

Revisiting Viewing Graph Solvability: an Effective Approach Based on Cycle Consistency

Federica Arrigoni, Andrea Fusiello, Romeo Rizzi, Elisa Ricci and Tomas Pajdla

Abstract—In the structure from motion, the viewing graph is a graph where the vertices correspond to cameras (or images) and the edges represent the fundamental matrices. We provide a new formulation and an algorithm for determining whether a viewing graph is solvable, i.e., uniquely determines a set of projective cameras. The known theoretical conditions either do not fully characterize the solvability of all viewing graphs, or are extremely difficult to compute because they involve solving a system of polynomial equations with a large number of unknowns. The main result of this paper is a method to reduce the number of unknowns by exploiting cycle consistency. We advance the understanding of solvability by (i) finishing the classification of all minimal graphs up to 9 nodes, (ii) extending the practical verification of solvability to minimal graphs with up to 90 nodes, (iii) finally answering an open research question by showing that finite solvability is not equivalent to solvability, and (iv) formally drawing the connection with the calibrated case (i.e., parallel rigidity). Finally, we present an experiment on real data that shows that unsolvable graphs may appear in practice.

Index Terms—Solvability, viewing graph, structure from motion, fundamental matrix, uncalibrated camera.

1 INTRODUCTION

STRUCTURE-FROM-MOTION (SfM) is a fundamental task in computer vision, due to its relevance in many important problems such as 3D reconstruction from image sets [1], [2], [3], image matching [4] and visual odometry [5], [6], [7], [8]. As a consequence, SfM algorithms are key enabling technologies in a wide range of application domains, such as autonomous driving, robotics and Augmented/Virtual Reality.

The fundamental problem in SfM, i.e., to determine which image sets can be reconstructed, can be addressed by analyzing the viewing graph [9]. The *viewing graph* of a set of images, also referred to as the *epipolar graph*, is a graph where vertices are images (or cameras) and edges correspond to fundamental matrices. With this formalism, an edge is present if and only if the fundamental matrix between the two images associated to the connecting vertices exists (i.e., if enough corresponding points between the two images can be found). As a consequence, in many real world applications, the viewing graph is not complete, as different images may depict different parts of a scene.

While several research efforts on SfM have focused on devising more accurate and efficient algorithms for 3D scene reconstruction and visual localization [10], much less attention have been devoted to investigate theoretical aspects. In this context, 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 words, an unsolvable viewing graph is graph for which *multiple* transformations exist that can be applied to the cameras without affecting the fundamental matrices. In the literature, the problem of viewing graph solvability has been previously studied by Levi and Werman [9]. In [9] a definition of solvability is provided which states 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.

Studying graph solvability is of utmost importance for 3D reconstruction. Indeed, viewing graphs are used in several projective SfM methods [11], [12], which operate by recovering the projective cameras starting from multiple fundamental matrices. Assessing the solvability of a viewing graph is a necessary, preliminary step for reconstruction. In fact, no existing SfM method operating on fundamental matrices will be able to provide a reasonable solution if the reconstruction problem is unsolvable.

Previous studies [13], [14] demonstrated that, in the calibrated case, i.e. when essential matrices are considered rather than the fundamental ones, the solvable graphs are exactly those that are *parallel rigid*. The topic of parallel rigidity, also referred as *bearing rigidity*, has been extensively studied in the literature [15]. However, in this paper, we focus on the more challenging situation of uncalibrated cameras, which has received little attention so far. For the sake of completeness, in Table 1 we provide an illustration of the relevant literature on graph solvability and reconstruction, considering both the calibrated and the uncalibrated case.

1.1 Related Work

Due to its relevance in several applications, Structure-from-Motion has been deeply studied in computer vision. However, over the years, most research efforts on SfM have focused on devising algorithmic solutions for accurate and efficient 3D scene reconstruction and visual localization. On the contrary, the underlying theoretical problems have been much less studied.

- F. Arrigoni is with the Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB), Politecnico di Milano, Italy.
E-mail: federica.arrigoni@polimi.it
- A. Fusiello is with the Polytechnic Department of Engineering and Architecture (DPIA), University of Udine, Italy.
E-mail: andrea.fusiello@uniud.it
- R. Rizzi is with the Department of Computer Science, University of Verona, Italy. E-mail: romeo.rizzi@univr.it
- E. Ricci is with the Department of Information Engineering and Computer Science (DISI), University of Trento, Italy and with the Deep Visual Learning Group, Fondazione Bruno Kessler (FBK), Italy.
E-mail: e.ricci@unitn.it
- T. Pajdla is with the Czech Institute of Informatics, Robotics and Cybernetics (CIIRC), Czech Technical University in Prague, Czech Republic.
E-mail: pajdla@cvut.cz

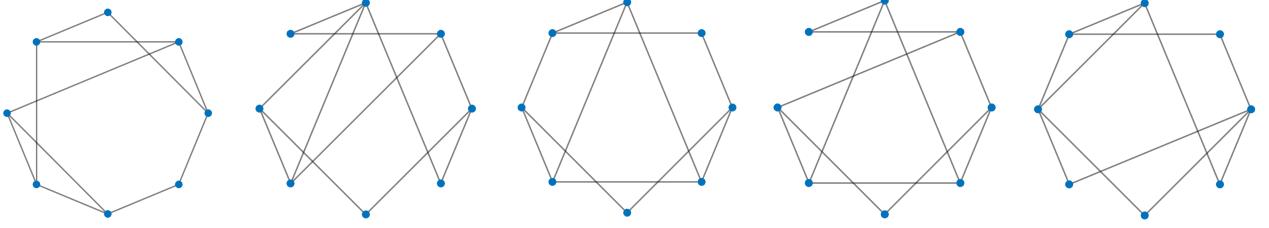


Fig. 1: Viewing graphs with eight vertices that were left undecided in [16] and that we determined to be solvable.

Focusing on graph solvability, Levi and Werman [9] were the first to characterize the solvability of graphs with at most six vertices and further provided some insights about how to analyze larger graphs. Nardi *et al.* [17] described a constructive method to complete such view graphs. Rudi *et al.* [18] analyzed the conditions under which a linear method can solve a viewing graph. This analysis was later extended in [19], where it was shown that some solvable graphs can be constructed starting from a triangle and adding vertices of degree two one at a time. More recently, Trager *et al.* [16] provided some necessary conditions and a new sufficient condition for graph solvability. Their results permit to completely characterize graphs with at most seven vertices. However, with the results of [19], graphs with eight or nine vertices such as those in Figure 1 cannot be classified into solvable or non solvable. Additionally, larger graphs were not studied. Trager *et al.* [16] also showed that any viewing graph can be classified via a system of polynomial equations. However, while theoretically important, this study does not provide an effective test for solvability. Indeed, solving a system of polynomial equations is computationally expensive due to nonlinearity and to a large number of unknowns.

Another open issue in this research area is related to the concept of *finite 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 a nutshell, solvability implies finite solvability. Trager *et al.* [16] did not consider the reverse implication, namely: *Does finite solvability imply solvability?* Our work advances the state of the art in SfM by addressing those research questions about viewing graph solvability that were left open in [16].

	calibrated	uncalibrated
solvability	[15]	[9], [16], [18]+ours
reconstruction	[10]	[11], [12]

TABLE 1: Problems taxonomy. We address the uncalibrated case. Surveys [10], [15] address the calibrated situation.

1.2 Contribution

We derive a new formulation of viewing graph solvability that is much simpler than [16] thanks to a substantial reduction in the number of unknowns involved. Our formulation is based on the *cycle consistency* property, namely the fact that the composition of (invertible) transformations along any cycle in the graph should be the identity.

This leads to a new algorithm, based on computational algebraic geometry, which implements a characterization of solvability. Previous techniques could only verify certain 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 [16].

In 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.

Furthermore, thanks to our algorithm, it is possible to exhibit concrete examples of graphs that are finite solvable but not solvable, thus answering the research question opened by [16].

