






# NeuRoRA: Neural Robust Rotation Averaging

Pulak Purkait<sup>(✉)</sup> , Tat-Jun Chin , and Ian Reid 

The University of Adelaide, Adelaide, SA 5005, Australia  
pulak.isi@gmail.com  
<https://github.com/pulak09/NeuRoRA>

**Abstract.** Multiple rotation averaging is an essential task for structure from motion, mapping, and robot navigation. The conventional methods for this task seek parameters of the absolute orientations that agree best with the observed noisy measurements according to a robust cost function. These robust cost functions are highly nonlinear and are designed based on certain assumptions about the noise and outlier distributions. In this work, we aim to build a neural network that learns the noise patterns from the data and predict/regress the model parameters from the noisy relative orientations. The proposed network is a combination of two networks: (1) a view-graph cleaning network, which detects outlier edges in the view-graph and rectifies noisy measurements; and (2) a fine-tuning network, which fine-tunes an initialization of absolute orientations bootstrapped from the cleaned graph, in a single step. The proposed combined network is very fast, moreover, being trained on a large number of synthetic graphs, it is more accurate than the conventional iterative optimization methods.

**Keywords:** Robust rotation averaging · Message passing neural networks

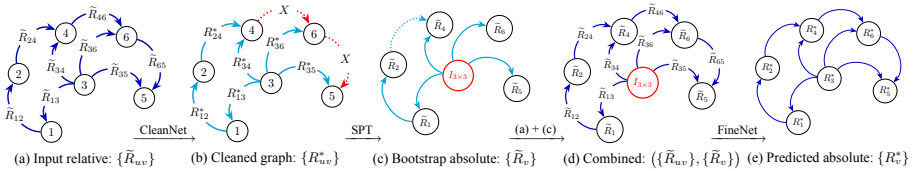
## 1 Introduction

Recently, we have witnessed a surge of interest in applying neural networks in various computer vision and robotics problems, such as, single-view depth estimation [16], absolute pose regression [28] and 3D point-cloud classification [35]. However, we still rely on robust optimizations at different steps of geometric problems, for example, robot navigation and mapping. The reason is that neural networks have not yet proven to be effective in solving constrained optimization problems. Some classic examples of the test-time geometric optimization include rotation averaging [8, 14, 15, 23, 26, 45] (a.k.a. rotation synchronization [5, 38, 44]), pose-graph optimization [30, 42], local bundle adjustment [33]

**Electronic supplementary material** The online version of this chapter ([https://doi.org/10.1007/978-3-030-58586-0\\_9](https://doi.org/10.1007/978-3-030-58586-0_9)) contains supplementary material, which is available to authorized users.

and global structure from motion [46]. These optimization methods estimate the model parameters that agree best with the observed noisy measurements by minimizing a robust (typically non-convex) cost function. Often, these loss functions are designed based on certain assumptions about the sensor noise and outlier distributions. However, the observed noise distribution in a real-world application could be far from those assumptions. A few such examples of noise patterns in real datasets are displayed in Fig. 2. Furthermore the nature and the structure of the objective loss function is the same for different problem instances of a specific task. Nonetheless, existing methods optimize the loss function for each instance. Moreover, an optimization during test-time could be slow for a target task involving a large number of parameters, and often forestalls a real-time solution to the problem.

In this work, with the advancement of machine learning, we address the following question: “can we learn the noise patterns in data, given thousands of different problem instances of a specific task, and regress the target parameters instead of optimizing them during test-time?” The answer is affirmative for some specific applications, and we propose a learning framework that exceeds baseline optimization methods for a geometric problem. We choose *multiple rotation averaging* (MRA) as a target application to validate our claim.



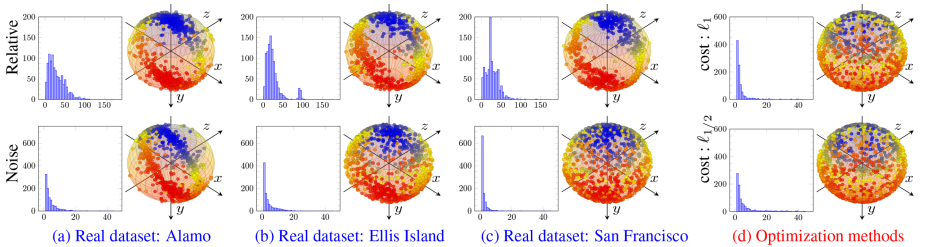
**Fig. 1.** The proposed method NeuRoRA is a two-step approach: in the first step a graph-based network (CleanNet) is utilized to clean the view-graph by removing outliers and rectifying noisy measurements. An initialization from the cleaned view-graph, instantiated from a shortest path tree (SPT), is then further fine-tuned using a separate graph-based network (FineNet). The notations are outlined in Table 1.

In MRA, the task is to estimate the absolute orientations of cameras given some of their pairwise noisy relative orientations defined on a view-graph. There are a different number of cameras for each problem instance of MRA, and usually sparsely connected to each other. Further, the observed relative orientations are often corrupted by outliers. The conventional methods for this task [5, 8, 14, 23, 44] optimize the parameters of the absolute orientations of the cameras that are most compatible (up to a robust cost) with the observed noisy relative orientations.

We propose a neural network for robust MRA. Our network is a combination of two simple four-layered message-passing neural networks defined on the view-graphs, summarized in Fig. 1. We name our method *Neural Robust Rotation Averaging*, which is abbreviated as NeuRoRA in the rest of the manuscript.

## Contribution and Findings

- A graph-based neural network NeuRoRA is proposed as an alternative to conventional optimizations for MRA.
- NeuRoRA requires *no* explicit optimization at test-time and hence it is **much faster** (10–50× on CPUs and 500–2000× on GPUs) than the baselines.
- The proposed NeuRoRA is **more accurate** than the conventional optimization methods of MRA. The mean/median orientation error of the predicted absolute orientations by the proposed method is  $1.45^\circ/0.74^\circ$ , compared to  $2.17^\circ/1.25^\circ$  by an optimization method [8].
- Being a **small size** network, NeuRoRA is fast and can easily be deployed to real-world applications (network size <0.5 Mb).



**Fig. 2.** Here we qualitatively illustrate that the noise distribution in real data diverges considerably from the noise assumptions baked into most optimization methods. We plot the angle and axes of observed **relative** orientations (first row) and the same of **noise** (second row) in real datasets (for clarity only  $10^3$  random samples) are displayed. The noise orientation is calculated from the ground-truth absolute and the observed relative orientations. The view-graphs of (a)–(b) are shared by [46] and (c) is shared by [12]. We plotted histograms of the magnitudes of the angles (in degrees) and the axes of the orientations. Notice that the axes of the sampled relative and noise orientations for the real data in (a)–(c) are not uniformly distributed on the unit ball. The sampled noise orientations (somewhat vertical axes) are far from the typical distribution assumptions regarded by optimization algorithms. Samples from such noise distributions ( $\ell_1$  [44] and  $\ell_{1/2}$  [8]) are shown in (d).

## 2 Related Works

We separate the related methods into two separate sections—(i) learning based methods as an alternative to optimizations, (ii) relevant optimizations specific to MRA, and (iii) other related optimization methods.

(i) **Learning to optimize.** A neural network is proposed as an alternative to non-linear least square optimizations in [11] for camera tracking and mapping. It exploits the least square structure of the problem and uses a recurrent network to compute updated steps of the optimization variables. In a similar

direction, [31] relaxes the assumptions made by inverse compositional algorithms for dense image alignment by incorporating data-driven priors. [40] proposes a bundle adjustment layer that learns to predict the dampening parameter of the Levenberg-Marquardt algorithm to optimize depth and camera parameters. In contrast to the direct optimization-based methods that explicitly use regularizers to solve an ill-posed problem, [1] implicitly learn the prior through a data-driven method. Aoki *et al.* [3] proposed an iterative algorithm based on PointNet [35] for point-cloud registration as an alternative to direct optimization. Learning to predict an approximate solution to combinatorial optimization problem over graphs, *e.g.* minimum vertex cover, traveling salesman problem, etc., is proposed in [29]. Learning methods to optimize general black-box functions [10] have also received a lot of attention recently. These conventional learning-based methods are tailored to some specific problems, where in this work, we are interested in an alternative learning-based solution for geometric problems, *e.g.* SFM.

