Unifying Viewgraph Sparsification and Disambiguation of Repeated Structures in Structure-from-Motion

Lalit Manam[1] and Venu Madhav Govindu[1*]

[1*]Indian Institute of Science, Bengaluru - 560012, INDIA.

*Corresponding author(s). E-mail(s): venug@iisc.ac.in;
Contributing authors: lalitmanam@iisc.ac.in;

Abstract

In Structure-from-Motion (SfM), viewgraphs obtained from pairwise camera relationships generally have a high redundancy of edges that can be sparsified while maintaining reconstruction quality. Due to incorrect image matching and repeated structures in scenes (symmetries), false edges are often present in the viewgraphs, which gives rise to ghosting and superimposed reconstruction artifacts. In this paper, we present a unified method to simultaneously perform the tasks of viewgraph sparsification via redundant edge removal and disambiguation by removing false edges. We design an edge scoring mechanism to determine redundant and false edges based on triples of cameras that have either two or three edges connecting the three cameras. Our edge selection is formulated as an optimization problem whose optimum can be obtained using a simple thresholding scheme. This results in a highly efficient algorithm which can be applied to any viewgraph without any restriction on its topology. Our algorithm can be incorporated into any SfM pipeline as a pre-processing step, making it modular. We demonstrate the efficacy of our method on publicly available datasets, with a significant reduction in reconstruction time while maintaining reconstruction quality and removing ghosting artifacts on generic datasets. Our method also removes false edges from ambiguous datasets, thereby avoiding incorrect superimposed reconstructions.

Keywords: Structure-from-Motion, Viewgraph Sparsification, Disambiguation, Repeated Structures, Camera Triples.

1 Introduction

The goal of Structure-from-Motion (SfM) Hartley & Zisserman (2004) is to obtain a 3D reconstruction given images of a scene, which involves estimating camera motions (rotations and translations). The initial step in SfM is to identify images capturing the same part of the scene. For large-scale unordered image collections, images are matched based on the descriptors assigned to them Arandjelovic et al. (2016); Hausler et al. (2021); Kulis & Grauman (2009); Nistér & Stewenius (2006); Noh et al. (2017). This leads to a graph representation, known as a viewgraph in the SfM literature, where cameras capturing the images are represented as nodes and similarity of descriptors between the images gives rise to edges. Next, the keypoints are extracted from the images, which are then matched across images based on the viewgraph connectivity Bay et al. (2008); Darmon et al. (2020); DeTone et al. (2018); Lindenberger et al. (2023); C. Liu et al. (2010); Lowe (2004); Rublee et al. (2011); Sarlin et al. (2020); Tola et al. (2009); Tyszkiewicz et al.

(2020); Yi et al. (2018). Subsequently, from inlier keypoint matches, epipolar geometry Hartley & Zisserman (2004) is estimated for every edge to obtain a relative motion between the camera pair. We refer to such matches on edges as **epipolar inliers**. This graph is processed either by incremental Schonberger & Frahm (2016); Snavely et al. (2006, 2008a); Wu (2011) or global SfM approaches Enqvist et al. (2011); Govindu (2001, 2004); Sweeney et al. (2015) to recover a 3D reconstruction of the scene along with camera motions.

For unordered image collections, popular views of a scene are captured more often, leading to a redundant set of images (equivalently, nodes) in the viewgraphs Snavely et al. (2008b). Thus, the coverage of the scene by the images is uneven, with a large number of image pairs (equivalently, edges) capturing the same part of the scene. This results in a highly redundant set of edges in the viewgraphs Havlena et al. (2010). We refer to the datasets with the above-mentioned properties as **generic datasets**. To sparsify the viewgraph, many methods Havlena et al. (2009, 2010); Shah et al. (2018); Shen et al. (2016); Snavely et al. (2008b) have been proposed, which maintain the accuracy of the reconstructions by selecting nodes and edges that are identified to be important.

Scenes containing repetitive structures lead to different symmetries Y. Liu et al. (2010), and cameras capturing the repeated structures are connected by edges in the viewgraph. This is due to the fact that such images look similar and hence have similar descriptors, even when they belong to different parts of the scene. This leads to a significant number of false edges in the viewgraph, and we refer to such datasets as **ambiguous datasets**. The presence of such false edges results in a reconstruction with different parts of the scene superimposed on each other. To deal with repetitive structures, many methods identify false edges via loop consistency checks and other viewgraph properties Cai et al. (2023); Cohen et al. (2012); Heinly et al. (2014a, 2014b); Jiang et al. (2012); Kataria et al. (2020); Roberts et al. (2011); Shah et al. (2018); Shen et al. (2016); Wang et al. (2018); Wilson & Snavely (2013); Yan et al. (2017); Zach et al. (2008, 2010).

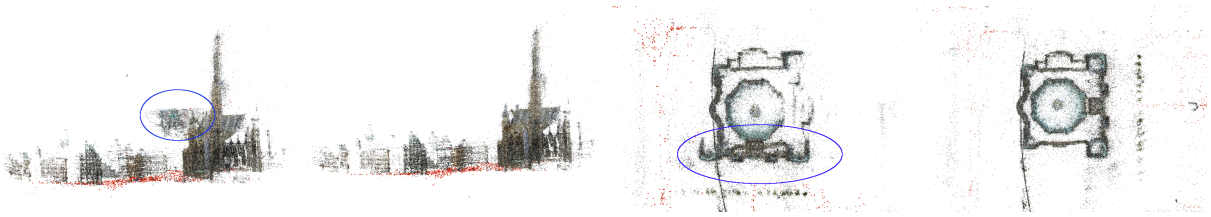
Table 1: Summary of different scenarios in viewgraphs handled by methods designed for specific tasks

Methods	Scenarios in Viewgraphs	
	Mostly True Edges	Many False Edges
Graph Sparsification	✓	✗
Disambiguation	✗	✓
Ours	✓	✓

In generic datasets, incorrect images are occasionally retrieved during the image matching phase, which results in a small number of false edges. These false edges arise for many reasons, such as low texture or different lighting conditions in the images, which result in keypoint descriptors that are not adequately discriminative. As a consequence, false edges are present in the viewgraphs, leading to **ghost artifacts** in reconstructions (Fig. 1a). On the other hand, repetitive structures in ambiguous datasets lead to a significant fraction of edges being false, due to cameras capturing unrelated parts of the scene being connected in the viewgraph. This results in **superimposed reconstructions** (Fig. 1b).

Most methods in the literature have been developed to deal with the two tasks of viewgraph sparsification and disambiguation of repeated structures in a standalone fashion. This is because the basic assumptions of both problems are contrasting. For sparsifying a viewgraph, most edges are considered to be reliable or true. Thus, the removal of some edges has minimal effect on reconstruction accuracy. In the case of disambiguating repeated structures, a significant fraction of the edges are false, which must be removed to obtain a correct reconstruction. Our objective is to handle both tasks simultaneously, which is summarized in Table 1.

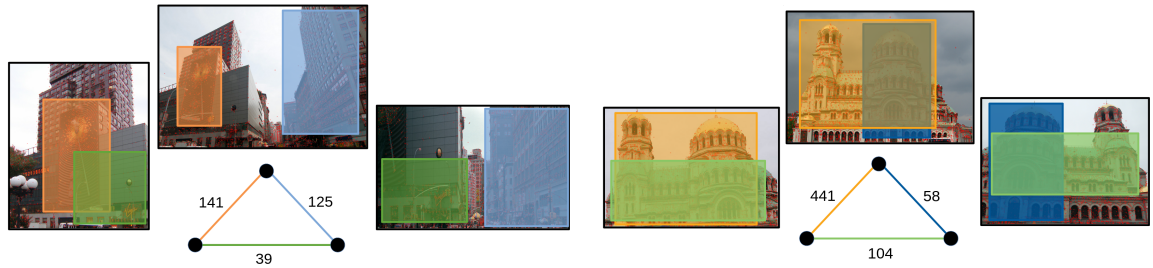
Our Contributions: Our aim is to solve both the problems of viewgraph sparsification and disambiguation of repeated structures in a unified manner. The central idea of our approach hinges on the fact that triples of cameras, connected by either two or three edges, are more informative than individual edges Snavely et al. (2008b); Zach et al. (2008). For the problems on hand, triples



(a) Ghost artifact
(Vienna Cathedral Wilson & Snavely (2014))

(b) Superimposed reconstruction
(Berliner Dom Heinly et al. (2014a))

Fig. 1: Incorrect reconstruction artifacts in different datasets. Artifacts are shown on the left of the subfigures and are marked in blue. (a) Generic dataset containing false edges resulting in ghost artifact. (b) Ambiguous dataset containing false edges due to repetitive structures resulting in superimposed reconstruction. Applying our method yields correct reconstructions, as shown on the right of the subfigures



(a) Generic case
(Union Square Wilson & Snavely (2014))

(b) Ambiguous case
(Alexander Nevsky Cathedral Heinly et al. (2014a))

Fig. 2: Camera triples in different cases. The number of epipolar inliers are shown for the respective edges. The part of the image where most of the inliers are matched is shown in shaded boxes with colours corresponding to the edge colours. The red dots on the images show the detected keypoints. (a) A triple in a generic dataset where the edge marked *green* has fewer inliers than other edges due to low overlap, thus, can potentially be removed to sparsify the viewgraph. (b) A triple in an ambiguous dataset where two different facades of a building are matched, giving false edges (marked *green* and *blue*). The false edges have a low number of inliers compared to the true edge (marked *orange*), thus giving a cue about ambiguity

provide an important indication of an edge being redundant or false based on matched keypoints (see Fig. 2). We propose a unified scheme based on camera triples to handle both tasks. Specifically, we design a scoring mechanism to identify redundant and false edges by aggregating information about the 3D point connectivity in the triples globally over the entire graph without relying on any intermediate partial reconstruction. Once edge scores are obtained, we formulate the task of edge removal as an optimization problem whose solution can be obtained using a simple thresholding scheme. Our previous work Manam & Govindu (2024) utilized the camera triples, which have all three edges present between them. This restricted the viewgraphs to contain edges that only participated in the camera triples with all three edges present in them.

In this paper, we extend our previous conference paper Manam & Govindu (2024) to handle general viewgraphs using triples. Instead of only considering camera triples containing all three edges (referred to as **strong triples**), we also consider camera triples with two edges connecting the three cameras (referred to as **weak triples**). Incorporating weak triples allows the scoring of all edges in a general viewgraph. This gives more

information about the 3D point connectivity of an edge with other connected edges. It also removes the necessity of preprocessing viewgraphs to contain only edges that contribute to strong triples, as required in Manam & Govindu (2024).

Moreover, we develop an algorithm that is parallelizable on the edges of a viewgraph, unlike Manam & Govindu (2024) that parallelizes on camera triples. This further reduces the compute time, in addition to the time saved by avoiding the extraction of strong triples, which is required in Manam & Govindu (2024). Our proposed method can be incorporated into any SfM pipeline as a preprocessing step, similar to Manam & Govindu (2024), making it practically usable. Experimental results on viewgraph sparsification reveal that the overall 3D scene structure is recovered and accuracy is maintained while reducing reconstruction time. Moreover, applying our method removes ghost artifacts in the reconstructions of generic datasets and disambiguates repeated structures on ambiguous datasets. In summary, our proposed method has the following advantages over Manam & Govindu (2024):

- can handle general viewgraphs with no restriction,
- does not require any preprocessing of viewgraphs,
- parallelizable algorithm on edges, which makes the process time efficient,
- results in more cameras and 3D points reconstructed.