Finally, we derive a new necessary condition for solvability, namely parallel rigidity. This results in a practical tool to detect non-solvable graphs, as parallel rigidity can be reduced to check the rank of a linear system [20]. Since essential matrices subsume fundamental ones, it seems reasonable that a graph that admits a unique solution when the edges are associated with fundamental matrices also does when the edges are associated with essential ones. Hence this result is largely expected, but it has never been proved before. 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;
- prove that solvability implies parallel rigidity.

The paper is organized as follows. We define the viewing graph solvability in Section 2, derive a new necessary condition in Section 3, present our main theoretical results in Section 4, and describe our algorithm in Section 5. Section 6 presents some experiments on synthetic and real data. Further examples are analyzed in Section 7. This paper is an extended version of [21].

2 BACKGROUND

Let us consider n projection matrices of uncalibrated cameras P_1, \dots, P_n , which are matrices in $\mathbb{R}^{3 \times 4}$ of rank 3. The center of a camera has homogeneous coordinates in vector $\mathbf{c}_i \in \mathbb{R}^4$, a non-null element of the kernel of P_i . 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 [22]. Conversely, the fundamental matrix of edge (i, j) uniquely determines the cameras of vertices i and j , up to a projective transformation [22].

In the following, we shall use uppercase letters to denote matrices, lowercase bold letters to denote vectors and lowercase letters to denote scalars¹. Projective quantities are represented as

1. Observe that this notation is different from the one used in [16].

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 ([16]). *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 [9] 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.

Different notions of solvability have been set forth in the literature. According to [9] a graph is solvable if and only if the given fundamental matrices uniquely determine the remaining ones that complete the graph. This definition is specific, but the results are given for generic configurations. Rudi et al. [18] define the notion of “solution set”, that is the set of camera matrices (modulo projective transformations and scalar multiplication) which yield the given fundamental matrices. Also this definition refers to a specific set of fundamental matrices. Clearly, if a graph is solvable with a specific set of fundamental matrices, it is also solvable in the sense of Definition 2, but the reverse is not true in general. Lemma 2 of [16] clarifies the link with the definition by [9].

Remark 1. Hereafter we will consider Definition 2 as our definition of solvability, as done in [16]. Observe that such a concept is based on the topology of the graph only. In fact, fundamental matrices are ignored altogether: generic cameras are assigned to the vertices in order to check solvability for a particular example, so one can *ideally* attach to the edges of the graph a set of noiseless fundamental matrices which can be derived straightforwardly from the cameras. Noisy fundamental matrices come into play when addressing the reconstruction, which takes place *after* checking solvability (see Table 1).

Solvability has only recently been characterised by [16] (see Section 4.1): previously only necessary or sufficient conditions were known. For example, any chordal (or triangulated) graph is solvable [19]. Other sufficient conditions are proposed in [16], [18], [19], where the idea is to check if the input graph can be transformed into a solvable one via suitable operations. Necessary conditions [9], [16] have a practical use, for they allow to quickly prune the candidates. For instance, a solvable graph:

- has at least $(11n - 15)/7$ edges [16];
- is biconnected [16], i.e., the removal of any single node leaves the graph connected;
- has the property that all the vertices have degree at least two and there are not two adjacent vertices with degree two (if $n > 3$) [9].

In addition, a solvable graph must satisfy the following condition, which will be exploited in Section 3.

Proposition 2 ([16]). *Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a solvable graph with $n = |\mathcal{V}|$ and $m = |\mathcal{E}|$. If $\mathcal{G}_1, \dots, \mathcal{G}_k$ are sub-graphs of \mathcal{G} with $\mathcal{G}_i = (\mathcal{E}_i, \mathcal{V}_i)$ and \mathcal{E}_i are pairwise disjoint, then it holds:*

$$\sum_{i=1}^k (11|\mathcal{V}_i| - 7|\mathcal{E}_i| - 15) \geq 11n - 7m - 15. \quad (1)$$

3 CONNECTION TO PARALLEL RIGIDITY

In this section we formally draw the connection to the calibrated case (i.e., parallel rigidity [13]). In particular, we prove that if a viewing graph is solvable, then it is also parallel rigid² in 3D space. Recall that solvability is equivalent to the well-posedness of an uncalibrated 3D reconstruction problem, represented as a graph with fundamental matrices on the edges. Parallel rigidity in 3D space, instead, is tantamount to solvability when cameras are calibrated and edges are associated with essential matrices (see Table 1). Hence, this result is largely expected. Nevertheless this is the first proof, to the best of our knowledge.

Among the available characterizations of parallel rigidity (see [15] for a comprehensive list), the following one best fits our goal:

Proposition 3 ([23]). *A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with $n = |\mathcal{V}|$ is parallel rigid in 3D space if and only if, for any collection $\mathcal{G}_1, \dots, \mathcal{G}_k$ of sub-graphs of \mathcal{G} with $\mathcal{G}_i = (\mathcal{E}_i, \mathcal{V}_i)$ such that $\mathcal{E}_1, \dots, \mathcal{E}_k$ form a partition of \mathcal{E} , it holds:*

$$\sum_{i=1}^k (3|\mathcal{V}_i| - 4) \geq 3n - 4. \quad (2)$$

Theorem 1. *If a graph \mathcal{G} is solvable then it is parallel rigid.*

Proof. Our proof builds on the characterization of parallel rigid graphs given in Proposition 3. As such, we are only due to show that the inequality

$$\sum_{i=1}^k (3|\mathcal{V}_i| - 4) \geq 3n - 4 \quad (3)$$

holds true for any partition $\{\mathcal{E}_1, \dots, \mathcal{E}_k\}$ of \mathcal{E} . In order to conveniently plug in the similar inequality which is known to hold for solvable graphs (i.e., Proposition 2), it is only convenient to verify the above inequality as multiplied by 11/3. Before we start, let us note that if the \mathcal{E}_i form a partition of \mathcal{E} then Proposition 2 writes:

$$\sum_{i=1}^k (11|\mathcal{V}_i| - 15) \geq 11n - 15 \quad (4)$$

because $\sum_{i=1}^k |\mathcal{E}_i| = m$. By computation we get:

$$\begin{aligned} \frac{11}{3} \sum_{i=1}^k (3|\mathcal{V}_i| - 4) &= \sum_{i=1}^k \left(11|\mathcal{V}_i| - \frac{44}{3} \right) = \\ &= \frac{k}{3} + \sum_{i=1}^k (11|\mathcal{V}_i| - 15) \geq \frac{1}{3} + (11n - 15) = 11n - \frac{44}{3} \end{aligned} \quad (5)$$

² Parallel rigidity can be defined in any dimension. We are interested here in the 3D case.

where the above inequality comes from (4) together with the assumption that the graph is solvable, and because $k \geq 1$. \square

Observe that the converse of the above result is not true. For example, a cycle of length four (namely a “square”) is rigid (see [14], [15]) but not solvable (see [9], [16]). In other terms, the set of solvable graphs is *strictly* contained in the set of parallel rigid graphs (see Figure 2). Note also that the property of being biconnected is a necessary condition for both (it was proved in [24] for rigid graphs and in [16] for solvable graphs). In this context, Theorem 1 has practical relevance as it provides a new necessary condition for solvability, namely testing parallel rigidity. Such a test is simple as it can be reduced to checking the rank of a properly constructed linear system (see [20], [25]).

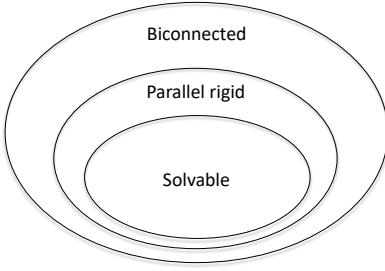


Fig. 2: Connection between solvability, parallel rigidity and biconnectivity. All the inclusions are strict.

4 CHARACTERIZATION OF SOLVABILITY

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 Definition 2. Throughout our discussion we assume that \mathcal{G} is connected and $n \geq 3$. The starting point of our derivations is the algebraic formulation by Trager *et al.* [16], which is the only available characterization of solvability and will be reviewed next.

