# Learning Structure-from-Motion with Graph Attention Networks

Lucas Brynte José Pedro Iglesias Carl Olsson Fredrik Kahl
Chalmers University of Technology
{brynte,jose.iglesias,caols,fredrik.kahl}@chalmers.se

## Abstract

*In this paper we tackle the problem of learning Structure-from-Motion (SfM) through the use of graph attention networks. SfM is a classic computer vision problem that is solved though iterative minimization of reprojection errors, referred to as Bundle Adjustment (BA), starting from a good initialization. In order to obtain a good enough initialization to BA, conventional methods rely on a sequence of sub-problems (such as pairwise pose estimation, pose averaging or triangulation) which provide an initial solution that can then be refined using BA. In this work we replace these sub-problems by learning a model that takes as input the 2D keypoints detected across multiple views, and outputs the corresponding camera poses and 3D keypoint coordinates. Our model takes advantage of graph neural networks to learn SfM-specific primitives, and we show that it can be used for fast inference of the reconstruction for new and unseen sequences. The experimental results show that the proposed model outperforms competing learning-based methods, and challenges COLMAP while having lower runtime. Our code is available at:* https://github.com/lucasbrynte/gasfm/.

## 1. Introduction

Structure-from-Motion (SfM) is a classic and still relevant problem in computer vision. The goal of SfM is to estimate camera poses and 3D coordinates of keypoints detected across multiple images, and can be formulated as an optimization over $m$ camera matrices $\{\mathbf{P}_i\}$, $i = 1, \dots, m$ and $n$ 3D points $\{\mathbf{X}_j\}$, $j = 1, \dots, n$ of the form

$$\begin{aligned} \underset{\{\mathbf{P}_i\},\{\mathbf{X}_j\}}{\text{minimize}} \quad & \sum_{ij} r\left(\mathbf{m}_{ij}, \mathbf{z}_{ij}\right) \\ \text{subject to} \quad & \mathbf{z}_{ij} = \mathbf{P}_i \bar{\mathbf{X}}_j, \ \forall i, j \end{aligned} \tag{1}$$

where $\mathbf{m}_{ij}$ holds the 2D coordinates of the $j^{\text{th}}$ keypoint in the $i^{\text{th}}$ image. The loss in (1) is generally chosen as the reprojection error

$$r\left(\mathbf{m}_{ij}, \mathbf{z}_{ij}\right) = ||\mathbf{m}_{ij} - \Pi\left(\mathbf{z}_{ij}\right)||^2 \tag{2}$$

where $\Pi(\mathbf{x}) = \left(\frac{x_1}{x_3}, \frac{x_2}{x_3}\right)$, and the nonlinear least squares problem (1), referred to as Bundle Adjustment (BA) [15], can be solved iteratively using second-order methods like Levenberg-Marquardt [15, 41]. Given the sparsity of the problem, sparse computation methods [27] can be used in order to increase the efficiency of the optimization, allowing BA to be used even for scenes with a large number of views or points. However, it is widely known that BA is highly non-convex and tends to converge to the nearest local minimum when not initialized close to the globally optimal solution. As a consequence, BA is typically the last step of a reconstruction pipeline, preceeding global SfM methods such as [9, 10, 20, 25], or incremental SfM methods such as [2, 37, 38] that solve a sequence of subproblems like pairwise pose estimation, pose averaging, triangulation or camera resection [13, 15, 22, 32]. A different approach consists of projective factorization methods [8, 17, 18, 28, 39, 48] which factorize the $2m \times n$ measurement matrix into two rank four matrices corresponding to the camera matrices and 3D points (in homogeneous coordinates). In particular, works like [17, 18, 48] allow initialization-free SfM given their wide basin of convergence, meaning that their methods can be initialized with random camera poses and still converge with a high rate of success to the desired global minimum. Even though these methods have been improving in terms of accuracy and robustness to missing data, factorization-based methods require the input data to be almost completely free of outliers which unfortunately cannot be guaranteed in most real world sequences or datasets, and hence severely compromises the usability of these methods.

A common challenge with all these approaches to solve SfM is their scalability as the number of views and keypoints increase. Incremental SfM tries to tackle this issue by starting with a subset of the views, estimate its reconstruction and incrementally adding more views. Some factorization-based methods can also take advantage of the same sparse computation methods used in BA, which significantly improves their ability to scale with sequence size. While this allows to reconstruct scenes with thousands of views and millions of points, it can still take hours to recover the reconstruction of a single scene.

More recently, deep learning methods for 3D reconstruction and SfM have been proposed [7, 29, 31, 35, 44–47]. Given their increased complexity, these models are able to learn complex relations between the input data, outperforming conventional methods and achieving state-of-the-art results in 3D reconstruction and novel view synthesis when the camera pose and calibration are known [7, 29, 46]. Works such as [31, 46, 47] tackle SfM with deep learning models which different degrees of success.

In DeepSfM [46] the authors propose an end-to-end approach to estimate camera pose and a dense depth map of a target view from multiple source views. Their method estimates a feature map for each of the source views, which are then combined with the feature map of the target view in order to estimate pose and depth cost volumes. These cost volumes are then fed to two heads that estimate the camera pose and depth map of the target view. In PoseDiffusion [44], a probabilistic diffusion framework is proposed to mirror the iterations of bundle adjustment, allowing it to incorporate geometric priors into the problem. The model is trained on object-centric scenes and it is shown that it can generalize to unseen scenes without further training.

In [31], which is the closest work to ours, the authors have pursued an approach similar to projective factorization, taking as input the 2D keypoints tracked along multiple views. These 2D point tracks are processed by a number of linear permutation-equivariant layers, i.e. layers such that a permutation on the rows and/or columns of the input tensor results in the same permutation on the output. The output of these layers is then fed to camera pose and 3D points coordinate regression heads. The model is learned using reprojection errors as loss combined with a hinge loss term to push points in front of the cameras. By using Adam [26] as optimizer along with gradient normalization, the authors show that their method has improved convergence properties compared to random initializations.

A major benefit of learning-based methods is their potential in generalizing to new data after being trained. This can be particularly beneficial to SfM problems since occlusions and missing measurements are common issues that make some scenes particularly challenging to reconstruct. In these cases, the learned model can act as a prior and help constrain the solution based on the data seen during training. However, most of the learning-based methods mentioned for 3D reconstruction or SfM require scene-specific optimization or training, which turns out to be particularly slow for large sequences. PoseDiffusion [44] achieves some degree of generalization to unseen scenes, but it is conditional on the categories and object-centric datasets seen during training. In [31] the authors actually show that their model can be trained on a large collection of scenes and then used for inference on unobserved scenes. However, without fine-tuning on these new scenes (which is time-consuming) the model's accuracy is significantly reduced.

In this paper we propose a graph attention network for initialization-free Structure-from-Motion that can be trained on multiple scenes and used for fast inference on new, unseen scenes. Similarly to [31], our network takes as input sparse measured image points, matched across multiple viewpoints, which are processed through a sequence of permutation-equivariant graph cross attention layers. The output of the graph cross attention layers is then processed by two regression heads, one for the 3D points and another for the camera poses. By using graph attention layers instead of linear equivariant maps as in [31], our model is able to better learn the implicit geometric primitives of SfM in the input data, leading to more expressiveness and better generalization to unseen sequences. Additionally, by not requiring time-consuming fine-tuning on unseen scenes, our model is able to provide fast inference, which can then be refined directly with BA with good convergence.