2 Literature Review

In this section, we review the literature on viewgraph sparsification and disambiguation.

2.1 Viewgraph Sparsification

An initial attempt to sparsify viewgraphs was presented in Snavely et al. (2008b), where skeletal graphs were constructed by utilizing approximate covariance of camera motions. Then, image level descriptors used for image matching were incorporated in Havlena et al. (2009) to choose camera triples for estimating partial reconstructions and then merge them. In Havlena et al. (2010), a minimally connected dominating set was chosen from the input images based on image level descriptors which was used for reconstruction. An online algorithm was used in Shen et al. (2016) to obtain a spanning tree and expand the tree based on strong triples with a community-based graph topology. A minimum cost network flow problem was used in Shah et al. (2018) to select edges on viewgraphs.

Many methods relied on intermediate partial reconstructions Havlena et al. (2009, 2010); Snavely et al. (2008b), designed their own specific SfM pipeline Havlena et al. (2009, 2010); Snavely et al. (2008b) or required multiple hyperparameters Shah et al. (2018); Shen et al. (2016) to be tuned. However, our solution is based on a single hyperparameter without any intermediate reconstruction and can be incorporated into any pipeline.

2.2 Disambiguation of Repeated Structures

The idea of analyzing missing correspondences was introduced in Zach et al. (2008), where loops in strong camera triples were employed in a Bayesian inference framework to detect false edges. This idea was further extended in Zach et al. (2010) to work on larger loops using relative camera motions. An expectation-maximization framework McLachlan & Krishnan (2008) was used in Roberts et al. (2011) to infer false edges. In Cohen et al. (2012), similarity transforms were estimated to detect symmetries and remove false edges, which reduced drift in reconstructions from image sequences. A measure based on missing correspondences was proposed in Jiang et al. (2012) to select edges in a viewgraph. In Wilson & Snavely (2013), false edges were inferred based on subgraphs of the camera-3D point relation graph. A post-processing step was suggested in Heinly et al. (2014a), where reconstructions were split based on conflicting observations and then merged with the help of epipolar inliers. In a similar spirit, the clustering behaviour of 3D point connectivity was analyzed in Heinly et al. (2014b) to isolate 3D points that incorrectly linked separate parts of the scene. Two-view reconstructions were merged in Cui & Tan (2015) to find missing correspondences. In Yan et al. (2017), a few images were selected as anchors and other images were related to them based on paths in a viewgraph to obtain

clean image correspondences. In Wang et al. (2018), edges on viewgraphs were scored based on their ability to expand the reconstruction. In Kataria et al. (2020), only the keypoints which were matched across a few images were considered for resectioning cameras in incremental SfM pipelines. Doppelgangers was proposed in Cai et al. (2023), where a neural network is trained to detect false edges based on images, keypoints and matches. Some of the graph sparsification methods were also able to disambiguate repeated structures using geometric consistency along loops Shen et al. (2016) or with modified cost functions Shah et al. (2018) to handle ambiguous datasets. Some image retrieval methods Achar et al. (2011); Knopp et al. (2010); Torii et al. (2013) were developed to avoid incorrect fetching of street images (which contain repetitive structures) during the image matching step by specific designs of image descriptors.

Many of these methods exploited intermediate reconstructions Cui & Tan (2015); Roberts et al. (2011), assumed specific probability distributions on missing correspondences Zach et al. (2008, 2010), involved multiple hyperparameters Heinly et al. (2014b); Jiang et al. (2012); Kataria et al. (2020); Shah et al. (2018); Shen et al. (2016); Wilson & Snavely (2013); Yan et al. (2017) or were designed for specific pipelines Kataria et al. (2020); Zach et al. (2008). Our method does not assume any probability distribution for missing correspondences, can be used in any pipeline, and uses only a single hyperparameter without requiring any intermediate reconstruction.

3 Proposed Method

Viewgraph sparsification aims to recover most of the cameras and 3D points with a sparser viewgraph in comparison to the original viewgraph while recovering overall 3D scene structure and maintaining reconstruction accuracy with reduced compute time. To handle visual ambiguities in the input images, it is required to identify repeated structures in the scene. We aim to solve both the problems of sparsifying viewgraphs and disambiguating repeated structures with a unified scheme. Our goal is to use as few hyperparameters as possible without the need for intermediate partial reconstructions. To achieve these objectives, we present a scheme to score edges based on camera triples and formulate an optimization problem whose solution can be obtained using a thresholding scheme. These lead to our proposed algorithm, which can be applied to viewgraphs without any restriction on their topology.

Scoring Edges in a Strong Triple: The importance of an edge in a graph cannot be validated by itself since it contains limited information about the pair of cameras it connects. Thus, a meaningful scoring of edges requires considering information from other edges connected via common nodes in the viewgraph. We take recourse to camera triples, which provide more information about an edge based on the other edges in a triple (Fig. 2). Camera triples with three edges connecting all pairs of cameras (or nodes), i.e. strong triples, have been widely used for sparsification Havlena et al. (2009, 2010); Shah et al. (2018); Shen et al. (2016); Snavely et al. (2008b) and disambiguation Jiang et al. (2012); Roberts et al. (2011); Zach et al. (2008, 2010).

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a viewgraph. $\mathcal{V}$ is the set of nodes representing cameras, and $\mathcal{E}$ is the set of edges representing epipolar inliers between camera pairs. Our objective is to assign a score, $q_{ij} > 0$, for each edge $(i, j) \in \mathcal{E}$, which is representative of its redundancy or it is a false edge. Since we aim to get a 3D reconstruction, to sparsify the viewgraph, we want to score the edges based on the connectivity via 3D points. Epipolar inliers on edges provide information about the 3D point connectivity without having access to 3D reconstruction since they are the candidate keypoints that will be used in the reconstruction process. One possible scoring mechanism would be to check for the fraction of common inliers in the edges connecting the strong triples and assign a score to each edge. However, such scoring can favour edges with a low number of inliers, which conflicts with our aim of scoring edges based on 3D point connectivity.

Therefore, to adhere to our aim of 3D point connectivity-based scores, we score edges in a strong triple using the number of inliers instead of common inliers across the three edges. As seen

in Fig 2, the indication for viewgraph sparsification and disambiguation in a strong triple is given by the relative number of inliers between the edges and not the actual number of inliers. Based on this idea, we score edges in a strong triple by taking the ratio of the number of inliers in each edge to the maximum number of inliers among all three edges in that triple. Given a strong triple $t = \{(i,j),(j,k),(i,k)\}$ with edges $(i,j),(j,k),(i,k) \in \mathcal{E}$ and their corresponding number of inliers n_{ij}, n_{jk} and n_{ik}, respectively, score for the edge (i,j) based on the strong triple t is computed as

$$q_{ij}^{t} = \frac{n_{ij}}{max(n_{ij}, n_{jk}, n_{ik})}. \tag{1}$$

Our edge scoring mechanism is based on the relative number of inliers in strong triples, which is intrinsically useful for disambiguation. It was observed in Cai et al. (2023) that the actual number of epipolar inliers does not provide useful information about false edges. In previous works, missing correspondences Jiang et al. (2012); Roberts et al. (2011); Yan et al. (2017); Zach et al. (2008) and the statistics of matched keypoints Yan et al. (2017) have been incorporated to detect the presence of false edges. Such false edges have been generally detected over loops of small lengths like a strong triple since the likelihood of observing the same 3D points on longer loops decreases. But utilizing small length loops for false edge detection requires the keypoints to be repeatable for the same 3D points across the three cameras, which can be violated in cases when images are noisy, occluded by other parts of the scene or a drastic change in viewpoint occurs Heinly et al. (2014a); Mikolajczyk & Schmid (2005); Zach et al. (2008). Hence, we choose to score edges based on the number of inliers instead of common inliers in strong triples. This decreases the effect of non-repeatable keypoints since there can be other matched keypoints on the edges, which will compensate for reduced edge scores due to unmatched keypoints. Such scoring makes sure that only keypoints, which are proven to have discriminative capabilities for matching, are used for inference instead of all the keypoints (as used in Cai et al. (2023); Roberts et al. (2011)).

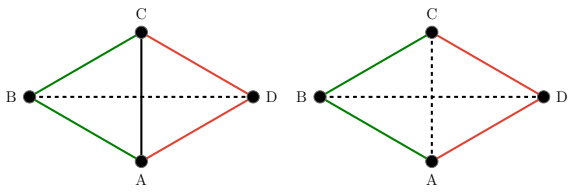
Scoring Edges in a Weak Triple: In our previous work Manam & Govindu (2024), strong triples were used to score edges which restricted the topology of viewgraphs where every edge must be a part of a strong triple. In general viewgraphs, there exist many edges which may not be a part of a strong triple. To enable scoring all edges, we also consider weak triples, i.e., three cameras connected by two edges. This definition of weak triples ensures that a camera triple cannot be a part of both sets of weak and strong triples. Thus, each set is exclusive of the other set.

The benefits of incorporating weak triples are three-fold. First, weak triples are more likely to occur in a viewgraph compared to strong triples for a given edge. Thus, more information about the relative number of epipolar inliers can be obtained via weak triples than by using only strong triples. Second, all edges in a connected viewgraph can be scored (with a minimum number of nodes in a viewgraph to be 3) because atleast one node of an edge will have another edge incident on it, resulting in a minimum of one weak or strong camera triple for the edge under consideration. Third, since all edges can be scored via both strong and weak camera triples, no preprocessing of viewgraphs is required, unlike Manam & Govindu (2024), which requires extraction of viewgraph containing camera triples with all three edges.

To utilize the scoring mechanism for strong triples, we consider a virtual edge in a weak triple between the two cameras, which are not connected by an edge, with 0 epipolar inliers. This converts a weak triple into a strong triple, and we can continue using the same scoring mechanism described for a strong triple in Eqn. 1. Given a weak triple $t = \{(i,j),(j,k)\}$ with edges $(i,j),(j,k) \in \mathcal{E}$ and their corresponding number of inliers as n_{ij} and n_{jk}, respectively, we add a virtual edge (i,k) with $n_{ik} = 0$. Now, score for the edge (i,j) based on the weak triple t is computed as

$$q_{ij}^{t} = \frac{n_{ij}}{max(n_{ij}, n_{jk}, n_{ik})} = \frac{n_{ij}}{max(n_{ij}, n_{jk})}. \tag{2}$$

In practice, we do not explicitly add virtual edges in weak triples since the edge scores in a weak



(a) Example viewgraph consisting of both triples (b) Example viewgraph consisting of weak triples

Fig. 3: Illustrative comparison of using only strong triples and using both weak and strong triples. Dashed edges are virtual edges to depict the presence of weak triples. (a) Scoring with strong triples considers $\{ABC, ADC\}$, whereas scoring with both weak and strong triples considers $\{ABC, ADC, ABD, CBD\}$. (b) The viewgraph cannot be scored with strong triples, and considering weak triples $\{ABC, ADC, ABD, CBD\}$ scores the edges

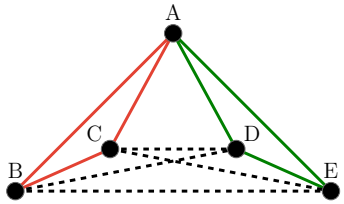


Fig. 4: Illustration of scoring a joint formed by ABC and ADE using our method. Dashed edges are not present in the input viewgraph and are shown only to represent weak triples. Strong triples, $\{ABC, ADE\}$, and weak triples, $\{ABD, ABE, ACD, ACE\}$, are used for scoring edges in our method. Scores for the *red* edges are dependent on the *green* edges and vice versa

triple can be computed using Eqn. 2.