4.1 Algebraic Formulation

Trager *et al.* [16] 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 4 ([16]). *Let $P \in \mathbb{R}^{3 \times 4}$ 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*

$$G = aI_4 + \mathbf{c}\mathbf{v}^\top \quad \forall a \in \mathbb{R}_{\neq 0}, \mathbf{v} \in \mathbb{R}^4 \quad (6)$$

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

Proof. Observe that $PG = aP \iff P(G - aI_4) = 0 \iff PB = 0$ with $B = (G - aI_4) \iff \text{im}(B)$ is a subspace of $\ker(P)$. Since $\dim(\ker(P)) = 1$ also $\dim(\text{im}(B)) = 1$, hence $\text{rank}(B) = 1$. So B must be the outer product of two vectors, and the left one must belong to $\ker(P)$, therefore $B = \mathbf{c}\mathbf{v}^\top$. \square

Please note that the condition $a + \mathbf{c}^\top \mathbf{v} \neq 0$ is necessary and sufficient for G to be invertible. This derives from the fact that 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 $\mathbf{w} = -\frac{1}{1 + \mathbf{c}^\top \mathbf{v}}\mathbf{v}$.

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.

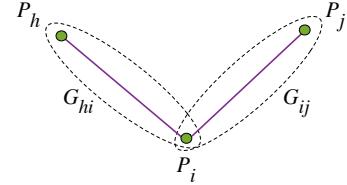


Fig. 3: Two adjacent edges in the viewing graph.

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 Figure 3). Such transformations must leave the camera at the common vertex fixed, resulting in the following *compatibility* constraint:

$$G_{hi}G_{ij}^{-1} = a_{hij}I_4 + \mathbf{c}_i\mathbf{v}_{hij}^\top \quad (7)$$

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 Equation (7) holds true for all adjacent edges, is called *compatible*.

Proposition 5 ([16]). *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*

$$G_{ij} = s_{ij}H \quad \forall (i, j) \in \mathcal{E} \quad (8)$$

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

The condition in Proposition 5 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.

Remark 2. Note that the centres can be sampled at random³, in order to satisfy the assumption of generic cameras. This is a standard procedure in solvability theory [16] (which is also used in the calibrated case [15]): it permits to check solvability in a generic sense, namely with cameras in a generic position, hence relying on the graph structure only (see Definition 2). In this respect, camera centres are treated as known variables hereafter.

Finding a compatible assignment of matrices, i.e., solving Equation (7) 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, Proposition 5 is given as a theoretical result in [16], without leading to a practical algorithm

³ Of course, there is a very small chance (with probability 0) to sample a degenerate configuration.

for checking viewing-graph solvability. We will explain how to alleviate this drawback later on.

4.2 Solvability on the Line Graph

Our formulation is inspired by the algebraic characterization detailed in Section 4.1. Specifically, we show how to reduce the number of unknowns in Equation (7), thus providing a more practical way for checking viewing graph solvability. To this end, we exploit the *line graph* associated with \mathcal{G} , which has been used also in [26] to handle multi-view geometry.

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 4 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 [27]:

$$\bar{m} = \frac{1}{2} \sum_{i=1}^n d_i^2 - m \quad (9)$$

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

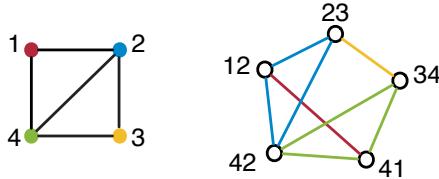


Fig. 4: Viewing graph with 4 vertices (left) and corresponding line graph (right). Please note that a vertex of the original graph (e.g., vertex 2) can appear multiple times as an edge of the line graph, as clarified by colors.

Let us rewrite Equation (7) 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 $v \in \bar{\mathcal{V}}$. Such vertices are connected by an edge $(\tau, v) \in \bar{\mathcal{E}}$ 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 Equation (7) becomes:

$$G_\tau G_v^{-1} = Z_{\tau v} \quad (10)$$

where:

$$Z_{\tau v} = a_{\tau v} I_4 + \mathbf{c}_i \mathbf{v}_{\tau v}^\top \quad (11)$$

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

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 Equation (10) holds for all the edges is called a *consistent labelling*.

Remark 3. A consistent labelling of the line graph corresponds to a compatible assignment on the original graph (see Definition 3). We give this equivalent definition here to outline the link with

the problem of *synchronization* [28], [29], [30] (see [31] 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 Equation (10), the task would be to compute G_τ, G_v, \dots starting from known $Z_{\tau v}$. In our case, however, the variables $Z_{\tau v} \in GL(4, \mathbb{R})$ are *unknown* for all $(\tau, v) \in \bar{\mathcal{E}}$. 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 Theorem 2).

Remark 4. The problem of finding a consistent labelling involves an equation of the form (10) for each edge in the line graph, which in turn spawns 16 equations when considered entry-wise. Thus, using (9), the total number of equations is given by:

$$16\bar{m} = 16 \left(\frac{1}{2} \sum_{i=1}^n d_i^2 - m \right). \quad (12)$$

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 $\mathbf{v}_{\tau v} \in \mathbb{R}^4$ and a scale $a_{\tau v} \in \mathbb{R}_{\neq 0}$. Thus, the total number of unknowns is given by:

$$16\bar{n} + 5\bar{m} = \frac{5}{2} \sum_{i=1}^n d_i^2 + 11m \quad (13)$$

where $\bar{n} = m$ by construction and \bar{m} is given by Equation (9). Recall that the camera centres are considered known as they are sampled at random in practice (see Remark 2).

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

Proposition 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. The graph \mathcal{G} is solvable if and only if any consistent labelling yields:

$$\mathbf{v}_{\tau v} = \mathbf{0} \quad \forall (\tau, v) \in \bar{\mathcal{E}}. \quad (14)$$

Proof. If \mathcal{G} is solvable, then, due to Proposition 5, all the matrices G_τ represent the same projective transformation, or equivalently, they are all multiples of each other. Hence, the product $G_\tau G_v^{-1}$ is a multiple of the identity (which is denoted by $b_{\tau v} I_4$ with $b_{\tau v} \in \mathbb{R}_{\neq 0}$). Hence Equation (11) becomes:

$$b_{\tau v} I_4 = a_{\tau v} I_4 + \mathbf{c}_i \mathbf{v}_{\tau v}^\top \Leftrightarrow (b_{\tau v} - a_{\tau v}) I_4 = \mathbf{c}_i \mathbf{v}_{\tau v}^\top. \quad (15)$$

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 $b_{\tau v} - a_{\tau v} = 0$ and $\mathbf{v}_{\tau v} = \mathbf{0}$, hence we get the result. In the opposite direction, if $\mathbf{v}_{\tau v} = \mathbf{0}$ then Equation (10) rewrites $G_\tau G_v^{-1} = a_{\tau v} I_4$ or, in other terms, $G_\tau = a_{\tau v} G_v$. 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 [32]). This means that all matrices G_τ are multiples of each other, hence the graph is solvable, due to Proposition 5. \square

Remark 5. Observe that one implication of the above proposition could be proved without the line graph, using Equation (7)-(8), whereas the reverse implication exploits the fact that connectivity on the original graph implies connectivity on the line graph. We will see later (see Theorem 2) that the synchronization formalism (which is at the basis of our development) requires an equation of the form (10), making the line graph indeed essential. Observe

also that the second part of the proof of Proposition 6 has actually found a consistent labelling for $\mathcal{L}(\mathcal{G})$, or, in other terms, it has solved synchronization [31]. Solving such a problem is straightforward in the absence of noise – as it is the case here: it is enough to arbitrarily fix one transformation (e.g., to the identity) and compute the others by propagating the relation $G_\tau = Z_{\tau v} G_v$ along a spanning tree, assuming a connected graph. Note that fixing one transformation to the identity corresponds to fixing the global projective ambiguity.

Remark 6. Observe that Proposition 5 gives a formulation of solvability in terms of the matrices G_τ whereas Proposition 6 considers the variables $v_{\tau v}$ 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 $v_{\tau v}$ and $a_{\tau v}$ only). In this respect, a formulation of solvability in terms of $v_{\tau v}$ (as given by Proposition 6) is indeed essential.

4.3 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

$$Z_{\tau_1 \tau_2} Z_{\tau_2 \tau_3} \cdots Z_{\tau_\ell \tau_1} = I_4. \quad (16)$$

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 [33] 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 [34] for details). An example is reported in Figure 5.

