Viewing Graph Solvability via Cycle Consistency

Federica Arrigoni[1], Andrea Fusiello[2], Elisa Ricci[1,3] and Tomas Pajdla[4]
[1]University of Trento [2]University of Udine [3]Fondazione Bruno Kessler [4]CIIRC CTU in Prague
federica.arrigoni@unitn.it, andrea.fusiello@uniud.it, e.ricci@unitn.it, pajdla@cvut.cz

Abstract

In structure-from-motion the viewing graph is a graph where vertices correspond to cameras and edges represent fundamental matrices. We provide a new formulation and an algorithm for establishing whether a viewing graph is solvable, i.e. it uniquely determines a set of projective cameras. Known theoretical conditions either do not fully characterize the solvability of all viewing graphs, or are exceedingly hard to compute for they involve solving a system of polynomial equations with a large number of unknowns. The main result of this paper is a method for reducing the number of unknowns by exploiting the cycle consistency. We advance the understanding of the solvability by (i) finishing the classification of all previously undecided minimal graphs up to 9 nodes, (ii) extending the practical solvability testing up to minimal graphs with up to 90 nodes, and (iii) definitely answering an open research question by showing that the finite solvability is not equivalent to the solvability. Finally, we present an experiment on real data showing that unsolvable graphs are appearing in practical situations.

1. Introduction

Solving structure-from-motion is important, e.g., for 3D reconstruction from images [33, 34, 29], image matching [26] and visual odometry and localization [22, 1, 28, 35]. The basic problem in structure-from-motion is to determine which image sets can be reconstructed. This can be done, e.g., by analyzing the viewing graph.

The *viewing graph* [19] (also known as the *epipolar graph*) of a set of images is a graph where vertices correspond to cameras/images and edges correspond to fundamental matrices. In other terms, an edge is present between two vertices if and only if the fundamental matrix between such image pair is available, meaning that there exist enough corresponding points. In most practical scenarios such graph is not complete due to the fact that different cameras may view different portions of the scene.

A relevant question is whether a viewing graph is *solvable*, i.e., if it uniquely determines a projective configuration of cameras, up to a *single* projective transformation. In other terms, for a non-solvable viewing graph there exist *multiple* transformations that can be applied to the cameras without affecting the fundamental matrices. An equivalent definition of solvability is given in [19], stating that a graph is solvable if and only if the available fundamental matrices uniquely determine the remaining ones, i.e., the input graph can be transformed into the complete graph.

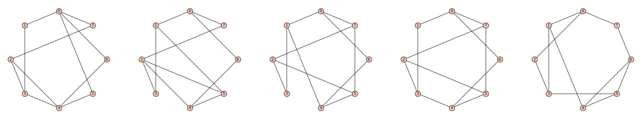
Figure 1: Viewing graphs with eight vertices that were left undecided in [37] and that we determined to be solvable.

The solvability is closely related to 3D reconstruction (see Tab. 1), since viewing graphs are used by a class of projective structure-from-motion methods [30, 16] that recover the projective cameras starting from multiple fundamental matrices. The solvability of a viewing graph should be assessed *before* addressing the reconstruction, since if a problem is non-solvable, then no method based on fundamental matrices only will provide a useful reconstruction.

Considering calibrated cameras, i.e. using the essential matrices instead of the fundamental ones, the solvable graphs are exactly those that are *parallel rigid* [24, 38]. The topic of parallel rigidity (also known as *bearing rigidity*) has been extensively studied in the literature (see [3] for a recent survey). Here we focus on the uncalibrated scenario, which has been much less understood.

Related Work. In [19], the solvability of graphs with at most six vertices is characterized, and some insights about how to analyze larger graphs are provided. A constructive method to actually complete such view graphs is discussed in [20]. Rudi *et al.* [27] studied conditions under which a viewing graph can be solved using a linear method. Further analysis is available in [36], where it is shown that some solvable graphs can be constructed starting from a triangle and adding vertices of degree two one at a time.

In a recent paper [37], some necessary conditions for

	calibrated	uncalibrated
solvability	[3]	[19, 27, 37]+ours
reconstruction	[25]	[30, 16]

Table 1: Problems taxonomy. We address the uncalibrated case. Surveys [3, 25] address the calibrated situation.

solvability are reported in addition to a new sufficient condition. Thanks to these results, graphs with at most seven vertices can be completely characterized, but there exist some graphs with eight or nine vertices that could not be classified into solvable or non solvable (see Fig. 1). Larger graphs were not studied. The authors of [37] also show that – *in principle* – it is possible to classify any viewing graph via a system of polynomial equations. However, this observation has only a theoretical value: solving such system is computationally expensive for it is nonlinear and it involves a large number of unknowns. Thus, an effective test for solvability is still missing, which motivates our work.

Another open issue has to do with the concept of *finite solvability*, which is tightly related to the notion of solvability. A viewing graph is called finite solvable if it determines at most a *finite* number of projective configurations of cameras. Obviously, if a graph is not finite solvable (i.e., it determines an *infinite* number of camera configurations) then it is not solvable. In other terms, solvability implies finite solvability. The reverse implication – namely, *does finite solvability imply solvability*? – was left as an open research question by [37].

Contribution. We derive a new formulation of viewing graph solvability that is much simpler than [37] thanks to a substantial reduction of the number of unknowns involved. Our formulation is based on the *cycle consistency* property of a graph, namely that the composition of (invertible) transformations along any cycle should be the identity.

This leads to a new algorithm, which is based on computational algebraic geometry, that implements a characterization of solvability. Previous techniques could only test either some sufficient or necessary conditions. Using this algorithm, we provide a complete characterization of *all* the minimal graphs up to nine nodes, including those that were left undecided by [37].

As a matter of fact, we are able to decide the solvability of minimal graphs with up to 90 nodes. In practice, our approach can be used to detect interesting solvable sub-graphs of dense/large viewing graphs coming from real data sets.

Finally, thanks to our algorithm, it is possible to exhibit concrete examples of graphs that are finite solvable but *not* solvable, thereby answering the open research question from [37]. In summary, we:

- propose a new simpler formulation of the solvability;
- build an effective algorithm for testing it;
- classify previously undecided viewing graphs;
- extend solvability testing up to graphs with 90 nodes;
- prove that finite solvability does not imply solvability.

The paper is organized as follows. We define the viewing graph solvability in Sec. 2, present theoretical results in Sec. 3, and describe our algorithm in Sec. 4. Sec. 5 shows some examples and presents experiments on real data.