Fig. 3 provides two example viewgraphs to illustrate the difference between using only strong triples and both weak and strong triples. We denote virtual edges with dashed lines. For the example in Fig. 3a, $\{ABC, ADC\}$ form strong triples and $\{ABD, CBD\}$ form weak triples. Manam & Govindu (2024) considers only strong triples, while our method considers both strong and weak triples, thus utilizing more information for scoring edges. Fig. 3b shows another example where there are no strong triples, and hence Manam & Govindu (2024) will not be able to score edges, but considering weak triples, our method can score all edges.

When two strong triples are connnected by a node with no common edge, then the edges in the two strong triples are scored independently. Such a connection between two strong triples is called a *joint* in Manam & Govindu (2024). In Fig. 4, we provide an example of a joint between two strong triples formed by red and green edges. Joints were separated in Manam & Govindu (2024) because it is not possible to infer via strong triples whether they are valid connections, which involved further preprocessing of graphs. By considering weak triples $\{ABD, ABE, ACD, ACE\}$ in Fig. 4, scores on the edges across the joint are dependent on each other. Thus, considering weak triples along with the strong triples allows us to score edges which are not a part of strong triple (as seen in Fig. 3b) as well as scores edges in a joint which are dependent on all other edges in that joint (as seen in Fig. 4).

Extracting Triples: Our method requires extracting weak and strong triples in $\mathcal{G}$, which is a combinatorial problem. In Manam & Govindu (2024), strong triples are explicitly extracted first, and then edge scoring is parallelized over the triples. Instead, we parallelize over the edges by using independent edge operations. Since the number of edges ($O(|\mathcal{V}|^2)$) is much smaller than the number of triples ($O(|\mathcal{V}|^3)$), with $|\mathcal{V}|$ being the number of nodes, our method is time-efficient compared to Manam & Govindu (2024).

For an edge $(i, j) \in \mathcal{E}$, we check the neighbours of i and j based on the connectivity in the graph, denoted as $\mathcal{N}(i)$ and $\mathcal{N}(j)$, respectively. The neighbours common to both the nodes form the set of strong triples, $ST(i, j) = \mathcal{N}(i) \cap \mathcal{N}(j)$, while the remaining neighbours form the set of weak triples, $WT(i, j) = (\mathcal{N}(i) \cup \mathcal{N}(j)) \setminus (\mathcal{N}(i) \cap \mathcal{N}(j))$. It can be seen that $ST(i, j) \cap WT(i, j) = \phi$, which is also implied from the definition of strong and weak triples. Now, for every $t \in ST(i, j) \cup WT(i, j)$, we compute the edge scores q_{ij}^t.

Aggregating Edge Scores: To obtain a final score for an edge $(i, j) \in \mathcal{E}$, denoted as q_{ij}, from

the scores obtained from weak and strong triples, $q_{ij}^t, t \in ST(i,j) \cup WT(i,j)$, we take the average of scores obtained from both types of triples. The edge scores q_{ij} can be written as

$$q_{ij} = \frac{\sum_{t \in ST(i,j) \cup WT(i,j)} q_{ij}^t}{|ST(i,j) \cup WT(i,j)|}. \tag{3}$$

This aggregates the global information about an edge from the whole viewgraph via weak and strong camera triples.

Edge Selection: Given edge scores q_{ij} for all edges, we follow the approach presented in Manam & Govindu (2024). We aim to select those edges which will increase the average edge score taken across all the edges while penalizing the loss of edges to avoid a degenerate solution of selecting edges only with the highest score (which could be as low as one edge in realistic scenarios). Let $s_{ij} \in \{0,1\}$ be the binary variables denoting selection ($s_{ij} = 1$) or removal ($s_{ij} = 0$) of edges. Then, the optimization problem can be formulated as

$$\max_{\substack{s_{ij} \in \{0,1\}, \\ (i,j) \in \mathcal{E}}} \frac{\sum_{(i,j) \in \mathcal{E}} s_{ij} q_{ij}}{\sum_{(i,j) \in \mathcal{E}} s_{ij}} - \lambda \frac{\sum_{(i,j) \in \mathcal{E}} (1 - s_{ij}) q_{ij}}{\sum_{(i,j) \in \mathcal{E}} (1 - s_{ij})}, \tag{4}$$

where $\lambda \geq 0$ is a regularization parameter, where reducing λ increases the number of edges removed. The second term in Eqn. 4 suggests the average of edge scores of the removed edges should not be high, thus regularizing the first term. The strict positiveness of q_{ij} ensures that edges are either selected or removed for any given $\lambda \geq 0$.

Although the problem in Eqn. 4 does not enforce the topology of the viewgraph, the scores obtained for edges are based on all the triples they are members of. Eqn. 3 reveals that Eqn. 4 is dependent on the graph topology. While the optimization problem in Eqn. 4 is combinatorial, our previous work Manam & Govindu (2024) provided a simple thresholding scheme to obtain its optimum, which is shown as follows:

$$s_{ij} = \begin{cases} 1 & \text{if } q_{ij} \geq \tau, \\ 0 & \text{otherwise,} \end{cases} \tag{5}$$

where τ denotes a threshold. This thresholding scheme is equivalent to optimizing the problem in Eqn. 4 with τ acting as a regularizer, as shown by the following theorem, presented in Manam & Govindu (2024):

Theorem 1. *For a given λ in Eqn. 4, there exists a threshold τ such that the values of s_{ij} obtained by solving the problems given by Eqn. 4 and Eqn. 5 are the same.*

In practice, finding a threshold is not easy for a given dataset, and a single threshold is not suitable for all datasets. We choose a threshold adaptively based on the connectivity of the graph. Specifically, given maximum node degree of the graph as d_{max}, we get a threshold as

$$\tau = m \left(1 - \frac{d_{max}}{|\mathcal{V}|}\right) + \left(\frac{d_{max}}{|\mathcal{V}|}\right), \tag{6}$$

where m is the minimum score which edges should satisfy. We deliberately choose the denominator as $|\mathcal{V}|$ since for a connected graph, $0 \leq d_{max} \leq |\mathcal{V}| - 1$, thus $m \leq \tau < 1$. This ensures that $\tau = 1$ is never chosen, which only retains edges scored exactly as 1. Such a scenario with many edges scored as 1 is seldom true in practice. We now summarize our method in Algo. 1. We note that steps 2-10 in Algo. 1 are performed using CPU parallelization (20 threads) and vectorized operations to reduce computation time.

4 Experiments

In this section, we present experimental results on publicly available generic and ambiguous datasets in individual subsections. We use COLMAP Schonberger & Frahm (2016) to obtain the original viewgraph and also use it to get 3D reconstructions under different scenarios. All experiments are performed on a PC with an Intel Xeon Silver 4210 processor with 128 GB RAM and two RTX 2080Ti GPUs. Our code is implemented in MATLAB and does not require any GPU[1]. Our proposed method and the method in Manam & Govindu (2024) use CPU parallelization with 20 threads. We use the methods in Chatterjee & Govindu (2017) and Wilson & Snavely (2014) to align camera motions obtained in reconstructions

[1]GPUs are used only for Doppelgangers Cai et al. (2023) and COLMAP Schonberger & Frahm (2016).

Algorithm 1 Edge Selection for Viewgraphs

Input: Viewgraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, with geometrically verified edges and a minimum edge score (m).
Output: Sparsified viewgraph $\mathcal{G}_{FG} = (\mathcal{V}_{FG}, \mathcal{E}_{FG})$ with ambiguous edges removed.

```
Extract the largest connected component of G.
for each edge (i, j) ∈ E do
    Check neighbours of nodes i and j as N(i) and N(j), respectively.
    Get lists of strong ST(i, j) and weak triples WT(i, j) from N(i) and N(j).
    for each triple t in ST(i, j) ∪ WT(i, j) do
        Get the number of inliers for all edges as n_ij, (i, j) ∈ t.
        Assign q_ij^t using Eqn. 1 or Eqn. 2.
    end for
    Assign q_ij to each edge using Eqn. 3.
end for
Compute τ from m using Eqn. 6.
Remove edges in G with q_ij < τ giving G_FG.
Extract the largest connected component of G_FG.
```

to the known reference, and then compute errors. We only consider the largest connected component of the viewgraphs for reporting values. We state the **notation** below for further usage:

- $\mathcal{G}$: Original viewgraph obtained using COLMAP Schonberger & Frahm (2016).
- $\mathcal{G}_{\#in}$: Graph obtained after thresholding number of inliers in $\mathcal{G}$ (threshold set to 150 as in Cai et al. (2023)).
- $\mathcal{G}_{Dopp}$: Graph obtained after applying Doppelgangers Cai et al. (2023) on $\mathcal{G}$.
- $\mathcal{G}_{FG}(m)$: Graph obtained after applying Algo. 1 with a minimum edge score m.
- $\#N_{CR}$: Number of cameras reconstructed.
- t_R: Reconstruction time using COLMAP Schonberger & Frahm (2016).
- RRE: Recall (in %) of camera rotation errors at 5°.
- RTE: Recall (in %) of camera translation errors at 10 meters for datasets with ground truth available, otherwise 10 units, where unit is defined by reference.

Finally, in the tables, **bold** values indicate best in the respective category.

4.1 Generic Datasets

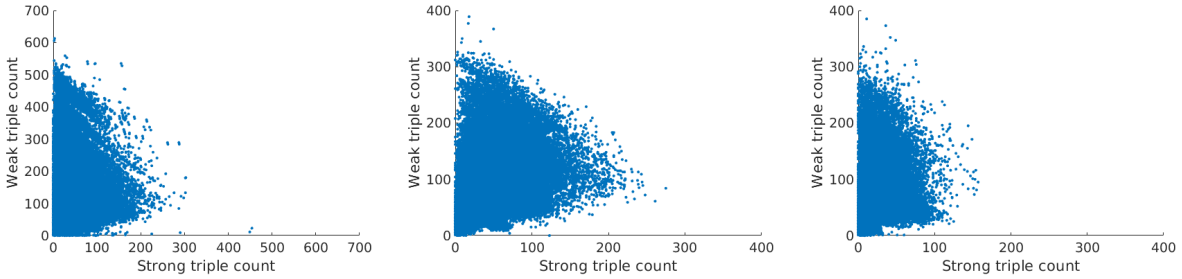
In this subsection, we first provide a list of generic datasets from Crandall et al. (2011); Howard et al. (2022); Li et al. (2010); Schops et al. (2017); Wilson & Snavely (2014) and their abbreviations in Table 2, which will be used further. Then, we analyze the datasets in terms of the presence of weak and strong triples that influence our edge scoring mechanism. Subsequently, we provide reconstruction results on these datasets. Finally, we compare using both strong and weak triples to that of using only strong triples Manam & Govindu (2024) in terms of reconstruction statistics. We note that we did not report results for the Pipes dataset from Schops et al. (2017) as no reconstructions were obtained from COLMAP.

Analysis of Generic Datasets: Fig. 5 provides scatter plots between the number of weak and strong triples (both types of triples being exclusive) formed by viewgraph edges on three generic datasets with different edge densities. It can be seen that, for most edges, the number of weak triples is generally higher than the number of strong triples. This suggests that the contribution of weak triples to score an edge is generally more than that of strong triples. An edge is more likely to be scored higher in a weak triple than a strong triple. Thus, the overall scores of edges will be higher when both strong and weak triples are considered compared to using only strong triples (as in Manam & Govindu (2024)) and will require a higher threshold for a similar fraction of edges to be removed.