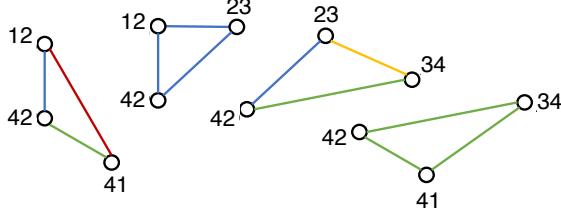


Fig. 5: Example of a cycle consistency basis (comprising four cycles) associated with the line graph in Figure 4. Observe that, given a graph, a cycle consistency basis may not be unique.

Theorem 2. 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 (16) for all the cycles in the basis. \mathcal{G} is solvable if and only if the solution to such system yields $\mathbf{v}_{\tau v} = \mathbf{0}$ for all $(\tau, v) \in \bar{\mathcal{E}}$.

Proof. It is known that there exists a consistent labelling if and only if *all* the cycles are null/consistent (see Lemma 8 in [35] and Corollary 1 in [31]). Clearly, if all cycles are consistent, then – in particular – the cycles in a basis are also consistent. The opposite does not hold in general [34]. 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 Equation (10), 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 (10) for all the edges in $\mathcal{L}(\mathcal{G})$, is equivalent to the system obtained by stacking equations of the form (16) for all the cycles in a cycle consistency basis of $\mathcal{L}(\mathcal{G})$. According to Proposition 6, a graph is solvable if and only if the solution to the former yields $\mathbf{v}_{\tau v} = \mathbf{0}$ for all $(\tau, v) \in \bar{\mathcal{E}}$, hence we prove the thesis. \square

Remark 7. The formulation of Theorem 2 comprises five unknowns for each edge $(\tau, v) \in \bar{\mathcal{E}}$ in the line graph, representing a vector $\mathbf{v}_{\tau v} \in \mathbb{R}^4$ and a scale $a_\tau \in \mathbb{R}_{\neq 0}$. Thus, using Equation (9), the total number of unknowns is given by

$$5\bar{m} = 5\left(\frac{1}{2} \sum_{i=1}^n d_i^2 - m\right). \quad (17)$$

Observe also that each cycle originates an equation of the form (16), 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 [34]), we get the total number of equations as

$$16(\bar{m} - \bar{n} + 1) = 16\left(\frac{1}{2} \sum_{i=1}^n d_i^2 - 2m + 1\right). \quad (18)$$

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 Equation (9). Note that Equation (16) 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 Equation (10), where the amount of unknowns is given in Equation (13). Observe also that Proposition 6 and the cycle space trick jointly contribute to reduce the number of variables: the former states that only the “ \mathbf{v} ” variables matter, whereas the latter provides an equation involving only the “ \mathbf{v} ”.

Remark 8. 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* edges⁴. Indeed, $Z_{\tau v} = G_\tau G_v^{-1} = G_{hi} G_{ij}^{-1}$ and $Z_{v\tau} = G_v G_\tau^{-1} = G_{ij} G_{hi}^{-1}$ are different transformations (we are considering here $\tau = (h, i)$ and $v = (i, j)$). However, from the practical point of view, it is convenient to reduce the number of unknowns. More precisely, for a given *oriented* edge $(\tau, v) \in \bar{\mathcal{E}}$ we consider $a_{\tau v} \in \mathbb{R}_{\neq 0}$ and $\mathbf{v}_{\tau v} \in \mathbb{R}^4$ as unknowns, and we use the relation $Z_{v\tau} = Z_{\tau v}^{-1}$ to

4. 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.

	#Eq.	#Var.										
Our formulation	64	36	64	40	112	63	112	67	192	100	208	109
Trager et al. [16]	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 Equation (18), (22)) and the one proposed in [16] (see Equation (12), (13)). Recall that the latter (described in Equation (7), (10)) is given as a theoretical result in [16] due to its computational complexity, without giving rise to an effective algorithm. Our formulation is more practical as it involves *fewer unknowns*.

handle the opposite edge $(v, \tau) \in \bar{\mathcal{E}}$, where the inverse can be easily computed.

4.4 A Simplified Formulation

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

Proposition 7. *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:*

$$W_{\tau_1 \tau_2} W_{\tau_2 \tau_3} \cdots W_{\tau_\ell \tau_1} = b_k I_4 \quad (19)$$

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

$$W_{\tau v} = I_4 + \mathbf{c}_v \mathbf{u}_{\tau v}^\top \quad (20)$$

where $\mathbf{u}_{\tau v} \in \mathbb{R}^4$ is unknown and $\{i\} = \tau \cap v$.

\mathcal{G} is solvable if and only if the solution to the above system yields $\mathbf{u}_{\tau v} = \mathbf{0}$ for all $(\tau, v) \in \bar{\mathcal{E}}$.

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

$$\mathbf{u}_{\tau v} = \mathbf{v}_{\tau v} / a_{\tau v} \quad (21)$$

which is well defined since $a_{\tau v} \neq 0$. \square

Remark 9. Thanks to Proposition 7, we have four unknowns for each edge $(\tau, v) \in \bar{\mathcal{E}}$ in the line graph, representing a vector $\mathbf{u}_{\tau v} \in \mathbb{R}^4$, plus one unknown scale for each cycle. Thus, the total number of unknowns becomes

$$4\bar{m} + 1(\bar{m} - \bar{n} + 1) = 5\bar{m} - m + 1 \quad (22)$$

which is lower than the formulation related to Theorem 2 where the number of unknowns is $5\bar{m}$ (see Equation (17)). The number of equations remains unchanged and it is given by Equation (18). Table 2 reports a comparison between our simplified formulation and the one in Equation (10) 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 (19) 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 $\mathbf{u}_{\tau v} = \mathbf{0}$ (thanks to Proposition 7), hence Equation (19) 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 $\mathbf{u}_{\tau v} = \mathbf{0}$, then we *always* get a solution to Equation (19). If we assume that there is a unique solution, then it must be equal to $b_k = 1$ and $\mathbf{u}_{\tau v} = \mathbf{0}$, i.e., the graph is solvable thanks to Proposition 7. If the graph is non-solvable, there will be also other solutions. \square

Remark 10. Corollary 1 means that the formulation given in Equation (19) 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 Equation (16), instead, is subject to scale ambiguity, for it involves an unknown scale $a_{\tau v}$ 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 $G_\tau G_v^{-1} = Z_{\tau v}$. Therefore, projective ambiguity is not present in the formulations given in Equation (16) and (19) (that do not involve the matrices G_τ).

5 PROPOSED ALGORITHM

Our algorithm (summarized in Algorithm 1) is a direct consequence of the theoretical results from Section 4; in particular, we follow the simplified formulation derived in Section 4.4, which is based on Equation (19). 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 (19) and (25)
- 5) compute the number s of real solutions
- 6) if $s = 1$ then solvable; else not solvable

Remark 11. Concerning Step 3, we focus on a particular type of cycle consistency basis [34], 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

Nodes	3	4	5	6	7	8	9
Graphs	1	1	1	4	3	36	27
Method	Alg. 1 [16]						
Solvable	1	1	1	4	3	36	17
Not solvable	0	0	0	0	0	0	5
Unknown	0	0	0	0	0	5	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 [16], while our approach provides a complete characterization of all the minimal graphs up to 9 nodes.

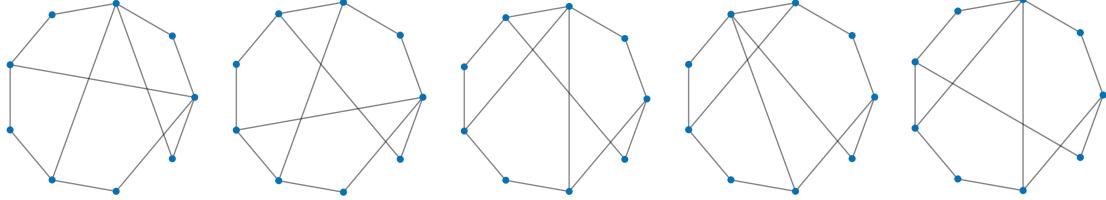


Fig. 6: Some solvable minimal viewing graphs with 9 nodes.