**(ii) Robust optimization for rotation averaging.** MRA was first introduced in [18] where a linear solution was proposed using quaternion averaging and later in [19] using Lie group based averaging. The solutions were non-robust in both the cases. Recently, there has been progress in designing robust algorithms [9, 22] for rotation averaging. Most of the algorithms are based on iterative methods for optimizing a robust loss function. MRA is also exploited using sparse matrix decomposition, for example [5, 44]. The state of the art methods are listed below:

- Chatterjee and Govindu [8] fine-tune an initialization by first performing an iterative  $\ell_1$  minimization, followed by another iterative reweighted least squares with a more robust loss function  $\ell_{\frac{1}{2}}$ .
- Hartley *et al.* [22] propose a straight-forward method. It fine-tunes an initialization by the Weiszfeld algorithm of  $\ell_1$  averaging [7]. At every iteration, the absolute orientations of each camera are updated by the median of those computed from its neighbors.
- Arrigoni *et al.* [5] formulate the problem as a low-rank and sparse matrix decomposition and utilizes existing decomposition algorithms that caters for missing data, outliers and noise in the pairwise observations.
- Wang *et al.* [44] employ the alternating direction method to minimize a robust cost function involving the sum of unsquared deviations.

Rotation averaging is surveyed recently in a vast amount of literature [4, 6, 34, 43].

**(iii) Other related optimization methods.** DISCO [12] employs a two-step approach. In the first step, a loopy belief propagation is used for an initial estimation of the absolute orientations which are fine-tuned by Levenberg-Marquardt method in the second step. The problem of detecting outliers in the view-graph has been extensively studied in the literature [13, 20, 27, 32, 36, 47]. Optimizing/cleaning the view-graph for sfm also proposed in [21, 37, 39].

Huang *et al.* [25] proposed a neural network solving the pairwise matching problem (*c.f.* page 2, 2nd col) accurately. The key component of [25] is a network that takes two 3D scans and a relative transformation between them as input

and outputs a score indicating the goodness of the scan alignment which is iteratively employed to fine-tune the absolute pose. Therefore, [25] is only valid (and tailored) for alignments of multiple scans.

### 3 Multiple Rotation Averaging

Consider  $N$  cameras with  $M$  pairwise relative orientation measurements forming a directed view-graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ . A vertex  $\mathcal{V}_v \in \mathcal{V}$  corresponds to the absolute orientation  $\widehat{R}_v$  (to be estimated) of the  $v$ th camera and an edge  $\mathcal{E}_{uv} \in \mathcal{E}$  corresponds to the observed relative orientation  $\widetilde{R}_{uv}$  from  $u$ th camera to  $v$ th camera. Conventionally, the task is to estimate the absolute orientations  $\{\widehat{R}_v\}$ , with respect to a global reference of orientations, such that the estimated orientations are most consistent with the observed noisy relative orientation measurements, *i.e.*  $\widetilde{R}_{uv} \approx \widehat{R}_v \widehat{R}_u^{-1}, \forall \mathcal{E}_{uv} \in \mathcal{E}$ . Further, the observed measurements are corrupted by outliers, *i.e.* some of the orientations  $\widetilde{R}_{uv}$  are far from  $\widehat{R}_v \widehat{R}_u^{-1}$ . Conventionally, the solution is obtained by minimizing a robust cost function that penalizes the discrepancy between observed noisy relative orientations  $\{\widetilde{R}_{uv}\}$  and the estimated relative orientations  $\{R_{uv}^*\} := \{R_v^* R_u^{*-1}\}$ . The corresponding optimization problem can then be expressed as

$$\arg \min_{\{R_v^*\}} \sum_{\mathcal{E}_{uv} \in \mathcal{E}} \rho\left(d(R_{uv}^*, \widetilde{R}_{uv})\right) \quad (1)$$

where  $\rho(\cdot)$  is a robust cost and  $d(\cdot, \cdot)$  is a distance measure between the orientations. The nature of the above optimization is a typical complex multi-variable nonlinear optimization problem with thousands of variables (for thousands of cameras) and there seems to be no direct method (closed-form solution) minimizing the above cost even without outliers [23].

**The Choice of Distance Measure  $d(\widetilde{R}, R)$ .** There are three commonly used distance measurements in the rotation group  $\text{SO}(3)$ : (i) the geodesic or angle metric  $d_\theta = \angle(\widetilde{R}, R)$ , (ii) the chordal metric  $d_C = \|\widetilde{R} - R\|_F$  and (iii) the quaternion metric  $d_Q = \min\{\|q_{\widetilde{R}} - q_R\|, \|q_{\widetilde{R}} + q_R\|\}$  where  $q_R$  and  $q_{\widetilde{R}}$  are quaternion representations of  $R$  and  $\widetilde{R}$  respectively, and  $\|\cdot\|_F$  is the Frobenius norm. The metrics  $d_C$  and  $d_Q$  are proven to be  $2\sqrt{2} \sin(d_\theta/2)$  and  $2 \sin(d_\theta/4)$  respectively [23], thus, all the metrics are the same to the first order. In our implementation, we employ the quaternion representations (with non-negative scalars).

**The Choice of Robust Cost  $\rho(\cdot)$ .** In practical applications, *e.g.* robot navigation, the agent usually ends up with some corrupt measurements (outliers), due to symmetric and repetitive man-made structures, in addition to the sensor noise. To estimate the absolute orientations of the cameras that are immune to those outliers, the conventional methods optimize a robust cost  $\rho(\cdot)$  as discussed above. An exhaustive list of such robust functions can be found in [8].

The noise and outliers in the observed relative orientations is assumed to follow some distributions subject to the cost function with mean identity orientation [23, 44]. However, in real data, we observe very different noise distributions

and a few such examples are shown in Fig. 2. Further, optical axis of most of the cameras are horizontal and hence the axes of the relative orientations are vertical. By training a neural network to perform the task, our aim is for the neural network to capture these patterns while predicting the absolute orientations.

## 4 Learning to Predict Absolute Orientations

Let  $\mathcal{D} := \{\mathcal{G}\}$  be a dataset of ground-truth view-graphs. Each view-graph  $\mathcal{G} := (\mathcal{V}, \mathcal{E})$  contains a noisy relative orientation measurement  $\tilde{R}_{uv}$  for each edge  $\mathcal{E}_{uv} \in \mathcal{E}$  and a ground-truth absolute orientation  $\hat{R}_v$  for each camera  $\mathcal{V}_v \in \mathcal{V}$ . The desired neural network learns a mapping  $\Phi$  that takes noisy relative measurements  $\{\tilde{R}_{uv}\}$  as input and predicts the absolute orientations  $\{R_v^\Phi\} := \Phi(\{\tilde{R}_{uv}\}; \Theta)$  as output, where  $\Theta$  is the set of network parameters. To train the parameters of such network, one could minimize the discrepancy between the ground-truth  $\hat{R}_{uv} := \hat{R}_v \hat{R}_u^{-1}$  and the estimated  $R_{uv}^\Phi := R_v^\Phi R_u^{\Phi^{-1}}$  relative orientations (cf. Eq. (1)), *i.e.*

$$\arg \min_{\Theta} \sum_{\mathcal{G} \in \mathcal{D}} \sum_{\mathcal{E}_{uv} \in \mathcal{E}} d(R_{uv}^\Phi, \hat{R}_{uv}) \quad (2)$$

In contrast to (1), where conventional methods optimize the orientation parameters for each instance of the view-graph  $\mathcal{G} \in \mathcal{D}$ , here in (2), the network parameters are optimized during training that learn the mapping  $\Phi$  effectively from observed relative orientations  $\{\tilde{R}_{uv}\}$  to the target absolute orientations  $\{\hat{R}_v\}$ , *i.e.*  $\{\hat{R}_v\} \approx \Phi(\{\tilde{R}_{uv}\}; \Theta)$  over the entire dataset of view-graphs  $\mathcal{D}$ .

