results. Note, while the CleanNet also produces a weight for each view-graph edge, these weights cannot be refined as the SPT-based initialization is not differentiable.

The results of this improved NeuRoRA are listed in Table 1 as (NeuRoRA + Refine). Although this improved version brings better results over NeuRoRA, its performance is still inferior to our pipeline. Moreover, the results of scene `palace_of_versa_chapel` and `big_ben_1` deteriorated (in terms of mean or median error).

To better demonstrate the 're-weighting', we visualize two histograms of edge wegiths on the data `Madrid` in Fig. 2, where (a) and (b) are the histogram of the inlier (relative error less than $10°$) and outlier edges (relative error larger than $45°$) respectively. These histograms show the number of edges with specific weights w before and after refinement. In histogram (a), the bin at $w = 1$ indicates true-positives. Similarly, in histogram (b), the bin at $w = 0$ indicates true negatives. After the edge weight refinement, both the number of true positive and true negative increase. Furthermore, false positive in histogram (b) (the bins at $w > 0$) are filtered out.Note that the number of false negatives in histogram (a) increases a bit (the bin at $w = 0$). Although not ideal, it does not deter from our purpose of removing outliers. As the model successfully exclude false positives, as a side benefit, it increases false negatives. This verifies the effectiveness of iterative edge refinement.

Additional execution time and outlier impact study are included in supplementary document.

5. Conclusion

This paper presents an end-to-end trainable neural network for Multiple Rotation Averaging. Our network takes image context and corresponding keypoints into consideration to weight each edge of the view-graph according to the reliability of the relative rotation defined on that edge. We then design a differentiable Multi-Source Propagator (MSP) module to initialize the camera orientations by graph convolution, which is further refined in the FineNet. Eventually, the view-graph edge weights are iteratively refined to exclude unreliable edges and improve the results. Experiments demonstrate that our method achieves state-of-the-art performance on two Internet datasets.

References

[1] Relja Arandjelović, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. Netvlad: Cnn architecture for weakly supervised place recognition. *IEEE Conf. Comput. Vis. Pattern Recog.*, 2016. 4

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

[3] 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, 6, 7

[4] Tolga Birdal, Michael Arbel, Umut Simsekli, and Leonidas J Guibas. Synchronizing probability measures on rotations via optimal transport. *IEEE Conf. Comput. Vis. Pattern Recog.*, 2020. 2

[5] Ramaswamy Chandrasekaran and Arie Tamir. Open questions concerning weiszfeld's algorithm for the fermat-weber location problem. *Mathematical Programming*, 44:293–295, 1989. 2

[6] Avishek Chatterjee and Venu Madhav Govindu. Efficient and robust large-scale rotation averaging. *Int. Conf. Comput. Vis.*, 2013. 1, 2, 5, 6, 7

[7] 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

[8] Zhaopeng Cui and Ping Tan. Global structure-from-motion by similarity averaging. *Int. Conf. Comput. Vis.*, 2015. 1, 3

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

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

[11] Anders Eriksson, Carl Olsson, Fredrik Kahl, and Tat-Jun Chin. Rotation averaging and strong duality. *IEEE Conf. Comput. Vis. Pattern Recog.*, 2018. 2

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

[13] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. 2017. 2, 4

[14] 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

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

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

[17] Venu Madhav Govindu. Robustness in motion averaging. *Asian Conf. on Comput. Vis.*, 2006. 2

[18] Leonidas J Guibas, Qixing Huang, and Zhenxiao Liang. A condition number for joint optimization of cycle-consistent networks. *Adv. Neural Inform. Process. Syst.*, 2019. 2

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

[20] Richard Hartley, Jochen Trumpf, Yuchao Dai, and Hongdong Li. Rotation averaging. *Int. J. Comput. Vis.*, 103(3):267–305, 2013. 1, 5

[21] 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. 6

[22] Sina Honari, Pavlo Molchanov, Stephen Tyree, Pascal Vincent, Christopher Pal, and Jan Kautz. Improving landmark localization with semi-supervised learning. *IEEE Conf. Comput. Vis. Pattern Recog.*, 2018. 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, 2

