Leveraging Camera Triplets for Efficient and Accurate Structure-from-Motion

Lalit Manam and Venu Madhav Govindu
Indian Institute of Science
Bengaluru - 560012, INDIA
{lalitmanam,venug}@iisc.ac.in

Abstract

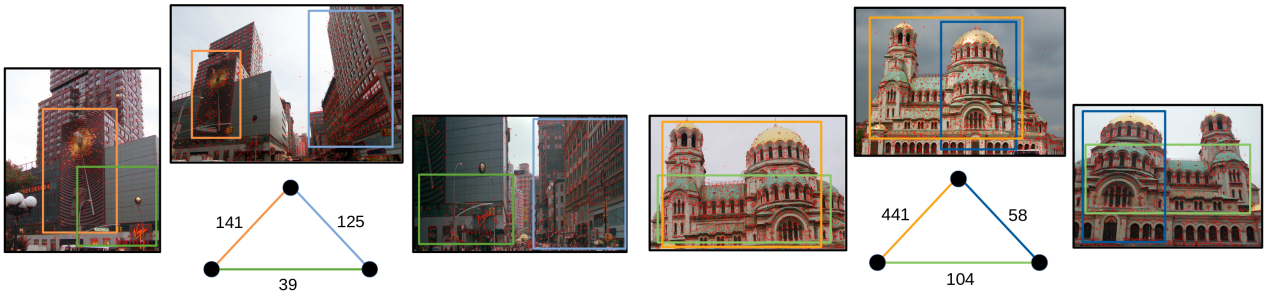
In Structure-from-Motion (SfM), the underlying viewgraphs of unordered image collections generally have a highly redundant set of edges that can be sparsified for efficiency without significant loss of reconstruction quality. Often, there are also false edges due to incorrect image retrieval and repeated structures (symmetries) that give rise to ghosting and superimposed reconstruction artifacts. We present a unified method to simultaneously sparsify the viewgraph and remove false edges. We propose a scoring mechanism based on camera triplets that identifies edge redundancy as well as false edges. Our edge selection is formulated as an optimization problem which can be provably solved using a simple thresholding scheme. This results in a highly efficient algorithm which can be incorporated as a pre-processing step into any SfM pipeline, making it practically usable. We demonstrate the utility of our method on generic and ambiguous datasets that cover the range of small, medium and large-scale datasets, all with different statistical properties. Sparsification of generic datasets using our method significantly reduces reconstruction time while maintaining the accuracy of the reconstructions as well as removing ghosting artifacts. For ambiguous datasets, our method removes false edges, thereby avoiding incorrect superimposed reconstructions.

1. Introduction

Structure-from-Motion (SfM) [13] has been a long-studied problem in computer vision. It aims to reconstruct a 3D structure and estimate camera motions given images of a scene. The first step in SfM is to identify the pairs of images that capture the same part of the scene. For unordered image collections, especially for large-scale datasets, image retrieval techniques [1, 14, 23, 31, 32] are used where each image is assigned with a descriptor and then matched with other images. The relationships between the pairs of images can be represented as a graph (also called a viewgraph in the SfM literature), with nodes representing cameras and edges representing an overlap of the scene in the images captured by the two cameras, implying that their relative geometry can be recovered. This graph is further processed to extract keypoints in the images, which are then matched with keypoints in other images based on the connectivity of the graph [2, 7, 8, 25, 27, 34, 35, 44, 50]. Subsequently, the relative motion for every edge is obtained by estimating epipolar geometry [13] using inlier keypoint matches. We refer to such matches as **epipolar inliers**. The obtained graph in this fashion is used by either incremental [36, 39, 40, 48] or global methods [9–11, 42] to get a 3D reconstruction of the scene along with camera motions.

The underlying viewgraphs obtained from unordered image collections generally contain a redundant set of images (equivalently nodes) since popular views of the scene are captured more frequently [41]. As a consequence, a large number of pairs of images (equivalently edges) are identified to capture the same part of the scene, resulting in a highly redundant set of edges in the viewgraphs [16]. We refer to the datasets with the above-mentioned properties as **generic datasets**. Many methods [15, 16, 37, 38, 41] have been proposed to sparsify the viewgraphs while maintaining the accuracy of the reconstructions by considering only the nodes and edges which are distinguished to be important.

When scenes contain repeated structures leading to different symmetries [26], a significant number of false edges occur between the cameras viewing repeated structures. This is because such images look similar and, thus, are identified as belonging to the same part of the scene, even if they capture physically different parts of the scene. The presence of such false edges leads to superimposed reconstructions. We refer to these datasets as **ambiguous datasets**. Methods based on viewgraph properties and loop consistency checks in the viewgraphs [3, 4, 17–19, 21, 33, 37, 38, 45, 46, 49, 51, 52] have been proposed to tackle false edges.



(a) Generic case (Union Square [47]).

(b) Ambiguous case (Alexander Nevsky Cathedral [17]).

Figure 1. Camera triplets 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 rectangular boxes with colours corresponding to the edge colours. The red dots on the images show the detected keypoints. (a) A triplet 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 triplet 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.

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

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

We note that generic datasets can also contain a small fraction of false edges due to incorrect images retrieved, which leads to ghost artifacts in reconstructions. On the other hand, ambiguous datasets contain repeated structures which give rise to a significant number of false edges.

Both the problems, viewgraph sparsification and disambiguation of repeated structures, have mostly been dealt with independently in the literature. The underlying assumptions of both the problems are very much opposed. For viewgraph sparsification, most of the edges are considered to be reliable (or true). Thus, the removal of a few edges does not severely affect the accuracy of reconstruction. On the other hand, for the disambiguation problem, a significant number of edges are false, which must be weeded out to obtain a proper reconstruction. Our objective is to solve both problems together by a single scheme, which is summarized in Table 1.

Our contributions: In this paper, we simultaneously target both the problems of viewgraph sparsification and disambiguation of repeated structures. A fundamental design attribute of our approach is the idea that camera triplets are more informative than individual edges [41, 51] and are easy to analyze since they form loops of the smallest possible length. For the problems on hand, triplets provide an important indication of an edge being redundant or ambiguous based on matched keypoints (see Fig. 1). We propose a single scheme for both tasks by leveraging camera triplets. We identify redundant and false edges based on a scoring mechanism that aggregates information about 3D point connectivity of the triplets globally over the whole graph without requiring intermediate partial reconstructions and remove them based on a single threshold. The proposed method can be incorporated into any SfM pipeline, either incremental or global, making it practical for usage. Experiments on viewgraph sparsification show a significant improvement in reconstruction time while maintaining accuracy on large-scale datasets. Our method also removes ghost artifacts in the reconstructions of generic datasets and shows disambiguation capability on ambiguous datasets.