**Direct Training of  $\Phi$  and Gauge Freedom.** For an arbitrary orientation  $R$ ,

$$R_{uv}^* := R_v^* R_u^{*-1} = (R_v^* R) (R_u^* R)^{-1}, \quad \forall \mathcal{E}_{uv} \in \mathcal{E} \quad (3)$$

Therefore,  $\{R_v^*\}$  and  $\{R_v^* R\}$  essentially represent the same solution to the MRA problem (1) and there is a gauge freedom of degree 3. The mapping  $\Phi$  is thus one-to-many as  $\{R_v^\Phi\}$  and  $\{R_v^\Phi R\}$  correspond to the same cost (2). This gauge freedom makes it difficult to train such a network. Further, one could choose a direct cost (no associated gauge freedom) to learn an one-to-one mapping  $\Phi$ , *e.g.*

$$\arg \min_{\Theta} \sum_{\mathcal{G} \in \mathcal{D}} \sum_{\mathcal{V}_v \in \mathcal{V}} d(R_v^\Phi, \hat{R}_v) \quad (4)$$

where the reference orientation is fixed according to the ground-truth. Again,  $\{\hat{R}_v\}$  and  $\{\hat{R}_v R\}$  represent the same ground-truth where the reference orientations are fixed at different directions. One could fix the issue by fixing the reference orientation to the orientation of the first camera in all the view-graphs in  $\mathcal{D}$ . However, in a graph (set representation), the nodes are permutation invariant. Thus the choice of the first camera, and hence the reference orientation, is arbitrary. Therefore, one needs to pass the reference orientation or the index of

the first camera (possibly via a binary encoding) to the network as an additional input to be able to train such a network. However, we employ an alternative strategy adopted from the conventional optimization methods [8, 23], *i.e.* initialize a solution of the absolute orientations under a fixed reference and pass the initialization to the network to fine-tune the solution. The network gets the reference orientation as an additional input via initialization (see Fig. 1(d)) and regress the parameters, *i.e.*  $\{\widehat{R}_v\} \approx \Phi(\{\widetilde{R}_{uv}\}, \{\widetilde{R}_v\}; \Theta)$ . Further, we train the network by minimizing a combined cost where the 1st term (2) enforces the consistency over the entire graph and the 2nd term (4) enforces a unique solution, *i.e.*

$$\arg \min_{\Theta} \sum_{\mathcal{G} \in \mathcal{D}} \left( \sum_{\mathcal{E}_{uv} \in \mathcal{E}} d(R_{uv}^{\Phi}, \widehat{R}_{uv}) + \beta \sum_{\mathcal{V}_v \in \mathcal{V}} d(R_v^{\Phi}, \widehat{R}_v) \right) \quad (5)$$

where  $\beta$  is a weight parameter. Note that the reference orientation are now fixed at the orientation of a certain camera  $c$  in the initialization  $\{\widetilde{R}_v\}$  as well as in the ground-truth absolute orientations  $\{\widehat{R}_v\}$ . Although, the choice of  $c$  is not critical, in practice, the camera  $c$  with most neighboring cameras is chosen as the reference, *i.e.*  $\widetilde{R}_c = \widehat{R}_c = I_{3 \times 3}$ .

The above mapping  $\Phi$  is now one-to-one. However, it requires an initialization  $\{\widetilde{R}_v\}$  as an additional input. Conventional methods initialize the absolute orientations using a spanning tree of the view graph. However even a single outlier in that spanning tree can lead to a very poor initialization, so it is very important to identify these outliers beforehand. Further, noise in the relative orientation along each edge of the spanning tree will also propagate at the subsequent nodes while computing the initial absolute orientations. Thus, we first clean the view-graph by removing the outliers and rectifying the noisy measurements, and then bootstrap an initialization from the cleaned view-graph.

**Cleaning the View-Graph.** Given the local structure in the view-graph, *i.e.* measurements of all the edges that the pair of adjacent nodes  $\{\mathcal{V}_u, \mathcal{V}_v\}$  are connected to (and possibly subsequent edges), an outlier edge  $\mathcal{E}_{uv}$  can be detected. To be specific, chaining the relative orientations along a cycle in the local structure of the view-graph forms an orientation close to the identity orientation and an indication of an outlier in the cycle otherwise. The presence of an outliers in multiple such cycles through the current edge indicates that the edge to be an outlier. Instead of designing such explicit algorithms, we use another neural network to clean the graph. The proposed method can be summarized as follows:

- A graph-based network is employed to clean the view-graph by removing outlier measurements and rectifying noisy measurements (see Sect. 4.2).
- The cleaned view-graph is then utilized to initialize the absolute orientations (see Sect. 4.3).
- The initialization is fine-tuned using a separate network (see Sect. 4.4).

For clarity of the rest of the paper, the notations are outlined in Table 1.

## 4.1 The Network Design Choice

Generalizing convolution operators to irregular domains, such as graphs, is typically expressed as neighborhood aggregation or a message-passing scheme. The proposed network is built using such Message-Passing Neural Networks (MPNN) [17], directly operating on view-graphs  $\mathcal{G}$ . A MPNN is defined in terms of message functions  $m_v^{(t)}$  and update functions  $\gamma^{(t)}$  that run for  $T$  time-steps (layers). At each time-step, the hidden state  $h_v^{(t)}$  at each node (feature) in the graph is updated according to

$$h_v^{(t)} = \gamma^{(t)}(h_v^{(t-1)}, m_v^{(t)}) \quad (6)$$

where  $m_v^{(t)}$  is the condensed message at node  $v$ , coming from the neighboring nodes  $u \in \mathcal{N}_v$ , and can be expressed as follows:

$$m_v^{(t)} = \square_{\mathcal{V}_u \in \mathcal{N}_v} \phi^{(t)}(h_v^{(t-1)}, h_u^{(t-1)}, e_{uv}) \quad (7)$$

where  $\square$  denotes a differentiable, permutation invariant symmetric function, *e.g.* *mean*, *soft-max*, etc.;  $\gamma^{(t)}$  and  $\phi^{(t)}$  are concatenation operations followed by 1-D convolutions and ReLUs;  $e_{uv}$  is the edge feature of the edge  $\mathcal{E}_{uv}$ ,  $h_{u \rightarrow v}^{(t)} := \phi^{(t)}(h_v^{(t-1)}, h_u^{(t-1)}, e_{uv})$  is the accumulated message for the edge  $\mathcal{E}_{uv}$  at time-step ( $t$ ); and  $\mathcal{N}_v$  is the set of all neighboring cameras connected to  $\mathcal{V}_v$ . A diagram of the elements involved in computing the next-level features is shown in Fig. 3.