The contributions of the paper can be summarized as:

- We propose graph attention networks for initialization-free Structure-from-Motion that estimates camera parameters and 3D coordinates of 2D keypoints tracked across multiple views;
- The method takes advantage of the graph cross-attention layers, which are equivariant to permutation on the input data regarding order of views and points. We show that these layers can model more complex relations between input data and when combined with data augmentation result in significantly improved performance when compared to baseline methods;
- We evaluate the proposed learned method on unobserved scenes and show that it outperforms competing state-of-the art learning methods at inference in terms of accuracy without the need of scene-specific model fine-tuning;
- We show that our method achieves competitive results when compared to state-of-the-art conventional Structure-from-Motion pipelines while having a significantly lower runtime.

## 2. Graph Attention Network Preliminaries

Before introducing our proposed method, we will here briefly outline the graph attention networks that will be utilized – the GATv2 model [4] based on [43].

The GATv2 model is a type of graph neural network (GNN) that carries out aggregation from neighboring nodes with a dynamic attention-based weighted average. The input to a GATv2 layer is a set of current node features $\{\mathbf{h}_i \in \mathbb{R}^d \mid i \in \mathcal{V}\}$ along with a set of directed edges $\mathcal{E}$ defining the graph connectivity. The layer outputs a new set of node features $\{\mathbf{h}'_i \in \mathbb{R}^{d'} \mid i \in \mathcal{V}\}$, by applying a single shared learned function on every node $h_i$ together with its neighborhood $\mathcal{N}_i = \{j \in \mathcal{V} \mid (j, i) \in \mathcal{E}\}$.

First, a scoring function $e : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$ is applied on

all edges, by feeding the respective source and target node features as input to a shared 2-layer Multi-Layer Perceptron (MLP):

$$e\left(\mathbf{h}_i, \mathbf{h}_j\right) = \mathbf{a}^\top \cdot \mathrm{LeakyReLU}\left(\mathbf{W}\left[\mathbf{h}_i \| \mathbf{h}_j\right]\right), \quad (3)$$

where $\mathbf{a} \in \mathbb{R}^{d'}$ and $\mathbf{W} \in \mathbb{R}^{d' \times 2d}$ are learned parameters, and $\|$ denotes vector concatenation. The attention scores are then normalized across all neighbors:

$$\alpha_{ij} = \mathrm{softmax}_j\left(e\left(\mathbf{h}_i, \mathbf{h}_j\right)\right) = \frac{\exp\left(e\left(\mathbf{h}_i, \mathbf{h}_j\right)\right)}{\sum_{j' \in \mathcal{N}_i} \exp\left(e\left(\mathbf{h}_i, \mathbf{h}_{j'}\right)\right)}, \quad (4)$$

and the layer is completed by carrying out a weighted average of linear projections of all source node features:

$$\mathbf{h}'_i = \sum\nolimits_{j \in \mathcal{N}_i} \alpha_{ij} \cdot \mathbf{W}\mathbf{h}_j. \quad (5)$$

## 3. Method

We apply a graph attention network model to perform Structure-from-Motion on a desired scene based on its measured image point correspondences. The model then outputs a reconstruction, represented by all $m$ camera matrices and all $n$ scene points. Structure-from-Motion is in general a large and complex optimization problem, and it is not trivially addressed as a learning problem. Our architecture combines the power and expressivity of the attention mechanism, and exploits its inherent permutation equivariance in a meaningful way, while balancing expressivity with efficiency, resulting in a very limited computational and memory footprint considering the task at hand.

The input data for a particular scene consists of a sparse set of measured image point correspondences (point tracks). To represent this data, we define a binary observability matrix $\mathbf{O} \in \{0,1\}^{m \times n}$, where $m$ is the number of views and $n$ is the number of scene points, and where $\mathbf{O}_{ij} = 1$ if and only if scene point $j$ is observed in image $i$. Furthermore, let $\mathcal{P} = \{\mathbf{m}_{ij} \in \mathbb{R}^2 \mid \mathbf{O}_{ij} = 1\}$ be the collection of observed image point tracks, where $\mathbf{m}_{ij}$ is the (normalized) measured image point for the projection of scene point $j$ in camera $i$.

### 3.1. Graph Attention Network Architecture

In this section, we will outline our graph attention network architecture, starting by introducing our feature representation. Then we proceed to define update operations for different feature types, and finally define the entire network architecture based on these operations.

**Feature representation.** Throughout the network architecture, we maintain and continuously update a set of features of various types[1]:

[1]For simplicity, we use a slight abuse of set notation here. If there are feature vector duplicates, they are still considered as separate elements.

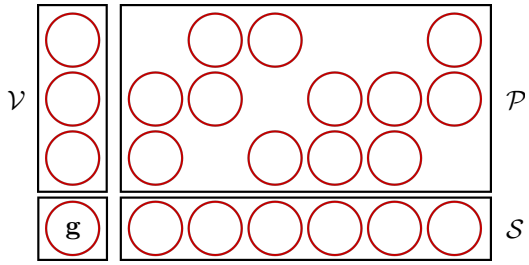


Figure 1. Illustration of the different types of features in the network architecture, for $m = 3$ cameras and $n = 6$ scene points. The projection features $\mathcal{P}$ are represented by a sparse vector-valued matrix, with its sparsity pattern determined by the point track measurements, while view features $\mathcal{V}$ and scene point features $\mathcal{S}$ are represented by dense vector-values column and row vectors. A single vector $\mathbf{g}$ holds global features.

- *Projection features*, $\mathcal{P} = \{\mathbf{p}_{ij} \in \mathbb{R}^{d_p} \mid \mathbf{O}_{ij} = 1\}$.
- *View features*, $\mathcal{V} = \{\mathbf{v}_i \in \mathbb{R}^{d_v} \mid i = 1, 2, \ldots, m\}$.
- *Scene point features*, $\mathcal{S} = \{\mathbf{s}_j \in \mathbb{R}^{d_s} \mid j = 1, 2, \ldots, n\}$.
- *Global features*, $\mathbf{g} \in \mathbb{R}^{d_g}$.

The *projection features* $\mathbf{p}_{ij}$ are feature vectors specific to every observed image point. In practice, these are bundled together and organized in a sparse vector-valued matrix, as illustrated by the large top-right matrix in Figure 1. The sparsity pattern of $\mathcal{P}$ will be fixed throughout the network and is determined by $\mathbf{O}$, which however depends on the scene. The *view features* $\mathcal{V}$ and *scene point features* $\mathcal{S}$, on their part, are organized in dense vector-valued column and row vectors, respectively, also illustrated in Figure 1. Finally, the *global features* $\mathbf{g}$, consist of a single globally shared feature vector, and is illustrated in the bottom-left corner of Figure 1.