2. Literature Review

Viewgraph sparsification: Snavely *et al.* [41] constructed skeletal graphs based on the approximate covariance of camera motions. Havlena *et al.* [15] used image level descriptors to carefully choose camera triplets for independent reconstructions and then merged them. Havlena *et al.* [16] also relied on image level descriptors and selected a minimally connected dominating set of the image collection. Shen *et al.* [38] used an online algorithm to get a spanning tree, expanded the graph based on triplets and finally enforced a community-based graph structure. Shah *et al.* [37] selected edges on the graph based on a minimum cost network flow problem. Many methods rely on intermediate partial reconstructions [15, 16, 41], have multiple hyperparameters [37, 38], or design their own specific SfM pipeline [15, 16, 41] but we provide a solution

based on a single hyperparameter without any intermediate reconstruction, and can be used in any pipeline.

Disambiguation of repeated structures: Zach *et al.* [51] reasoned about false edges based on missing correspondences in triplets using Bayesian inference. Zach *et al.* [52] further extended [51] on longer loops using relative camera motions. Roberts *et al.* [33] also inferred false edges based on relative motions using an expectation-maximization framework [29]. Cohen *et al.* [4] detected symmetries by estimating similarity transforms and removing false edges to reduce drift. Jiang *et al.* [19] measured reconstruction quality based on missing correspondences and used it for edge selection. Wilson *et al.* [46] split the camera-point relation graph based on its covering subgraphs. Heinly *et al.* [17] proposed a post-processing step to split the reconstruction based on conflicting observations and then merge the disconnected epipolar inliers. Heinly *et al.* [18] also proposed a post-processing step to isolate 3D points connecting separate parts of the reconstruction incorrectly by analyzing the clustering behaviour of 3D point connectivity. Cui *et al.* [6] merged two-view reconstructions locally to find missing correspondences. Yan *et al.* [49] constructed a path network where a few anchor images are selected and other images are related to them with paths for clean image correspondences. Wang *et al.* [45] grouped the images based on viewgraph and scored edges on their ability to expand the reconstruction. Kataria *et al.* [21] considered keypoints which are only matched across a few images for resectioning cameras to obtain a reliable reconstruction. Cai *et al.* [3] proposed Doppelgangers, where a neural network detects false edges based on images, keypoints and matches. Some graph sparsification methods also disambiguated repeated structures either by checking geometric consistency along loops [38] or by modifying their cost function for handling ambiguous datasets [37]. Some other methods [22, 43] avoided incorrect fetching of images containing repetitive structures during the image retrieval phase by specific designs of image descriptors. Many of these methods assume specific probability distributions on missing correspondences [51, 52], exploit intermediate reconstructions [6, 33], involve multiple hyperparameters [18, 19, 21, 37, 38, 46, 49] or are designed for specific pipelines [21, 51]. 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

The goal of graph sparsification is to recover most of the cameras and 3D points with a sparsified graph compared to the original graph while maintaining accuracy and significantly reducing reconstruction time. Dealing with ambiguities in the scene requires distinguishing between repeated structures. Our aim is to handle both the tasks of graph sparsification and disambiguation of repeated structures using a single scheme with as few hyperparameters as possible without needing any intermediate partial reconstructions. We also seek to build a scheme which can be used as a pre-processing step for any pipeline, either incremental or global. To achieve our goals, we present a scheme for scoring edge quality and then formulate an optimization problem which can be provably solved based on thresholding leading to our proposed algorithm.

Edge quality score: Edges contain limited information about the pairs of cameras they connect. An individual edge alone cannot validate its importance in the graph, and thus, scoring them meaningfully requires considering other connected edges in the graph. We take recourse to camera triplets, which provide much more information about an edge based on the other two edges in a triplet (see Fig. 1). Triplets have been widely used for sparsification [15, 16, 37, 38, 41] and disambiguation [19, 33, 51, 52]. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a viewgraph, with $\mathcal{V}$ and $\mathcal{E}$ denote the set of nodes and edges in $\mathcal{G}$, respectively. We first determine a quality score, $q_{ij} > 0$, for each edge $(i, j) \in \mathcal{E}$ before processing the viewgraph. Since we aim to get a 3D reconstruction, to sparsify the viewgraph, we want to score the edges based on how well they are connected via 3D points. One way to get an idea of 3D point connectivity without access to 3D reconstruction is to check for epipolar inliers on the edges since inliers are the candidate keypoints for reconstruction. One could check for common inliers in all three cameras and assign a score to each edge on the triplet based on the fraction of common inliers to that of the total inliers belonging to that edge. Such a scoring mechanism favours edges with a low number of inliers, which conflicts with our aim to score based on 3D point connectivity.

Therefore, we score edges based on the number of inliers instead of common inliers across all three edges of the triplet, which adheres to our aim of 3D point connectivity-based scores. The indication for graph sparsification and disambiguation in the triplets is given by the relative number of inliers between the three edges (also seen in Fig. 1) and not by the actual number of inliers. To score edges, we take the ratio of the number of inliers in each edge to the maximum number of inliers among all three edges in a triplet. This assigns scores to edges based on relative 3D point connectivity in the triplet and is also considerate of the cases where all images in triplets have a low number of inliers due to occlusion or lack of texture.

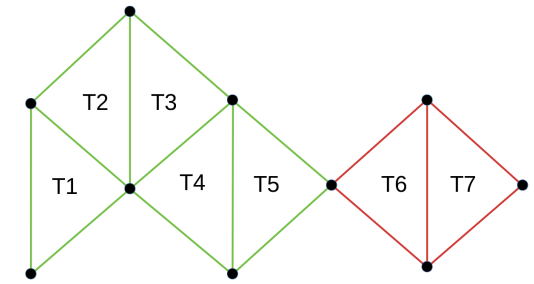
Inherently, our scoring mechanism based on the relative

number of inliers in triplets is also useful for disambiguation. [3] empirically noted that the actual number of epipolar inliers does not provide useful information about ambiguity. Missing correspondences [19, 33, 49, 51] and the statistics of matched keypoints [49] provide a cue about the presence of false edges. They are generally detected using loops of small lengths like triplets since the likelihood of viewing the same 3D points on larger loops reduces. But the implicit assumption is the repeatability of the keypoints for the same 3D point, which can be violated in case of noisy images, occlusion or drastic changes in viewpoint [17, 30, 51]. Scoring based on the number of inliers instead of common inliers in the loops reduces the effect of unmatched keypoints due to repeatability concerns. This is because there can be other matched keypoints on the edges which will compensate for reduced edge scores due to non-repeatable keypoints. Moreover, we consider only inliers on the edges for computing scores. This ensures that only keypoints, which are proven to have discriminative capabilities for matching, are used for inference instead of all the keypoints (as used in [3, 33]).