2. Background

Let us consider n uncalibrated cameras $P_1, \dots P_n \in \mathbb{R}^{3\times 4}$ with centres $\mathbf{c}_1, \dots, \mathbf{c}_n \in \mathbb{R}^4$. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an undirected graph with vertex set $\mathcal{V} = \{1, \dots, n\}$ and edge set $\mathcal{E} \subseteq \{1, \dots, n\} \times \{1, \dots, n\}$. Let $m = |\mathcal{E}|$ be the number of edges. Recall that for each edge $(i, j) \in \mathcal{E}$ we can compute the fundamental matrix F_{ij} relating cameras i and j in a closed-form [15]. Conversely, the fundamental matrix of edge (i, j) uniquely determines the cameras of vertices i and j, up to a projective transformation [15].

In the following, we shall use uppercase letters to denote matrices, lowercase bold letters to denote vectors and lowercase letters to denote scalars[1]. Projective quantities are represented as non-homogeneous variables and suitable scales are introduced to handle the scale ambiguity.

Definition 1. Let $\mathcal{G}$ be a viewing graph and $\mathcal{P} = \{P_1, \dots, P_n\}$ be a set of cameras. The pair $(\mathcal{G}, \mathcal{P})$ is called *solvable* if all the camera configurations yielding the same fundamental matrices as $\mathcal{P}$ are projectively equivalent, i.e. they are related by the *same* projective transformation.

Proposition 1 ([37]). *Let $\mathcal{G}$ be a viewing graph and $\mathcal{P} = \{P_1, \dots, P_n\}$ be a set of cameras with centres $\mathbf{c}_1, \dots, \mathbf{c}_n \in \mathbb{R}^4$. The solvability of the pair $(\mathcal{G}, \mathcal{P})$ only depends on the graph $\mathcal{G}$ and on the camera centres $\mathbf{c}_1, \dots, \mathbf{c}_n$.*

According to the above result, if a problem is non-solvable, then the cause can be either the topology of the graph or the actual coordinates of the centres. For instance, if the centres are all aligned, then the problem is not solvable (see [19] for more examples). The following concept – which is the main focus of this paper – permits to predicate the solvability of a problem based on the graph topology only.

Definition 2. A graph $\mathcal{G}$ is called *solvable* if it is solvable for a *generic* configuration of cameras.

Necessary conditions for viewing graph solvability [19, 37] allow to quickly prune the solvability candidates. For instance, a solvable graph must satisfy:

- it has at least $(11n - 15)/7$ edges [37];

[1]Observe that this notation is different from the one used in [37].

- it is biconnected [37];
- all the vertices have degree at least two and no two adjacent vertices have degree two (if $n > 3$) [19].

Concerning sufficient conditions, it is known that any chordal (or triangulated) graph is solvable [36]. Other constructive approaches are proposed in [27, 36, 37], where the idea is to check if the input graph can be transformed into a solvable one via suitable operations. The only condition that is *both* necessary and sufficient is derived in [37], and it will be reviewed next.

2.1. Algebraic characterization

Trager *et al.* [37] linked viewing graph solvability to the characterization of the set of projective transformations that can be applied to all cameras without affecting the fundamental matrices. First of all, they identify the family of transformations that leave a camera matrix fixed.

Proposition 2 ([37]). *Let $P \in \mathbb{R}^{3\times4}$ be a camera with centre $\mathbf{c} \in \mathbb{R}^4$. All the solutions to $PG = aP$ for $G \in GL(4,\mathbb{R})$ and $a \in \mathbb{R}_{\neq 0}$ are described by*

$$G = aI_4 + \mathbf{c}\mathbf{v}^\mathsf{T} \quad \forall a \in \mathbb{R}_{\neq 0}, \mathbf{v} \in \mathbb{R}^4 \tag{1}$$

where I_4 denotes the 4×4 identity matrix and $GL(4,\mathbb{R})$ denotes the group of real 4×4 invertible matrices.

Let us consider a viewing graph $\mathcal{G} = (\mathcal{V},\mathcal{E})$ and let us assign a projective transformation $G_{ij} \in GL(4,\mathbb{R})$ to every edge $(i,j) \in \mathcal{E}$. It is understood that this transformation is to be applied to the two adjacent cameras P_i and P_j. Clearly this does not change the fundamental matrix F_{ij}, for it is invariant under projective transformations.

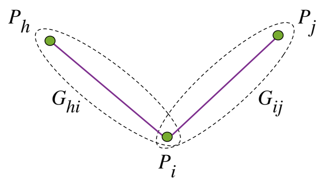

Figure 2: Two adjacent edges in the viewing graph.

If we were dealing with a single edge (i.e., a pair of cameras), we would be free to choose any G_{ij}. However, when dealing with multiple edges (i.e., when considering the whole viewing graph), these matrices must satisfy global compatibility constraints. Specifically, let us consider two adjacent edges $(h,i) \in \mathcal{E}$ and $(i,j) \in \mathcal{E}$ that are both incident to vertex $i \in \mathcal{V}$ and to which the two transformations G_{hi} and G_{ij} are assigned (see Fig. 2). Such transformations must leave the camera at the common vertex fixed, resulting in the following *compatibility* constraint:

$$G_{hi}G_{ij}^{-1} = a_{hij}I_4 + \mathbf{c}_i\mathbf{v}_{hij}^\mathsf{T} \tag{2}$$

where $a_{hij} \in \mathbb{R}_{\neq 0}$ and $\mathbf{v}_{hij} \in \mathbb{R}^4$ are unknown.

Definition 3. Let $\mathcal{G}$ be a graph and let $\mathbf{c}_1,\dots,\mathbf{c}_n \in \mathbb{R}^4$ be n generic camera centres. Any assignment of transformations $G_{ij} \in GL(4,\mathbb{R})$ to the edges of the graph, such that Eq. (2) holds true for all adjacent edges, is called *compatible*.

Proposition 3 ([37]). *Let $\mathcal{G}$ be a graph and let $\mathbf{c}_1,\dots,\mathbf{c}_n \in \mathbb{R}^4$ be n generic camera centres. $\mathcal{G}$ is solvable if and only if any compatible assignment is of the form*

$$G_{ij} = s_{ij}H \quad \forall (i,j) \in \mathcal{E} \tag{3}$$

where $H \in GL(4,\mathbb{R})$ and $s_{ij} \in \mathbb{R}_{\neq 0}$.

The condition in Prop. 3 means that, for a solvable graph, the only way to act on all the cameras (without affecting the fixed fundamental matrices) is to apply a *single* projective transformation. Note that the centres can be sampled at random, in order to satisfy the assumption of generic cameras.

Finding a compatible assignment of matrices, i.e., solving Eq. (2) for all adjacent edges simultaneously, is very challenging since it defines a non-linear algebraic system with a large number of unknowns. For this reason, Prop. 3 is given as a theoretical result in [37], without leading to a practical algorithm for checking viewing-graph solvability. The next section explains how we alleviate this drawback.