In order to define graph operations, we see every feature vector as being associated with a corresponding graph node. At this point, note that a simple approach would be to connect all nodes $\mathcal{P}$, $\mathcal{V}$, $\mathcal{S}$, and $\{\mathbf{g}\}$ in one unified graph, and directly apply a graph neural network with global weight sharing. Instead, since the nodes are semantically different, we consider these semantics when choosing the level of weight sharing, and even vary the feature dimension depending on the type of node. In practice, we construct different graphs for propagation between different types of nodes.

Having defined the different feature types, we proceed to define corresponding update operations, which are repeatedly carried out throughout the network. In all pseudo-code, ReLU denotes to the rectified linear unit and LN denotes layer normalization [3]. Linear denotes a learned affine layer and FFN denotes a number of stacked Linear layers interleaved with ReLU activations. Subscripts may occur to emphasize feature dimensions. Note that all of these operations may be applied on multiple nodes at once, with shared parameters. The $+$ notation implies node-wise addition.

**Graph Cross-Attention.** Procedure 1 is utilized for all feature aggregations throughout the network. It carries out cross-attention [42], attending to and propagating information from one set of input / source nodes $\mathcal{H}_1$ to another set of output / target nodes $\widehat{\mathcal{H}}_2$. The previous target node features $\mathcal{H}_2$, if provided, will act as query features when computing attention scores. In the first few layers these may not be available, and initial zero-features are used instead.

In our context, the source nodes and target nodes correspond to different feature types, possibly with different feature dimension. As the source and target nodes are disjoint, we may represent the connectivity by a directed bipartite graph. Furthermore, in our context this graph will often consist of a number of disconnected subgraphs. Procedure 1 essentially acts as a wrapper around a GATv2 layer [4] (for which we use the PyTorch Geometric [11] implementation), while carrying out input normalization, and feature projections as necessary. We consistently use 4 attention heads for the GATv2 layers.

**Procedure 1** GRAPHCROSSATTENTION

**Input:** $\mathcal{H}_1$ Source node features.
**Input:** $\mathcal{H}_2$ Previous target node features (optional).
**Input:** $\mathcal{E}_{\mathcal{H}_1 \to \mathcal{H}_2}$ Set of edges from $\mathcal{H}_1 \to \mathcal{H}_2$ .
**Input:** $d_1$ Source node feature dimension.
**Input:** $d_2$ Target node feature dimension.
**Output:** $\widehat{\mathcal{H}}_2$ Updated target node features.

$\widehat{\mathcal{H}}_1 \leftarrow \text{ReLU}(\text{LN}(\mathcal{H}_1))$
**if** $\mathcal{H}_2$ provided **then**
  $\widehat{\mathcal{H}}_2 \leftarrow \text{ReLU}(\text{LN}(\mathcal{H}_2))$
  **if** $d_1 \neq d_2$ **then**
    $\widehat{\mathcal{H}}_2 \leftarrow \text{Linear}_{d_2 \to d_1}(\widehat{\mathcal{H}}_2)$
**else**
  $\widehat{\mathcal{H}}_2 \leftarrow$ zero features $\vec{0} \in \mathbb{R}^{d_1}$
$\widehat{\mathcal{H}}_2 \leftarrow \text{GATv2Conv}(\widehat{\mathcal{H}}_1, \widehat{\mathcal{H}}_2, \mathcal{E}_{\mathcal{H}_1 \to \mathcal{H}_2})$
**if** $d_1 \neq d_2$ **then**
  $\widehat{\mathcal{H}}_2 \leftarrow \text{Linear}_{d_1 \to d_2}(\widehat{\mathcal{H}}_2)$

**View & Scene Point Feature Update.** View features and scene point features are updated by Procedure 2 and 3, respectively, entirely symmetrical to one another. A directed bipartite graph is constructed by taking $\mathcal{P}$ as source nodes, and $\mathcal{V}$ (or $\mathcal{S}$) as target nodes, and connecting every projection node with its corresponding view (or scene point) node, corresponding to rows (or columns) of $\mathcal{P}$. Graph cross-attention according to Procedure 1 is then carried out on the graph, and used as a residual mapping. This is followed by another residual mapping with a feed-forward network. Illustrations of these update operations are provided in Figures 2 and 3.

**Global Feature Update.** The global feature update, outlined in Procedure 4 is carried out in almost the same manner, but now aggregating from all view features $\mathcal{V}$ and scene point features $\mathcal{S}$ instead of $\mathcal{P}$. This is carried out with two independent aggregations, with different learned parameters, and the sum of both aggregations constitute the residual mapping. Figure 4 provides an illustration for the update

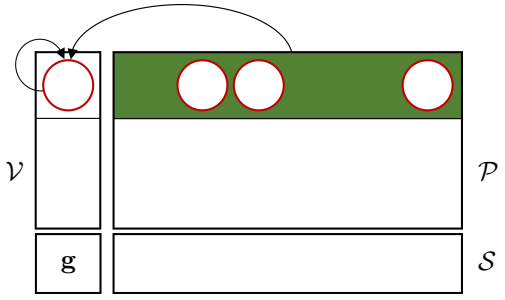


Figure 2. Illustration of the update of a single view feature. All $\mathcal{V}$ features are updated based on their previous value and the corresponding rows of $\mathcal{P}$.

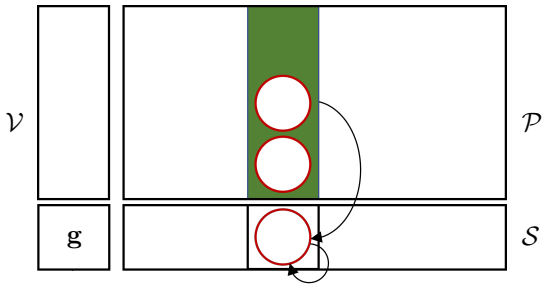


Figure 3. Illustration of the update of a single scene point feature. All $\mathcal{S}$ features are updated based on their previous value and the corresponding columns of $\mathcal{P}$.

**Procedure 2** UPDATEVIEWFEAT

**Input:** $\mathcal{P}, \mathbf{O}$ Projection features + observability matrix.
**Input:** $\mathcal{V}$ Previous view features (optional).
**Input:** $d_p, d_v$ Feature dimensions.
**Output:** $\widehat{\mathcal{V}}$ Updated view features.

$\mathcal{E}_{\mathcal{P} \to \mathcal{V}} \leftarrow \{(\mathcal{P}_{ij}, \mathcal{V}_i) \mid \mathbf{O}_{ij} = 1, \quad i = 1, 2, \ldots, m\}$
**if** $\mathcal{V}$ provided **then**
  $\widehat{\mathcal{V}} \leftarrow \mathcal{V} + \text{GRAPHCROSSATTENTION}(\mathcal{P}, \mathcal{V}, \mathcal{E}_{\mathcal{P} \to \mathcal{V}}, d_p, d_v)$
**else**
  $\widehat{\mathcal{V}} \leftarrow \text{GRAPHCROSSATTENTION}(\mathcal{P}, \mathcal{E}_{\mathcal{P} \to \mathcal{V}}, d_p, d_v)$
$\widehat{\mathcal{V}} \leftarrow \widehat{\mathcal{V}} + \text{FFN}(\text{ReLU}(\text{LN}(\widehat{\mathcal{V}})))$

**Procedure 3** UPDATESCENEPOINTFEAT

**Input:** $\mathcal{P}, \mathbf{O}$ Projection features + observability matrix.
**Input:** $\mathcal{S}$ Previous scene point features (optional).
**Input:** $d_p, d_s$ Feature dimensions.
**Output:** $\widehat{\mathcal{S}}$ Updated scene point features.