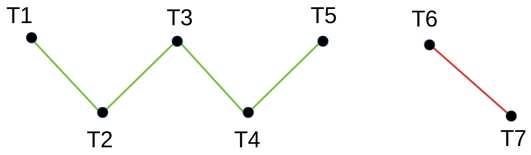
A given edge can be a part of many triplets and, thus, have as many scores. To aggregate information about the edge over the whole graph, for each edge, we take the average of the scores obtained from all the triplets it participates in. This ensures that edge scores reflect global information about the edge in the graph. Since our scoring mechanism is based only on triplets, we remove edges which are not a part of any triplet, as such edges cannot be scored. In practice, the viewgraphs for unordered image collections are well connected and losing edges not contributing to triplets has minimal impact on the reconstructions (please refer to Sec. 4 for details).

Handling graphs consisting only of triplets: For graphs containing only triplets, there can be cases when two triplets are connected by a node and do not share a common edge. We call such a connection between two triplets a **joint**. In Fig. 2a, we present a scenario where the triplets $T5$ and $T6$ have a common node but no common edge. Also, there are no other triplets that share common edges with both *green* and *red* edges. In such cases, the edges marked in the two colours are effectively being scored independently since edges marked in each colour do not participate in the scoring of edges in the other colour. Although, we can continue with such independent scoring, for ambiguous datasets, it is not possible to infer whether the joints are valid connections or not. So, we separate the joints, which ensures that inference on edges is not based on independent scores.

We construct a triplet graph $\mathcal{G}_T = (\mathcal{V}_T, \mathcal{E}_T)$, as done in [20, 28], where nodes, $\mathcal{V}_T$, denote triplets in $\mathcal{G}$ and edges, $\mathcal{E}_T$, connect the nodes if an edge is common between the triplets in $\mathcal{G}$. By construction, the triplets connected by the joints in the viewgraph $\mathcal{G}$ are not connected in the triplet graph $\mathcal{G}_T$. If there are no common triplets between the set of edges separated by joints, it results in multiple connected components in the triplet graph $\mathcal{G}_T$, as can be seen in Fig. 2b. For each connected component of $\mathcal{G}_T$, the triplets in $\mathcal{G}$ will have at least one common edge with the other triplets in the same connected component and no common edges across different components of $\mathcal{G}_T$. Thus, the edges participating in different connected components of $\mathcal{G}_T$ can be scored independently. So, we only retain the edges in $\mathcal{G}$ corresponding to the largest connected component of $\mathcal{G}_T$ (denoting it as $\mathcal{G}_{LCT}$) since our aim is to assign scores based on global reasoning. We empirically observe that separating the joints has a minor effect on the reconstructions (see Sec. 4).



(a) Graph ($\mathcal{G}$) containing only triplets.



(b) Corresponding triplet graph ($\mathcal{G}_T$).

Figure 2. Graph ($\mathcal{G}$) containing only triplets with a joint and its corresponding triplet graph ($\mathcal{G}_T$). The edges marked in *green* and *red* can be scored independently due to the absence of triplets sharing common edges across them. The triplet graph separates the joint and disconnects such graphs.

Formulation: Given a quality score q_{ij} for each edge, we aim to select those edges which will increase the average quality score taken across all the edges. To avoid having the degenerate case of selecting edges only with the highest score (which could be as low as one edge in realistic scenarios), we need to regularize the cost by penalizing the loss of edges. 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

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

where $\lambda \geq 0$ is a regularization parameter which avoids degenerate solutions for the optimization problem. λ also acts as a tuning parameter for the aggressiveness of edge removal, where reducing λ increases the number of edges removed. The second term in Eqn. 1 suggests the average of quality 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. 1 does not enforce the graph structure, the quality scores obtained for edges are based on all the triplets they are members of. Let Trp be the set of all triplets in $\mathcal{G}_{LCT}$, $trp(i,j) \subseteq Trp$ be the set of triplets for which the edge (i,j) is a member, and q_{ij}^t be the score obtained from the triplet $t \in trp(i,j)$. Replacing the scores q_{ij} in Eqn. 1 with $\frac{\sum_{t\in trp(i,j)} q_{ij}^t}{|trp(i,j)|}$ discloses the dependence of Eqn. 1 on the graph structure more explicitly. While the optimization problem in Eqn. 1 is combinatorial, we provide a simple thresholding scheme to solve it, which is shown as follows:

$$s_{ij} = \begin{cases} 1 & \text{if } q_{ij} \geq \tau, \\ 0 & \text{otherwise,} \end{cases} \quad (2)$$

where τ denotes a threshold. This thresholding scheme is equivalent to optimizing the problem in Eqn. 1 with τ acting as a regularizer, as shown by the following theorem:

Theorem 1. *For a given λ in Eqn. 1, there exists a threshold τ such that the values of s_{ij} obtained by solving the problems given by Eqn. 1 and Eqn. 2 are the same.*

Please refer to supplementary material for the proof of Thm. 1. 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

$$\tau = m\left(1 - \frac{d_{max}}{|\mathcal{V}|}\right) + \left(\frac{d_{max}}{|\mathcal{V}|}\right), \quad (3)$$

where $|\mathcal{V}|$ is the number of nodes and m is the minimum score which edges should satisfy. We deliberately choose $|\mathcal{V}|$ as the denominator 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 procedure in Algo. 1. Efficient implementations exist to get the list of all triplets [12, 42]. We note that steps 4-8 in Algo. 1 are performed using CPU parallelization (20 threads) and vectorized operations, which makes it efficient.

Algorithm 1: Edge Selection using Triplets

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

1 Construct triplet graph $\mathcal{G}_T$ from $\mathcal{G}$.
2 Retain edges in $\mathcal{G}$ participating in largest connected component of $\mathcal{G}_T$, denoting it $\mathcal{G}_{LCT}$.
3 Find the list of all triplets Trp in $\mathcal{G}_{LCT}$.
4 **for** each triplet t in Trp **do**
5 Get the number of inliers for all three edges as $n_{ij}, (i,j) \in t$.
6 Assign a score to each edge based on triplet t: $q_{ij}^t = \frac{n_{ij}}{\max_{(k,l)\in t} n_{kl}}$.
7 **end**
8 Assign q_{ij} to each edge: $q_{ij} = \frac{\sum_{t\in trp(i,j)} q_{ij}^t}{|trp(i,j)|}$.
9 Compute τ from m using Eqn. 3.
10 Remove edges in $\mathcal{G}_{LCT}$ with $q_{ij} < \tau$ giving $\mathcal{G}_F$.
11 Extract the largest connected component of $\mathcal{G}_F$.