3. Proposed Formulation

Given a graph $\mathcal{G} = (\mathcal{V},\mathcal{E})$ with n vertices and m edges, our task is to establish whether such a graph is solvable according to Def. 2. Throughout our discussion we assume that $\mathcal{G}$ is connected and $n \geqslant 3$. The proposed formulation is inspired by the algebraic characterization detailed in Sec. 2.1. Specifically, we show how to reduce the number of unknowns in Eq. (2), thus providing a more practical way for checking viewing graph solvability.

3.1. Solvability on the Line Graph

Our formulation involves the *line graph* associated with $\mathcal{G}$, which is constructed as in the following definition.

Definition 4. Given an undirected graph $\mathcal{G}$, its *line graph* (also called *edge-to-vertex dual graph*) is denoted by $\mathcal{L}(\mathcal{G}) = (\bar{\mathcal{V}},\bar{\mathcal{E}})$ and it is another undirected graph such that:

- each vertex of $\mathcal{L}(\mathcal{G})$ represents an edge of $\mathcal{G}$;
- two vertices of $\mathcal{L}(\mathcal{G})$ are adjacent if and only if their corresponding edges are adjacent in $\mathcal{G}$, i.e, they are incident to the same vertex.

Figure 3 shows an example. The number of vertices in the line graph coincides with the number of edges in the original graph, i.e., $\bar{n} = m$, whereas the number of edges in the line graph is given by the following formula [14]:

$$\bar{m} = \frac{1}{2}\sum_{i=1}^{n} d_i^2 - m \tag{4}$$

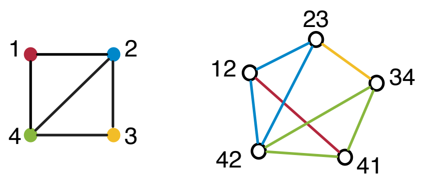

Figure 3: Viewing graph with 4 vertices (left) and corresponding line graph (right). Please note that a vertex of the original graph (e.g., vertex 2) can appear multiple times as an edge of the line graph, as clarified by colors.

where d_i denotes the degree of vertex $i \in \mathcal{V}$.

Let us rewrite Eq. (2) in terms of the line graph $\mathcal{L}(\mathcal{G}) = (\bar{\mathcal{V}}, \bar{\mathcal{E}})$. Note that the edge $(h, i) \in \mathcal{E}$ is a vertex $\tau \in \bar{\mathcal{V}}$ and similarly $(i, j) \in \mathcal{E}$ represents a vertex $\upsilon \in \bar{\mathcal{V}}$. Such vertices are connected by an edge $(\tau, \upsilon) \in \bar{\mathcal{E}}$ by construction (as they share vertex $i \in \mathcal{V}$ in the input graph). Hereafter we use Greek letters to denote vertices/edges in the line graph. Using this notation Eq. (2) becomes:

$$G_\tau G_\upsilon^{-1} = Z_{\tau\upsilon} \tag{5}$$

where:

$$Z_{\tau\upsilon} = a_{\tau\upsilon} I_4 + \mathbf{c}_i \mathbf{v}_{\tau\upsilon}^\mathsf{T} \tag{6}$$

and the index i of the camera is defined as $\{i\}{=}\tau \cap \upsilon$.

Definition 5. Let $\mathcal{G}$ be a graph and let $\mathbf{c}_1, \dots, \mathbf{c}_n \in \mathbb{R}^4$ be n generic camera centres. Any assignment of transformations $G_\tau \in GL(4, \mathbb{R})$ to the vertices of the line graph $\mathcal{L}(\mathcal{G})$ such that Eq. (5) holds for all the edges is called a *consistent labelling*.

Remark 1. A consistent labelling of the line graph corresponds to a compatible assignment on the original graph (see Def. 3). We give this equivalent definition here to outline the link with the problem of *synchronization* [32, 10, 5] (see [4] for a recent survey), where the task is finding a consistent labelling (of vertices) starting from measured ratios on the edges. Specifically, with reference to Eq. (5), the task would be to compute $G_\tau, G_\upsilon, \dots$ starting from known $Z_{\tau\upsilon}$. In our case, however, the variables $Z_{\tau\upsilon} \in GL(4)$ are *unknown* for all $(\tau, \upsilon) \in \bar{\mathcal{E}}$. Nevertheless, the framework of synchronization is useful to derive a new formulation of solvability, as it will be clarified in the next subsection (see the proof of Thm. 1).

Remark 2. The problem of finding a consistent labelling involves an equation of the form (5) for each edge in the line graph, which in turn spawns 16 equations when considered entry-wise. Thus, using Eq. (4), the total number of equations is given by:

$$16\bar{m} = 16\big(\frac{1}{2}\sum_{i=1}^{n} d_i^2 - m\big). \tag{7}$$

Observe that there are 16 unknowns for each node in the line graph, representing a matrix $G_\tau \in GL(4, \mathbb{R})$. In addition, there are five unknowns for each edge in the line graph, representing a vector $\mathbf{v}_{\tau\upsilon} \in \mathbb{R}^4$ and a scale $a_{\tau\upsilon} \in \mathbb{R}_{\neq 0}$. Thus, the total number of unknowns is given by:

$$16\bar{n} + 5\bar{m} = \frac{5}{2}\sum_{i=1}^{n} d_i^2 + 11m \tag{8}$$

where $\bar{n} = m$ by construction and $\bar{m}$ is given by Eq. (4). Recall that the camera centres are considered known as they are sampled at random in practice (see [37]).

Reasoning on the line graph, we are able to prove the following result, which gives a characterization of solvability in terms of the variables $\mathbf{v}_{\tau\upsilon} \in \mathbb{R}^4$ only.