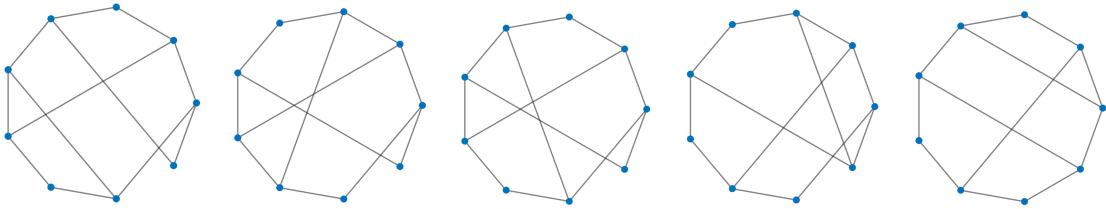


Fig. 7: Some unsolvable minimal viewing graphs with 9 nodes.

from $\bar{\mathcal{E}} \setminus \mathcal{T}$ to \mathcal{T} generates a cycle; the set of such cycles constitutes the *fundamental cycle basis* [33].

Remark 12. As for Step 4, recall that our unknowns comprise one scale $b_k \in \mathbb{R}$ for each cycle and one vector $\mathbf{u}_{\tau v} \in \mathbb{R}^4$ for each edge in the line graph. Such variables must satisfy the following constraints:

$$b_k \neq 0 \quad (23)$$

$$W_{\tau v} = I_4 + \mathbf{c}_i \mathbf{u}_{\tau v}^\top \in GL(4, \mathbb{R}). \quad (24)$$

Instead of explicitly enforcing them, we add the following equation

$$z_{\tau v} \det(I_4 + \mathbf{c}_i \mathbf{u}_{\tau v}^\top) + 1 = 0 \quad (25)$$

for each edge in the line graph, where $z_{\tau v} \in \mathbb{R}$ is an auxiliary variable. Clearly, if $\det(I_4 + \mathbf{c}_i \mathbf{u}_{\tau v}^\top) = 0$ 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 $W_{\tau v}$ are invertible, then the product of a subset of them is also invertible. In other terms, the left term in Equation (19) is invertible for each cycle and hence $b_k \neq 0$. Thus (25) implies both (24) and (23).

Remark 13. Step 5 is based on computational algebraic geometry. In particular, we employ *Gröbner basis* computation [36], 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 [37].

Remark 14. Although our problem is stated over \mathbb{R} , for the sake of efficiency [38] 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} [39],

which is greater or equal to the sought number s of solutions in \mathbb{R} . Recall that $s \geq 1$, since there always exists at least one trivial real solution (given by $\mathbf{u}_{\tau v} = 0$, $b_k = 1$ and $z_{\tau v} = -1$). Several cases are given: i) if $c = \infty$ then $s = \infty$ [16]; ii) if $c = 1$ then $s = 1$; iii) if $c > 1$ then $s \geq 1$. Note that if c is even then $s \geq 2$ since the solutions must come in conjugated pairs.

6 EXPERIMENTS

In this section, we show that our method can be profitably used to check the viewing graph solvability on several examples. Our algorithm is implemented in Macaulay2 [40] and the code is publicly available⁵.

6.1 Synthetic Data

We follow the protocol used in [16] 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 [16] (see Table 2 in [16]), as they satisfy the necessary but not sufficient conditions⁶. 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 Table 3. In particular, the five cases with eight nodes (shown in Figure 1) were found to be all *solvable*.

As for the minimal graphs with nine nodes, there are 22 undecided graphs in [16], which, in particular, are finite solvable (i.e., they identify a finite number of camera configurations).

5. <https://github.com/federica-arrigoni/solvability>

6. Actually, Trager et al [16] manually worked out that one of those graphs is solvable.

# Solutions	Finite		Infinite
	1	≥ 2	∞
Is solvable?	yes	no	no
Is finite solvable?	yes	yes	no

TABLE 4: Connection between solvability and finite solvability. By definition, a graph is finite solvable if it entails a finite number of solutions: in the case of exactly one solution, it is also solvable; in the case of at least 2 solutions, it is non solvable. A graph with an infinite number of solutions is clearly neither solvable nor finite solvable.

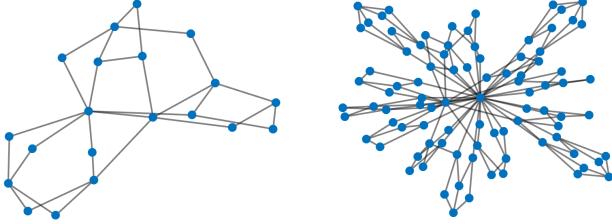


Fig. 8: Examples of solvable minimal viewing graphs with 20 nodes (left) and 90 nodes (right).

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 5: Execution times of Algorithm 1 on some minimal graphs.

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 Figure 6 for some examples). Surprisingly, there exist also some non-solvable graphs among those candidates (see Figure 7 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. [16].

Remark 15. Proving that finite solvability is not equivalent to solvability is of great relevance: it implies that we can not use the finite solvability tools for checking solvability. As explained in [16], finite solvability can be established by simply looking at the rank of a properly constructed linear system. In other terms, finite solvability is much simpler than solvability as it involves a system of linear equations instead of polynomial equations. Table 4 summarizes the connection between these concepts.

Viewing graphs with more than nine nodes are not studied in [16]. Our approach, instead, is able to handle minimal graphs with up to 90 nodes. For instance, we can prove that the graphs reported in Figure 8 are solvable.

Table 5 reports the execution times of Algorithm 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 [36]. 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, as shown in the next paragraph.

Data set	Solvable			Unsolvable		
	by suff.	by Alg. 1	Tot.	by nec.	by Alg. 1	Tot.
Alcatraz Courtyard	200	0	200	0	0	0
Buddah Tooth	182	15	197	3	0	3
Pumpkin	172	17	189	11	0	11
Skansen Kronan	179	11	190	10	0	10
Tsar Nikolai I	181	0	181	19	0	19
Alamo	125	18	143	57	0	57
Ellis Island	139	23	162	38	0	38
Gendarmenmarkt	124	12	136	64	0	64
Madrid Metropolis	100	20	120	80	0	80
Montreal Notre Dame	125	9	134	66	0	66
Notre Dame	158	13	171	29	0	29
NYC Library	93	33	126	74	0	74
Piazza del Popolo	117	15	132	68	0	68
Piccadilly	112	15	127	73	0	73
Roman Forum	110	13	123	76	1	77
Tower of London	108	11	119	81	0	81
Trafalgar	87	9	96	104	0	104
Union Square	73	21	94	106	0	106
Vienna Cathedral	107	6	113	87	0	87
Yorkminster	127	6	133	67	0	67
Cornell Arts Quad	80	28	108	91	1	92

TABLE 6: Characterization of sub-graphs with **eight nodes** sampled from some real viewing graphs [41], [42], [43]. Solvable by sufficiency means that the graph satisfies a sufficient condition, namely being chordal [19]. Unsolvable by necessity means that the graph fails to satisfy some necessary conditions (namely counting the number of edges [16] or parallel rigidity). All the other cases are resolved by our approach (Algorithm 1).

6.2 Real Data

In our experiment, we randomly sample small sub-graphs of large viewing graphs coming from real data sets. In particular, we consider the following datasets comprising either structured or unordered image collections typically used for structure from motion: the Cornell Arts Quad dataset [42]; 13 sequences from the 1DSfM benchmark [43]; 5 sequences taken from [41]. In all the cases, the viewing graph is made available by the authors of the datasets. In particular, for each image sequence, we use only the associated graph and we discard other information (e.g., point correspondences and fundamental matrices), as we only need the graph structure to check solvability.

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 7 nodes within the neighborhood. This, in addition to the original node, yields an eight-node subgraph. Following this procedure, we sample 200 subgraphs from each real graph, without replacement. Observe that these examples are not minimal graphs, in contrast to experiments in Section 6.1. The results, reported in Table 6, tell us that most local sub-graphs 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.