Reconstruction Results: Here, we provide results for viewgraph sparsification. We apply Algo. 1 on generic datasets to sparsify viewgraphs. We use the minimum edge score $m = 0.7$ for large-scale datasets (Crandall et al. (2011); Li et al. (2010); Wilson & Snavely (2014)) and $m = 0.3$ for medium (Howard et al. (2022)) and small-scale (Schops et al. (2017)) datasets. In Table 3, we compare results of the original viewgraphs $\mathcal{G}$ and the sparsified viewgraphs $\mathcal{G}_{FG}$. It can be seen that for large-scale datasets, most of the cameras and 3D points are reconstructed after sparsifying the viewgraphs ($\mathcal{G}_{FG}$) with our method compared to the original viewgraphs $\mathcal{G}$, with atleast 30% reduction in reconstruction time

Table 2: List of generic datasets and their abbreviations

Dataset	Abbreviation	Dataset	Abbreviation	Dataset	Abbreviation
Wilson & Snavely (2014)		Howard et al. (2022)		Schops et al. (2017)	
Alamo	ALM	Brandenburg Gate	BDG	Courtyard	CTY
Gendarmenmarkt	GMM	British Museum	BRM	Delivery Area	DYA
Madrid Metropolis	MDR	Buckingham Palace	BKP	Electro	ELT
Montreal Notre Dame	MND	Colosseum Exterior	COE	Facade	FCD
Notre Dame	ND	Grand Place Brussels	GPB	Kicker	KKR
NYC Library	NYC	Lincoln Memorial Statue	LMS	Meadow	MDW
Piccadilly	PIC	Notre Dame Front Facade	NDF	Office	OFC
Roman Forum	ROF	Pantheon Exterior	PNE	Playground	PYG
Tower of London	TOL	Piazza San Marco	PSM	Relief	RLF
Trafalgar	TFG	Sacre Coeur	SCR	Relief 2	RLF2
Union Square	USQ	Sagrada Familia	SAF	Terrace	TRC
Vienna Cathedral	VNC	St Pauls Cathedral	SPC	Terrains	TRN
Li et al. (2010)		St Peters Square	SPS	Crandall et al. (2011)	
Dubrovnik	DVK	Taj Mahal	TAJ	Quad	QAD
Rome	ROM	Temple Nara Japan	TNJ		
		Trevi Fountain	TRF		



DVK ($ed = 0.92\%$) MND ($ed = 2.82\%$) ROF ($ed = 1.88\%$)

Fig. 5: Scatter plots of the number of weak triples vs strong triples at every edge on generic datasets. *ed*: edge density

on most datasets. Moreover, our method ($\mathcal{G}_{FG}$) reconstructs significantly more cameras and 3D points compared to thresholding edges based on the number of inliers ($\mathcal{G}_{\#in}$). This suggests that our method of scoring edges based on the relative proportion of inliers (Eqns. 1 and 2) is useful for viewgraph sparsification compared to an inlier thresholding scheme. The reprojection errors also decrease for the sparsified viewgraphs, indicating that estimated parameters (camera intrinsics, motions and 3D points) are more consistent with the observations (epipolar inliers) than the original graph.

We also provide recall values for camera motion errors obtained after applying our method. For large-scale datasets, we use COLMAP reconstructions as the reference. For datasets from Wilson & Snavely (2014), we align the COLMAP solution to the ground truth provided, to obtain the errors approximately in meters (as done in Manam & Govindu (2023)). From Table 3, it is observed that the number of cameras reconstructed is not similar across different methods. When the errors for cameras not reconstructed are considered infinity, the recall values favour the method with most cameras reconstructed. This is because the recall curve (or the cumulative error distribution curve) saturates at a lower value for

Table 3: Reconstruction statistics on generic datasets. ✓/✓*/ ✗: No ghost artifact/no ghost artifact but over-split reconstruction/contains ghost artifacts. -: No reconstruction obtained from COLMAP. -^: No camera motion errors computed as COLMAP reference or our method contains ghost artifacts. The best reprojection error is checked only across reconstructions with no ghost artifact/no ghost artifact but oversplit reconstructions. Our method ($\mathcal{G}_{FG}$) sparsifies the graphs, reconstructing most cameras and 3D points with reduced reprojection errors

Dataset	No Ghost Artifact			# Cameras ($\#N_{CR}$)			# 3D Points (in 10^3)			Reprojection Errors (px)			Reconstruction Time (mins) (t_R)			Cam. Motion Error Recall (%) of $\mathcal{G}_{FG}$	
	$\mathcal{G}$	$\mathcal{G}_{\#in}$	$\mathcal{G}_{FG}$	$\mathcal{G}$	$\mathcal{G}_{\#in}$	$\mathcal{G}_{FG}$	$\mathcal{G}$	$\mathcal{G}_{\#in}$	$\mathcal{G}_{FG}$	$\mathcal{G}$	$\mathcal{G}_{\#in}$	$\mathcal{G}_{FG}$	$\mathcal{G}$	$\mathcal{G}_{\#in}$	$\mathcal{G}_{FG}$	*RRE*	*RTE*
ALM	✓	✓	✓	899	685	770	161	139	143	0.66	0.66	**0.62**	44	25	28	100.0	99.7
GMM	✗	✓	✓	1042	889	957	207	171	177	0.76	**0.75**	**0.75**	372	189	211	-^	-^
MDR	✓	✓	✓	471	250	425	74	40	69	0.62	**0.59**	**0.59**	88	28	42	96.8	97.2
MND	✓	✓	✓	575	490	551	145	128	135	0.86	0.85	**0.83**	114	80	89	100.0	100.0
ND	✗	✓	✓	1407	1378	1353	348	337	338	0.76	0.75	**0.71**	2054	1746	1229	-^	-^
NYC	✓	✓	✓	635	408	555	110	91	100	0.74	0.72	**0.71**	105	48	55	99.7	100.0
PIC	✗	✓	✓	3208	2416	2923	374	296	338	0.76	**0.72**	0.74	1969	941	1026	-^	-^
ROF	✓	✓	✓	1587	1297	1480	324	290	306	0.76	0.75	**0.74**	784	449	508	99.8	99.9
TOL	✓	✓	✓	735	574	626	165	151	152	0.63	0.63	**0.62**	95	60	66	100.0	99.5
TFG	✓	✓	✓	7867	5506	7275	706	550	647	0.74	**0.73**	**0.73**	6875	3622	4855	99.5	99.1
USQ	✓	✓*	✓	1160	444	1027	82	39	74	0.74	**0.72**	0.73	220	74	147	99.5	98.6
VNC	✗	✓	✓	1197	1032	1090	304	274	279	0.75	0.75	**0.73**	793	546	558	-^	-^
DVK	✓	✓	✓	5883	5592	5671	1252	1162	1170	0.76	0.75	**0.74**	582	396	406	99.9	100.0
ROM	✓	✓	✓	1812	1812	1809	395	392	390	0.90	0.90	**0.88**	284	280	217	99.9	100.0
QAD	✓	✗	✓	5729	4092	4950	1295	1046	1151	0.69	**0.68**	**0.68**	417	272	290	96.5	100.0
BDG	✓	✓	✓	349	330	290	40	35	32	0.74	0.71	**0.56**	21	12	9	100.0	100.0
BRM	✓	✓	✓	176	173	75	30	29	11	0.75	0.74	**0.46**	8	8	2	100.0	100.0
BKP	✓	✓	✓	446	442	425	96	91	84	0.66	0.65	**0.54**	51	50	43	100.0	99.8
COE	✓	✓	✓	499	499	493	110	110	107	0.64	0.64	**0.63**	31	32	26	100.0	98.2
GPB	✓	✓	✓	235	231	226	79	76	75	**0.60**	**0.60**	0.78	18	15	12	100.0	97.7
LMS	✓	✓	✓	214	213	153	22	21	18	0.80	0.80	**0.60**	6	6	3	100.0	100.0
NDF	✓	✓	✓	910	903	893	194	190	188	0.69	0.69	**0.67**	124	115	96	100.0	100.0
PNE	✓	✓	✓	320	319	310	70	69	62	0.70	0.69	**0.60**	29	21	19	100.0	99.7
PSM	✗	✗	✗	65	66	55	24	24	17	1.06	1.06	0.84	7	7	5	-^	-^
SCR	✓	✓	✓	281	281	273	55	52	46	0.63	0.62	**0.54**	18	11	9	100.0	100.0
SAF	✓	✓	✓*	90	90	29	43	42	16	0.64	0.64	**0.46**	9	10	2	100.0	100.0
SPC	✓	✓	✓*	142	141	18	38	35	6	0.70	0.69	**0.45**	7	6	1	100.0	100.0
SPS	✓	✓	✓	622	618	584	93	89	84	0.68	0.67	**0.58**	53	42	36	100.0	98.6
TAJ	✓	✓	✓	399	391	335	46	44	37	0.67	0.65	**0.42**	61	47	23	100.0	99.7
TNJ	✓	✓	✓	217	213	111	35	34	16	0.67	0.67	**0.47**	17	17	6	100.0	96.4
TRF	✓	✓	✓	703	699	696	200	198	196	0.83	0.83	**0.82**	69	66	60	100.0	99.9
CTY	✓	✓	✓	38	38	27	21	21	16	1.25	1.26	**1.16**	2	2	1	100.0	100.0
DYA	✓	✓	✓	44	44	44	13	13	13	**1.20**	**1.20**	**1.20**	1	1	1	99.8	100.0
ELT	✓	✓	✓	38	20	32	8	5	7	1.15	**1.11**	1.15	2	2	2	100.0	100.0
FCD	✓	✓	✓	75	75	69	57	57	52	1.27	1.28	**1.22**	7	6	4	100.0	100.0
KKR	✓	✓	✓	30	25	23	7	7	6	1.23	1.21	**1.20**	1	1	1	100.0	100.0
MDW	✓	-	✓*	12	-	6	0.5	-	0.4	1.18	-	**1.08**	1	1	1	100.0	100.0
OFC	✓	✓	✓	20	11	14	1	0.7	0.8	1.25	**1.21**	1.27	1	1	1	100.0	100.0
PYG	✓	✓	✓	18	19	19	4	4	4	1.21	1.21	**1.18**	1	1	1	100.0	100.0
RLF	✓	✓	✓	31	31	20	15	15	14	1.03	**1.02**	0.95	2	2	1	100.0	100.0
RLF2	✓	✓	✓	31	31	31	9	9	9	1.05	1.06	**1.02**	1	1	1	100.0	100.0
TRC	✓	✓	✓	23	23	23	5	5	5	1.30	1.30	**1.27**	1	1	1	100.0	100.0
TRN	✓	✓*	✓*	42	24	31	8	6	6	1.23	1.23	**1.22**	2	1	1	88.0	100.0

the method having fewer reconstructed cameras, compared to other methods. For viewgraph sparsification, the aim is to maintain the reconstruction accuracy and overall scene structure even with a reduced number of reconstructed cameras, which is not entirely captured when cameras not reconstructed are considered to have infinite error. To have a fair comparison of the methods in the context of viewgraph sparsification, we only compute recall values with common reconstructed cameras across all methods. We observe from Table 3 that the recall for camera motion errors is high, indicating that the reconstruction accuracy is