4. Experiments

We present experimental results on publicly available generic and ambiguous datasets in individual subsections. We use COLMAP [36] to obtain the original viewgraph and also use it to get 3D reconstructions under different scenarios. All experiments are performed on a PC with 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]. We only consider the largest connected component of the viewgraphs for reporting values. We state the **notations for graphs** below (same as used in Algo. 1), for further usage:

- $\mathcal{G}$: Original viewgraph.
- $\mathcal{G}_{LCT}$: Graph containing edges contributing to the largest connected component of the triplet graph $\mathcal{G}_T$.
- $\mathcal{G}_{Dopp}$: Graph obtained after applying Doppelgangers [3] on the original graph $\mathcal{G}$.
- $\mathcal{G}_F(m)$: Graph obtained after applying Algo. 1 with minimum edge score m.

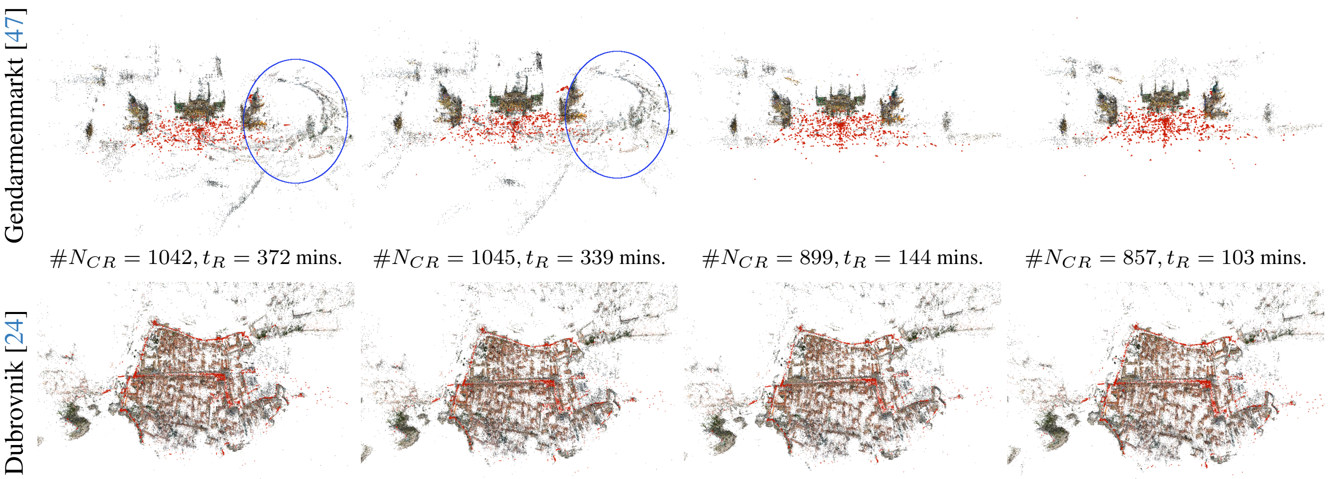
4.1. Generic Datasets

In this subsection, we provide results for graph sparsification on generic datasets. We consider datasets provided by [5, 24, 47] and apply Algo. 1 to sparsify viewgraphs.

[1] GPUs are used only for Doppelgangers [3] and COLMAP [36].

Dataset	Clean Reconstruction				# Cameras Reconstructed ($\#N_{CR}$)				# 3D Points Reconstructed (in 10^3)				Mean Reprojection Errors (px)				Reconstruction Time (mins) (t_R)			
	$\mathcal{G}$	$\mathcal{G}_{LCT}$	$\mathcal{G}_F(0.6)$	$\mathcal{G}_F(0.7)$	$\mathcal{G}$	$\mathcal{G}_{LCT}$	$\mathcal{G}_F(0.6)$	$\mathcal{G}_F(0.7)$	$\mathcal{G}$	$\mathcal{G}_{LCT}$	$\mathcal{G}_F(0.6)$	$\mathcal{G}_F(0.7)$	$\mathcal{G}$	$\mathcal{G}_{LCT}$	$\mathcal{G}_F(0.6)$	$\mathcal{G}_F(0.7)$	$\mathcal{G}$	$\mathcal{G}_{LCT}$	$\mathcal{G}_F(0.6)$	$\mathcal{G}_F(0.7)$
Alamo [47]	✓	✓	✓	✓	899	875	652	612	161	157	128	126	0.66	0.66	0.62	**0.59**	44	44	28	**25**
Gendarmenmarkt [47]	✗	✗	✓	✓	1042	1045	899	857	207	206	160	151	0.76	0.76	**0.61**	0.74	372	339	144	**103**
Madrid Metropolis [47]	✓	✓	✓	✓	471	453	274	232	74	72	47	33	0.62	0.61	**0.57**	0.58	88	90	22	**21**
Montreal Notre Dame [47]	✓	✓	✓	✓	575	570	411	398	145	146	108	103	0.86	0.86	0.80	**0.79**	114	100	**50**	52
Notre Dame [47]	✗	✗	✓	✓	1407	1406	1295	1093	348	348	332	288	0.76	0.76	0.70	**0.68**	2054	2131	1323	**758**
NYC Library [47]	✓	✓	✓	✓	635	614	509	396	110	109	93	84	0.74	0.74	0.71	**0.70**	105	101	**40**	**40**
Piccadilly [47]	✗	✗	✓	✓	3208	3110	2786	2645	374	368	311	288	0.76	0.76	0.74	**0.72**	1969	1409	1060	**1000**
Roman Forum [47]	✓	✓	✓	✓	1587	1565	1365	1295	324	323	278	263	0.76	0.76	0.73	**0.72**	784	810	477	**453**
Tower of London [47]	✓	✓	✓	✓	735	718	535	495	165	165	137	129	0.63	0.63	0.62	**0.61**	95	86	44	**42**
Trafalgar [47]	✓	✓	✓	✓	7867	7532	6538	6256	706	695	573	545	0.74	0.74	0.73	**0.72**	6875	5601	3852	**3698**
Union Square [47]	✓	✓	✓	✓	1160	1150	891	805	82	81	62	55	0.74	0.73	**0.69**	**0.69**	220	185	**80**	86
Vienna Cathedral [47]	✗	✗	✓	✓	1197	1161	1010	965	304	301	258	248	0.75	0.75	0.74	**0.73**	793	703	398	**376**
Dubrovnik [24]	✓	✓	✓	✓	5883	5850	5576	5464	1252	1250	1127	1073	0.76	0.76	0.73	**0.72**	582	556	521	**428**
Rome [24]	✓	✓	✓	✓	1812	1812	1807	1728	395	395	389	389	0.90	0.90	0.87	**0.85**	284	280	279	**265**
Quad [5]	✓	✓	✓	✓	5729	5569	4360	3843	1295	1277	1041	882	0.69	0.69	0.67	**0.66**	417	418	318	**249**

Table 2. Reconstruction statistics on generic datasets. ✓/ ✗: Removed/Existing ghost artifacts. **Bold** values indicate best in the respective category. Our method sparsifies the viewgraphs, reconstructing most cameras and 3D points with reduced reprojection errors and reconstruction time.