**Table 1.** The notations and symbols used in the manuscript

Orientation parameters in the view-graph	
$\tilde{R}_{uv} R_{uv}^*$ : Observed Noise-rectified relative	$\tilde{R}_v R_v^\Phi R_v^*$ : Initial Refined predicted absolute
$\tilde{R}_{uv} \tilde{R}_v$ : Ground-truth relative absolute	$\{R_{uv}\} \{R_v\}$ : Set of all relative absolute
The network parameters and symbols	
$\alpha_{uv}^* \hat{\alpha}_{uv}$ : Predicted Ground-truth outlier-score	$h_v^{(t)} m_v^{(t)}$ : Features Message at node $v$
$\phi^{(t)} \gamma^{(t)}$ : Feature update Accumulated message	$lp_1, lp_2, lp_3$ : Single layers of linear perceptrons

## 4.2 View-Graph Cleaning Network

The view-graph cleaning network (CleanNet) is built on a MPNN. The input to CleanNet is a noisy view-graph and the output is a clean one, *i.e.* the network takes noisy relative orientations  $\tilde{R}_{uv}$  as the edge features  $e_{uv}$  and predicts the noise-rectified relative orientations  $R_{uv}^*$  from the accumulated message  $h_{u \rightarrow v}^{(T)} := \phi^{(T)}(h_v^{(T-1)}, h_u^{(T-1)}, e_{uv})$  at the last layer. It also predicts a score  $\alpha_{uv}^*$  depicting the probability of the edge  $\mathcal{E}_{uv}$  to be an outlier. *i.e.*

$$R_{uv}^* = lp_1(h_{u \rightarrow v}^{(T)}) \star \tilde{R}_{uv} \quad \text{and} \quad \alpha_{uv}^* = lp_2(h_{u \rightarrow v}^{(T)}) \quad (8)$$

where  $lp_1(\cdot)$  and  $lp_2(\cdot)$  are single-layered linear perceptrons that map the accumulated messages to the edge noise orientation and outlier score respectively.



‘ $\star$ ’ is the matrix multiplication. The hidden states are initialized by null vectors, *i.e.*  $h_v^{(0)} = \emptyset$ . Note that instead of directly estimating the rectified orientations, we predict the noise in the relative orientation measurements, which are then multiplied to obtain the rectified orientations. The loss is chosen as the weighted combination of mean orientation error  $\mathcal{L}_{mre}$  of the rectified  $R_{uv}^*$  and ground-truth  $\widehat{R}_{uv} := \widehat{R}_v \widehat{R}_u^{-1}$  relative orientations, and mean binary cross-entropy error  $\mathcal{L}_{bce}$  of the predicted  $\alpha_{uv}^*$  and the ground-truth outlier score  $\widehat{\alpha}_{uv}$ , *i.e.*

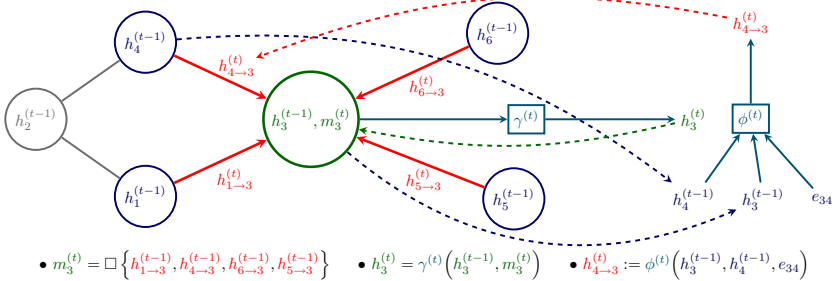
$$\mathcal{L} = \sum_{\mathcal{G} \in \mathcal{D}} \sum_{\mathcal{E}_{uv} \in \mathcal{E}} \left( \mathcal{L}_{mre}(R_{uv}^*, \widehat{R}_{uv}) + \lambda \mathcal{L}_{bce}(\alpha_{uv}^*, \widehat{\alpha}_{uv}) \right) \quad (9)$$

where  $\lambda$  is a weight parameter (fixed as  $\lambda = 10$ ). We formulate the orientations using unit quaternions and the predictions are normalized accordingly. The error in the prediction is also normalized by the degree of the node, *i.e.*

$$\mathcal{L}_{mre}(R_{uv}^*, \widehat{R}_{uv}) = \frac{1}{|\mathcal{N}_v| |\mathcal{N}_u|} d_Q \left( \frac{R_{uv}^*}{\|R_{uv}^*\|_2}, \widehat{R}_{uv} \right) \quad (10)$$

Experimentally, we observed the above loss produces superior performance than the standard discrepancy loss (2). Note that the ground-truth outlier score  $\widehat{\alpha}_{uv}$  is generated based on the amount of noise in the relative orientations. Specifically, if the amount of noise in the relative orientation  $\widetilde{R}_v^{-1} \widehat{R}_{uv} \widetilde{R}_u > 20^\circ$ , the ground-truth edge label is assigned as an outlier, *i.e.*  $\widehat{\alpha}_{uv} = 1$  and  $\widehat{\alpha}_{uv} = 0$  otherwise.

An edge  $\mathcal{E}_{uv}$  is marked as an outlier edge if the predicted outlier score  $\alpha_{uv}^*$  is greater than a predefined threshold  $\epsilon$ . In all of our experiments, we choose the threshold  $\epsilon = 0.75$ <sup>1</sup>. A cleaned view-graph  $\mathcal{G}^*$  is then generated by removing outlier edges from  $\mathcal{G}$  and replacing noisy relative orientations  $\widetilde{R}_{uv}$  by the rectified orientations  $R_{uv}^*$ . Note that the cleaned graph  $\mathcal{G}^*$  is only employed to bootstrap an initialization of the absolute orientations.



**Fig. 3.** An illustration of computing next level features of a message-passing network.

<sup>1</sup> The choice is not critical in the range  $\epsilon \in [0.35, 0.8]$ .

### 4.3 Bootstrapping Absolute Orientations

Hartley *et al.* [22] proposed generating a spanning tree by setting the camera with the maximum number of neighbors as the root and recursively adding adjacent cameras without forming a cycle. The reference orientation is fixed at the camera at the root of the spanning tree. The orientations of the rest of the cameras in the tree are computed by propagating away the rectified orientations  $R_{uv}^*$  from the root node along the edges, *i.e.*  $\tilde{R}_v = R_{uv}^* \tilde{R}_u$ .

As discussed before, the noise in the relative orientation along each edge  $R_{uv}^* \tilde{R}_{uv}^{-1}$  propagates at the subsequent nodes while computing the initial absolute orientations of the cameras. Therefore, the spanning tree that minimizes the sum of depths of all the nodes (a.k.a. shortest path tree [41]) is the best spanning tree for the initialization. Starting with a root node, a shortest path tree could be computed by greedily connecting nodes to each neighboring node in the breadth-first order. The best shortest path tree can be found by applying the same procedure with each one of the nodes as a root node (time complexity  $\mathcal{O}(n^2)$ ) [24]. However, we employed the procedure just once (time complexity  $\mathcal{O}(n)$ ) with the root at the node with the maximum number of adjacent nodes (similar to Hartley *et al.* [22]) and observed similar results as with the best spanning tree. The reference orientation of the initialization and the ground-truth is fixed at the root of the tree. This procedure is very fast and it takes only a fraction of a second for a large view-graph with thousands of cameras. We abbreviate this procedure as SPT and it is the default initializer in all of our experiments.