$\mathcal{E}_{\mathcal{P} \to \mathcal{S}} \leftarrow \{(\mathcal{P}_{ij}, \mathcal{S}_i) \mid \mathbf{O}_{ij} = 1, \quad i = 1, 2, \ldots, m\}$
**if** $\mathcal{S}$ provided **then**
  $\widehat{\mathcal{S}} \leftarrow \mathcal{S} + \text{GRAPHCROSSATTENTION}(\mathcal{P}, \mathcal{S}, \mathcal{E}_{\mathcal{P} \to \mathcal{S}}, d_p, d_s)$
**else**
  $\widehat{\mathcal{S}} \leftarrow \text{GRAPHCROSSATTENTION}(\mathcal{P}, \mathcal{E}_{\mathcal{P} \to \mathcal{S}}, d_p, d_s)$
$\widehat{\mathcal{S}} \leftarrow \widehat{\mathcal{S}} + \text{FFN}(\text{ReLU}(\text{LN}(\widehat{\mathcal{S}})))$

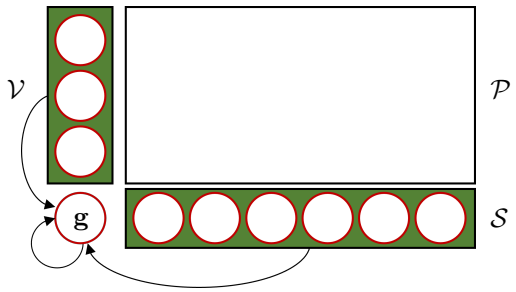


Figure 4. Illustration of the global feature update, where $\mathbf{g}$ is updated based on its previous value, together with an aggregation of all $\mathcal{V}$ and $\mathcal{S}$ features.

operation.

**Procedure 4** UPDATEGLOBALFEAT

**Input:** $\mathcal{V}$ View features.
**Input:** $\mathcal{S}$ Scene point features.
**Input:** $\mathbf{g}$ Previous global features (optional).
**Input:** $d_v, d_s, d_g$ Feature dimensions.
**Output:** $\widehat{\mathbf{g}}$ Updated global features.

$\mathcal{E}_{\mathcal{V}\to\mathbf{g}} \leftarrow \{(\mathcal{V}_i, \mathbf{g}) \mid i = 1, 2, \ldots, m\}$
$\mathcal{E}_{\mathcal{S}\to\mathbf{g}} \leftarrow \{(\mathcal{S}_j, \mathbf{g}) \mid j = 1, 2, \ldots, n\}$
**if** $\mathbf{g}$ provided **then**
  $\widehat{\mathbf{g}}_v \leftarrow \text{GRAPHCROSSATTENTION}(\mathcal{V}, \{\mathbf{g}\}, \mathcal{E}_{\mathcal{V}\to\mathbf{g}}, d_v, d_g)$
  $\widehat{\mathbf{g}}_s \leftarrow \text{GRAPHCROSSATTENTION}(\mathcal{S}, \{\mathbf{g}\}, \mathcal{E}_{\mathcal{S}\to\mathbf{g}}, d_s, d_g)$
  $\widehat{\mathbf{g}} \leftarrow \mathbf{g} + \widehat{\mathbf{g}}_v + \widehat{\mathbf{g}}_s$
**else**
  $\widehat{\mathbf{g}}_v \leftarrow \text{GRAPHCROSSATTENTION}(\mathcal{V}, -, \mathcal{E}_{\mathcal{V}\to\mathbf{g}}, d_v, d_g)$
  $\widehat{\mathbf{g}}_s \leftarrow +\text{GRAPHCROSSATTENTION}(\mathcal{S}, -, \mathcal{E}_{\mathcal{S}\to\mathbf{g}}, d_s, d_g)$
  $\widehat{\mathbf{g}} \leftarrow \widehat{\mathbf{g}}_v + \widehat{\mathbf{g}}_s$
$\widehat{\mathbf{g}} \leftarrow \widehat{\mathbf{g}} + \text{FFN}(\text{ReLU}(\text{LN}(\widehat{\mathbf{g}})))$

**Projection Feature Update.** The projection features are updated according to Procedure 5. No aggregation is involved during this step. Instead, at every projection node the corresponding $\mathcal{V}$, $\mathcal{S}$ and $\mathbf{g}$ features are all collected along with the current projection features $\mathcal{P}$, as illustrated in Figure 5. Each of these sources then undergoes a ReLU activation and normalization with LN, before being concatenated. Note that the initial projection features $\mathcal{P}_0$ are typically also added to the input, and in this case they are simply concatenated to the previous projection features before the procedure is called. Finally, a residual mapping is applied, by feeding the concatenated features to a shared feed-forward network.

**Entire Network Architecture.** Now we have all the pieces to define the complete network architecture, which is outlined in Procedure 6. The structure is straightforward: 1) UPDATEVIEWFEAT and UPDATESCENEPOINTFEAT (Procedures 2 and 3) are called to aggregate projection features. 2) UPDATEGLOBALFEAT (Procedure 4) is called to aggregate the resulting features further, to a single global feature vector. 3) UPDATEPROJFEAT (Procedure 5) is called in order to again distribute the aggregated information across the projection features. These steps are repeated $L$ times, where we refer to $L$ as the number of layers. Then, a final call to UPDATEVIEWFEAT and UPDATESCENEPOINTFEAT is carried out, and the resulting view and scene point features are further processed by two corresponding regression heads, consisting of shared 3-layer feed-forward networks (FFN) with hidden neurons of the same dimension as the input ($d_v$ or $d_s$). The scene point head simply outputs the 3 coordinates of every estimated scene point. For Euclidean reconstruction, the view head outputs the 3 coordinates of the location of the camera center, together with a unit-norm quaternion representing its orientation. For projective reconstruction, all 12 camera matrix elements are regressed, followed by a normalization in the same manner as [31]. Except for the regression heads, we use 1-layer FFNs throughout the network, which in general seemed slightly easier to train.

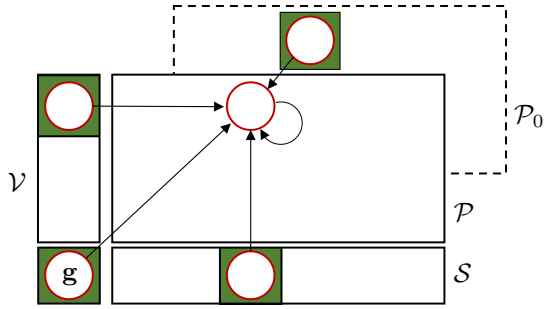


Figure 5. Illustration of the update of a single projection feature. All $\mathcal{P}$ features are updated based on their previous value, as well as initial projection features $\mathcal{P}_0$, the global feature $\mathbf{g}$, and the corresponding $\mathcal{V}$ and $\mathcal{S}$ features.