$\#N_{CR} = 1042, t_R = 372$ mins. $\#N_{CR} = 1045, t_R = 339$ mins. $\#N_{CR} = 899, t_R = 144$ mins. $\#N_{CR} = 857, t_R = 103$ mins.

$\#N_{CR} = 5883, t_R = 582$ mins. $\#N_{CR} = 5850, t_R = 556$ mins. $\#N_{CR} = 5576, t_R = 521$ mins. $\#N_{CR} = 5464, t_R = 428$ mins.

Original Graph ($\mathcal{G}$). Graph with Triplets ($\mathcal{G}_{LCT}$). Sparsified Graph [Ours] ($\mathcal{G}_F(0.6)$). Sparsified Graph [Ours] ($\mathcal{G}_F(0.7)$).

Figure 3. Reconstructions obtained with different graphs on generic datasets. $\#N_{CR}$: Number of cameras reconstructed, t_R: Reconstruction time using COLMAP [36]. Top row: Dataset creating ghost artifacts in reconstructions (marked in *blue*). Applying our method removes such artifacts with other parts of the reconstructions intact. Bottom row: Dataset with many redundant cameras and edges. Our method sparsifies the graphs, giving visually similar reconstructions in reduced reconstruction time.

In Table 2, we compare results of the original viewgraph $\mathcal{G}$ and the sparsified viewgraphs $\mathcal{G}_F$ with different values of minimum edge score, $m = \{0.6, 0.7\}$. We also show the results for the viewgraph $\mathcal{G}_{LCT}$, since Algo. 1 extracts this graph before scoring the edges. It can be seen that the number of cameras and 3D points reconstructed with $\mathcal{G}$ and $\mathcal{G}_{LCT}$ are similar, with mean reprojection errors being close. This shows that the removal of edges not participating in the largest connected component of the triplet graph has minimal impact on the 3D reconstructions. The sparsified viewgraphs ($\mathcal{G}_F$), using Algo. 1, also have most cameras and 3D points reconstructed compared to $\mathcal{G}_{LCT}$. Although the number of 3D points reconstructed is reduced compared to $\mathcal{G}_{LCT}$, they have lower reprojection errors, indicating that the estimated camera parameters (intrinsics and motions) and 3D points in $\mathcal{G}_F$ are more consistent with the observations (epipolar inliers on the edges) compared to the original graph. Moreover, the ghost artifacts in the reconstructions are also removed in sparsified graphs, along with a significant reduction in the reconstruction time using COLMAP, by a margin of 50% to 80%. Our method takes $< 1\%$ of the reconstruction time to execute in most cases. More details are provided in the supplementary material. These results are in accordance with the goals of sparsification to obtain a reconstruction with most cameras and 3D points recovered with similar or improved accuracy with reduced reconstruction time using a time-efficient method.

Dataset	Disambiguated				# Cameras Reconstructed ($\#N_{CR}$)				# 3D Points Reconstructed (in 10^3)				Mean Reprojection Errors (px)				Reconstruction Time (mins) (t_R)			
	$\mathcal{G}$	$\mathcal{G}_{LCT}$	$\mathcal{G}_{Dopp}$ [3]	$\mathcal{G}_F$ Ours	$\mathcal{G}$	$\mathcal{G}_{LCT}$	$\mathcal{G}_{Dopp}$ [3]	$\mathcal{G}_F$ Ours	$\mathcal{G}$	$\mathcal{G}_{LCT}$	$\mathcal{G}_{Dopp}$ [3]	$\mathcal{G}_F$ Ours	$\mathcal{G}$	$\mathcal{G}_{LCT}$	$\mathcal{G}_{Dopp}$ [3]	$\mathcal{G}_F$ Ours	$\mathcal{G}$	$\mathcal{G}_{LCT}$	$\mathcal{G}_{Dopp}$ [3]	$\mathcal{G}_F$ Ours
Louvre [46]	✓	✓	✓	✓	367	359	320	186	128	129	108	113	**0.60**	**0.60**	**0.60**	**0.60**	95	90	79	**52**
Notre Dame [46]	✗	✗	✓	✓	7952	7778	5481	5420	1785	1775	1314	1306	0.73	0.74	0.74	**0.72**	1558	1686	1160	**1059**
Sacre Coeur [46]	✗	✗	✓	✓	4492	4443	3796	1984	711	706	548	347	0.70	0.71	0.71	**0.61**	385	405	347	**156**
Seville [46]	✗	✗	✓	✓	1510	1498	447	475	353	353	109	117	0.68	0.68	0.64	**0.62**	203	113	87	**24**
Ellis Island [47]	✗	✗	✓	✓	880	910	314	333	164	171	84	89	0.76	0.75	0.80	**0.79**	212	195	33	**26**
Piazza del Popolo [47]	✗	✗	✓	✓	1023	1012	922	865	138	136	123	110	0.69	0.69	0.68	**0.67**	178	184	128	**99**
Yorkminster [47]	✗	✗	✓	✓	1065	1027	585	520	284	280	173	162	0.75	0.75	0.77	**0.76**	383	412	113	**86**
Alexander Nevsky Cathedral [17]	✗	✗	✓	✓	447	447	445	429	100	100	90	83	0.67	0.67	0.68	**0.62**	32	28	31	**13**
Arc de Triomphe [17]	✗	✗	✓	✓	427	425	392	394	81	81	69	70	0.66	0.66	0.66	**0.64**	22	18	17	**15**
Berliner Dom [17]	✗	✗	✓	✓	1603	1603	1600	1588	242	242	238	234	0.70	0.70	0.70	**0.68**	142	118	126	**104**
Big Ben [17]	✗	✗	✓	✓	397	398	394	379	74	74	73	67	0.64	0.64	0.64	**0.61**	40	36	43	**33**
Brandenburg Gate [17]	✗	✗	✓*	✓*	172	172	151	129	24	24	21	14	0.84	0.84	0.87	**0.73**	8	8	9	**2**
Church on Spilled Blood [17]	✗	✗	✗	✓	273	274	258	136	69	69	64	30	0.61	0.61	0.61	**0.52**	18	18	16	**5**
Indoor [17]	✗	✗	✓	✓	152	152	152	42	73	73	58	9	0.52	0.55	0.53	**0.40**	9	8	8	**1**
Radcliffe Camera [17]	✗	✗	✓*	✓*	279	280	94	177	76	76	28	40	0.64	0.64	0.60	**0.58**	18	18	**6**	**6**
Books [49]	✗	✗	✗	✓*	21	21	21	9	8	8	7	2	0.41	0.41	0.41	**0.31**	0.3	0.3	0.3	**0.1**
Cereal [49]	✗	✗	✗	✓*	25	25	25	7	12	12	12	3	0.41	0.41	0.41	**0.30**	0.5	0.5	0.5	**0.1**
Cup [49]	✗	✗	✓	✗	64	64	63	40	6	6	6	5	0.61	0.61	**0.39**	0.42	0.5	0.6	**0.3**	0.3
Desk [49]	✓	✓	✓	✓*	31	31	31	12	14	14	12	3	0.49	0.49	0.48	**0.40**	1.0	1.1	0.8	**0.2**
Oats [49]	✗	✗	✗	✓*	23	23	23	9	8	8	7	3	0.40	0.39	0.37	**0.25**	0.4	0.5	0.4	**0.1**
Street [49]	✗	✗	✓*	✓	19	19	10	19	4	4	1	2	0.50	0.50	0.37	**0.35**	0.2	0.2	**0.1**	**0.1**
Temple of Heaven [49]	✓	✓	✓	✓	338	338	338	338	192	192	185	196	0.65	0.65	**0.63**	0.65	78	76	**32**	54