### 4.4 Fine-Tuning Network

The fine-tuning network (FineNet) is again built on a MPNN. It takes the initial absolute orientations  $\{\tilde{R}_v\}$  and the relative orientation measurements  $\{\tilde{R}_{uv}\}$  as inputs, and predicts the refined absolute orientations  $\{R_v^*\}$  as the output. The refined orientations are estimated from the hidden states  $h_v^{(T)}$  of the nodes at the last layer of the network, *i.e.*

$$R_v^* = lp_3(h_v^{(T)}) \star \tilde{R}_v \quad (11)$$

where  $lp_3$  is a single layer of linear perceptron. We initialize the hidden states of the MPNN by the initial orientations, *i.e.*  $h_v^{(0)} = \tilde{R}_v$ . The edge attributes are chosen as the relative discrepancy of the initial and the observed relative orientations, *i.e.*  $e_{uv} = \tilde{R}_v^{-1} \tilde{R}_{uv} \tilde{R}_u$ . The loss for the fine-tuning network is computed as the weighted sum of edge consistency loss and the rotational distance between the predicted orientation  $\tilde{R}_v$  and the ground-truth orientation  $\tilde{R}_v$ , *i.e.*

$$\mathcal{L} = \sum_{\mathcal{G} \in \mathcal{D}} \left( \sum_{\mathcal{E}_{uv} \in \mathcal{E}} \mathcal{L}_{mre}(R_{uv}^*, \hat{R}_{uv}) + \frac{\beta}{|\mathcal{N}_v|} \sum_{\mathcal{V}_v \in \mathcal{V}} d_Q\left(\frac{R_v^*}{\|R_v^*\|}, \hat{R}_v\right) \right) \quad (12)$$

where  $\mathcal{L}_{mre}$  is chosen as the quaternion distance (10). This is a combination of two loss functions chosen according to (5). We value consistency of the entire graph (enforced via relative orientations in the first term) over individual accuracy (second term), and so choose  $\beta = 0.1$ .

## 4.5 Training

The view-graph cleaning network and the fine-tuning network are trained separately. For each edge  $\mathcal{E}_{uv}$  in the view-graph with observed orientation  $\tilde{R}_{uv}$ , an additional edge  $\mathcal{E}_{vu}$  is included in the view-graph in the opposite direction with orientation  $\tilde{R}_{vu} := \tilde{R}_{uv}^{-1}$ . This will ensure the messages flow in both directions of an edge. In both of the above networks, the parameters are chosen as: the number of time-steps  $T = 4$ , the permutation invariant function  $\square$  as the *mean*, and the length of the message  $m_v^{(t)}$  and hidden state  $h_v^{(t)}$  are 32.

**Network Parameters  $\Theta$ :** The parameters are involved with: (i) 1-D convolutions of  $\gamma^{(t)}$  in (6) and  $\phi^{(t)}$  in (7), and (ii) linear perceptrons  $lp_1$ ,  $lp_2$  in (8) and  $lp_3$  in (11). With the above hyper-parameters (*i.e.* time-steps, length of the messages, etc.), the total number of parameters of NeuRoRA becomes  $\approx 49.8\text{K}$ . Note that increasing the hyper-parameters could lead to a bigger network (more parameters) and that results a slower performance. A small size network is much faster but is not capable of predicting accurate outputs for larger view-graphs. We have tried different network sizes and found the current network size is a good balance between speed and accuracy.

**Architecture Setup:** The networks are implemented in PyTorch Toolbox<sup>2</sup> and trained on a GTX 1080 Ti GPU with a learning rate of  $0.5 \times 10^{-4}$  and weight decay  $10^{-4}$ . Each of CleanNet and FineNet are trained for 250 epochs (takes  $\sim 4\text{--}6$  h) to learn the network parameters. To prevent the networks from over-fitting on the training dataset, we randomly drop 25% of the edges of each view-graph along with observed noisy relative orientations in each epoch. During testing all the edges were kept active. The parameters that yielded the minimum validation loss were kept for evaluation. All the baselines including the proposed networks were evaluated on an Intel Core i7 CPU.

## 5 Results

Experiments were carried out on synthetic as well as real datasets to demonstrate the true potential of our approach.

**Baseline Methods.** We evaluated NeuRoRA against the following baseline methods (also described in Sect. 2):

- Chatterjee and Govindu [8]: the latest implementation with the default parameters and cost function in the shared scripts<sup>3</sup> were employed. We also employed their evaluation strategy to compare the predicted orientations.
- Weiszfeld algorithm [22]: the algorithm is straightforward but computationally expensive and we only ran it for 50 iterations of  $\ell_1$  averaging.
- Arrigoni *et al.* [5]: the authors shared the code with the optimal parameters<sup>4</sup>.

<sup>2</sup> <https://pytorch-geometric.readthedocs.io>.

<sup>3</sup> <http://www.ee.iisc.ac.in/labs/cvl/research/rotaveraging/>.

<sup>4</sup> <http://www.diegm.uniud.it/fusiello/demo/gmf/>.

– Wang and Singer [44]: this is employed with a publicly available scripts<sup>5</sup>.

We also ran the graph cleaning network (CleanNet) followed by bootstrapping initial orientation (using SPT) as a baseline CleanNet-SPT, and ran SPT on the noisy graph followed by fine-tuning network (FineNet) as another baseline SPT-FineNet. Note that the proposed network NeuRoRA takes CleanNet-SPT as an initialization and then fine-tunes the initialization in a single step by FineNet. NeuRoRA-*v2* is a variation of the proposed method where an initialization from CleanNet-SPT is fine-tuned in two steps of FineNet, *i.e.* the output of FineNet in the first step is fed as an initialization of FineNet in the second step.

**Synthetic Dataset.** We carefully designed a synthetic dataset that closely resembles the real-world datasets. Since the amount of noise in observed relative measurements changes with the sensor type (*e.g.* camera device), the structure of the connections in the view-graphs and the outlier ratios are varied with the scene (Fig. 2). A single view-graph was generated as follows: (1) the number of cameras were sampled in the range 250–1000 and their orientations were generated randomly on a horizontal plane (yaw only), (2) pairwise edges and corresponding relative orientations were randomly introduced between the cameras that amounted to (10–30)% of all possible pairs, (3) the relative orientations were then corrupted by a noise with a std  $\sigma$  where  $\sigma$  is chosen uniformly in the range ( $5^\circ$ – $30^\circ$ ) once for the entire view-graph, and the directions are chosen randomly on the vertical plane (to emulate realistic distributions 2), and (4) the relative orientations were further corrupted by (0–30)% of outliers with random orientations. Our synthetic dataset consisted of 1200 sparse view-graphs. The dataset was divided into training (80%), validation (10%), and testing (10%).

The results are furnished in Table 2. The average angular error on all the view-graphs in the dataset is displayed. The proposed method NeuRoRA performs remarkably well compared to the baselines in terms of accuracy and speed. NeuRoRA-*v2* further improves the results. Overall, Chatterjee [8] performs well but the performance does not improve with a better initialization. Unlike Wang [44], Weiszfeld [22] improves the performance with a better initialization given by CleanNet-SPT, but, it can not improve the solution further given an even better initialization by NeuRoRA. Notice that the proposed NeuRoRA is three orders of magnitude faster with a GPU than the baseline methods.

**Real Dataset.** We summarize the real datasets and display in Table 3. There are a total of 19 publicly available view-graphs with observed noisy relative orientations and the ground-truth absolute orientations. The ground-truth orientations were obtained by applying incremental bundle adjustment [2] on the view-graphs. The TNotreDame dataset is shared by Chatterjee *et al.* [8]<sup>6</sup>. The Artsquad and SanFrancisco datasets are provided by DISCO [12]<sup>7</sup>. The rest of the view-graphs are publicly shared by 1DSFM [46]<sup>8</sup>. The ground-truth orienta-

<sup>5</sup> [https://github.com/huangqx/map\\_synchronization](https://github.com/huangqx/map_synchronization).

<sup>6</sup> <http://www.ee.iisc.ac.in/labs/cvl/research/rotaveraging/>.

<sup>7</sup> <http://vision.soic.indiana.edu/projects/disco/>.

<sup>8</sup> <http://www.cs.cornell.edu/projects/1dsfm/>.

**Table 2.** Results of Rotation averaging on a test synthetic dataset. The average angular error on all the view-graphs in our dataset is displayed. The proposed method NeuRoRA is remarkably faster than the baselines while producing better results. There is no GPU implementations of [5, 8, 22, 44] available, thus the runtime comparisons on cuda are excluded. Note that NeuRoRA takes only **0.0016 s** on average on a GPU.