**Procedure 5** UPDATEPROJFEAT

**Input:** $\mathcal{P}$ Previous projection features.
**Input:** $\mathcal{V}$ View features.
**Input:** $\mathcal{S}$ Scene point features.
**Input:** $\mathbf{g}$ Global features.
**Input:** $d_p^{in}, d_v, d_s, d_g, d_p^{out}$ Feature dimensions.
**Output:** $\widehat{\mathcal{P}}$ Updated projection features.

$\widehat{\mathcal{P}} \leftarrow \text{ReLU}(\text{LN}(\mathcal{V})) \parallel \text{ReLU}(\text{LN}(\mathcal{S})) \parallel \text{ReLU}(\text{LN}(\mathbf{g})) \parallel \text{ReLU}(\text{LN}(\mathcal{P}))$ `// View, scene point, and global features are implicitly distributed to their corresponding projection features before feature concatenation.`
$\widehat{\mathcal{P}} \leftarrow \mathcal{P} + \text{FFN}_{(d_v+d_s+d_g+d_p^{in})\to d_p^{out}}(\widehat{\mathcal{P}})$

## 3.2. Loss Function

The network is trained using the Adam optimizer, and as loss function we use the (non-squared) reprojection error, averaged over all observed image points. To handle negative

**Procedure 6** GRAPHATTENTIONSFM

**Input:** $\mathcal{M}$ Image point tracks, normalized.
**Input:** $\mathbf{O}$ Observability matrix.
**Input:** $d_p, d_v, d_s, d_g$ Feature dimensions.
**Input:** $L$ Number of layers.
**Output:** $\mathcal{V}$ Estimated camera parameters.
**Output:** $\mathcal{S}$ Estimated scene points.

$\mathcal{P}_0 \leftarrow \text{Linear}_{2\to 2}(\mathcal{M})$ `// Initial projection features: linear embedding of image points.`
$\mathcal{V} \leftarrow \text{UPDATEVIEWFEAT}(\mathcal{P}_0, \mathbf{O}, -, 2, d_v)$
$\mathcal{S} \leftarrow \text{UPDATESCENEPOINTFEAT}(\mathcal{P}_0, \mathbf{O}, -, 2, d_s)$
$\mathbf{g} \leftarrow \text{UPDATEGLOBALFEAT}(\mathcal{V}, \mathcal{S}, -, d_v, d_s, d_g)$
**for** $l = 1, 2, \ldots, L$ **do**
  **if** $l = 1$ **then**
    $\mathcal{P} \leftarrow \text{UPDATEPROJFEAT}(\mathcal{P}_0, \mathcal{V}, \mathcal{S}, \mathbf{g}, 2, d_v, d_s, d_g, d_p)$
  **else**
    $\mathcal{P} \leftarrow \text{UPDATEPROJFEAT}(\mathcal{P} \| \mathcal{P}_0, \mathcal{V}, \mathcal{S}, \mathbf{g}, d_p + 2, d_v, d_s, d_g, d_p)$
  $\mathcal{V} \leftarrow \text{UPDATEVIEWFEAT}(\mathcal{P}, \mathbf{O}, \mathcal{V}, d_p, d_v)$
  $\mathcal{S} \leftarrow \text{UPDATESCENEPOINTFEAT}(\mathcal{P}, \mathbf{O}, \mathcal{S}, d_p, d_s)$
  **if** $l < L$ **then**
    $\mathbf{g} \leftarrow \text{UPDATEGLOBALFEAT}(\mathcal{V}, \mathcal{S}, \mathbf{g}, d_v, d_s, d_g)$
$\{\mathbf{P}_i\} \leftarrow \text{FFN}(\text{ReLU}(\mathcal{V}))$ `// Camera regression head`
$\{\mathbf{X}_j\} \leftarrow \text{FFN}(\text{ReLU}(\mathcal{S}))$ `// Scene point regression head`

depths and overcome the singularity at the principal plane, we substitute the reprojection error with minus the depth in the loss function for any projection with depth smaller than a threshold $h = 1e-4$. We also apply gradient normalization in the same manner as [31].

### 3.3. Data Augmentation

For Euclidean reconstruction, where the image points are normalized using the intrinsic camera parameters, and therefore geometrically interpretable, we experiment with augmenting the training scenes with slightly transformed variations. While the scene points remain fixed, each camera is randomly rotated about its center, first by a uniformly sampled angle $\alpha \in [-15, 15]$ about the principal axis, followed by a uniformly sampled angle $\gamma \in [-20, 20]$ about an axis orthogonal to the principal axis with a random direction. The image points are transformed accordingly.

### 3.4. Artificial Outlier Injection

We also consider the presence of measurement outliers, by artificially corrupting the data. This is done by: 1) Selecting a subset of the 2D keypoint measurements to be replaced by artificial outliers. 2) For each view, replace all measurements marked to be outliers with random samples from a bivariate normal distribution, which is fit to all remaining (inlier) points. For step 1), we carefully select points for outlier injection such that we avoid getting any scene points visible in $< 2$ views or any views with $< 8$ scene points visible, if regarding only the inlier projections. In the supplementary material, the details are given on how this is achieved. Note that while the network input is corrupted by outlier injection, the uncorrupted data is still used as learning targets for the loss function.

## 4. Results

### 4.1. Experimental Setup

We take a particular interest in Euclidean reconstruction, due to the relevance of metric reconstructions for photogrammetry applications. Furthermore, we focus mainly on the task of learning and generalizing from a set of training scenes, due to the great potential in having a fast learned model acquire reconstructions of completely novel scenes. We use the DPESFM method of Moran et al. [31] as a baseline, as they have presented experiments on precisely this. To this end, we also use Olsson's dataset [33] (together with the VGG dataset [12] for additional projective reconstruction experiments in the supplement), and we use the same random partition of 10 test scenes and 3 validation scenes.

We base our implementation on that of [31], implemented in PyTorch [34], but replace the network architecture with our proposed GRAPHATTENTIONSFM network, implemented using GATv2Conv layers [4] from PyTorch Geometric [11]. The implementation details are in general consistent with [31], for more details see the supplement. The DPESFM results reproduced by us are obtained by running the method of [31] with our code base, with the hyper-parameters published on the official GitHub repository of the method [30]. For our model, we use $L = 12$ layers and feature dimensions $d_p = 32, d_v = 1024, d_s = 64, d_g = 2048$, which corresponds to 145M parameters. By using small dimensions for the projection features and scene point features the otherwise high memory consumption is significantly reduced. Hyper-parameters have been chosen based on the performance on the validation data, but without any rigorous hyper-parameter tuning.