Proposition 4. *Let $\mathcal{G}$ be a graph and let $\mathbf{c}_1, \dots, \mathbf{c}_n \in \mathbb{R}^4$ be n generic camera centres. The graph $\mathcal{G}$ is solvable if and only if any consistent labelling yields:*

$$\mathbf{v}_{\tau\upsilon} = \mathbf{0} \quad \forall (\tau, \upsilon) \in \bar{\mathcal{E}}. \tag{9}$$

Proof. If $\mathcal{G}$ is solvable, then, due to Prop. 3, all the matrices G_τ represent the same projective transformation, or equivalently, they are all multiples of each other. Hence, the product $G_\tau G_\upsilon^{-1}$ is a multiple of the identity (which is denoted by $b_{\tau\upsilon} I_4$ with $b_{\tau\upsilon} \in \mathbb{R}_{\neq 0}$). Hence Eq. (6) becomes:

$$b_{\tau\upsilon} I_4 = a_{\tau\upsilon} I_4 + \mathbf{c}_i \mathbf{v}_{\tau\upsilon}^\mathsf{T} \Leftrightarrow (b_{\tau\upsilon} - a_{\tau\upsilon}) I_4 = \mathbf{c}_i \mathbf{v}_{\tau\upsilon}^\mathsf{T}. \tag{10}$$

Since the right term in the above equation is a rank-1 matrix, whereas I_4 is full rank, the only way to let the equation true is to set $b_{\tau\upsilon} - a_{\tau\upsilon} = 0$ and $\mathbf{v}_{\tau\upsilon} = \mathbf{0}$, hence we get the result. In the opposite direction, if $\mathbf{v}_{\tau\upsilon} = \mathbf{0}$ then Eq. (5) rewrites $G_\tau G_\upsilon^{-1} = a_{\tau\upsilon} I_4$ or, in other terms, $G_\tau = a_{\tau\upsilon} G_\upsilon$. Such an equation can be propagated through all vertices $\tau \in \bar{\mathcal{V}}$ as soon as the line graph is connected (which is true if the original graph is connected [7]). This means that all matrices G_τ are multiples of each other, hence the graph is solvable, thanks to Prop. 3. □

Remark 3. Observe that Prop. 3 gives a formulation of solvability in terms of the matrices G_τ whereas Prop. 4 considers the variables $\mathbf{v}_{\tau\upsilon}$ only. We will show in the next subsection that the problem of finding a consistent labelling can be expressed via a system of equations *not* involving the matrices G_τ (but involving the variables $\mathbf{v}_{\tau\upsilon}$ and $a_{\tau\upsilon}$ only). In this respect, a formulation of solvability in terms of $\mathbf{v}_{\tau\upsilon}$ (as given by Prop. 4) is indeed essential.

3.2. Cycle Consistency

To derive the main result of our paper, we introduce the notion of "consistent cycle". A *cycle* is a non-empty path in which the only repeated vertices are the first and last ones. A *consistent cycle* is a cycle satisfying an algebraic constraint, as given in the following definition.

Definition 6. Let $\mathcal{G}$ be a graph and let $\mathbf{c}_1, \dots, \mathbf{c}_n \in \mathbb{R}^4$ be n generic camera centres. Let $\mathcal{C} = \{\tau_1, \tau_2, \tau_3, \dots, \tau_\ell, \tau_1\}$ be a cycle in the line graph $\mathcal{L}(\mathcal{G})$. We say that $\mathcal{C}$ is a *consistent cycle* (or a *null cycle*) if and only if the composition of the edge labels along the cycle returns the identity, namely

$$Z_{\tau_1\tau_2} Z_{\tau_2\tau_3} \cdot \dots \cdot Z_{\tau_\ell\tau_1} = I_4. \tag{11}$$

A *cycle basis* is a minimal set of cycles such that every cycle can be written as a sum of the cycles in the basis, where the sum of two cycles is a cycle where the common edges vanish. There exist several types of cycle bases (see [17] for a survey). We are interested here in a *cycle consistency basis* for the line graph, that is a cycle basis such that: if the cycles in the basis are consistent, then consistency also holds on *all* the other cycles (see [12] for details).

Theorem 1. *Let $\mathcal{G}$ be a graph and let $\mathbf{c}_1, \dots, \mathbf{c}_n \in \mathbb{R}^4$ be n generic camera centres. Let $\{\mathcal{C}_1, \mathcal{C}_2, \dots, \mathcal{C}_f\}$ be a cycle consistency basis for the line graph $\mathcal{L}(\mathcal{G})$. Let us collect in a unique system the equations of the form (11) for all the cycles in the basis. $\mathcal{G}$ is solvable if and only if the solution to such system yields $\mathbf{v}_{\tau\upsilon} = \mathbf{0}$ for all $(\tau, \upsilon) \in \bar{\mathcal{E}}$.*

Proof. It is known that there exists a consistent labelling if and only if *all* the cycles are null/consistent (see Lemma 8 in [13] and Corollary 1 in [4]). Clearly, if all cycles are consistent, then – in particular – the cycles in a basis are also consistent. The opposite does not hold in general [12]. However, if we consider a *cycle consistency basis*, then consistency on the basis implies consistency on all cycles by definition. Thus, there exists a consistent labelling if and only if all cycles in a cycle consistency basis are consistent. We now apply this general result to our problem: finding a consistent labelling, i.e., an assignment of transformations satisfying Eq. (5), is equivalent to imposing that all cycles in a cycle consistency basis of the line graph are consistent. In other terms, the system obtained by stacking equations of the form (5) for all the edges in $\mathcal{L}(\mathcal{G})$, is equivalent to the system obtained by stacking equations of the form (11) for all the cycles in a cycle consistency basis of $\mathcal{L}(\mathcal{G})$. According to Prop. 4, a graph is solvable if and only if the solution to the former yields $\mathbf{v}_{\tau\upsilon} = \mathbf{0}$ for all $(\tau, \upsilon) \in \bar{\mathcal{E}}$, hence we get the thesis. □

Remark 4. The formulation of Thm. 1 comprises five unknowns for each edge $(\tau, \upsilon) \in \bar{\mathcal{E}}$ in the line graph, representing a vector $\mathbf{v}_{\tau\upsilon} \in \mathbb{R}^4$ and a scale $a_\tau \in \mathbb{R}_{\neq 0}$. Thus, using Eq. (4), the total number of unknowns is given by

$$5\bar{m} = 5\Big(\frac{1}{2}\sum_{i=1}^{n} d_i^2 - m\Big). \tag{12}$$

Observe also that each cycle originates an equation of the form (11), which in turn spawns 16 equations when considered entry-wise. Considering the fact that the cardinality of a cycle consistency basis is the number of edges minus the number of vertices plus one (see [12]), we get the total number of equations as

$$16(\bar{m} - \bar{n} + 1) = 16\Big(\frac{1}{2}\sum_{i=1}^{n} d_i^2 - 2m + 1\Big). \tag{13}$$

Recall that the number of vertices in the line graph satisfies $\bar{n} = m$ by construction, and the number of edges $\bar{m}$ is given by Eq. (4). Note that Eq. (11) is still nonlinear, but it has the advantage of not involving the unknowns G_τ for $\tau \in \bar{\mathcal{V}}$, thus reducing the difficulty of the problem compared to Eq. (5), where the amount of unknowns is given in Eq. (8).