Baseline methods	mn	md	cpu			mn	md	cpu	
Chatterjee [8]	2.17°	1.25°	5.38 s	( 1×)	Arrigoni [5]	2.92°	1.42°	8.20 s	(0.65×)
Weiszfeld [22]	3.35°	1.02°	50.92 s	(0.11×)	Wang [44]	2.77°	1.40°	9.75 s	(0.55×)
Proposed methods									
CleanNet-SPT + [8]	2.11°	1.26°	5.41 s	(0.99×)	NeuRoRA	<b>1.45°</b>	<b>0.74°</b>	<b>0.21 s</b>	( <b>24×</b> )
CleanNet-SPT + [22]	1.74°	1.01°	50.36 s	(0.11×)	NeuRoRA-v2	<b>1.30°</b>	<b>0.68°</b>	<b>0.30 s</b>	( <b>18×</b> )
Other methods									
CleanNet-SPT	2.93°	1.47°	<b>0.11 s</b>	( <b>47×</b> )	SPT-FineNet	3.00°	1.57°	<b>0.11 s</b>	( <b>47×</b> )
CleanNet-SPT + [44]	2.77°	1.40°	9.86 s	(0.53×)	SPT-FineNet + [44]	2.78°	1.40°	9.86 s	(0.53×)
SPT-FineNet + [8]	2.12°	1.26°	5.41 s	(0.99×)	SPT-FineNet + [22]	1.78°	1.01°	50.36 s	(0.11×)
NeuRoRA + [8]	2.11°	1.26°	5.51 s	(0.97×)	NeuRoRA + [22]	1.73°	1.01°	50.46 s	(0.10×)

mn: mean of the angular error; md: median of the angular error; cpu: the average runtime of the method; MethodA + MethodB: MethodB is initialized by the solution of MethodA.

tions are available for some of those cameras (indicated in parenthesis) and the training, validation and testing are performed only on those cameras.

Due to limited availability of real datasets for training, we employed network parameters pre-trained on the above synthetic dataset and further fine-tuned on the real datasets in round-robin fashion (leave one out). Such evaluation protocol is employed because we did not want to divide the sequences into training and testing sequences that might favor one particular method. The finetuning is done for each round of the round-robin using the real-data apart from the held-out test sequence. Overall, the proposed NeuRoRA outperformed the baselines for this task in terms of accuracy and efficiency (Table 3). The Artsquad and SanFrancisco datasets have different orientation patterns as shared from a different source [12]. In particular SanFrancisco dataset is captured along a road which is significantly different from others. Thus, the performance of NeuRoRA falls short to Chatterjee [8] and Wang [44] only on those two sequences, but, is still better than Weiszfeld [22] and Arrigoni [5]. Nonetheless, the proposed NeuRoRA is much faster than others.

**Robustness Check.** In this experiment we study the generalization capability of NeuRoRA. To check the individual effects of different sensor settings, we generate a number of synthetic dataset varying (i) #cameras (ii) #edges, (iii) amount of noise and outliers, and (iv) planar/random camera motion. NeuRoRA is then trained on one of such datasets and evaluated on the others. Each dataset consists of 1000 view-graphs (large ones contain only 100). Results are furnished in Table 4. Notice that NeuRoRa generalizes well across dataset changes except when the network is trained on planar cameras and tested on random. We therefore advice to use two separate networks for planar and non-planar scenes. Notice that Chatterjee [8] demands a large memory for large

**Table 3.** Results of MRA on real datasets. The proposed method NeuRoRA is much faster than the baselines while producing overall similar or better results. The number of cameras, for which ground-truths are available, is shown within parenthesis.

Datasets	Chatterjee [8]			Weiszfeld [22]			Arrigoni [5]			Wang [44]			NeuRoRA				
	#cameras	#edges	mn	md	cpu	mn	md	cpu	mn	md	cpu	mn	md	cpu	mn	md	cpu
Alamo	627(577)	49.5%	<b>4.2</b>	<b>1.1</b>	20.5s	4.9	1.4	84.0s	6.2	1.6	2.7s	5.3	1.4	20.6s	4.9	1.2	<b>2.2s</b>
EllisIsland	247(227)	66.8%	2.8	<b>0.5</b>	2.5s	4.4	1.0	8.9s	3.9	1.2	<b>0.2s</b>	3.6	1.1	2.6s	<b>2.6</b>	0.6	0.4s
GendrmMarkt	742(677)	17.5%	<b>37.6</b>	<b>7.7</b>	11.1s	<b>29.4</b>	9.6	53.7s	<b>41.6</b>	13.3	8.9s	<b>32.6</b>	6.1	12.5s	<b>4.5</b>	<b>2.9</b>	<b>0.5s</b>
MadridMetrop	394(341)	30.7%	6.9	1.2	3.2s	7.5	2.7	14.5s	6.0	1.7	0.9s	5.0	1.4	3.6s	<b>2.5</b>	<b>1.1</b>	<b>0.2s</b>
MontrealNotre	474(450)	46.8%	1.5	<b>0.5</b>	9.1s	2.1	0.7	41.5s	4.8	0.9	2.9s	2.0	0.8	10.1s	<b>1.2</b>	0.6	<b>1.0s</b>
NYCLibrary	376(332)	29.3%	3.0	1.3	4.8s	3.8	2.1	14.4s	3.9	1.5	1.4s	2.9	1.4	3.2s	<b>1.9</b>	<b>1.1</b>	<b>0.2s</b>
NotreDame	553(553)	68.1%	3.5	<b>0.6</b>	23.3s	4.7	0.8	80.8s	3.9	1.0	4.2s	3.5	0.9	19.5s	<b>1.6</b>	<b>0.6</b>	<b>2.0s</b>
PiazzaDelP	354(338)	39.5%	4.0	0.8	3.3s	4.8	1.3	16.7s	10.8	1.2	0.6s	6.2	1.1	3.6s	<b>3.0</b>	<b>0.7</b>	<b>0.4s</b>
Piccadilly	2508(2152)	10.2%	6.9	2.9	<b>449.0s</b>	<b>26.4</b>	7.5	<b>~20m</b>	<b>22.0</b>	9.7	43.7s	10.1	3.9	118.1s	<b>4.7</b>	<b>1.9</b>	<b>5.9s</b>
RomanForum	1134(1084)	10.9%	3.1	1.5	20.2s	4.8	1.8	115.0s	13.2	8.2	16.8s	4.6	3.5	19.6s	<b>2.3</b>	<b>1.3</b>	<b>1.3s</b>
TowerLondon	508(472)	18.5%	3.9	2.4	1.9s	4.7	2.9	17.1s	4.6	1.8	3.9s	2.9	1.5	3.6s	<b>2.6</b>	<b>1.4</b>	<b>0.3s</b>
Trafalgar	5433(5058)	4.6%	<b>3.5</b>	<b>2.0</b>	<b>858.4s</b>	15.6	<b>11.3</b>	<b>~92m</b>	<b>48.6</b>	13.2	<b>167.4s</b>	17.2	16.0	<b>319.2s</b>	5.3	2.2	<b>15.5s</b>
UnionSquare	930(789)	5.9%	9.3	3.9	6.8s	<b>40.9</b>	10.3	42.8s	9.2	4.4	12.1s	6.8	3.2	4.1s	<b>5.9</b>	<b>2.0</b>	<b>0.6s</b>
ViennaCath	918(836)	24.6%	8.2	1.2	48.1s	11.7	1.9	<b>158.3s</b>	19.3	2.39	6.0s	10.1	1.8	25.7s	<b>3.9</b>	<b>1.5</b>	<b>2.1s</b>
Yorkminster	458(437)	26.5%	3.5	1.6	4.0s	5.7	2.0	32.0s	4.5	1.6	2.5s	3.5	1.3	4.9s	<b>2.5</b>	<b>0.9</b>	<b>0.4s</b>
Acropolis	463(463)	10.7%	1.1	0.7	1.5s	<b>0.6</b>	<b>0.3</b>	15.0s	2.7	1.7	3.2s	2.4	1.6	1.7s	0.8	0.5	<b>0.2s</b>
ArtsQuad	5530(4978)	1.4%	<b>4.8</b>	3.5	116.1s	<b>34.4</b>	<b>23.1</b>	<b>~32m</b>	<b>35.2</b>	15.8	<b>189.4s</b>	6.0	<b>3.2</b>	73.9s	<b>27.5</b>	7.3	<b>5.0s</b>
SanFran	7866(7866)	0.3%	<b>3.6</b>	<b>3.4</b>	15.2s	18.8	16.4	<b>~22m</b>	<b>66.8</b>	<b>43.9</b>	<b>354.7s</b>	<b>89.2</b>	<b>75.5</b>	27.2s	17.6	12.6	<b>2.6s</b>
TNotreDame	715(715)	25.3%	<b>1.0</b>	<b>0.4</b>	10.6s	1.4	0.6	72.5s	2.4	0.9	5.7s	1.7	0.8	14.8s	1.7	0.7	<b>1.4s</b>