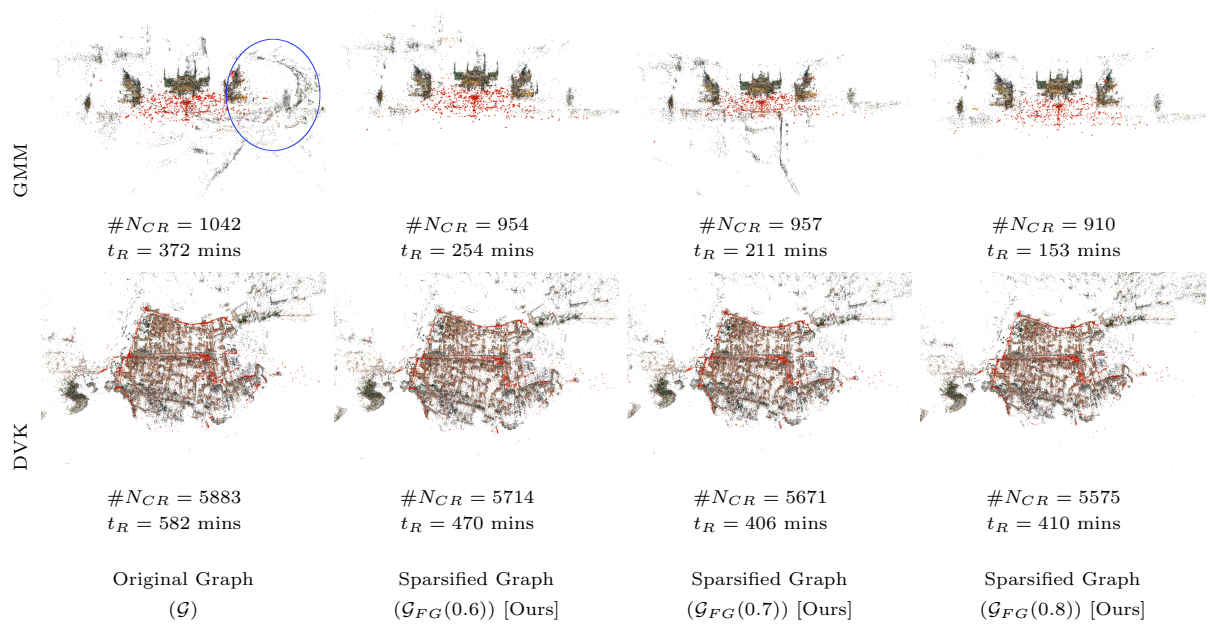


Fig. 6: Reconstructions obtained on generic datasets after applying our method. Top row: Dataset creating ghost artifacts in reconstructions (marked in *blue*). Bottom row: Dataset with redundant edges. Applying our method removes ghost artifacts (top row) and maintains reconstruction quality (bottom row) in reduced reconstruction time. We note that the extra wall in GMM reconstruction with $\mathcal{G}_{FG}(0.7)$ is not a ghosting artifact

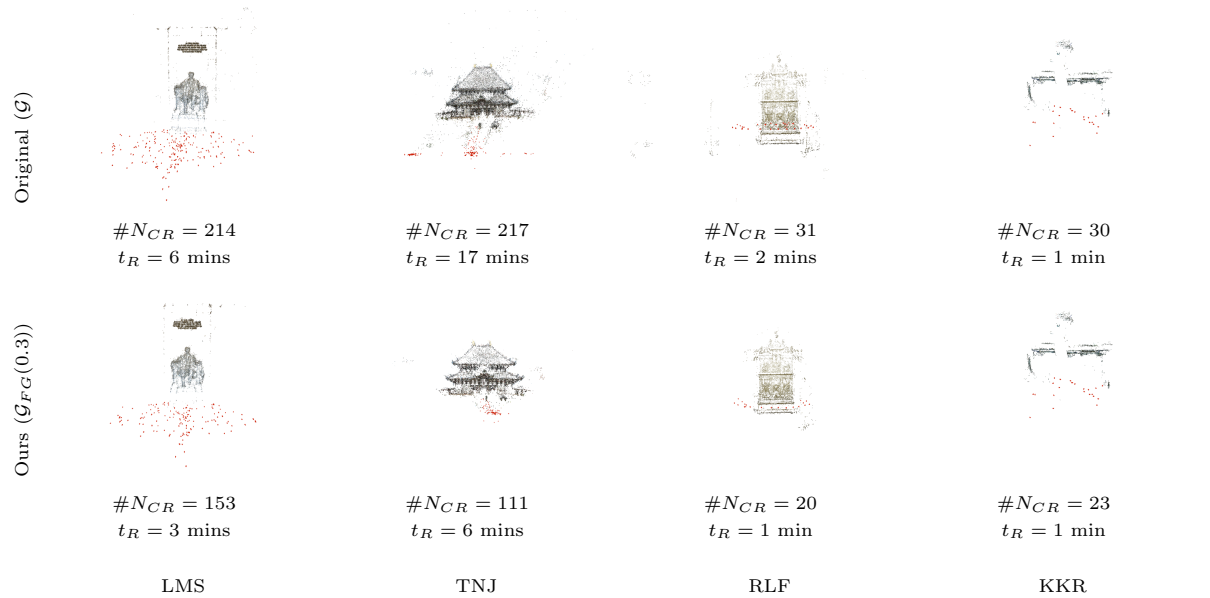


Fig. 7: Reconstructions obtained on generic datasets after applying our method. These are some examples of the datasets where overall scene structure is recovered even with less reconstructed cameras

maintained. We also compare recall values of different methods in Table S2 of the supplementary material. In Fig. 6, we provide reconstructions on GMM, which contains a ghost artifact, and DVK,

Table 4: Comparison of our method with Manam & Govindu (2024) on generic datasets. ✓/✓*/ ✗: No ghost artifact/no ghost artifact but over-split reconstruction/contains ghost artifacts. ST and WT indicate the usage of strong and weak triples, respectively. Using both strong and weak triples (our method [ST+WT]) reconstructs more cameras and 3D points compared to using only strong triples Manam & Govindu (2024) [ST]. Our method also takes less time compared to Manam & Govindu (2024) due to efficient and parallelizable Algo 1

Dataset	No Ghost Artifact				# Cameras ($\#N_{CR}$)				# 3D Points (in 10^3)				Time Taken (sec)	
	$m = 0.6$		$m = 0.7$		$m = 0.6$		$m = 0.7$		$m = 0.6$		$m = 0.7$			
Triples →	ST	ST+WT	ST	ST+WT	ST	ST+WT	ST	ST+WT	ST	ST+WT	ST	ST+WT	ST	ST+WT
ALM	✓	✓	✓	✓	652	**839**	612	**770**	128	**151**	126	**143**	23	**12**
GMM	✓	✓	✓	✓	899	**954**	857	**957**	160	**178**	151	**177**	25	**16**
MDR	✓	✓	✓	✓	274	**447**	232	**425**	47	**71**	33	**69**	4	**3**
MND	✓	✓	✓	✓	411	**557**	398	**551**	108	**139**	103	**135**	33	**25**
ND	✓	✓	✓	✓	1295	**1383**	1093	**1353**	332	**343**	288	**338**	**4**	6
NYC	✓	✓	✓	✓	509	**573**	396	**555**	93	**103**	84	**100**	238	**99**
PIC	✓	✗	✓	✓	**2786**	2993	2645	**2923**	**311**	349	288	**338**	170	**161**
ROF	✓	✓	✓	✓	1365	**1522**	1295	**1480**	278	**313**	263	**306**	**14**	15
TOL	✓	✓	✓	✓	535	**665**	495	**626**	137	**157**	129	**152**	4	**3**
TFG	✓	✓	✓	✓	6538	**7461**	6256	**7275**	573	**666**	545	**647**	844	**787**
USQ	✓	✓	✓	✓	891	**1078**	805	**1027**	62	**81**	55	**74**	23	**15**
VNC	✓	✓	✓	✓	1010	**1125**	965	**1090**	258	**285**	248	**279**	94	**57**
DVK	✓	✓	✓	✓	5576	**5714**	5464	**5671**	1127	**1195**	1073	**1170**	**405**	550
ROM	✓	✓	✓	✓	1807	**1810**	1728	**1809**	389	**392**	389	**390**	**9478**	11414
QAD	✓	✓	✓	✓	4360	**5277**	3843	**4950**	1041	**1205**	882	**1151**	152	**143**

which gives a correct reconstruction. The reconstructions obtained with sparsified viewgraphs $\mathcal{G}_{FG}$ result in the removal of ghosting artifact in GMM and visually similar reconstructions for DVK compared to the original graph, with most cameras reconstructed.

For medium and small-scale datasets, the number of cameras and 3D points reconstructed and the reprojection errors have a similar trend as observed for the large-scale ones, which can be seen in Table 3. Although fewer cameras and 3D points are reconstructed for BRM, LMS, TNJ, RFL and KKR, we observe that sparsifed viewgraphs recover the overall scene structure, as seen in Fig. 7. For SAF, SPC, MDW and TRN datasets, the reconstructions are over-split due to low redundancy in the viewgraphs. For PSM, the scene consists of similar texture due to which false edges are present, and thus, none of the methods obtains a correct reconstruction. Our method leads to atleast 20% reduction in reconstruction time for medium-scale datasets, while for small-scale datasets, the gain is not significant. We also note that recall values of camera motion errors are high with ground truth provided in the datasets as the reference, suggesting that the reconstruction accuracy is maintained. A comparison with different methods in provided in Table S2 of the supplementary material. This suggests that viewgraph sparsification is beneficial for large and medium-scale datasets. These show that the goals of viewgraph sparsification are achieved, which is to recover most cameras and 3D points with overall 3D scene structure recovered and reconstruction accuracy maintained while reducing the reconstruction time. Additional visual results are provided in the supplementary material.

Comparison with using only Strong Triples Manam & Govindu (2024)**:** We compare our method, which uses both strong and weak triples, to that of the method proposed in Manam & Govindu (2024), which utilizes only strong triples. In Table 4, we compare both methods for the minimum edge scores $m = \{0.6, 0.7\}$. We observe that using both strong and weak triples reconstructs significantly more cameras and 3D points for most datasets when compared to using only strong triples for a given m. We report the reconstruction time and reprojection errors in Table S3 of the supplementary material, which suggests that reconstructing more cameras results in more reconstruction time than Manam & Govindu (2024) while keeping the reprojection errors similar. Moreover, a higher minimum edge score m is required for the removal of the ghosting artifact in PIC when using both strong and weak triples compared to using only strong

triples. This is expected since weak triples are larger in number than strong triples, resulting in higher edge scores. Thus, a higher threshold is required to remove a considerable number of edges, including the false edges.

We also report the time taken for our method (Algo. 1) and compare it with the method presented in Manam & Govindu (2024) in Table 4. It can be seen that our method takes less time than that of Manam & Govindu (2024) on most datasets. This is because, unlike Manam & Govindu (2024), our method does not require the extraction of strong triples from viewgraphs as a preprocessing step. The ROM Li et al. (2010) dataset is well-connected, due to which the triples are large in number, resulting in a large computation time for both methods.

Additionally, it can be seen that our method takes less than 2% of the reconstruction time with the original graph ($\mathcal{G}$) on most large-scale datasets, which is reported in minutes in Table 3. This shows the benefits of using our method, which has a short compute time and yields significant time savings in the reconstruction process for large-scale datasets.