Table 3. Reconstruction statistics on ambiguous datasets. Please refer to text for details on probability threshold (p) used for Doppelgangers [3] and minimmum edge score (m) for our method (Algo. 1). ✓/✓*/✗: Disambiguated/Disambiguated but oversplit/Non-disambiguated reconstructions. **Bold** values indicate best in the respective category. The best value is checked only across disambiguated/disambiguated but oversplit reconstructions. Our method disambiguates all datasets except Cup (and also sparsifies them), reconstructing most cameras and 3D points with reduced reprojection errors and reconstruction time.

In Fig. 3, we show the reconstructions of two datasets, with Gendarmenmarkt containing a ghost artifact. It can be seen that extracting $\mathcal{G}_{LCT}$ from $\mathcal{G}$ does not remove the ghost artifact and requires further processing to avoid it. The reconstructions obtained with sparsified viewgraphs are visually similar to those from $\mathcal{G}$ and $\mathcal{G}_{LCT}$ (after removing ghost artifacts, if present), even with some lost cameras, in significantly less time. This reveals the advantage of scoring edges based on the number of epipolar inliers.

4.2. Ambiguous Datasets

In this subsection, we evaluate the performance of our method on ambiguous datasets. At first, we study how well the edge scores reflect the presence of false edges. We take datasets provided by Doppelgangers [3] since they provide ground truth labels for true and false edges. We obtain viewgraphs using COLMAP and compute the edge quality scores based on Algo. 1. In Fig. 4, we provide separate histograms of the edge quality scores for true (non-ambiguous) and false (ambiguous) edges for all the datasets. We only use those edges for the histograms whose labels are provided in the datasets. It can be seen that the ambiguous edges are scored low, which is desired for their removal using our method. For non-ambiguous edges, the edge scores are distributed more uniformly, which helps in the uniform removal of edges for graph sparsification with our method. Thus, our edge scoring mechanism is able to capture the presence of false edges and redundancy of the graph based on connectivity via 3D points.

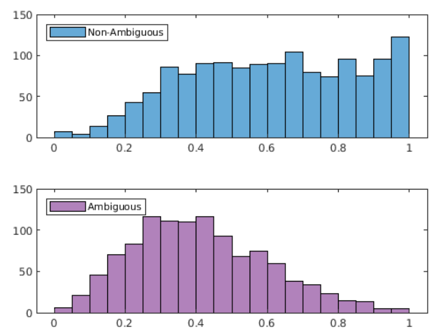


Figure 4. Histograms of edge quality scores over all datasets from Doppelgangers [3]. Top: Only non-ambiguous edges. Bottom: Only ambiguous edges. Ambiguous edges have low scores. Non-ambiguous edges are scored more uniformly, which aids uniform edge removal for graph sparsification.

Next, we present results on ambiguous datasets provided by [17, 46, 47, 49]. In Table 3, we compare results of the disambiguated viewgraph $\mathcal{G}_F$ to those of viewgraphs $\mathcal{G}$ and $\mathcal{G}_{LCT}$. We use $m = 0.6$ for large-scale datasets [46, 47], the same as used for generic datasets, except for highly ambiguous datasets (Louvre and Sacre Coeur), where we use $m = 0.9$. We use $m = 0.3$ for medium and small-scale datasets [17, 49]. We also provide reconstruction statistics after applying Doppelgangers [3] on $\mathcal{G}$, where we use probability threshold $p = 0.8$ for all datasets (as suggested in [3]) except for Louvre, where $p = 0.9$. No value of p worked for the datasets not disambiguated by [3]. We