Remark 5. Observe that the input graph $\mathcal{G}$ is an *undirected* graph. Indeed, given a pair of cameras, or, equivalently, an edge $(i, j) \in \mathcal{E}$, the projective transformation that fixes the fundamental matrix of that camera pair is independent of the order of the cameras. In other words, $G_{ij} = G_{ji}$. When considering the line graph, instead, we are concerned with *directed* edges[2]. Indeed, $Z_{\tau\upsilon} = G_\tau G_\upsilon^{-1} = G_{hi} G_{ij}^{-1}$ and $Z_{\upsilon\tau} = G_\upsilon G_\tau^{-1} = G_{ij} G_{hi}^{-1}$ are different transformations (we are considering here $\tau = (h, i)$ and $\upsilon = (i, j)$). However, from the practical point of view, it is convenient to reduce the number of unknowns. More precisely, for a given *oriented* edge $(\tau, \upsilon) \in \bar{\mathcal{E}}$ we consider $a_{\tau\upsilon} \in \mathbb{R}_{\neq 0}$ and $\mathbf{v}_{\tau\upsilon} \in \mathbb{R}^4$ as unknowns, and we use the relation $Z_{\upsilon\tau} = Z_{\tau\upsilon}^{-1}$ to handle the opposite edge $(\upsilon, \tau) \in \bar{\mathcal{E}}$, where the inverse can be easily computed[3].

3.3. A Simplified Formulation

We now derive a simpler equivalent formulation by exploiting the change of variables.

Proposition 5. *Let $\mathcal{G}$ be a graph and let $\mathbf{c}_1, \dots, \mathbf{c}_n \in \mathbb{R}^4$ be n generic camera centres. Let $\{\mathcal{C}_1, \mathcal{C}_2, \dots, \mathcal{C}_f\}$ be a cycle consistency basis for the line graph $\mathcal{L}(\mathcal{G})$. For each cycle $\mathcal{C}_k = (\tau_1, \tau_2, \dots, \tau_\ell, \tau_1)$ in the basis, let us form the following equation:*

$$W_{\tau_1\tau_2} W_{\tau_2\tau_3} \cdot \dots \cdot W_{\tau_\ell\tau_1} = b_k I_4 \tag{14}$$

where $b_k \in \mathbb{R}_{\neq 0}$ is an unknown scale and

$$W_{\tau\upsilon} = I_4 + \mathbf{c}_i \mathbf{u}_{\tau\upsilon}^\top \tag{15}$$

where $\mathbf{u}_{\tau\upsilon} \in \mathbb{R}^4$ is unknown and $\{i\} = \tau \cap \upsilon$.
$\mathcal{G}$ is solvable if and only if the solution to the above system yields $\mathbf{u}_{\tau\upsilon} = \mathbf{0}$ for all $(\tau, \upsilon) \in \bar{\mathcal{E}}$.

[2]The line graph of an undirected graph is undirected by construction. However, it can be easily transformed into a directed graph by orienting the edges arbitrarily.

[3]The inverse of a matrix of the form $I_4 + \mathbf{c}\mathbf{v}^\top$ is given by $I_4 + \mathbf{c}\mathbf{w}^\top$ where $\mathbf{w} = -\frac{1}{1+\mathbf{c}^\top\mathbf{v}}\mathbf{v}$. Observe that $\mathbf{w}$ is a scalar multiple of $\mathbf{v}$.

	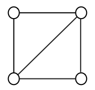		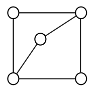		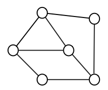		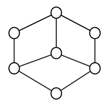		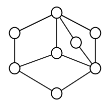		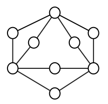	
	#Eq.	#Var.	#Eq.	#Var.	#Eq.	#Var.	#Eq.	#Var.	#Eq.	#Var.	#Eq.	#Var.
Our formulation	64	36	64	40	112	63	112	67	192	100	208	109
Trager et al. [37]	128	120	144	141	224	198	240	219	352	286	384	312

Table 2: The number of equations and variables are reported on some minimal examples for our formulation (see Eq. (13), (17)) and the one proposed in [37] (see Eq. (7), (8)). Recall that the latter (described in Eq. (2) and (5)) is given as a theoretical result in [37] due to its computational complexity, without giving rise to an effective algorithm. Our formulation is more practical as it involves *fewer unknowns*.

Proof. The thesis derives from the following change of variables for each edge in the line graph:

$$\mathbf{u}_{\tau\upsilon} = \mathbf{v}_{\tau\upsilon}/a_{\tau\upsilon} \tag{16}$$

which is well defined since $a_{\tau\upsilon} \neq 0$. □

Remark 6. Thanks to Prop. 5, we have four unknowns for each edge $(\tau, \upsilon) \in \bar{\mathcal{E}}$ in the line graph, representing a vector $\mathbf{u}_{\tau\upsilon} \in \mathbb{R}^4$, plus one unknown scale for each cycle. Thus, the total number of unknowns becomes

$$4\bar{m} + 1(\bar{m} - \bar{n} + 1) = 5\bar{m} - m + 1 \tag{17}$$

which is lower than the formulation related to Thm. 1 where the number of unknowns is $5\bar{m}$ (see Eq. (12)). The number of equations remains unchanged and it is given by Eq. (13). Table 2 reports a comparison between our simplified formulation and the one in Eq. (5) for some examples.

Corollary 1. *Let $\mathcal{G}$ be a graph and let $\mathbf{c}_1, \dots, \mathbf{c}_n \in \mathbb{R}^4$ be n generic camera centres. Let $\{\mathcal{C}_1, \mathcal{C}_2, \dots, \mathcal{C}_f\}$ be a cycle consistency basis for the line graph $\mathcal{L}(\mathcal{G})$. Let us collect in a unique system the equations of the form* (14) *for all the cycles in the basis. $\mathcal{G}$ is solvable if and only if such a system admits* exactly one *solution.*

Proof. In one direction. If $\mathcal{G}$ is solvable then $\mathbf{u}_{\tau\upsilon} = \mathbf{0}$ (thanks to Prop. 5), hence Eq. (14) gives $b_k = 1$ for each cycle $\mathcal{C}_k$ in the basis, i.e., there is *exactly* one solution. In the opposite direction. It is easy to see that if we set all the scales $b_k = 1$ and all the vectors $\mathbf{u}_{\tau\upsilon} = \mathbf{0}$, then we *always* get a solution to Eq. (14). If we assume that there is a unique solution, then it must be equal to $b_k = 1$ and $\mathbf{u}_{\tau\upsilon} = \mathbf{0}$, i.e., the graph is solvable thanks to Prop. 5. If the graph is non-solvable, there will be also other solutions. □

Remark 7. Corollary 1 means that the formulation given in Eq. (14) permits to fix all ambiguities, so that the solution is exactly one (for a solvable graph). It also implies that one does not need to explicitly compute the solution(s) in practice, but it is enough to recover the *number* of solutions. Note that the formulation in Eq. (11), instead, is subject to scale ambiguity, for it involves an unknown scale $a_{\tau\upsilon}$ for each edge in $\mathcal{L}(\mathcal{G})$: when considering a single cycle, for instance, the product of such scales is fixed but all of them are free. Concerning the global projective ambiguity (which is inherent to the problem), observe that a global change in the coordinate system affects the matrices G_τ only, but it does not affect the product $G_\tau G_\upsilon^{-1} = Z_{\tau\upsilon}$. Therefore, projective ambiguity is not present in the formulations given in Eq. (11) and (14) (that do not involve the matrices G_τ).