Table 5: List of ambiguous datasets and their abbreviations

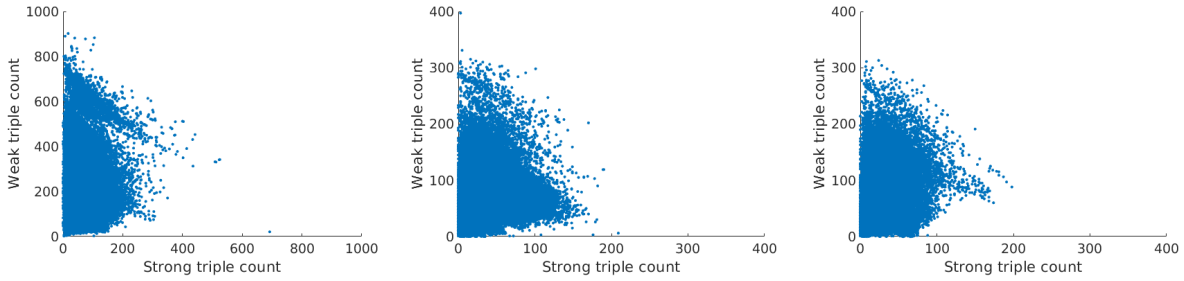
Dataset	Abbreviation
Wilson & Snavely (2013)	
Louvre	LVE
Notre Dame	NDE
Sacre Coeur	SCO
Seville	SEV
Wilson & Snavely (2014)	
Ellis Island	ELS
Piazza del Popolo	PDP
Yorkminster	YKM
Heinly et al. (2014a)	
Alexander Nevsky Cathedral	ANC
Arc de Triomphe	ADT
Berliner Dom	BLD
Big Ben	BBN
Brandenburg Gate	BRG
Church on Spilled Blood	CSB
Indoor	IDR
Radcliffe Camera	RAC
Yan et al. (2017)	
Books	BOK
Cereal	CER
Cup	CUP
Desk	DSK
Oats	OTS
Street	STR
Temple of Heaven	TOH

4.2 Ambiguous Datasets

In this subsection, we present results on ambiguous datasets Heinly et al. (2014a); Wilson & Snavely (2013, 2014); Yan et al. (2017). We provide a list of ambiguous datasets along with their abbreviations in Table 5. Similar to Sec. 4.1, we analyze the presence of weak and strong triples, which are used in scoring edges. We also analyze the distribution of edge scores on non-ambiguous and ambiguous edges. Then, we provide reconstruction results and compare using both strong and weak triples, used by our method, to that of using only strong triples, used in Manam & Govindu (2024).

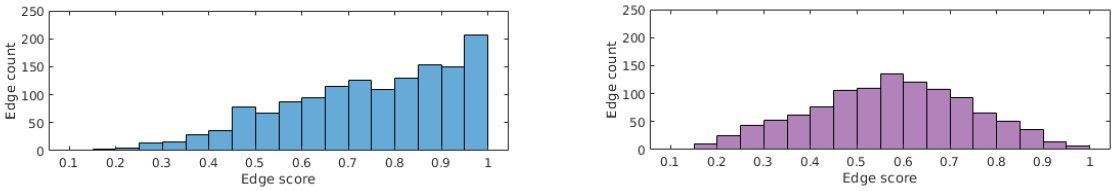
Analysis of Ambiguous Datasets: Fig. 8 provides scatter plots between the number of weak and strong triples formed by viewgraph edges on three ambiguous datasets with different edge densities. We observe that the number of weak triples an edge contributes to is generally higher than the number of strong triples. Thus, the overall scores of edges will be higher when our method is considered, which uses both strong and weak triples, compared to the method in Manam & Govindu (2024), which uses only strong triples. Thus, our method will require a higher threshold for false edges to be removed.

Analysis of Edge Scores: We study how well the edge scores reflect the presence of redundant and false edges with our method. Similar to Manam & Govindu (2024), we compute edge scores on Doppelgangers Cai et al. (2023) datasets and provide histograms in Fig. 9. We observe that for non-ambiguous edges, the number of edges increases with increasing edge score, while for ambiguous edges, most of the edges are scored between 0.5 to 0.7. Thus, choosing a high threshold removes many redundant edges and most of the false edges. Moreover, our method uses both strong and weak triples, due to which



BLD ($ed = 7.05\%$) LVE ($ed = 0.79\%$) YKM ($ed = 1.26\%$)

Fig. 8: Scatter plots of the number of weak triples vs strong triples at every edge on ambiguous datasets. *ed*: edge density



Non-ambiguous Edges Ambiguous Edges

Fig. 9: Histograms of edge scores with our method. Scores are computed over all datasets from Doppelgangers Cai et al. (2023). For non-ambiguous edges, the number of edges increases with increasing edge scores. For ambiguous edges, most of the edges are scored between 0.5 to 0.7. Choosing a high threshold removes many redundant edges and most of the false edges

the distribution of scores for both ambiguous and non-ambiguous edges have shifted towards the right compared to using only strong triples (see Fig. 4 in Manam & Govindu (2024)). This is because edges are more likely to be scored higher in a weak triple than in a strong triple, and the number of weak triples is generally higher than the strong triples for an edge (as seen in Figs. 5 and 8), which results in shifting both distributions towards right.

Reconstruction Results: Here, we present reconstruction results on ambiguous datasets. In Table 6, we provide reconstruction statistics on ambiguous datasets. We use $m = 0.7$ for large-scale datasets Wilson & Snavely (2013, 2014), the same as used for generic datasets, except for SCO, SEV and ELS, where we use $m = 0.9$. We use $m = 0.5$ for medium and small-scale datasets Heinly et al. (2014a); Yan et al. (2017). We also provide reconstruction statistics after applying Doppelgangers Cai et al. (2023) on $\mathcal{G}$, with probability threshold $p = 0.8$ for all datasets (as suggested in Cai et al. (2023)) except for LVE, where $p = 0.9$. No value of p worked for the datasets not disambiguated by Cai et al. (2023). It can be seen that thresholding the number of inliers ($\mathcal{G}_{\#in}$) does not yield disambiguated reconstructions for most datasets. Thus, we do not report its reconstruction statistics. Our method ($\mathcal{G}_{FG}$) is able to disambiguate repeated structures for all large, medium, and small-scale datasets, showing the importance of scoring edges based on the relative proportion of inliers (Eqns. 1 and 2). Moreover, a similar number of cameras and 3D points are reconstructed with our method for many datasets when compared to Doppelgangers Cai et al. (2023) ($\mathcal{G}_{Dopp}$). We also compare camera motions of common reconstructed cameras (as done for generic datasets), keeping $\mathcal{G}_{Dopp}$ as reference, and observe that its recall values are high. This shows that our

Table 6: Reconstruction statistics on ambiguous datasets with our method. ✓/✓*/✗: Disambiguated/Disambiguated but oversplit/Non-disambiguated reconstructions. -ˆ: No camera motion errors computed as Doppelgangers Cai et al. (2023) reference is not disambiguated. -#: No camera motion errors computed as $\mathcal{G}_{Dopp}$ and $\mathcal{G}_{FG}$ do not have common cameras. The best value is checked only across disambiguated/disambiguated but oversplit reconstructions. Our method ($\mathcal{G}_{FG}$) disambiguates all datasets and also sparsifies them, reconstructing most cameras and 3D points with reduced reprojection errors

Dataset	Disambiguated				# Cameras ($\#N_{CR}$)			# 3D Points (in 10^3)			Reprojection Errors (px)			Cam. Motion Error Recall (%) of $\mathcal{G}_{FG}$	
	$\mathcal{G}$	$\mathcal{G}_{\#in}$	$\mathcal{G}_{Dopp}$	$\mathcal{G}_{FG}$	$\mathcal{G}$	$\mathcal{G}_{Dopp}$	$\mathcal{G}_{FG}$	$\mathcal{G}$	$\mathcal{G}_{Dopp}$	$\mathcal{G}_{FG}$	$\mathcal{G}$	$\mathcal{G}_{Dopp}$	$\mathcal{G}_{FG}$	*RRE*	*RTE*
LVE	✓	✓	✓	✓	367	320	360	128	108	121	0.60	**0.60**	**0.60**	99.6	100.0
NDE	✗	✗	✓	✓	7952	5481	5640	1785	1314	1357	0.73	0.74	**0.69**	99.9	99.9
SCO	✗	✗	✓	✓	4492	3796	2206	711	548	366	0.70	0.71	**0.69**	98.6	99.9
SEV	✗	✗	✓	✓*	1510	447	239	353	109	72	0.68	0.64	**0.62**	99.5	100.0
ELS	✗	✓	✓	✓	880	314	333	164	84	92	0.76	0.80	**0.73**	98.9	98.9
PDP	✗	✓*	✓	✓	1023	922	947	138	123	124	0.69	0.68	**0.67**	100.0	100.0
YKM	✗	✓	✓	✓	1065	585	567	284	173	170	0.75	0.77	**0.76**	100.0	99.8
ANC	✗	✗	✓	✓	447	445	434	100	90	85	0.67	0.68	**0.64**	100.0	100.0
ADT	✗	✓	✓	✓	427	392	422	81	69	75	0.66	0.66	**0.65**	99.1	100.0
BLD	✗	✓	✓	✓	1603	1600	1596	242	238	236	0.70	0.70	**0.68**	100.0	100.0
BBN	✗	✗	✓	✓	397	394	387	74	73	69	0.64	0.64	**0.62**	99.2	100.0
BRG	✗	✓*	✓*	✓*	172	151	137	24	21	14	0.84	0.87	**0.72**	100.0	100.0
CSB	✗	✗	✗	✓	273	258	140	69	64	30	0.61	0.61	**0.53**	-ˆ	-ˆ
IDR	✗	✓	✓	✓	152	152	42	73	58	9	0.52	0.53	**0.40**	100.0	100.0
RAC	✗	✗	✓*	✓*	279	94	125	76	28	29	0.64	**0.60**	0.61	-#	-#
BOK	✗	✗	✗	✓*	21	21	7	8	7	2	0.41	0.41	**0.31**	-ˆ	-ˆ
CER	✗	✗	✗	✓*	25	25	7	12	12	3	0.41	0.41	**0.27**	-ˆ	-ˆ
CUP	✗	✗	✓	✓*	64	63	40	6	6	3	0.61	0.39	**0.37**	100.0	100.0
DSK	✗	✗	✓	✓*	31	31	11	14	12	3	0.49	0.48	**0.39**	100.0	100.0
OTS	✗	✗	✗	✓*	23	23	9	8	7	3	0.40	0.37	**0.24**	-ˆ	-ˆ
STR	✗	✗	✓*	✓*	19	10	10	4	1	1	0.50	0.37	**0.33**	100.0	100.0
TOH	✓	✓	✓	✓	338	338	338	192	185	197	0.65	**0.63**	0.65	99.7	100.0

method is able to maintain the reconstruction accuracy similar to the current best method for removing only false edges. A comparison of recall values of different methods is provided in Table S5 of the supplementary material. Our method also reduces reconstruction time due to its sparsification behaviour, details of which are provided in Table S6 of the supplementary material. For small-scale datasets Yan et al. (2017), scenes are over-split after applying our method. This is because our method is dependent on the redundancy of edges for scoring them, which is low for small-scale datasets, thus leading to over-split reconstructions. More details about the datasets and the impact of different choices of m are provided in the supplementary material.