### 4.2. Euclidean Reconstruction of Novel Scenes

In Table 1, we report the Euclidean reconstruction results of our method, when applied on novel test scenes, and measure the performance by reprojection errors, as well as rotation and translation errors for the estimated camera poses. For calculating the latter, the predicted and ground truth poses are first aligned to a common reference frame, as described in Section C in the supplement. The results of DPESFM as reported in [31] can be seen in parentheses, when available, but we also retrain their model and incorporate further results of the model, not reported in [31], in particular results without bundle adjustment, and additional metrics to reprojection errors. Experiments are carried out both with and without data augmentation with random rotational cameras perturbations, as described in Section 3.3, and the columns are partitioned accordingly. Consistently with [31], *Inference* refers to the network output followed

| | | Without data augmentation | | | | With data augmentation | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Inference | | Inference + BA | | Inference | | Inference + BA | | |
| | | Ours | DPESFM | Ours | DPESFM | Ours | DPESFM | Ours | DPESFM | Colmap |
| Reprojection error (px) | Alcatraz Courtyard | 68.41 | 68.50 | 1.70 | **0.81** (0.82) | 36.01 | 92.37 | **0.81** | 0.92 | **0.81** |
| | Alcatraz Water Tower | 36.30 | 50.47 | 1.05 | 1.13 (**0.55**) | 87.67 | 2831.94 | 0.88 | 10.16 | **0.55** |
| | Drinking Fountain Somewhere in Zurich | 36.02 | 45.87 | 0.55 | **0.31** (7.21) | 219.75 | 234.90 | **0.31** | 6.73 | **0.31** |
| | Nijo Castle Gate | 65.71 | 64.53 | **0.73** | **0.73** (5.81) | 61.41 | 68.19 | 0.88 | 0.89 | **0.73** |
| | Porta San Donato Bologna | 74.32 | 94.26 | **0.74** | **0.74** (1.10) | 52.15 | 84.46 | 0.76 | 0.75 | 0.75 |
| | Round Church Cambridge | 61.18 | 55.51 | **0.39** | **0.39** (0.50) | 29.80 | 59.54 | **0.39** | 1.49 | **0.39** |
| | Smolny Cathedral St Petersburg | 156.16 | 120.85 | **0.81** | **0.81** (15.15) | 85.38 | 87.81 | **0.81** | **0.81** | **0.81** |
| | Some Cathedral in Barcelona | 150.01 | 146.10 | 10.42 | 12.71 (21.46) | 125.68 | 687.83 | **0.89** | 16.77 | **0.89** |
| | Sri Veeramakaliamman Singapore | 121.02 | 157.39 | 2.19 | 16.87 (16.92) | 83.50 | 166.68 | 2.13 | 9.30 | **0.71** |
| | Yueh Hai Ching Temple Singapore | 37.46 | 52.59 | **0.65** | **0.65** (1.16) | 25.60 | 51.35 | **0.65** | 0.73 | **0.65** |
| | Average | 80.66 | 85.61 | 1.92 | 3.52 (7.07) | 80.69 | 436.51 | 0.85 | 4.86 | **0.66** |
| Rotation error (deg) | Alcatraz Courtyard | 10.102 | 13.201 | 2.607 | 0.035 | 6.093 | 10.946 | 0.038 | **0.030** | 0.043 |
| | Alcatraz Water Tower | 10.637 | 11.053 | 0.499 | 0.764 | 11.501 | 10.641 | 0.699 | 19.351 | **0.228** |
| | Drinking Fountain Somewhere in Zurich | 15.846 | 16.014 | 0.003 | **0.001** | 15.415 | 15.704 | **0.001** | 22.759 | 0.007 |
| | Nijo Castle Gate | 16.751 | 10.546 | 0.062 | 0.062 | 17.347 | 20.032 | 0.038 | **0.036** | 0.064 |
| | Porta San Donato Bologna | 23.839 | 24.120 | 0.095 | **0.094** | 18.411 | 25.004 | **0.094** | **0.094** | 0.099 |
| | Round Church Cambridge | 18.906 | 14.473 | 0.029 | **0.026** | 10.295 | 18.685 | 0.029 | 1.086 | 0.035 |
| | Smolny Cathedral St Petersburg | 19.387 | 17.971 | 0.023 | 0.022 | 11.662 | 14.380 | 0.023 | **0.019** | 0.029 |
| | Some Cathedral in Barcelona | 27.270 | 30.471 | 10.009 | 20.050 | 27.908 | 29.119 | **0.020** | 47.892 | 0.025 |
| | Sri Veeramakaliamman Singapore | 28.275 | 36.903 | 0.549 | 4.871 | 23.702 | 36.176 | 0.457 | 2.759 | **0.169** |
| | Yueh Hai Ching Temple Singapore | 15.733 | 22.706 | **0.038** | **0.038** | 9.515 | 21.561 | **0.038** | **0.038** | 0.043 |
| | Average | 18.675 | 19.746 | 1.391 | 2.596 | 15.185 | 20.225 | 0.144 | 9.406 | **0.074** |
| Translation error (m) | Alcatraz Courtyard | 4.82 | 4.93 | 1.09 | **0.01** | 2.73 | 5.74 | **0.01** | **0.01** | **0.01** |
| | Alcatraz Water Tower | 8.66 | 7.35 | 0.31 | 0.44 | 7.53 | 7.77 | 0.41 | 9.05 | **0.12** |
| | Drinking Fountain Somewhere in Zurich | 4.44 | 4.44 | **0.00** | **0.00** | 4.45 | 4.46 | **0.00** | 1.38 | **0.00** |
| | Nijo Castle Gate | 5.07 | 3.08 | **0.01** | **0.01** | 5.67 | 6.95 | **0.01** | **0.01** | **0.01** |
| | Porta San Donato Bologna | 9.50 | 10.72 | **0.05** | **0.05** | 4.80 | 10.48 | **0.05** | **0.05** | **0.05** |
| | Round Church Cambridge | 8.94 | 7.19 | **0.01** | **0.01** | 5.28 | 9.00 | **0.01** | 0.56 | **0.01** |
| | Smolny Cathedral St Petersburg | 2.70 | 2.43 | **0.01** | **0.01** | 2.33 | 2.52 | **0.01** | **0.01** | **0.01** |
| | Some Cathedral in Barcelona | 12.64 | 12.69 | 3.22 | 6.38 | 12.32 | 12.66 | **0.01** | 11.93 | **0.01** |
| | Sri Veeramakaliamman Singapore | 4.94 | 4.90 | 0.16 | 1.32 | 4.93 | 4.90 | 0.14 | 0.77 | **0.04** |
| | Yueh Hai Ching Temple Singapore | 4.12 | 4.27 | **0.01** | **0.01** | 2.44 | 4.28 | **0.01** | **0.01** | **0.01** |
| | Average | 6.58 | 6.20 | 0.49 | 0.82 | 5.25 | 6.88 | 0.07 | 2.38 | **0.03** |

Table 1. Results of Euclidean reconstruction of novel test scenes, with and without data augmentation. The results of DPESFM [31] have been acquired by us training the model, along with the results reported by [31] in parentheses, if available. The result of Colmap, as reported by [31], is also added for reference.