We repeat the same experiment by sampling sub-graphs with nine nodes from real datasets. Results are reported in Table 7, which confirms the outcome of our previous experiment. It is worth observing that the cases of unsolvable graphs detected by our approach are really rare (only one case in Table 7 and two cases in Table 6). This means that the necessary condition derived in [16] (namely counting the number of edges and bi-connectivity) together with our new necessary condition (namely parallel rigidity) are indeed strong and can be used in practice to

Data set	Solvable			Unsolvable		
	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 7: Characterization of sub-graphs with **nine nodes** sampled from some real viewing graphs [41], [42], [43]. Solvable by sufficiency means that the graph satisfies a sufficient condition, namely being chordal [19]. Unsolvable by necessity means that the graph fails to satisfy some necessary conditions (namely counting the number of edges [16] or parallel rigidity). All the other cases are resolved by our approach (Algorithm 1).

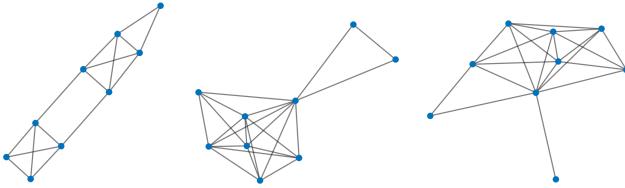


Fig. 9: Examples of unsolvable sub-graphs with nine nodes sampled from the viewing graph of the Pumpkin data set [41]. The left one has been detected with Algorithm 1, whereas the others are not biconnected (hence they do not satisfy a necessary condition for solvability [16]).

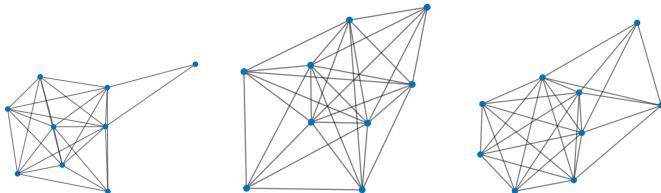


Fig. 10: Examples of solvable sub-graphs with nine nodes sampled from the viewing graph of the Pumpkin data set [41].

successfully detect many unsolvable graphs. The same conclusion, however, does not hold for the solvable examples: previous results are indeed unable to prove that many real graphs are solvable, as already observed in Table 3, thus motivating the need of Algorithm 1. Some examples of unsolvable cases are given in Figure 9 whereas Figure 10 reports some solvable examples.

7 EXAMPLES

This section is not essential for understanding the method and can be skipped at first reading. We focus here on the most relevant equations related to our formulation, and we show how they look

like on some examples, where we also analyze the output of our algorithm.

7.1 Non-solvable graph (infinite number of solutions).

Suppose that \mathcal{G} is a cycle of length 4, represented in Figure 11. Let $\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3, \mathbf{c}_4 \in \mathbb{R}^4$ represent known (generic) camera centres. Hereafter, for a given edge $(\tau, v) \in \mathcal{E}$ in the line graph – which corresponds to two adjacent edges in the original graph (i.e., $\tau = (h, i) \in \mathcal{E}$ and $v = (i, j) \in \mathcal{E}$) – we will use the triplet (h, i, j) instead of (τ, v) for simplicity of exposition. Equation (10) rewrites

$$\begin{aligned} G_{12}G_{23}^{-1} &= a_{123}I_4 + \mathbf{c}_2\mathbf{v}_{123}^\top \\ G_{23}G_{34}^{-1} &= a_{234}I_4 + \mathbf{c}_3\mathbf{v}_{234}^\top \\ G_{34}G_{41}^{-1} &= a_{341}I_4 + \mathbf{c}_4\mathbf{v}_{341}^\top \\ G_{41}G_{12}^{-1} &= a_{412}I_4 + \mathbf{c}_1\mathbf{v}_{412}^\top \end{aligned} \quad (26)$$

where the following variables are unknown

$$\begin{aligned} G_{12}, G_{23}, G_{34}, G_{41} &\in GL(4, \mathbb{R}) \\ a_{123}, a_{234}, a_{341}, a_{412} &\in \mathbb{R}_{\neq 0} \\ \mathbf{v}_{123}, \mathbf{v}_{234}, \mathbf{v}_{341}, \mathbf{v}_{412} &\in \mathbb{R}^4. \end{aligned} \quad (27)$$

The line graph consists of a single cycle (of length 4), which is also a fundamental cycle basis (associated, e.g., with the spanning tree $\mathcal{T} = \{(12, 23), (23, 34), (34, 41)\}$), as shown in Figure 11. Equation (16) rewrites

$$\begin{aligned} I_4 &= (a_{123}I_4 + \mathbf{c}_2\mathbf{v}_{123}^\top)(a_{234}I_4 + \mathbf{c}_3\mathbf{v}_{234}^\top) \\ &\quad (a_{341}I_4 + \mathbf{c}_4\mathbf{v}_{341}^\top)(a_{412}I_4 + \mathbf{c}_1\mathbf{v}_{412}^\top) \end{aligned} \quad (28)$$

where the following variables are unknown

$$\begin{aligned} a_{123}, a_{234}, a_{341}, a_{412} &\in \mathbb{R}_{\neq 0} \\ \mathbf{v}_{123}, \mathbf{v}_{234}, \mathbf{v}_{341}, \mathbf{v}_{412} &\in \mathbb{R}^4. \end{aligned} \quad (29)$$

Equation (19) rewrites

$$bI_4 = (I_4 + \mathbf{c}_2\mathbf{u}_{123}^\top)(I_4 + \mathbf{c}_3\mathbf{u}_{234}^\top)(I_4 + \mathbf{c}_4\mathbf{u}_{341}^\top)(I_4 + \mathbf{c}_1\mathbf{u}_{412}^\top) \quad (30)$$

where the following variables are unknown

$$\begin{aligned} b &\in \mathbb{R}_{\neq 0} \\ \mathbf{u}_{123}, \mathbf{u}_{234}, \mathbf{u}_{341}, \mathbf{u}_{412} &\in \mathbb{R}^4. \end{aligned} \quad (31)$$

Observe that Equation (30) involves less unknowns than (28), which in turn involves less unknowns than (26), as already

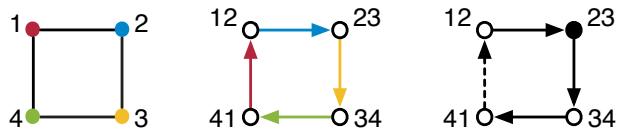


Fig. 11: Non-solvable viewing graph with 4 vertices (left) and corresponding line graph (middle), where edges are oriented arbitrarily. Colors clarify correspondences between edges in the line graph and vertices in the original graph. On the right a spanning tree is reported, where the root is coloured in black and the only non-tree edge is drawn with a dashed arrow.

$$\begin{aligned}
& z_{_1-z_3}, z_{_0-z_2}, u_{_15-559z_2-559}, u_{_14-10838z_3-10838}, u_{_13+7287z_2+7287}, u_{_12-7283z_3-7283}, u_{_11+7762z_2+7762}, u_{_10-12207z_3-12207}, \\
& u_{_9-12548z_2-12548}, u_{_8-7348z_3-7348}, u_{_7-14623z_2-14623}, u_{_6-6734z_3-6734}, u_{_5+11215z_2+11215}, u_{_4-2201z_3-2201}, u_{_3-4301z_2-4301}, \\
& u_{_2+14484z_3+14484}, u_{_1+11757z_2+11757}, u_{_0-11403z_3-11403}, b_{_0+z_3}, z_{_2z_3-1}
\end{aligned}$$

Fig. 12: Gröbner basis associated with the polynomial system in Equation (30) and (32), for a specific set of camera centres. For coherence with our Macaulay2 implementation, here variables are linearly (0-based) indexed. Each term represents a polynomial which should be equal to zero for the sought solution(s).

observed in Section 4. Finally, the auxiliary equations given in (25) become

$$\begin{aligned}
& z_{123} \det(I_4 + \mathbf{c}_2 \mathbf{u}_{123}^\top) + 1 = 0 \\
& z_{234} \det(I_4 + \mathbf{c}_3 \mathbf{u}_{234}^\top) + 1 = 0 \\
& z_{341} \det(I_4 + \mathbf{c}_4 \mathbf{u}_{341}^\top) + 1 = 0 \\
& z_{412} \det(I_4 + \mathbf{c}_1 \mathbf{u}_{412}^\top) + 1 = 0
\end{aligned} \tag{32}$$

where $z_{123}, z_{234}, z_{341}, z_{412} \in \mathbb{R}$ are unknown.