mn: mean of the angular error (in deg); md: median of the angular error (in deg); cpu: the runtime of the method on a cpu (s; in sec, m: in minute); entries with  $>20^\circ$  or  $>120s$  are marked in red.

**Table 4.** Robustness check: results of Rotation averaging on multiple datasets where NeuRoRA is trained on one and evaluated on the other datasets.

Robustness	Training datasets			Evaluation datasets			NeuRoRA			Chatterjee [8]				
	$ \mathcal{V} $	$ \mathcal{E} $	E&O	P	$ \mathcal{V} $	$ \mathcal{E} $	E&O	P	mn	md	cpu	mn	md	cpu
#cameras ( $ \mathcal{V} $ )	1000	25.0%	30°&10%	✓	<b>250</b>	25.0%	30°&10%	✓	<b>1.1°</b>	<b>0.9°</b>	<b>0.1 s</b>	1.8°	1.7°	0.3 s
	<b>250</b>	25.0%	30°&10%	✓	<b>5000</b>	<b>2.5%</b>	30°&10%	✓	<b>1.1°</b>	<b>1.0°</b>	<b>4.9 s</b>	1.4°	1.3°	~12 m
	<b>250</b>	25.0%	30°&10%	✓	<b>10000</b>	<b>2.5%</b>	30°&10%	✓	<b>0.7°</b>	<b>0.6°</b>	<b>18.7 s</b>	Out of memory		
	<b>250</b>	25.0%	30°&10%	✓	<b>25000</b>	<b>2.5%</b>	30°&10%	✓	<b>0.6°</b>	<b>0.5°</b>	<b>142.6 s</b>	Out of memory		
#edges ( $ \mathcal{E} $ )	1000	25.0%	30°&10%	✓	1000	<b>2.5%</b>	30°&10%	✓	<b>2.4°</b>	<b>2.1°</b>	<b>0.1 s</b>	3.0°	2.8°	3.3 s
	1000	<b>2.5%</b>	30°&10%	✓	1000	25.0%	30°&10%	✓	<b>0.5°</b>	<b>0.4°</b>	<b>2.5 s</b>	0.9°	0.8°	43.2 s
Noise & outliers (E&O)	1000	25.0%	30°&10%	✓	1000	25.0%	<b>10°&amp;5%</b>	✓	0.4°	<b>0.3°</b>	<b>2.5 s</b>	<b>0.3°</b>	<b>0.3°</b>	31.6 s
	1000	25.0%	<b>10°&amp;5%</b>	✓	1000	25.0%	30°&10%	✓	<b>0.6°</b>	<b>0.5°</b>	<b>2.5 s</b>	0.9°	0.8°	43.2 s
Planar (P)	1000	25.0%	30°&10%	✓	1000	25.0%	30°&10%	✓	2.2°	1.6°	<b>2.5 s</b>	<b>0.9°</b>	<b>0.8°</b>	26.3 s
	1000	25.0%	30°&10%	<b>X</b>	1000	25.0%	30°&10%	✓	<b>0.9°</b>	<b>0.7°</b>	<b>2.5 s</b>	<b>0.9°</b>	<b>0.8°</b>	26.3 s

E: noise with a std chosen uniformly in the range (0°-E°); O: percentage of outliers; P: flag for planar cameras.

view-graphs and failed to execute on a system of 64 Gb of RAM. We tested our method on view-graphs upto 25K vertices and 24M edges on the same system.

## 6 Discussion

We have proposed a graph-based neural network for absolute orientation regression of a number of cameras from their observed relative orientations. The proposed network is exceptionally faster than the strong optimization-based baselines while producing better results on most datasets. The outstanding performance of the current work and the relevant neural networks for test-time optimization leads to the following question: “*can we then replace all the optimizations in robotics/computer vision by a suitable neural network-based regression?*” The answer is obviously *No*. For instance, if an optimization at test-time requires solving a simpler convex cost with a few parameters to optimize, a naive gradient descent will find the globally optimal parameters, while a network-based regression would only estimate sub-optimal parameters. To date, neural nets have been proven to be consistently better at solving pattern recognition problems than solving a constraint optimization problems. A few neural network-based solutions are proposed recently that can exploit the patterns in the data while solving a test-time optimization. Therefore the current work also opens up many questions related to the right tool for a specific application.

**Acknowledgement.** We gratefully acknowledge the support of the Australian Research Council through the Centre of Excellence for Robotic Vision, CE140100016 and the Australian Research Council Discovery Project DP200101675.

## References

1. Adler, J., Öktem, O.: Solving ill-posed inverse problems using iterative deep neural networks. *Inverse Prob.* **33**(12), 124007 (2017)
2. Agarwal, S., Snavely, N., Seitz, S.M., Szeliski, R.: Bundle adjustment in the large. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) *ECCV 2010*. LNCS, vol. 6312, pp. 29–42. Springer, Heidelberg (2010). [https://doi.org/10.1007/978-3-642-15552-9\\_3](https://doi.org/10.1007/978-3-642-15552-9_3)
3. Aoki, Y., Goforth, H., Srivatsan, R.A., Lucey, S.: PointNetLK: robust & efficient point cloud registration using PointNet. In: *Proceedings of CVPR*, pp. 7163–7172 (2019)
4. Arrigoni, F., Fusiello, A.: Synchronization problems in computer vision with closed-form solutions. *IJCV* **128**(1), 26–52 (2020)
5. Arrigoni, F., Rossi, B., Fragneto, P., Fusiello, A.: Robust synchronization in  $SO(3)$  and  $SE(3)$  via low-rank and sparse matrix decomposition. *CVIU* **174**, 95–113 (2018)
6. Carlone, L., Tron, R., Daniilidis, K., Dellaert, F.: Initialization techniques for 3D SLAM: a survey on rotation estimation and its use in pose graph optimization. In: *Proceedings of ICRA*, pp. 4597–4604 (2015)
7. Chandrasekaran, R., Tamir, A.: Open questions concerning Weiszfeld’s algorithm for the Fermat-Weber location problem. *Math. Program.* **44**(1–3), 293–295 (1989)