| Scene | Infer. | BA | Colmap |
|---|---|---|---|
| Alcatraz Courtyard | 0.24 | 45.54 | 286 |
| Alcatraz Water Tower | 0.13 | 31.11 | 130 |
| Drinking Fountain Somewhere In Zurich | 0.06 | 1.98 | 16 |
| Nijo Castle Gate | 0.09 | 3.97 | 21 |
| Porta San Donato Bologna | 0.18 | 27.02 | 170 |
| Round Church Cambridge | 0.43 | 56.47 | 229 |
| Smolny Cathedral St Petersburg | 0.49 | 86.09 | 516 |
| Some Cathedral In Barcelona | 0.24 | 47.05 | 451 |
| Sri Veeramakaliamman Singapore | 0.63 | 115.80 | 583 |
| Yueh Hai Ching Temple Singapore | 0.08 | 8.54 | 106 |

Table 2. Runtime (s) of our method for Euclidean reconstruction on test scenes, in comparison with Colmap (measured by [31]).

by a computationally cheap triangulation and *Inference + BA* refers to the network output followed by bundle adjustment, which is carried out in the same manner as [31], using the Ceres solver [1]. Moran et al. [31] also performed experiments with fine-tuning the network parameters on the test scenes. In contrast, we advocate against this, since this approach is relatively costly, and one might as well acquire high-quality reconstructions from traditional SfM pipelines such as Colmap, in similar execution time. Nevertheless, the supplementary material includes such experiments, for completeness.

Without data augmentation, there is no clear difference between the *Inference* results of both methods. Neither reconstruction is good, and it is not so meaningful to compare minor differences in the metrics under these circumstances. In particular, calculating rotation and translation errors requires aligning the predicted and ground truth camera poses in a common reference frame, as described in Section C in the supplement, and when the predicted configuration of camera poses is not very consistent with the ground truth, determining this alignment itself can be sensitive. In all other scenarios, we outperform DPESFM, and when using both data augmentation and applying bundle adjustment, the solutions we acquire for the novel test scenes are in general very good. In terms of reprojection error and translation error, we match the performance of Colmap for almost every test scene, leading to an average reprojection

error of 0.85 (vs 0.66) px, and an average translation error of 0.07 (vs 0.03) m. In terms of rotation error, the error is actually lower than Colmap's for 8 out of 10 scenes, but the average error is slightly larger: 0.144 (vs 0.074) degrees.

### 4.3. Artificial Outlier Injection

In Table 3 we present additional results where, on top of data augmentation (see Section 3.3), we also apply artificial outlier injection during training according to Section 3.4.

We evaluate inference of the model in two settings: 1) Applied on the uncorrupted outlier-free test scenes as-is. 2) Applied to the test scenes followed by random injection of $10\%$, in the same manner as during training[2]. One can immediately observe significantly reduced reprojection errors for our model, suggesting that the outlier injection during training has a regularizing effect. Beyond that, this experiment is intended to serve as a teaser for the utility and promise of learning-based methods for SfM. Constant challenges such as the presence of outliers may not be as big of a challenge for learning-based as for conventional methods, and the phenomenon may even be exploited to develop effective training strategies. In fact, for any real-world scenario, it is crucial for learning-based methods applied on image point correspondences to be resilient to outliers, as we do not have the luxary of facing high-quality curated data such as Olsson's [33] in the wild. Note that it should be possible to apply bundle adjustment on the corrupted test data as well, when combined with a robust loss function.

Similarly as for the data augmentation, we note that DPESFM appears to struggle with the presence of outliers during training, and its performance on the test scenes at inference is not very good. It is very probable that the non-linear attention layers make our model more powerful and expressive than DPESFM.

### 4.4. Additional Results

In the supplementary material we provide additional results on projective reconstruction, as well as single-scene optimization, for completeness. We also include statistics of the number of views and scene points for each scene.

## 5. Conclusion

With this paper we have added to the research direction of learned initialization-free Structure-from-Motion by introducing a novel and expressive graph attention network outperforming previous learning-based methods for Euclidean reconstruction of novel scenes. When succeeded by bundle adjustment, we are able to reconstruct novel test scenes to perfection or of very decent precision, at a speedup of about $5-10\times$ compared to Colmap. Moreover, we illustrate great potential in coping with the presence of outliers, and even show artificial outlier injection to provide an effective regularizing training strategy, combined with data augmentation by random rotational camera perturbations.

For future work, it could definitely be promising to train on larger datasets, since model generalization is still a challenge. While the proposed model does not require fine-tuning, and thus is much faster to execute than previous work, it still relies on bundle adjustment to acquire a good reconstruction, which constitutes the computational bottleneck. Exploring alternative formulations where the network regresses relative camera poses is also an interesting direction, as the current absolute pose prediction may be a weakness due to the reconstruction ambiguity up to a similarity transformation. Other possible research directions include unrolling the architecture to multiple learned iterations / refinement steps, extensions to non-rigid Structure-from-Motion, and incorporating modern learned image matching pipelines such as [5, 6, 40] in an end-to-end fashion.

| | Uncorrupted | | Outlier-injected | |
|---|---|---|---|---|
| | Ours | DPESFM | Ours | DPESFM |
| Alcatraz Courtyard | **47.74** | 85.81 | **52.99** | 94.24 |
| Alcatraz Water Tower | **35.96** | 72.84 | **37.89** | 83.55 |
| Drinking Fountain Somewhere in Zurich | **52.08** | 1012.14 | **46.65** | 1453.31 |
| Nijo Castle Gate | **46.48** | 72.99 | **62.52** | 126.18 |
| Porta San Donato Bologna | **53.12** | 88.02 | **65.08** | 94.72 |
| Round Church Cambridge | **36.09** | 63.72 | **48.63** | 90.63 |
| Smolny Cathedral St Petersburg | **47.28** | 91.03 | **59.52** | 98.05 |
| Some Cathedral in Barcelona | **109.86** | 397.75 | **123.75** | 462.28 |
| Sri Veeramakali-amman Singapore | **63.60** | 169.63 | **70.90** | 146.98 |
| Yueh Hai Ching Temple Singapore | **26.83** | 51.41 | **36.69** | 57.59 |
| Average | **51.91** | 210.53 | **60.46** | 270.75 |

Table 3. Reprojection errors of Euclidean reconstruction of novel test scenes, with model trained with data augmentation as well as artificial outlier injection. The results of DPESFM [31] have been acquired by us training the model.

**Acknowledgements** The work was supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation, and the Swedish Research Council grants no. 2016-04445, 2018-05375, and 2023-05341. Computations were enabled by resources provided by the National Academic Infrastructure for Supercomputing in Sweden (NAISS) at Chalmers Centre for Computational Science and Engineering (C3SE) and National Supercomputer Centre (NSC) Berzelius at Linköping University partially funded by the Swedish Research Council through grant no. 2022-06725.

[2] Yet, during evaluation, we use all true measurements as ground truth targets when calculating reprojection errors, even in the corrupted case.