The fact that the graph in Figure 11 is not solvable can be easily deduced by counting the number of edges: the necessary condition $m \geq (11n - 15)/7$ is not satisfied here (see [16]). However, it is useful to analyze the output of our algorithm on this simple example. Specifically, Figure 12 shows the generators of the Gröbner basis [36] associated with the polynomial system in Equation (30) and (32), for a specific configuration of camera centers (sampled at random). Such generators encode a set of equations which is equivalent to the original system but at the same time it is much simpler. Let us consider the last generator (i.e. $z_{_2z_3-1}=0$): note that the product of $z_{_2}$ and $z_{_3}$ is fixed (it is equal to 1), but there is an *infinite* number of solutions that satisfy such equation. Observe also that the remaining variables are uniquely determined – given $z_{_2}$ and $z_{_3}$ – as all other equations are linear and they involve one unknown at a time (in addition to $z_{_2}$ or $z_{_3}$).

7.2 Non-solvable graph (finite number of solutions).

The previous example refers to a non-solvable graph with an infinite number of solutions. We now consider an example of a non-solvable graph where the number of solutions is finite but strictly greater than one. Specifically, let us consider the graph with 9 nodes reported in Figure 7 (left). Our algorithm computed two solutions on this example, meaning that the graph is not solvable. Since a lot of variables are involved here, we do not explicitly write all the equations for this example, but we only analyze the associated Gröbner basis obtained for a specific configuration of random cameras (see Figure 13). Let us consider the last generator (i.e., $z_{_20^2-4598z_20-4599}=0$): observe that it defines an equation of degree two in one unknown. Observe also that all other equations are linear and they involve one unknown at a time (in addition to $z_{_20}$). Since we know that one of the solutions is $z_{_20} = -1$, we see that the other solution for $z_{_20}$ has to be real as well (namely $z_{_20} = 4599$). Thus, all other variables have to be real too, since they are linear functions of $z_{_20}$. Hence, the whole polynomial system admits two distinct real solutions.

7.3 Solvable graph.

Suppose that \mathcal{G} is the graph reported in Figure 15. Let $\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3, \mathbf{c}_4 \in \mathbb{R}^4$ represent known (generic) camera centres.

Equation (10) rewrites

$$\begin{aligned}
G_{41}G_{12}^{-1} &= a_{412}I_4 + \mathbf{c}_1 \mathbf{v}_{412}^\top \\
G_{12}G_{23}^{-1} &= a_{123}I_4 + \mathbf{c}_2 \mathbf{v}_{123}^\top \\
G_{23}G_{34}^{-1} &= a_{234}I_4 + \mathbf{c}_3 \mathbf{v}_{234}^\top \\
G_{34}G_{41}^{-1} &= a_{341}I_4 + \mathbf{c}_4 \mathbf{v}_{341}^\top \\
G_{12}G_{42}^{-1} &= a_{124}I_4 + \mathbf{c}_2 \mathbf{v}_{124}^\top \\
G_{42}G_{23}^{-1} &= a_{423}I_4 + \mathbf{c}_2 \mathbf{v}_{423}^\top \\
G_{41}G_{42}^{-1} &= a_{142}I_4 + \mathbf{c}_4 \mathbf{v}_{142}^\top \\
G_{42}G_{34}^{-1} &= a_{243}I_4 + \mathbf{c}_4 \mathbf{v}_{243}^\top
\end{aligned} \tag{33}$$

where the following variables are unknown

$$\begin{aligned}
G_{12}, G_{23}, G_{34}, G_{41}, G_{42} &\in GL(4, \mathbb{R}) \\
a_{412}, a_{123}, a_{234}, a_{341}, a_{124}, a_{423}, a_{142}, a_{243} &\in \mathbb{R}_{\neq 0} \\
\mathbf{v}_{412}, \mathbf{v}_{123}, \mathbf{v}_{234}, \mathbf{v}_{341}, \mathbf{v}_{124}, \mathbf{v}_{423}, \mathbf{v}_{142}, \mathbf{v}_{243} &\in \mathbb{R}^4.
\end{aligned} \tag{34}$$

If we consider the spanning tree $\mathcal{T} = \{(12, 42), (42, 23), (23, 34), (34, 12)\}$, then the line graph admits a fundamental cycle basis composed of four cycles (see Figure 15): $C_1 = (12, 23, 42)$, $C_2 = (42, 23, 34)$, $C_3 = (23, 34, 12)$ and $C_4 = (12, 34, 42)$. Observe that each cycle consists of a sequence of vertices that is traversed in a cyclic order (clockwise or anticlockwise): for each edge in the cycle, we consider the associated matrix or its inverse if the edge is traversed in forward or backward direction, respectively. Thus Equation (16) becomes

$$\begin{aligned}
I_4 &= (a_{123}I_4 + \mathbf{c}_2 \mathbf{v}_{123}^\top)(a_{423}I_4 + \mathbf{c}_2 \mathbf{v}_{423}^\top)^{-1}(a_{124}I_4 + \mathbf{c}_2 \mathbf{v}_{124}^\top)^{-1} \\
I_4 &= (a_{423}I_4 + \mathbf{c}_2 \mathbf{v}_{423}^\top)(a_{234}I_4 + \mathbf{c}_3 \mathbf{v}_{234}^\top)(a_{243}I_4 + \mathbf{c}_4 \mathbf{v}_{243}^\top)^{-1} \\
I_4 &= (a_{243}I_4 + \mathbf{c}_4 \mathbf{v}_{243}^\top)(a_{341}I_4 + \mathbf{c}_4 \mathbf{v}_{341}^\top)(a_{142}I_4 + \mathbf{c}_4 \mathbf{v}_{142}^\top)^{-1} \\
I_4 &= (a_{412}I_4 + \mathbf{c}_1 \mathbf{v}_{412}^\top)(a_{124}I_4 + \mathbf{c}_2 \mathbf{v}_{124}^\top)(a_{142}I_4 + \mathbf{c}_4 \mathbf{v}_{142}^\top)^{-1}
\end{aligned} \tag{35}$$

where the following variables are unknown

$$\begin{aligned}
a_{412}, a_{123}, a_{234}, a_{341}, a_{124}, a_{423}, a_{142}, a_{243} &\in \mathbb{R}_{\neq 0} \\
\mathbf{v}_{412}, \mathbf{v}_{123}, \mathbf{v}_{234}, \mathbf{v}_{341}, \mathbf{v}_{124}, \mathbf{v}_{423}, \mathbf{v}_{142}, \mathbf{v}_{243} &\in \mathbb{R}^4.
\end{aligned} \tag{36}$$

Equation (19) rewrites

$$\begin{aligned}
b_1 I_4 &= (I_4 + \mathbf{c}_2 \mathbf{u}_{123}^\top)(I_4 + \mathbf{c}_2 \mathbf{u}_{423}^\top)^{-1}(I_4 + \mathbf{c}_2 \mathbf{u}_{124}^\top)^{-1} \\
b_2 I_4 &= (I_4 + \mathbf{c}_2 \mathbf{u}_{423}^\top)(I_4 + \mathbf{c}_3 \mathbf{u}_{234}^\top)(I_4 + \mathbf{c}_4 \mathbf{u}_{243}^\top)^{-1} \\
b_3 I_4 &= (I_4 + \mathbf{c}_4 \mathbf{u}_{243}^\top)(I_4 + \mathbf{c}_4 \mathbf{u}_{341}^\top)(I_4 + \mathbf{c}_4 \mathbf{u}_{142}^\top)^{-1} \\
b_4 I_4 &= (I_4 + \mathbf{c}_1 \mathbf{u}_{412}^\top)(I_4 + \mathbf{c}_2 \mathbf{u}_{124}^\top)(I_4 + \mathbf{c}_4 \mathbf{u}_{142}^\top)^{-1}
\end{aligned} \tag{37}$$

where the following variables are unknown

$$\begin{aligned}
b_1, b_2, b_3, b_4 &\in \mathbb{R}_{\neq 0} \\
\mathbf{u}_{412}, \mathbf{u}_{123}, \mathbf{u}_{234}, \mathbf{u}_{341}, \mathbf{u}_{124}, \mathbf{u}_{423}, \mathbf{u}_{142}, \mathbf{u}_{243} &\in \mathbb{R}^4.
\end{aligned} \tag{38}$$