8. Chatterjee, A., Govindu, V.M.: Robust relative rotation averaging. *TPAMI* **40**(4), 958–972 (2017)
9. Chatterjee, A., Madhav Govindu, V.: Efficient and robust large-scale rotation averaging. In: *Proceedings of ICCV*, pp. 521–528 (2013)
10. Chen, Y., et al.: Learning to learn without gradient descent by gradient descent. In: *Proceedings of ICML*, pp. 748–756. *JMLR.org* (2017)
11. Clark, R., Bloesch, M., Czarnowski, J., Leutenegger, S., Davison, A.J.: Learning to solve nonlinear least squares for monocular stereo. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) *ECCV 2018*. LNCS, vol. 11212, pp. 291–306. Springer, Cham (2018). [https://doi.org/10.1007/978-3-030-01237-3\\_18](https://doi.org/10.1007/978-3-030-01237-3_18)
12. Crandall, D., Owens, A., Snavely, N., Huttenlocher, D.: Discrete-continuous optimization for large-scale structure from motion. In: *Proceedings of CVPR*, pp. 3001–3008 (2011)
13. Enqvist, O., Kahl, F., Olsson, C.: Non-sequential structure from motion. In: *Proceedings of ICCV Workshops*, pp. 264–271 (2011)
14. Eriksson, A., Olsson, C., Kahl, F., Chin, T.J.: Rotation averaging and strong duality. In: *Proceedings of CVPR*, pp. 127–135 (2018)
15. Fredriksson, J., Olsson, C.: Simultaneous multiple rotation averaging using Lagrangian duality. In: Lee, K.M., Matsushita, Y., Rehg, J.M., Hu, Z. (eds.) *ACCV 2012*. LNCS, vol. 7726, pp. 245–258. Springer, Heidelberg (2013). [https://doi.org/10.1007/978-3-642-37431-9\\_19](https://doi.org/10.1007/978-3-642-37431-9_19)
16. Garg, R., Vijay Kumar, B.G., Carneiro, G., Reid, I.: Unsupervised CNN for single view depth estimation: geometry to the rescue. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) *ECCV 2016*. LNCS, vol. 9912, pp. 740–756. Springer, Cham (2016). [https://doi.org/10.1007/978-3-319-46484-8\\_45](https://doi.org/10.1007/978-3-319-46484-8_45)
17. Gilmer, J., Schoenholz, S.S., Riley, P.F., Vinyals, O., Dahl, G.E.: Neural message passing for quantum chemistry. In: *Proceedings of ICML*, pp. 1263–1272. *JMLR.org* (2017)
18. Govindu, V.M.: Combining two-view constraints for motion estimation. In: *Proceedings of CVPR*, vol. 2, p. II (2001)
19. Govindu, V.M.: Lie-algebraic averaging for globally consistent motion estimation. In: *Proceedings of CVPR*, vol. 1, p. I (2004)
20. Govindu, V.M.: Robustness in motion averaging. In: Narayanan, P.J., Nayar, S.K., Shum, H.-Y. (eds.) *ACCV 2006*. LNCS, vol. 3852, pp. 457–466. Springer, Heidelberg (2006). [https://doi.org/10.1007/11612704\\_46](https://doi.org/10.1007/11612704_46)
21. Guibas, L.J., Huang, Q., Liang, Z.: A condition number for joint optimization of cycle-consistent networks. In: *Proceedings of NeurIPS*, pp. 1005–1015 (2019)
22. Hartley, R., Aftab, K., Trunpf, J.: L1 rotation averaging using the Weiszfeld algorithm. In: *Proceedings of CVPR*, pp. 3041–3048 (2011)
23. Hartley, R., Trunpf, J., Dai, Y., Li, H.: Rotation averaging. *IJCV* **103**(3), 267–305 (2013)
24. Hassin, R., Tamir, A.: On the minimum diameter spanning tree problem. *Inf. Process. Lett.* **53**(2), 109–111 (1995)
25. Huang, X., Liang, Z., Zhou, X., Xie, Y., Guibas, L.J., Huang, Q.: Learning transformation synchronization. In: *Proceedings of CVPR*, pp. 8082–8091 (2019)
26. Huynh, D.Q.: Metrics for 3D rotations: comparison and analysis. *J. Math. Imaging Vis.* **35**(2), 155–164 (2009)
27. Jiang, N., Cui, Z., Tan, P.: A global linear method for camera pose registration. In: *Proceedings of ICCV*, pp. 481–488 (2013)
28. Kendall, A., Grimes, M., Cipolla, R.: PoseNet: a convolutional network for real-time 6-DOF camera relocalization. In: *Proceedings of ICCV*, pp. 2938–2946 (2015)

29. Khalil, E., Dai, H., Zhang, Y., Dilkina, B., Song, L.: Learning combinatorial optimization algorithms over graphs. In: Proceedings of NIPS, pp. 6348–6358 (2017)
30. Kümmerle, R., Grisetti, G., Strasdat, H., Konolige, K., Burgard, W.: g2o: a general framework for graph optimization. In: Proceedings of ICRA, pp. 3607–3613 (2011)
31. Lv, Z., Dellaert, F., Rehg, J.M., Geiger, A.: Taking a deeper look at the inverse compositional algorithm. In: Proceedings of CVPR, pp. 4581–4590 (2019)
32. Moulon, P., Monasse, P., Marlet, R.: Global fusion of relative motions for robust, accurate and scalable structure from motion. In: Proceedings of ICCV, pp. 3248–3255 (2013)
33. Mouragnon, E., Lhuillier, M., Dhome, M., Dekeyser, F., Sayd, P.: Generic and real-time structure from motion using local bundle adjustment. *Image Vis. Comput.* **27**(8), 1178–1193 (2009)
34. Özyeşil, O., Voroninski, V., Basri, R., Singer, A.: A survey of structure from motion\*. *Acta Numerica* **26**, 305–364 (2017)
35. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: PointNet: deep learning on point sets for 3D classification and segmentation. In: Proceedings of CVPR, pp. 652–660 (2017)
36. Shah, R., Chari, V., Narayanan, P.J.: View-graph selection framework for SfM. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV 2018. LNCS, vol. 11209, pp. 553–568. Springer, Cham (2018). [https://doi.org/10.1007/978-3-030-01228-1\\_33](https://doi.org/10.1007/978-3-030-01228-1_33)
37. Shen, T., Zhu, S., Fang, T., Zhang, R., Quan, L.: Graph-based consistent matching for structure-from-motion. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9907, pp. 139–155. Springer, Cham (2016). [https://doi.org/10.1007/978-3-319-46487-9\\_9](https://doi.org/10.1007/978-3-319-46487-9_9)
38. Singer, A.: Angular synchronization by eigenvectors and semidefinite programming. *Appl. Comput. Harmonic Anal.* **30**(1), 20–36 (2011)
39. Sweeney, C., Sattler, T., Hollerer, T., Turk, M., Pollefeys, M.: Optimizing the viewing graph for structure-from-motion. In: Proceedings of ICCV, pp. 801–809 (2015)
40. Tang, C., Tan, P.: BA-Net: dense bundle adjustment network. In: ICLR 2019 (2019)
41. Tarjan, R.E.: Sensitivity analysis of minimum spanning trees and shortest path trees. *Inf. Process. Lett.* **14**(1), 30–33 (1982)
42. Tron, R., Vidal, R.: Distributed image-based 3-D localization of camera sensor networks. In: Proceedings of CDC, pp. 901–908 (2009)
43. Tron, R., Zhou, X., Daniilidis, K.: A survey on rotation optimization in structure from motion. In: Proceedings of CVPR Workshops, pp. 77–85 (2016)
44. Wang, L., Singer, A.: Exact and stable recovery of rotations for robust synchronization. *Inf. Infer. J. IMA* **2**(2), 145–193 (2013)
45. Wilson, K., Bindel, D., Snavely, N.: When is rotations averaging hard? In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9911, pp. 255–270. Springer, Cham (2016). [https://doi.org/10.1007/978-3-319-46478-7\\_16](https://doi.org/10.1007/978-3-319-46478-7_16)
46. Wilson, K., Snavely, N.: Robust global translations with 1DSfM. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8691, pp. 61–75. Springer, Cham (2014). [https://doi.org/10.1007/978-3-319-10578-9\\_5](https://doi.org/10.1007/978-3-319-10578-9_5)
47. Zach, C., Klopschitz, M., Pollefeys, M.: Disambiguating visual relations using loop constraints. In: Proceedings of CVPR, pp. 1426–1433 (2010)