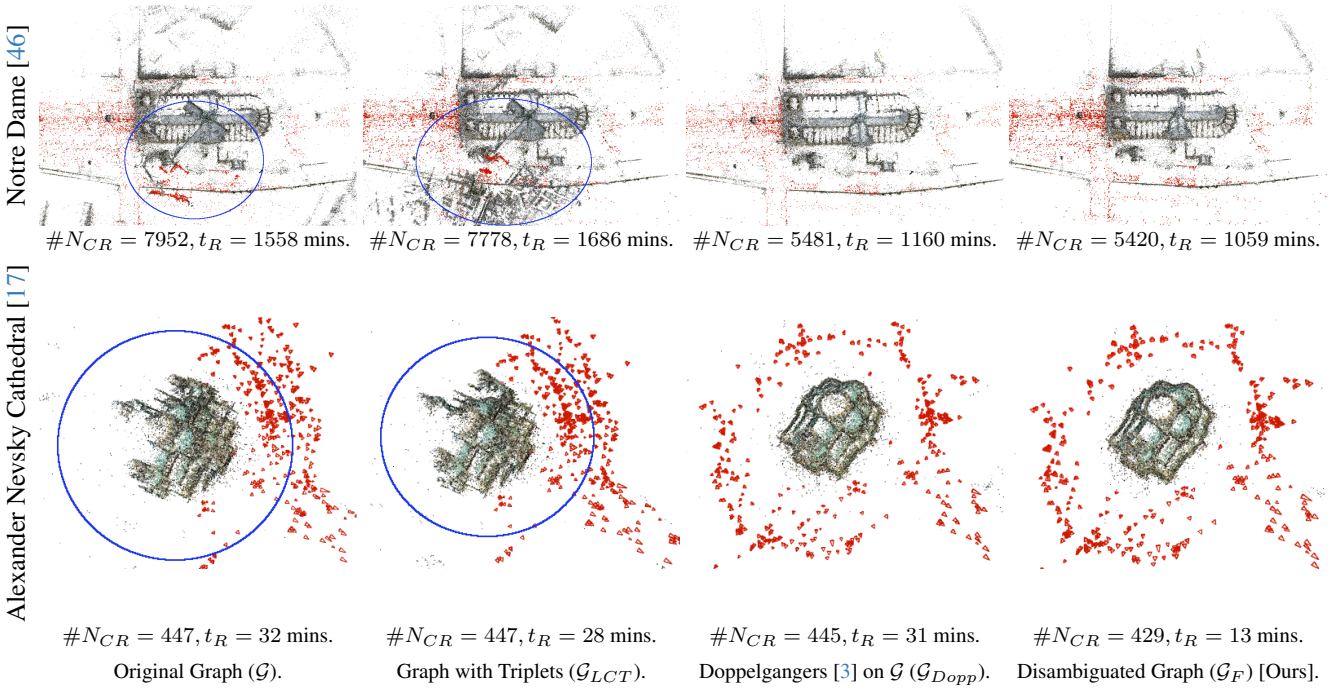


Figure 5. Reconstructions obtained with different graphs on ambiguous datasets. $\#N_{CR}$: Number of cameras reconstructed, t_R: Reconstruction time using COLMAP [36]. Datasets resulting in superimposed reconstructions (marked in *blue*) are corrected after applying our method with faster reconstruction time compared to Doppelgangers [3] due to the sparsification behaviour of our method.

note that $\mathcal{G}_{LCT}$ still contains false edges, due to which no dataset is disambiguated by extracting $\mathcal{G}_{LCT}$ from $\mathcal{G}$. Our method is able to disambiguate repeated structures for all large and medium-scale datasets, recovering most of the cameras and 3D points when compared to Doppelgangers reconstructions, with similar or better reprojection errors for almost all the datasets. For Louvre, although many cameras are lost, we still reconstruct more 3D points than Doppelgangers with reprojection errors close to each other, showing the effectiveness of our edge-scoring mechanism. Additionally, apart from removing false edges, our method also sparsifies the graphs, which reduces the reconstruction times for most datasets. More details are given in the supplementary material. Although similar observations can be made for small-scale datasets [49], some scenes are over-split due to graph sparsification by our method because these datasets [49], being sequential, have a low redundancy of edges.

In Fig. 5, we show reconstructions of two datasets, one large-scale and another medium-scale. Both $\mathcal{G}$ and $\mathcal{G}_{LCT}$ lead to superimposed reconstructions due to the presence of false edges. Our method is able to disambiguate different facades of the buildings, leading to correct reconstruction, as validated by Doppelgangers reconstruction. Moreover, due to graph sparsification, our method significantly reduces reconstruction time compared to Doppelgangers. More results are provided in the supplementary material.

5. Conclusion

In this paper, we present a single method based on camera triplets to solve both the problems of 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 ambiguous edges based on global information of 3D point connectivity. We formulate an optimization problem whose solution is proven to be a thresholding scheme. This leads us to an efficient algorithm which can be incorporated into any SfM pipeline as a pre-processing step. Applying our method reduces the reconstruction time significantly while maintaining accuracy and also removes ghost artifacts on generic datasets. Our method also disambiguates repeated structures in the scenes along with sparsifying viewgraphs of ambiguous datasets, resulting in reduced reconstruction time.

Acknowledgments: Lalit Manam is 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.

References

[1] Relja Arandjelovic, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. Netvlad: Cnn architecture for weakly supervised place recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 5297–5307, 2016. 1

[2] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). Computer vision and image understanding, 110(3):346–359, 2008. 1

[3] Ruojin Cai, Joseph Tung, Qianqian Wang, Hadar Averbuch-Elor, Bharath Hariharan, and Noah Snavely. Doppelgangers: Learning to disambiguate images of similar structures. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 34–44, 2023. 1, 3, 4, 5, 7, 8

[4] Andrea Cohen, Christopher Zach, Sudipta N Sinha, and Marc Pollefeys. Discovering and exploiting 3d symmetries in structure from motion. In 2012 IEEE Conference on Computer Vision and Pattern Recognition, pages 1514–1521. IEEE, 2012. 1, 3

[5] David Crandall, Andrew Owens, Noah Snavely, and Dan Huttenlocher. Discrete-continuous optimization for large-scale structure from motion. In CVPR 2011, pages 3001–3008. IEEE, 2011. 5, 6

[6] Zhaopeng Cui and Ping Tan. Global structure-from-motion by similarity averaging. In Proceedings of the IEEE International Conference on Computer Vision, pages 864–872, 2015. 3

[7] François Darmon, Mathieu Aubry, and Pascal Monasse. Learning to guide local feature matches. In 2020 International Conference on 3D Vision (3DV), pages 1127–1136. IEEE, 2020. 1

[8] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. In Proceedings of the IEEE conference on computer vision and pattern recognition workshops, pages 224–236, 2018. 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. IEEE, 2011. 1

[10] Venu Madhav Govindu. Combining two-view constraints for motion estimation. In Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001, pages II–II. IEEE, 2001.

[11] Venu Madhav Govindu. Lie-algebraic averaging for globally consistent motion estimation. In Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004., pages I–I. IEEE, 2004. 1

[12] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. Exploring network structure, dynamics, and function using networkx. In Proceedings of the 7th Python in Science Conference, pages 11 – 15, Pasadena, CA USA, 2008. 5

[13] R. I. Hartley and A. Zisserman. Multiple View Geometry in Computer Vision. Cambridge University Press, ISBN: 0521540518, second edition, 2004. 1

[14] Stephen Hausler, Sourav Garg, Ming Xu, Michael Milford, and Tobias Fischer. Patch-netvlad: Multi-scale fusion of locally-global descriptors for place recognition. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 14141–14152, 2021. 1

[15] Michal Havlena, Akihiko Torii, Jan Knopp, and Tomas Pajdla. Randomized structure from motion based on atomic 3d models from camera triplets. In 2009 IEEE Conference on Computer Vision and Pattern Recognition, pages 2874–2881. IEEE, 2009. 1, 2, 3

[16] Michal Havlena, Akihiko Torii, and Tomáš Pajdla. Efficient structure from motion by graph optimization. In Computer Vision–ECCV 2010: 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part II 11, pages 100–113. Springer, 2010. 1, 2, 3

[17] Jared Heinly, Enrique Dunn, and Jan-Michael Frahm. Correcting for duplicate scene structure in sparse 3d reconstruction. In Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part IV 13, pages 780–795. Springer, 2014. 1, 2, 3, 4, 7, 8

[18] Jared Heinly, Enrique Dunn, and Jan-Michael Frahm. Recovering correct reconstructions from indistinguishable geometry. In 2014 2nd International Conference on 3D Vision, pages 377–384. IEEE, 2014. 3