## References


[1] Sameer Agarwal, Keir Mierle, and Others. Ceres solver. http://ceres-solver.org/. 7

[2] Sameer Agarwal, Noah Snavely, Ian Simon, Steven M. Seitz, and Richard Szeliski. Building rome in a day. In *2009 IEEE 12th International Conference on Computer Vision*, pages 72–79, 2009. 1

[3] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016. 3

[4] Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks? In *International Conference on Learning Representations*, 2022. 2, 4, 6

[5] Georg Bökman and Fredrik Kahl. A case for using rotation invariant features in state of the art feature matchers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5110–5119, 2022. 8

[6] Georg Bökman, Johan Edstedt, Michael Felsberg, and Fredrik Kahl. Steerers: A framework for rotation equivariant keypoint descriptors. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2024. 8

[7] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. *TensoRF: Tensorial Radiance Fields*, pages 333–350. 2022. 2

[8] Yuchao Dai, Hongdong Li, and Mingyi He. Projective multiview structure and motion from element-wise factorization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(9):2238–2251, 2013. 1

[9] Olof Enqvist, Fredrik Kahl, and Carl Olsson. Non-sequential structure from motion. In *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 264–271, 2011. 1

[10] Anders Eriksson, Carl Olsson, Fredrik Kahl, and Tat-Jun Chin. Rotation averaging with the chordal distance: Global minimizers and strong duality. *IEEE Trans. Pattern Anal. Mach. Intell.*, 43(1):256–268, 2021. 1

[11] Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019. 4, 6

[12] Oxford Visual Geometry Group. Multi-view datasets. https://www.robots.ox.ac.uk/~vgg/data/mview/. 6

[13] Richard Hartley. In defense of the eight-point algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(6):580–593, 1997. 1, 2

[14] Richard Hartley and Fredrik Kahl. Critical configurations for projective reconstruction from multiple views. *Int. J. Comput. Vis.*, 71(1):5–47, 2007.

[15] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2 edition, 2004. 1

[16] Je Hyeong Hong, Christopher Zach, Andrew Fitzgibbon, and Roberto Cipolla. Projective bundle adjustment from arbitrary initialization using the variable projection method. In *ECCV 2016*, pages 477–493. Springer, 2016. 2, 9

[17] Je Hyeong Hong, Christopher Zach, and Andrew Fitzgibbon. Revisiting the variable projection method for separable nonlinear least squares problems. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5939–5947, 2017. 1

[18] José Pedro Iglesias, Amanda Nilsson, and Carl Olsson. expOSE: Accurate initialization-free projective factorization using exponential regularization. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 8959–8968, 2023. 1

[19] Nianjuan Jiang, Zhaopeng Cui, and Ping Tan. A global linear method for camera pose registration. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2013. 3, 6, 7, 8

[20] Fredrik Kahl and Richard Hartley. Multiple view geometry under the $L_\infty$-norm. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(9):1603–1617, 2008. 1

[21] Fredrik Kahl, Richard Hartley, and Kalle Astrom. Critical configurations for n-view projective reconstruction. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2001.

[22] Fredrik Kahl, Sameer Agarwal, Manmohan Chandraker, David Kriegman, and Serge Belongie. Practical global optimization for multiview geometry. *Int. J. Comput. Vis.*, 79 (3):271–284, 2008. 1

[23] Yoni Kasten, Amnon Geifman, Meirav Galun, and Ronen Basri. Gpsfm: Global projective sfm using algebraic constraints on multi-view fundamental matrices. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 9

[24] Yoni Kasten, Amnon Geifman, Meirav Galun, and Ronen Basri. Algebraic characterization of essential matrices and their averaging in multiview settings. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019. 3, 6, 7, 8

[25] Q. Ke and T. Kanade. Quasiconvex optimization for robust geometric reconstruction. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(10):1834–1847, 2007. 1

[26] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. 2

[27] Kurt Konolige. Sparse sparse bundle adjustment. In *Brit. Mach. Vis. Conf.*, pages 1–11, 2010. 1

[28] Ludovic Magerand and Alessio Del Bue. Practical projective structure from motion (p2sfm). In *Int. Conf. Comput. Vis.*, pages 39–47, 2017. 1, 9

[29] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis, 2020. cite arxiv:2003.08934Comment: ECCV 2020 (oral). Project page with videos and code: http://tancik.com/nerf. 2

[30] Dror Moran, Hodaya Koslowsky, Yoni Kasten, Haggai Maron, Meirav Galun, and Ronen Basri. Deep permutation-equivariant sfm: Github repository. https://github.com/drormoran/Equivariant-SFM/. 6

[31] Dror Moran, Hodaya Koslowsky, Yoni Kasten, Haggai Maron, Meirav Galun, and Ronen Basri. Deep permutation equivariant structure from motion. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*

*(ICCV)*, pages 5976–5986, 2021. 2, 5, 6, 7, 8, 1, 3, 4, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19

[32] David Nister. An efficient solution to the five-point relative pose problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6):756–770, 2004. 1

[33] Carl Olsson and Olof Enqvist. Stable structure from motion for unordered image collections. In *Scand. Conf. Image Analysis*, pages 524–535. Springer, 2011. 6, 8

[34] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. 6

[35] Pulak Purkait, Tat-Jun Chin, and Ian Reid. Neurora: Neural robust rotation averaging. In *Computer Vision – ECCV 2020*, pages 137–154, Cham, 2020. Springer International Publishing. 2

[36] Johannes L. Schönberger. Colmap code. https://colmap.github.io/. 3, 6, 7, 8

[37] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1

[38] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)*, 2016. 1, 3, 6, 7, 8

[39] Peter Sturm and Bill Triggs. A factorization based algorithm for multi-image projective structure and motion. In *Computer Vision — ECCV '96*, pages 709–720, Berlin, Heidelberg, 1996. Springer Berlin Heidelberg. 1

[40] Jiaming Sun, Zehong Shen, Yuang Wang, Hujun Bao, and Xiaowei Zhou. LoFTR: Detector-free local feature matching with transformers. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 8922–8931, 2021. 8

[41] Bill Triggs, Philip F. McLauchlan, Richard I. Hartley, and Andrew W. Fitzgibbon. Bundle adjustment - a modern synthesis. page 298–372, Berlin, Heidelberg, 1999. Springer-Verlag. 1

[42] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2017. 4

[43] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018. 2

[44] Jianyuan Wang, Christian Rupprecht, and David Novotny. PoseDiffusion: Solving pose estimation via diffusion-aided bundle adjustment. 2023. 2

[45] Zirui Wang, Shangzhe Wu, Weidi Xie, Min Chen, and Victor Adrian Prisacariu. NeRF−−: Neural radiance fields without known camera parameters. *arXiv preprint arXiv:2102.07064*, 2021.

[46] Xingkui Wei, Yinda Zhang, Zhuwen Li, Yanwei Fu, and Xiangyang Xue. Deepsfm: Structure from motion via deep bundle adjustment. In *ECCV*, 2020. 2

[47] Yi Wei, Shaohui Liu, Yongming Rao, Wang Zhao, Jiwen Lu, and Jie Zhou. Nerfingmvs: Guided optimization of neural radiance fields for indoor multi-view stereo. In *ICCV*, 2021. 2

[48] Christopher Zach and Je Hyeong Hong. pOSE: Pseudo object space error for initialization-free bundle adjustment. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1876–1885, 2018. 1