[25] Nianjuan Jiang, Ping Tan, and Loong-Fah Cheong. Seeing double without confusion: Structure-from-motion in highly ambiguous scenes. *IEEE Conf. Comput. Vis. Pattern Recog.*, 2012. 3

[26] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5

[27] Daniel Martinec and Tomas Pajdla. Robust rotation and translation estimation in multiview reconstruction. *IEEE Conf. Comput. Vis. Pattern Recog.*, 2007. 1

[28] Kwang Moo Yi, Eduard Trulls, Yuki Ono, Vincent Lepetit, Mathieu Salzmann, and Pascal Fua. Learning to find good correspondences. *IEEE Conf. Comput. Vis. Pattern Recog.*, 2018. 3

[29] Pierre Moulon, Pascal Monasse, and Renaud Marlet. Global fusion of relative motions for robust, accurate and scalable structure from motion. *Int. Conf. Comput. Vis.*, 2013. 2

[30] 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

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

[32] Pulak Purkait, Tat-Jun Chin, and Ian Reid. Neurora: Neural robust rotation averaging. *Eur. Conf. Comput. Vis.*, 2020. 1, 2, 4, 5, 6, 7, 8

[33] Richard Roberts, Sudipta N Sinha, Richard Szeliski, and Drew Steedly. Structure from motion for scenes with large duplicate structures. *IEEE Conf. Comput. Vis. Pattern Recog.*, 2011. 1, 3

[34] David M Rosen, Luca Carlone, Afonso S Bandeira, and John J Leonard. Se-sync: A certifiably correct algorithm for

synchronization over the special euclidean group. *The International Journal of Robotics Research*, 38(2-3):95–125, 2019. 2

[35] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. *IEEE Conf. Comput. Vis. Pattern Recog.*, 2016. 6

[36] Tianwei Shen, Siyu Zhu, Tian Fang, Runze Zhang, and Long Quan. Graph-based consistent matching for structure-from-motion. *Eur. Conf. Comput. Vis.*, 2016. 2

[37] Yunpeng Shi and G. Lerman. Message passing least squares framework and its application to rotation synchronization. *Int. Conf. on Machine Learning*, 2020. 1, 2, 6, 7

[38] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 3

[39] A. Singer. Angular synchronization by eigenvectors and semidefinite programming. *Applied and Computational Harmonic Analysis*, 30(1):20–36, 2011. 2

[40] Noah Snavely, Steven M Seitz, and Richard Szeliski. Photo tourism: exploring photo collections in 3d. *ACM Siggraph*, 2006. 6

[41] Noah Snavely, Steven M Seitz, and Richard Szeliski. Skeletal graphs for efficient structure from motion. *IEEE Conf. Comput. Vis. Pattern Recog.*, 2008. 3

[42] Chris Sweeney, Torsten Sattler, Tobias Hollerer, Matthew Turk, and Marc Pollefeys. Optimizing the viewing graph for structure-from-motion. *Int. Conf. Comput. Vis.*, 2015. 2

[43] 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

[44] Robert Endre Tarjan. Sensitivity analysis of minimum spanning trees and shortest path trees. *Information Processing Letters*, 14(1):30 – 33, 1982. 8

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

[46] 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

[47] 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

[48] 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

[49] Minjie Wang, Da Zheng, Zihao Ye, Quan Gan, Mufei Li, Xiang Song, Jinjing Zhou, Chao Ma, Lingfan Yu, Yu Gai, Tianjun Xiao, Tong He, George Karypis, Jinyang Li, and Zheng Zhang. Deep graph library: A graph-centric, highly-performant package for graph neural networks. *arXiv preprint arXiv:1909.01315*, 2019. 6

[50] Kyle Wilson and Noah Snavely. Robust global translations with 1dsfm. *Eur. Conf. Comput. Vis.*, 2014. 6

[51] Christopher Zach, Manfred Klopschitz, and Marc Pollefeys. Disambiguating visual relations using loop constraints. *IEEE Conf. Comput. Vis. Pattern Recog.*, 2010. 2