[19] Nianjuan Jiang, Ping Tan, and Loong-Fah Cheong. Seeing double without confusion: Structure-from-motion in highly ambiguous scenes. In 2012 IEEE Conference on Computer Vision and Pattern Recognition, pages 1458–1465. IEEE, 2012. 1, 3, 4

[20] 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, pages 481–488, 2013. 4

[21] Rajbir Kataria, Joseph DeGol, and Derek Hoiem. Improving structure from motion with reliable resectioning. In 2020 international conference on 3D vision (3DV), pages 41–50. IEEE, 2020. 1, 3

[22] Jan Knopp, Josef Sivic, and Tomas Pajdla. Avoiding confusing features in place recognition. In Computer Vision–ECCV 2010: 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part I 11, pages 748–761. Springer, 2010. 3

[23] Brian Kulis and Kristen Grauman. Kernelized locality-sensitive hashing for scalable image search. In 2009 IEEE 12th international conference on computer vision, pages 2130–2137. IEEE, 2009. 1

[24] Yunpeng Li, Noah Snavely, and Daniel P Huttenlocher. Location recognition using prioritized feature matching. In Computer Vision–ECCV 2010: 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part II 11, pages 791–804. Springer, 2010. 5, 6

[25] Philipp Lindenberger, Paul-Edouard Sarlin, and Marc Pollefeys. Lightglue: Local feature matching at light speed. In

Proceedings of the IEEE/CVF International Conference on Computer Vision, 2023. 1

[26] Yanxi Liu, Hagit Hel-Or, Craig S Kaplan, Luc Van Gool, et al. Computational symmetry in computer vision and computer graphics. Foundations and Trends® in Computer Graphics and Vision, 5(1–2):1–195, 2010. 1

[27] David G Lowe. Distinctive image features from scale-invariant keypoints. International journal of computer vision, 60:91–110, 2004. 1

[28] Lalit Manam and Venu Madhav Govindu. Sensitivity in translation averaging. In Thirty-seventh Conference on Neural Information Processing Systems, 2023. 4

[29] Geoffrey J. McLachlan and Thriyambakam Krishnan. The EM algorithm and extensions. Wiley, 2. ed edition, 2008. 3

[30] Krystian Mikolajczyk and Cordelia Schmid. A performance evaluation of local descriptors. IEEE transactions on pattern analysis and machine intelligence, 27(10):1615–1630, 2005. 4

[31] David Nister and Henrik Stewenius. Scalable recognition with a vocabulary tree. In 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06), pages 2161–2168. Ieee, 2006. 1

[32] Hyeonwoo Noh, Andre Araujo, Jack Sim, Tobias Weyand, and Bohyung Han. Large-scale image retrieval with attentive deep local features. In Proceedings of the IEEE international conference on computer vision, pages 3456–3465, 2017. 1

[33] Richard Roberts, Sudipta N Sinha, Richard Szeliski, and Drew Steedly. Structure from motion for scenes with large duplicate structures. In CVPR 2011, pages 3137–3144. IEEE, 2011. 1, 3, 4

[34] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In 2011 International conference on computer vision, pages 2564–2571. IEEE, 2011. 1

[35] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superglue: Learning feature matching with graph neural networks. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pages 4938–4947, 2020. 1

[36] Johannes Lutz Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 4104–4113, 2016. 1, 5, 6, 8

[37] Rajvi Shah, Visesh Chari, and PJ Narayanan. View-graph selection framework for sfm. In Proceedings of the European Conference on Computer Vision (ECCV), pages 535–550, 2018. 1, 2, 3

[38] Tianwei Shen, Siyu Zhu, Tian Fang, Runze Zhang, and Long Quan. Graph-based consistent matching for structure-from-motion. In Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part III 14, pages 139–155. Springer, 2016. 1, 2, 3

[39] Noah Snavely, Steven M Seitz, and Richard Szeliski. Photo tourism: exploring photo collections in 3d. In ACM siggraph 2006 papers, pages 835–846. 2006. 1

[40] Noah Snavely, Steven M Seitz, and Richard Szeliski. Modeling the world from internet photo collections. International journal of computer vision, 80:189–210, 2008. 1

[41] Noah Snavely, Steven M Seitz, and Richard Szeliski. Skeletal graphs for efficient structure from motion. In 2008 IEEE Conference on Computer Vision and Pattern Recognition, pages 1–8. IEEE, 2008. 1, 2, 3

[42] Christopher Sweeney, Tobias Hollerer, and Matthew Turk. Theia: A fast and scalable structure-from-motion library. In Proceedings of the 23rd ACM international conference on Multimedia, pages 693–696, 2015. 1, 5

[43] Akihiko Torii, Josef Sivic, Tomas Pajdla, and Masatoshi Okutomi. Visual place recognition with repetitive structures. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 883–890, 2013. 3

[44] Michał Tyszkiewicz, Pascal Fua, and Eduard Trulls. Disk: Learning local features with policy gradient. Advances in Neural Information Processing Systems, 33:14254–14265, 2020. 1

[45] Fan Wang, Aditi Nayak, Yogesh Agrawal, and Roy Shilkrot. Hierachical image link selection scheme for duplicate structure disambiguation. In BMVC, page 221, 2018. 1, 3

[46] Kyle Wilson and Noah Snavely. Network principles for sfm: Disambiguating repeated structures with local context. In Proceedings of the IEEE International Conference on Computer Vision, pages 513–520, 2013. 1, 3, 7, 8

[47] Kyle Wilson and Noah Snavely. Robust global translations with 1dsfm. In European Conference on Computer Vision, pages 61–75. Springer, 2014. 2, 5, 6, 7

[48] Changchang Wu et al. Visualsfm: A visual structure from motion system. 2011. 1

[49] Qingan Yan, Long Yang, Ling Zhang, and Chunxia Xiao. Distinguishing the indistinguishable: Exploring structural ambiguities via geodesic context. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 3836–3844, 2017. 1, 3, 4, 7, 8

[50] Kwang Moo Yi, Eduard Trulls, Yuki Ono, Vincent Lepetit, Mathieu Salzmann, and Pascal Fua. Learning to find good correspondences. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 2666–2674, 2018. 1

[51] Christopher Zach, Arnold Irschara, and Horst Bischof. What can missing correspondences tell us about 3d structure and motion? In 2008 IEEE Conference on Computer Vision and Pattern Recognition, pages 1–8. IEEE, 2008. 1, 2, 3, 4

[52] Christopher Zach, Manfred Klopschitz, and Marc Pollefeys. Disambiguating visual relations using loop constraints. In 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pages 1426–1433. IEEE, 2010. 1, 3