4. Proposed Algorithm

Our algorithm (summarized in Alg. 1) is a direct consequence of the theoretical results from Sec. 3; in particular, we follow the simplified formulation derived in Sec. 3.3, which is based on Eq. (14). Some steps require additional explanations, which are given in the following remarks.

Algorithm 1 Viewing Graph Solvability

```
Input: undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
Output: solvable or not solvable
    1. randomly sample the camera centres
    2. compute the line graph $\mathcal{L}(\mathcal{G})$
    3. compute a cycle consistency basis for $\mathcal{L}(\mathcal{G})$
    4. set up equations of the form (14) and (20)
    5. compute the number $s$ of real solutions
    6. if $s = 1$ then solvable; else not solvable
```

Remark 8. Concerning Step 3, we focus on a particular type of cycle consistency basis [12], namely, we consider a *fundamental cycle basis*, due to its simplicity. In fact, this basis can be constructed starting from a spanning tree, which can be found in linear time by either depth-first search or breadth-first search. Let $\mathcal{T}$ be a spanning tree of $\mathcal{L}(\mathcal{G}) = (\bar{\mathcal{V}}, \bar{\mathcal{E}})$, then adding any edge from $\bar{\mathcal{E}} \backslash \mathcal{T}$ to $\mathcal{T}$ generates a cycle; the set of such cycles constitutes the *fundamental cycle basis* [17].

Remark 9. As for Step 4, recall that our unknowns comprise one scale $b_k \in \mathbb{R}$ for each cycle and one vector $\mathbf{u}_{\tau\upsilon} \in \mathbb{R}^4$ for each edge in the line graph. Such variables must satisfy

Nodes	3		4		5		6		7		8		9	
Graphs	1		1		1		4		3		36		27	
Method	Alg. 1	[37]	Alg. 1	[37]	Alg. 1	[37]	Alg. 1	[37]	Alg. 1	[37]	Alg. 1	[37]	Alg. 1	[37]
Solvable	1	1	1	1	1	1	4	4	3	3	36	31	17	5
Not solvable	0	0	0	0	0	0	0	0	0	0	0	0	10	0
Unknown	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	5	**0**	22

Table 3: Solvability of minimal viewing graphs. Candidates are connected graphs which are finite solvable and satisfy necessary conditions. Some cases were left undecided by [37], while our approach provides a complete characterization of all the minimal graphs up to 9 nodes.

the following constraints:

$$b_k \neq 0 \tag{18}$$

$$W_{\tau\upsilon} = I_4 + \mathbf{c}_i \mathbf{u}_{\tau\upsilon}^\top \in GL(4, \mathbb{R}). \tag{19}$$

Instead of explicitly enforcing them, we add the following equation

$$z_{\tau\upsilon} \det(I_4 + \mathbf{c}_i \mathbf{u}_{\tau\upsilon}^\top) + 1 = 0 \tag{20}$$

for each edge in the line graph, where $z_{\tau\upsilon} \in \mathbb{R}$ is an auxiliary variable. Clearly, if $\det(I_4 + \mathbf{c}_i \mathbf{u}_{\tau\upsilon}^\top) = 0$ then the above equation can not be satisfied over real numbers. In other words, this additional equation has the effect of automatically discarding non-invertible matrices. Observe also that if all matrices $W_{\tau\upsilon}$ are invertible, then the product of a subset of them is also invertible. In other terms, the left term in Eq. (14) is invertible for each cycle and hence $b_k \neq 0$. Thus equation (20) implies both (19) and (18).

Remark 10. Step 5 is based on computational algebraic geometry. In particular, we employ *Gröbner basis* computation [8], that is one of the main practical tools for solving systems of polynomial equations with coefficients in a field. A Gröbner basis can be viewed as a nonlinear generalization of the Gaussian elimination for linear systems [18].

Remark 11. Although our problem is stated over $\mathbb{R}$, for the sake of efficiency [2] we perform computations over $\mathbb{Z}_p$ (i.e., the integers modulo a large prime number p), as customary in applied algebraic geometry. This yields the same number c of solutions as in $\mathbb{C}$ [31], which is greater or equal to the sought number s of solutions in $\mathbb{R}$. Recall that $s \geqslant 1$, since there always exists at least one trivial real solution (given by $\mathbf{u}_{\tau\upsilon} = 0$, $b_k = 1$ and $z_{\tau\upsilon} = -1$). Several cases are given: i) if $c = \infty$ then $s = \infty$ [37]; ii) if $c = 1$ then $s = 1$; iii) if $c > 1$ then $s \geqslant 1$. Note that if c is even then $s \geqslant 2$ since the solutions must come in conjugated pairs.

5. Experiments

In this section, we show that our method can be profitably used to check the viewing graph solvability on several examples. See also the supplementary material. Our algorithm is implemented in Macaulay2 [11] and the code is publicly available[4].

[4]https://github.com/federica-arrigoni/solvability

We follow the protocol used in [37] where graphs with minimal number of edges (i.e., $m = \lceil (11n-15)/7 \rceil$) are analyzed. As already pointed out, there exist cases with eight and nine nodes that are left undecided in [37] (see Tab. 2 in [37]), as they satisfy the necessary but not sufficient conditions[5]. Our approach, instead, is an effective test for solvability, being based on a characterization of the problem (i.e., a condition, that is *both* necessary and sufficient); as such, it is able to classify all those undecided cases, as summarized in Tab. 3. In particular, the five cases with eight nodes (shown in Fig. 1) were found to be all *solvable*.

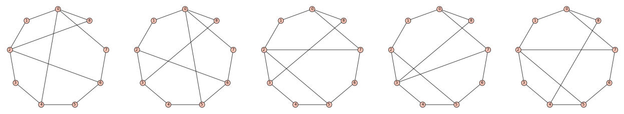
Figure 4: Some solvable minimal viewing graphs with 9 nodes.

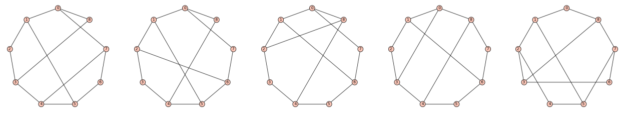
Figure 5: Some unsolvable minimal viewing graphs with 9 nodes.