In Table 7, we compare our method with other disambiguation methods Cai et al. (2023); Cui & Tan (2015); Heinly et al. (2014a); Wilson & Snavely (2013); Yan et al. (2017) by following the experimental procedure in Cai et al. (2023). COLMAP Schonberger & Frahm (2016) fails to disambiguate on all datasets except LVE Wilson & Snavely (2013) and TOH Yan et al. (2017). Other methods are able to disambiguate many datasets, with our method performing the best. In Fig. 10, we show reconstructions of two ambiguous datasets. The original viewgraphs $\mathcal{G}$ result in superimposed reconstructions. Applying Doppelgangers Cai et al. (2023), the method with only strong triples Manam & Govindu (2024), or our method with both strong and weak triples leads to correct reconstructions. Our method reconstructs more cameras with a similar amount of time taken by Manam & Govindu (2024). Applying our method or Manam & Govindu (2024) leads to faster reconstruction time than Doppelgangers because our method sparsifies the viewgraphs along with removing false edges. Time taken for different disambiguation methods is provided in Table S7 of the supplementary material, along

Table 7: Comparison of our method with other disambiguation methods on ambiguous datasets. ✓/✓*/✗: Disambiguated/Disambiguated but oversplit/Non-disambiguated reconstructions. Results of Heinly et al. (2014a) are shown as reported in Cai et al. (2023). '-': Results not reported in Cai et al. (2023). '-*': Code failed to execute

Method	Label
Schonberger & Frahm (2016)	COLMAP
Heinly et al. (2014a)	Heinly
Wilson & Snavely (2013)	Wilson
Cui & Tan (2015)	Cui
Yan et al. (2017)	Yan
Cai et al. (2023)	Cai
Manam & Govindu (2024)	Manam

Dataset	Methods							
	COLMAP	Heinly	Wilson	Cui	Yan	Cai	Manam	Ours
LVE	✓	-	✗	✓	✓	✓	✓	✓
NDE	✗	-	✗	✓	-*	✓	✓	✓
SCO	✗	✓	✓	✓	✓	✓	✓	✓
SEV	✗	✗	✓*	✓*	✓*	✓	✓	✓*
ELS	✗	-	✓	✓*	✓	✓	✓	✓
PDP	✗	-	✗	✓*	✓*	✓	✓	✓
YKM	✗	-	✓	✓	✓	✓	✓	✓
ANC	✗	✓	✗	✓	✓	✓	✓	✓
ADT	✗	✓	✗	✗	✓	✓	✓	✓
BLD	✗	✓	✓*	✓	✓	✓	✓	✓
BBN	✗	✓	✓	✗	✗	✓	✓	✓
BRG	✗	✓	✗	✓*	✓*	✓*	✓*	✓*
CSB	✗	✓	✗	✓	✗	✗	✓	✓
IDR	✗	✓	✗	✓	✓	✓	✓	✓
RAC	✗	✓	✓*	✓*	✓*	✓*	✓*	✓*
BOK	✗	-	✗	✗	✗	✗	✓*	✓*
CER	✗	✓	✗	✗	✓	✗	✓*	✓*
CUP	✗	✗	✗	✗	✗	✓	✗	✓*
DSK	✗	-	✗	✓	✓	✓	✓*	✓*
OTS	✗	-	✗	✗	✗	✗	✓*	✓*
STR	✗	✗	✗	✓	✗	✓*	✓	✓*
TOH	✓	-	✓	✓	✓*	✓	✓	✓

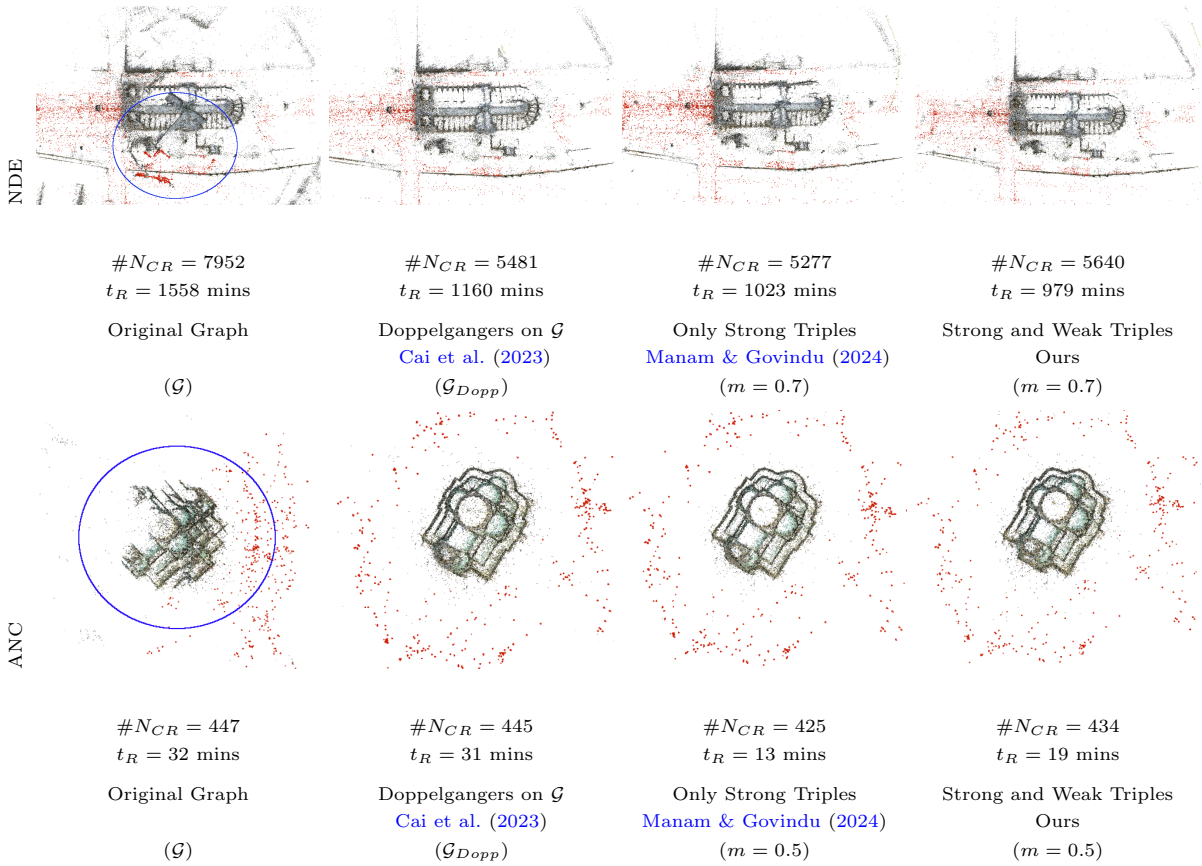


Fig. 10: Reconstructions obtained on ambiguous datasets with our method. Datasets resulting in superimposed reconstructions (marked in *blue*) are corrected after applying our method with faster reconstruction time compared to Doppelgangers Cai et al. (2023) due to the sparsification behaviour of our method

Table 8: Comparison of our method [ST+WT] with Manam & Govindu (2024) [ST] on ambiguous datasets. ✓/✓*/✗: Disambiguated/Disambiguated but oversplit/Non-disambiguated reconstructions. '-': No reconstruction was obtained. ST and WT indicate the usage of strong and weak triples, respectively. The best value is checked only across disambiguated/disambiguated but oversplit reconstructions

Dataset	Disambiguated				# Cameras ($\#N_{CR}$)				# 3D Points (in 10^3)				Time Taken (sec)	
	$m = 0.7$		$m = 0.9$		$m = 0.7$		$m = 0.9$		$m = 0.7$		$m = 0.9$			
Triples →	ST	ST+WT	ST	ST+WT	ST	ST+WT	ST	ST+WT	ST	ST+WT	ST	ST+WT	ST	ST+WT
LVE	✗	✓	✓	✓	317	**360**	186	**245**	110	**121**	69	**82**	48	**28**
NDE	✓	✓	✓	✓	5277	**5640**	3977	**5040**	1280	**1357**	1053	**1255**	5314	**2886**
SCO	✗	✗	✓	✓	2645	2811	1984	**2206**	450	491	347	**366**	1269	**665**
SEV	✓	✗	-	✓*	**253**	1425	-	**239**	**74**	339	-	**72**	22	**20**
ELS	✗	✗	✓	✓	614	801	288	**333**	122	155	75	**92**	34	**21**
PDP	✓	✓	✓*	✓*	818	**947**	299	**436**	101	**124**	40	**51**	13	**11**
YKM	✓	✓	✓*	✓	457	**567**	74	**344**	141	**170**	29	**112**	**19**	20
	$m = 0.4$		$m = 0.5$		$m = 0.4$		$m = 0.5$		$m = 0.4$		$m = 0.5$			
Triples →	ST	ST+WT	ST	ST+WT	ST	ST+WT	ST	ST+WT	ST	ST+WT	ST	ST+WT	ST	ST+WT
ANC	✓	✗	✓	✓	**426**	436	425	**434**	**82**	86	81	**85**	96	**10**
ADT	✓	✓	✓	✓	387	**423**	324	**422**	69	**76**	53	**75**	15	**3**
BLD	✓	✓	✓	✓	1585	**1601**	1575	**1596**	233	**237**	232	**236**	607	**107**
BBN	✓	✓	✓	✓	375	**392**	365	**387**	66	**70**	65	**69**	31	**5**
BRG	✓	✓*	✓	✓*	121	**139**	110	**137**	14	**14**	12	**14**	11	**1**
CSB	✓	✓	✓	✓	135	**142**	131	**140**	30	**31**	29	**30**	21	**3**
IDR	✓	✓	✓*	✓	**42**	**42**	33	**42**	**9**	**9**	**9**	**9**	11	**1**
RAC	✓	✓*	✓	✓*	120	**125**	115	**125**	28	**30**	27	**29**	17	**2**
BOK	✓*	✓*	✓*	✓*	**7**	**7**	**7**	**7**	**2**	**2**	**2**	**2**	**1**	**1**
CER	✓*	✓*	✓*	✓*	**7**	**7**	**7**	**7**	**3**	**3**	**3**	**3**	**1**	**1**
CUP	✗	✗	✓*	✓*	64	64	**40**	**40**	5	5	**3**	**3**	**1**	**1**
DSK	✓*	✓*	✓*	✓*	**11**	**11**	**11**	**11**	**3**	**3**	**3**	**3**	**1**	**1**
OTS	✓*	✓*	✓*	✓*	**9**	**9**	**9**	**9**	**3**	**3**	**3**	**3**	**1**	**1**
STR	✓	✓	✓*	✓*	**19**	**19**	**10**	**10**	**2**	**2**	**1**	**1**	**1**	**1**
TOH	✓	✓	✓	✓	**338**	**338**	**338**	**338**	**199**	195	**200**	197	35	**7**

with additional visual results.

Comparison with using only Strong Triples Manam & Govindu (2024)**:** In Table 8, we compare our method, which uses both strong and weak triples, to that of the method presented in Manam & Govindu (2024) that uses only strong triples. We use the minimum edge scores $m = \{0.7, 0.9\}$ for large-scale datasets and $m = \{0.4, 0.5\}$ for medium and small-scale datasets. We observe that a higher value of m using only strong triples Manam & Govindu (2024) can lead to no reconstruction, as seen for the SEV dataset, whereas using both strong and weak triples leads to reasonable reconstructions for all datasets. This is because edge scores are relatively high with our method compared to Manam & Govindu (2024). Hence, a high threshold removes both redundant and false edges. Similar to generic datasets, using both strong and weak triples reconstructs significantly more cameras and 3D points for most datasets when compared to using only strong triples for a given m. This leads to an increase in the reconstruction time when viewgraphs are sparsified with our method than Manam & Govindu (2024) for some datasets, which is shown in Table S8 of the supplementary material. Moreover, as observed with generic datasets, our method takes less compute time compared to Manam & Govindu (2024) for ambiguous datasets as well.