As for the minimal graphs with nine nodes, there are 22 undecided graphs in [37], which, in particular, are finite solvable (i.e., they identify a finite number of camera configurations). Finite solvability is a necessary condition for solvability, but it was unknown whether it is also sufficient, as all non-solvable graphs – studied so far – define an infinite number of solutions. Our algorithm is able to prove that a subset of those undecided cases are solvable graphs (see Fig. 4 for some examples). Surprisingly, there exist also some non-solvable graphs among those candidates (see Fig. 5 for some examples), where our algorithm finds *two* real solutions. Thus, *it is possible for a graph to be finite solvable without being solvable* (i.e., to have a finite number of real solutions strictly greater than one). This answers an open research question pointed out by Trager et al. [37].

Viewing graphs with more than nine nodes are not studied in [37]. Our approach, instead, is able to handle minimal graphs with up to 90 nodes. For instance, we can prove that the graphs reported in Fig. 6 are solvable.

[5]Actually, Trager et. al [37] *manually* worked out that one of those graphs is solvable.

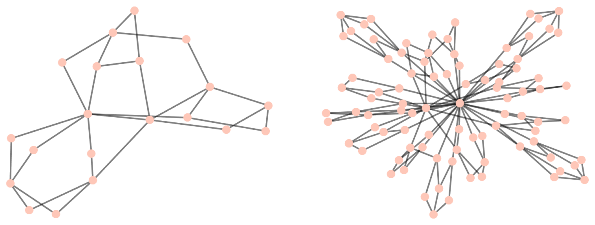

Figure 6: Examples of solvable minimal viewing graphs with 20 nodes (left) and 90 nodes (right).

Larger/denser graphs would require too much computational effort to be characterized with the computer used in our experiments (2020 MacBook Pro with 1.4 GHz processor, 8 GB RAM). Nevertheless, we can use our approach as a probe to study their local structure. Table 4 reports the execution times of Alg. 1 on some minimal graphs with increasing number of nodes. The computational complexity is dominated by Gröbner basis computation, whose worst-case complexity is doubly exponential in the number of variables [8]. In our experiment, we randomly sample small sub-graphs of large viewing graphs coming from real data sets. More precisely, we proceed as follows: i) we select at random one node of the graph; ii) we identify the first neighborhood of the sampled node, (if the first neighbors are not enough, we also consider the neighbors of neighbors and so on); iii) we randomly select 8 nodes within the neighborhood. This, in addition to the original node, yields a nine-node subgraph. Following this procedure, we sample 200 subgraphs from each real graph, without replacement. The results, reported in Tab. 5, tell us that most local subgraphs are solvable. This gives an indication about which sub-graphs could be used in practice as a starting point for an incremental pipeline for image-based 3D reconstruction.

Nodes	10	20	30	40	50	60	70	80	90
Time	1.6 s	9 s	93 s	3 min	15 min	35 min	1 h	≈ 2 h	> 4 h

Table 4: Execution times of Alg. 1 on some minimal graphs.

6. Conclusions and future work

We investigated the solvability of viewing graphs, i.e. whether they uniquely determine projective cameras, and made several important advances in the theory and practical use of viewing graphs. Building upon [37], we proposed a new characterization that involves fewer unknowns by exploiting cycle consistency. The resulting algorithm is an effective test (necessary and sufficient conditions) for solvability, thanks to which we classified all the cases left undecided by [37], and proved that *finite solvability does not imply solvability*, thereby answering an open research question. Moreover, we were able to process minimal graphs with up to 90 vertices, which sets the state-of-the-art in the

	Solvable			Unsolvable		
Data set	by suff.	by Alg. 1	Tot.	by nec.	by Alg. 1	Tot.
Alcatraz Courtyard	200	0	200	0	0	0
Buddah Tooth	178	20	198	2	0	2
Pumpkin	169	22	191	8	1	9
Skansen Kronan	179	8	187	13	0	13
Tsar Nikolai I	196	0	196	4	0	4
Alamo	136	16	152	48	0	48
Ellis Island	136	30	166	34	0	34
Gendarmenmarkt	128	11	139	61	0	61
Madrid Metropolis	88	28	116	84	0	84
Montreal Notre Dame	140	12	152	48	0	48
Notre Dame	165	18	183	17	0	17
NYC Library	110	19	129	71	0	71
Piazza del Popolo	105	22	127	73	0	73
Piccadilly	109	23	132	68	0	68
Roman Forum	114	28	142	58	0	58
Tower of London	123	18	141	59	0	59
Trafalgar	86	16	102	98	0	98
Union Square	74	19	93	107	0	107
Vienna Cathedral	122	8	130	70	0	70
Yorkminster	116	14	130	70	0	70
Cornell Arts Quad	76	23	99	101	0	101

Table 5: Characterization of sub-graphs with nine nodes sampled from some real viewing graphs [23, 6, 39]. Solvable by sufficiency means that the graph satisfies a sufficient condition, namely being chordal [36]. Unsolvable by necessity means that the graph fails to satisfy some necessary conditions [37]. All the other cases are resolved by our approach (Alg. 1).

uncalibrated case. Although this is still far from the level of maturity of the calibrated case, a careful analysis of small graphs is important as they are the building blocks of larger graphs. The maximum size we can manage is a matter of designing clever solvers and exploiting computational power: we are working on pushing this limit forward. For example, we plan to investigate numerical algebraic geometry (e.g., [9]), which gives good grounds for expecting to make the computation tractable for large-scale scenarios.

In this paper, we considered the concept of solvability given in Def. 2, which is based solely on the topology of the viewing graph. Using additional information (e.g., points) would give rise to a different solvability notion [21], that would be interesting to explore in the future. Drawing the connection to the calibrated case (parallel rigidity [3]) would also be an interesting topic of prospective research.

Besides being of theoretical interest, the solvability problem has a practical impact, for reconstruction methods such as [30, 16] will benefit from knowing in advance whether the graph at hand is solvable or not: if the problem is ill-posed, then any method will fail to return a useful solution. In this case, finding a maximal subgraph that is solvable would be of great interest.

Acknowledgements. This work was supported by the European Regional Development Fund under the project IMPACT (No. CZ.02.1.01/0.0/0.0/15_003/0000468) and by the EU H2020 Programme under the project SPRING (No. 871245).

References

[1] H. S. Alismail, B. Browning, and M. B. Dias. Evaluating pose estimation methods for stereo visual odometry on robots. In *the 11th International Conference on Intelligent Autonomous Systems (IAS-11)*, January 2011. 1

[2] E. A. Arnold. Modular algorithms for computing Gröbner bases. *Journal of Symbolic Computation*, 35(4):403 – 419, 2003. 7

[3] F. Arrigoni and A. Fusiello. Bearing-based network localizability: A unifying view. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(9):2049 – 2069, 2019. 1, 2, 8

[4] F. Arrigoni and A. Fusiello. Synchronization problems in computer vision with closed-form solutions. *International Journal of Computer Vision*, 128:26–52, 2020. 4, 5

[5] F. Bernard, J. Thunberg, P. Gemmar, F. Hertel, A. Husch, and J. Goncalves. A solution for multi-alignment by transformation synchronisation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015. 4

[6] D. Crandall, A. Owens, N. Snavely, and D. P. Huttenlocher. Discrete-continuous optimization for large-scale structure from motion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3001–3008, 2011. 8

[7] D. Cvetkovic, P. Rowlinson, and S. Simic. *Spectral Generalizations of Line Graphs: On Graphs with Least Eigenvalue -2*. London Mathematical Society Lecture Note Series. Cambridge University Press, 2004. 4

[8] T. W. Dubé. The structure of polynomial ideals and Gröbner bases. *SIAM Journal on Computing*, 19(4):750 – 773, 1990. 7, 8

[9] T. Duff, C. Hill, A. Jensen, K. Lee, A. Leykin, and J. Sommars. Solving polynomial systems via homotopy continuation and monodromy. *IMA Journal of Numerical Analysis*, 39(3):1421–1446, 2018. 8

[10] V. M. Govindu. Lie-algebraic averaging for globally consistent motion estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 684–691, 2004. 4

[11] D. R. Grayson and M. E. Stillman. Macaulay2, a software system for research in algebraic geometry. Available at https://math.uiuc.edu/Macaulay2/. 7

[12] L. J. Guibas, Q. Huang, and Z. Liang. A condition number for joint optimization of cycle-consistent networks. In *Neural Information Processing Systems*, volume 32, pages 1007–1017, 2019. 5, 6

[13] S. Guillemot. FPT algorithms for path-traversal and cycle-traversal problems. *Discrete Optimization*, 8(1):61 – 71, 2011. 5

[14] F. Harary. *Graph Theory*. Addison-Wesley, 1972. 3

[15] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition, 2004. 2

[16] Y. Kasten, A. Geifman, M. Galun, and R. Basri. GPSfM: Global projective SFM using algebraic constraints on multi-view fundamental matrices. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3259–3267, 2019. 1, 2, 8

[17] T. Kavitha, C. Liebchen, K. Mehlhorn, D. Michail, R. Rizzi, T. Ueckerdt, and K. Zweig. Cycle bases in graphs: Characterization, algorithms, complexity, and applications. *Computer Scienze Review*, 3(4):199–243, 2009. 5, 6

[18] D. Lazard. Gröbner bases, Gaussian elimination and resolution of systems of algebraic equations. In J. A. van Hulzen, editor, *Computer Algebra*, pages 146–156, Berlin, Heidelberg, 1983. Springer Berlin Heidelberg. 7

[19] N. Levi and M. Werman. The viewing graph. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 518 – 522, 2003. 1, 2, 3

[20] A. Nardi, D. Comanducci, and C. Colombo. Augmented vision: Seeing beyond field of view and occlusions via uncalibrated visual transfer from multiple viewpoints. In *Proceedings of the 2011 Irish Machine Vision and Image Processing Conference*, page 38–44, 2011. 1

[21] B. Nasihatkon, R. Hartley, and J. Trumpf. A generalized projective reconstruction theorem and depth constraints for projective factorization. *International Journal of Computer Vision*, 115:87 – 115, 2015. 8

[22] D. Nistér, O. Naroditsky, and J. Bergen. Visual odometry. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 652–659, 2004. 1

[23] C. Olsson and O. Enqvist. Stable structure from motion for unordered image collections. In *Proceedings of the 17th Scandinavian conference on Image analysis (SCIA'11)*, pages 524–535. Springer-Verlag, 2011. 8

[24] O. Ozyesil and A. Singer. Robust camera location estimation by convex programming. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2674 – 2683, 2015. 1

[25] O. Ozyesil, V. Voroninski, R. Basri, and A. Singer. A survey of structure from motion. *Acta Numerica*, 26:305 – 364, 2017. 2

[26] I. Rocco, M. Cimpoi, R. Arandjelović, A. Torii, T. Pajdla, and J. Sivic. Neighbourhood consensus networks. In *Neural Information Processing Systems (NeurIPS)*, 2018. 1

[27] A. Rudi, M. Pizzoli, and F. Pirri. Linear solvability in the viewing graph. In *Proceedings of the Asian Conference on Computer Vision*, pages 369–381, 2011. 1, 2, 3

[28] T. Sattler, B. Leibe, and L. Kobbelt. Efficient & effective prioritized matching for large-scale image-based localization. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(9):1744–1756, 2017. 1

[29] J. L. Schönberger and J.-M. Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1

[30] S. Sengupta, T. Amir, M. Galun, T. Goldstein, D. W. Jacobs, A. Singer, and R. Basri. A new rank constraint on multi-view fundamental matrices, and its application to camera location recovery. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2413–2421, 2017. 1, 2, 8

[31] I. Shafarevich and K. Hirsch. *Basic algebraic geometry*, volume 1. Springer, 1994. 7

[32] A. Singer. Angular synchronization by eigenvectors and semidefinite programming. *Applied and Computational Harmonic Analysis*, 30(1):20 – 36, 2011. 4

[33] N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: exploring photo collections in 3D. In *ACM SIGGRAPH*, 2006. 1

[34] N. Snavely, S. M. Seitz, and R. Szeliski. Modeling the world from internet photo collections. *International Journal of Computer Vision*, 80(2):189–210, 2008. 1

[35] H. Taira, M. Okutomi, T. Sattler, M. Cimpoi, M. Pollefeys, J. Sivic, T. Pajdla, and A. Torii. InLoc: Indoor visual localization with dense matching and view synthesis. In *CVPR*, 2018. 1

[36] M. Trager, M. Hebert, and J. Ponce. The joint image handbook. In *Proceedings of the International Conference on Computer Vision*, pages 909–917, 2015. 1, 3, 8

[37] M. Trager, B. Osserman, and J. Ponce. On the solvability of viewing graphs. In *Proceedings of the European Conference on Computer Vision*, pages 335–350, 2018. 1, 2, 3, 4, 6, 7, 8

[38] R. Tron, L. Carlone, F. Dellaert, and K. Daniilidis. Rigid components identification and rigidity enforcement in bearing-only localization using the graph cycle basis. In *IEEE American Control Conference*, 2015. 1

[39] K. Wilson and N. Snavely. Robust global translations with 1DSfM. In *Proceedings of the European Conference on Computer Vision*, pages 61–75, 2014. 8