5 Conclusion

In this paper, we present a unified view of two tasks in Structure-from-Motion, viewgraph sparsification and disambiguation of repeated structures. We propose a scoring mechanism for the edges in a viewgraph based on epipolar inliers, which captures both the redundancy of the edges and the presence of false edges based on global information of 3D point connectivity. We use triples of cameras connected by two or three edges to score edges based on the relative proportion of epipolar inliers in triples. We formulate our edge selection procedure as an optimization problem whose optimum can be obtained using a simple thresholding scheme. We propose an efficient algorithm to handle both tasks simultaneously, requiring a minimum edge score as the only hyperparameter as input. The proposed algorithm can be incorporated into any SfM pipeline as a preprocessing step, making it modular. Applying our method significantly reduces reconstruction time while recovering overall 3D scene structure, maintaining accuracy and removing ghost artifacts from generic datasets. Our method also disambiguates repeated structures in the scenes along with sparsifying viewgraphs of ambiguous datasets, resulting in reduced reconstruction time. Moreover, we also compare our method with Manam & Govindu (2024) that uses only strong triples, which reveals that our method recovers more cameras and 3D points with a faster compute time compared to the method that uses only strong triples.

Supplementary Information.
This paper has an accompanying supplementary file which provides additional details of the experiments shown here.

Acknowledgements.
Lalit Manam was supported by a Prime Minister's Research Fellowship, Government of India. This research was supported in part by a Core Research Grant from the Science and Engineering Research Board, Department of Science and Technology, Government of India.

Data Availability.
All datasets and source codes of published methods are available at the respective authors'/project websites, whose papers are cited in this paper. The source code of the proposed method will be released publicly.

References

Achar, S., Jawahar, C.V., Krishna, K.M. (2011). Large-scale visual localization in urban environments. *International conference on robotics and automation* (pp. 5642–5648).

Arandjelovic, R., Gronat, P., Torii, A., Pajdla, T., Sivic, J. (2016). NetVLAD: CNN architecture for weakly supervised place recognition. *Conference on computer vision and pattern recognition* (pp. 5297–5307).

Bay, H., Ess, A., Tuytelaars, T., Van Gool, L. (2008). Speeded-up robust features (SURF). *Computer Vision and Image Understanding*, *110*(3), 346–359,

Cai, R., Tung, J., Wang, Q., Averbuch-Elor, H., Hariharan, B., Snavely, N. (2023). Doppelgangers: Learning to disambiguate images of similar structures. *International conference on computer vision* (pp. 34–44). IEEE.

Chatterjee, A., & Govindu, V.M. (2017). Robust relative rotation averaging. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *40*(4), 958–972,

Cohen, A., Zach, C., Sinha, S.N., Pollefeys, M. (2012). Discovering and exploiting 3D symmetries in Structure-from-Motion. *Conference on computer vision and pattern recognition* (pp. 1514–1521).

Crandall, D., Owens, A., Snavely, N., Huttenlocher, D.P. (2011). Discrete-continuous optimization for large-scale Structure-from-Motion. *Conference on computer vision and pattern recognition* (pp. 3001–3008).

Cui, Z., & Tan, P. (2015). Global Structure-from-Motion by similarity averaging. *International conference on computer vision* (pp. 864–872). IEEE.

Darmon, F., Aubry, M., Monasse, P. (2020). Learning to guide local feature matches. *International conference on 3d vision* (pp. 1127–1136).

DeTone, D., Malisiewicz, T., Rabinovich, A. (2018). Superpoint: Self-supervised interest point detection and description. *Conference on computer vision and pattern recognition workshops* (pp. 224–236).

Enqvist, O., Kahl, F., Olsson, C. (2011). Non-sequential Structure-from-Motion. *International conference on computer vision workshops* (pp. 264–271).

Govindu, V.M. (2001). Combining two-view constraints for motion estimation. *Conference on computer vision and pattern recognition* (Vol. 2, p. 218-225).

Govindu, V.M. (2004). Lie-algebraic averaging for globally consistent motion estimation. *Conference on computer vision and pattern recognition* (Vol. 1).

Hartley, R., & Zisserman, A. (2004). *Multiple view geometry in computer vision* (Second ed.). Cambridge University Press.

Hausler, S., Garg, S., Xu, M., Milford, M., Fischer, T. (2021). Patch-NetVLAD: Multi-scale fusion of locally-global descriptors for place recognition. *Conference on computer vision and pattern recognition* (pp. 14141–14152).

Havlena, M., Torii, A., Knopp, J., Pajdla, T. (2009). Randomized Structure-from-Motion based on atomic 3D models from camera triplets. *Conference on computer vision and pattern recognition* (pp. 2874–2881).

Havlena, M., Torii, A., Pajdla, T. (2010). Efficient Structure-from-Motion by graph optimization. *European conference on computer vision* (pp. 100–113).

Heinly, J., Dunn, E., Frahm, J.-M. (2014a). Correcting for duplicate scene structure in sparse 3D reconstruction. *European conference on computer vision* (pp. 780–795).

Heinly, J., Dunn, E., Frahm, J.-M. (2014b). Recovering correct reconstructions from indistinguishable geometry. *International conference on 3d vision* (Vol. 1, pp. 377–384).

Howard, A., Trulls, E., etru1927, Yi, K.M., old ufo, Dane, S., Jin, Y. (2022). *Image matching challenge 2022.* https://kaggle.com/competitions/image-matching-challenge-2022. (Kaggle)

Jiang, N., Tan, P., Cheong, L.-F. (2012). Seeing double without confusion: Structure-from-Motion in highly ambiguous scenes. *Conference on computer vision and pattern recognition* (pp. 1458–1465).

Kataria, R., DeGol, J., Hoiem, D. (2020). Improving Structure-from-Motion with reliable resectioning. *International conference on 3d vision* (pp. 41–50).

Knopp, J., Sivic, J., Pajdla, T. (2010). Avoiding confusing features in place recognition. *European conference on computer vision* (pp. 748–761).

Kulis, B., & Grauman, K. (2009). Kernelized locality-sensitive hashing for scalable image search. *International conference on computer vision* (pp. 2130–2137).

Li, Y., Snavely, N., Huttenlocher, D.P. (2010). Location recognition using prioritized feature matching. *European conference on computer vision* (pp. 791–804).

Lindenberger, P., Sarlin, P.-E., Pollefeys, M. (2023). Lightglue: Local feature matching at light speed. *International conference on computer vision.* IEEE.

Liu, C., Yuen, J., Torralba, A. (2010). SIFT flow: Dense correspondence across scenes and its applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *33*(5), 978–994,

Liu, Y., Hel-Or, H., Kaplan, C.S., Van Gool, L., Others. (2010). Computational symmetry in computer vision and computer graphics. *Foundations and Trends in Computer Graphics and Vision*, *5*(1–2), 1–195,

Lowe, D.G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, *60*, 91–110,

Manam, L., & Govindu, V.M. (2023). Sensitivity in translation averaging. *Advances in neural information processing systems* (p. 62740-62763).

Manam, L., & Govindu, V.M. (2024). Leveraging camera triplets for efficient and accurate structure-from-motion. *Conference on computer vision and pattern recognition* (pp. 4959–4968).

McLachlan, G., & Krishnan, T. (2008). *The em algorithm and extensions* (second ed.). Wiley.

Mikolajczyk, K., & Schmid, C. (2005). A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *27*(10), 1615–1630,

Nistér, D., & Stewenius, H. (2006). Scalable recognition with a vocabulary tree. *Conference on computer vision and pattern recognition* (Vol. 2, pp. 2161–2168).

Noh, H., Araujo, A., Sim, J., Weyand, T., Han, B. (2017). Large-scale image retrieval with attentive deep local features. *International conference on computer vision* (pp. 3456–3465). IEEE.

Roberts, R., Sinha, S.N., Szeliski, R., Steedly, D. (2011). Structure-from-Motion for scenes with large duplicate structures. *Conference on computer vision and pattern recognition* (pp. 3137–3144).

Rublee, E., Rabaud, V., Konolige, K., Bradski, G. (2011). ORB: An efficient alternative to SIFT or SURF. *International conference on computer vision* (pp. 2564–2571).

Sarlin, P.-E., DeTone, D., Malisiewicz, T., Rabinovich, A. (2020). Superglue: Learning feature matching with graph neural networks. *Conference on computer vision and pattern recognition* (pp. 4938–4947).

Schonberger, J.L., & Frahm, J.-M. (2016). Structure-from-Motion revisited. *Conference on computer vision and pattern recognition* (pp. 4104–4113).

Schops, T., Schonberger, J.L., Galliani, S., Sattler, T., Schindler, K., Pollefeys, M., Geiger, A. (2017). A multi-view stereo benchmark with high-resolution images and multi-camera videos. *Conference on computer vision and pattern recognition* (pp. 3260–3269).

Shah, R., Chari, V., Narayanan, P.J. (2018). Viewgraph selection framework for SfM. *European conference on computer vision* (pp. 535–550).

Shen, T., Zhu, S., Fang, T., Zhang, R., Quan, L. (2016). Graph-based consistent matching for Structure-from-Motion. *European conference on computer vision* (pp. 139–155).

Snavely, N., Seitz, S.M., Szeliski, R. (2006). Photo tourism: Exploring photo collections in 3D. *Siggraph 2006 papers* (pp. 835–846). ACM.

Snavely, N., Seitz, S.M., Szeliski, R. (2008a). Modeling the world from internet photo collections. *International Journal of Computer Vision*, *80*, 189–210,

Snavely, N., Seitz, S.M., Szeliski, R. (2008b). Skeletal graphs for efficient Structure-from-Motion. *Conference on computer vision and pattern recognition* (pp. 1–8).

Sweeney, C., Hollerer, T., Turk, M. (2015). Theia: A fast and scalable Structure-from-Motion library. *International conference on multimedia* (pp. 693–696).

Tola, E., Lepetit, V., Fua, P. (2009). Daisy: An efficient dense descriptor applied to wide-baseline stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *32*(5), 815–830,

Torii, A., Sivic, J., Pajdla, T., Okutomi, M. (2013). Visual place recognition with repetitive structures. *Conference on computer vision and pattern recognition* (pp. 883–890).

Tyszkiewicz, M., Fua, P., Trulls, E. (2020). DISK: Learning local features with policy gradient. *Advances in neural information processing systems* (pp. 14254–14265).

Wang, F., Nayak, A., Agrawal, Y., Shilkrot, R. (2018). Hierachical image link selection scheme for duplicate structure disambiguation. *British machine vision conference* (p. 221).

Wilson, K., & Snavely, N. (2013). Network principles for SfM: Disambiguating repeated structures with local context. *International conference on computer vision* (pp. 513–520). IEEE.

Wilson, K., & Snavely, N. (2014). Robust global translations with 1DSfM. *European conference on computer vision* (pp. 61–75).

Wu, C. (2011). *VisualSfM: A visual Structure-from-Motion system.* http://www.cs.washington.edu/homes/ccwu/vsfm.

Yan, Q., Yang, L., Zhang, L., Xiao, C. (2017). Distinguishing the indistinguishable: Exploring structural ambiguities via geodesic context. *Conference on computer vision and pattern recognition* (pp. 3836–3844).

Yi, K.M., Trulls, E., Ono, Y., Lepetit, V., Salzmann, M., Fua, P. (2018). Learning to find good correspondences. *Conference on computer vision and pattern recognition* (pp. 2666–2674).

Zach, C., Irschara, A., Bischof, H. (2008). What can missing correspondences tell us about 3D structure and motion? *Conference on computer vision and pattern recognition* (pp. 1–8).

Zach, C., Klopschitz, M., Pollefeys, M. (2010). Disambiguating visual relations using loop constraints. *Conference on computer vision and pattern recognition* (pp. 1426–1433).