Observe that the formulation implemented by our method (given in Equation (37)) involves less unknowns than the one proposed

$z_{19+12604z_{20+12605}}, z_{18-11727z_{20-11726}}, z_{17+894z_{20+895}}, z_{16-z_{20}}, z_{15-334z_{20-3342}}, z_{14+894z_{20+895}}, z_{13+11565z_{20+11566}},$
 $z_{12-13791z_{20-13790}}, z_{11+8203z_{20+8204}}, z_{10-8522z_{20-8521}}, z_{9+7215z_{20+7216}}, z_{8-1877z_{20-1876}}, z_{7-11118z_{20-11117}}, z_{6-7063z_{20-7062}},$
 $z_{5-7412z_{20-7411}}, z_{4+8437z_{20+8438}}, z_{3+1796z_{20+1797}}, z_{2+5030z_{20+5031}}, z_{1-2848z_{20-2847}}, z_{0-9253z_{20-9252}}, u_{83-8414z_{20-8414}},$
 $u_{82+10987z_{20+10987}}, u_{81-3919z_{20-3919}}, u_{80+973z_{20+973}}, u_{79-10847z_{20-10847}}, u_{78-5390z_{20-5390}}, u_{77+431z_{20+431}}, u_{76-13455z_{20-13455}},$
 $u_{75-3615z_{20-3615}}, u_{74+3100z_{20+3100}}, u_{73-9420z_{20-9420}}, u_{72-10243z_{20-10243}}, u_{71+4517z_{20+4517}}, u_{70+1766z_{20+1766}}, u_{69+6198z_{20+6198}},$
 $u_{68+681z_{20+681}}, u_{67-7149z_{20-7149}}, u_{66-2977z_{20-2977}}, u_{65+11405z_{20+11405}}, u_{64-3437z_{20-3437}}, u_{63-7836z_{20-7836}}, u_{62+13583z_{20+13583}},$
 $u_{61+1799z_{20+1799}}, u_{60-8356z_{20-8356}}, u_{59-10495z_{20-10495}}, u_{58-9989z_{20-9989}}, u_{57+9771z_{20+9771}}, u_{56+12865z_{20+12865}}, u_{55+10476z_{20+10476}},$
 $u_{54-13684z_{20-13684}}, u_{53-14439z_{20-14439}}, u_{52-6145z_{20-6145}}, u_{51-11189z_{20-11189}}, u_{50-3733z_{20-3733}}, u_{49+10275z_{20+10275}}, u_{48-4355z_{20-4355}},$
 $u_{47+14504z_{20+14504}}, u_{46+13506z_{20+13506}}, u_{45+10191z_{20+10191}}, u_{44-1472z_{20-1472}}, u_{43+6220z_{20+6220}}, u_{42+9716z_{20+9716}}, u_{41-66z_{20-66}},$
 $u_{40+2077z_{20+2077}}, u_{39+9070z_{20+9070}}, u_{38+603z_{20+603}}, u_{37-7509z_{20-7509}}, u_{36-12738z_{20-12738}}, u_{35-10638z_{20-10638}}, u_{34+1764z_{20+1764}},$
 $u_{33-9009z_{20-9009}}, u_{32-8517z_{20-8517}}, u_{31+8333z_{20+8333}}, u_{30+1020z_{20+1020}}, u_{29-10090z_{20-10090}}, u_{28-14978z_{20-14978}}, u_{27-13238z_{20-13238}},$
 $u_{26-9412z_{20-9412}}, u_{25-13990z_{20-13990}}, u_{24-11470z_{20-11470}}, u_{23-14044z_{20-14044}}, u_{22-1292z_{20-1292}}, u_{21+1034z_{20+1034}}, u_{20-4116z_{20-4116}},$
 $u_{19-1995z_{20-1995}}, u_{18-14930z_{20-14930}}, u_{17+3697z_{20+3697}}, u_{16-10633z_{20-10633}}, u_{15-900z_{20-900}}, u_{14+458z_{20+458}}, u_{13+5440z_{20+5440}},$
 $u_{12-11618z_{20-11618}}, u_{11-2465z_{20-2465}}, u_{10-247z_{20-247}}, u_{9-10781z_{20-10781}}, u_{8-6754z_{20-6754}}, u_{7-4883z_{20-4883}}, u_{6-6092z_{20-6092}},$
 $u_{5+5548z_{20+5548}}, u_{4-2399z_{20-2399}}, u_{3-4745z_{20-4745}}, u_{2-3226z_{20-3226}}, u_{1-5879z_{20-5879}}, u_{0-1620z_{20-1620}}, b_{9-1}, b_{8+6278z_{20+6277}}, b_{7-1},$
 $b_{6+11118z_{20+11117}}, b_{5+7063z_{20+7062}}, b_{4+7167z_{20+7166}}, b_{3-1}, b_{2+8994z_{20+8993}}, b_{1-1}, b_{0-1}, z_{20^2-4598z_{20-4599}}$

Fig. 13: Gröbner basis for checking solvability for the graph in Figure 7 (left). For coherence with our Macaulay2 implementation, here variables are linearly (0-based) indexed. Each term represents a polynomial which should be equal to zero for the sought solution(s).

$z_{7+1}, z_{6+1}, z_{5+1}, z_{4+1}, z_{3+1}, z_{2+1}, z_{1+1}, z_{0+1}, u_{31}, u_{30}, u_{29}, u_{28}, u_{27}, u_{26}, u_{25}, u_{24}, u_{23}, u_{22}, u_{21}, u_{20}, u_{19}, u_{18}, u_{17},$
 $u_{16}, u_{15}, u_{14}, u_{13}, u_{12}, u_{11}, u_{10}, u_{9}, u_{8}, u_{7}, u_{6}, u_{5}, u_{4}, u_{3}, u_{2}, u_{1}, u_{0}, b_{3-1}, b_{2-1}, b_{1-1}, b_{0-1}$

Fig. 14: Gröbner basis associated with the polynomial system in Equation (39) and (40), for a specific set of camera centres. For coherence with our Macaulay2 implementation, here variables are linearly (0-based) indexed. Each term represents a polynomial which should be equal to zero for the sought solution.

in [16] (given in Equation (33)). By computing inverses explicitly, Equation (37) rewrites:

$$\begin{aligned}
 & (1 + \mathbf{c}_2^\top \mathbf{u}_{423})(1 + \mathbf{c}_2^\top \mathbf{u}_{124})b_1 I_4 = \\
 & = (I_4 + \mathbf{c}_2 \mathbf{u}_{123}^\top)((1 + \mathbf{c}_2^\top \mathbf{u}_{423})I_4 - \mathbf{c}_2 \mathbf{u}_{423}^\top)((1 + \mathbf{c}_2^\top \mathbf{u}_{124})I_4 - \mathbf{c}_2 \mathbf{u}_{124}^\top) \\
 & \quad (1 + \mathbf{c}_4^\top \mathbf{u}_{243})b_2 I_4 = \\
 & = (I_4 + \mathbf{c}_2 \mathbf{u}_{423}^\top)(I_4 + \mathbf{c}_3 \mathbf{u}_{234}^\top)((1 + \mathbf{c}_4^\top \mathbf{u}_{243})I_4 - \mathbf{c}_4 \mathbf{u}_{243}^\top) \\
 & \quad b_3 I_4 = (I_4 + \mathbf{c}_4 \mathbf{u}_{243}^\top)(I_4 + \mathbf{c}_4 \mathbf{u}_{341}^\top)(I_4 + \mathbf{c}_4 \mathbf{u}_{142}^\top) \\
 & \quad (1 + \mathbf{c}_4^\top \mathbf{u}_{142})b_4 I_4 = \\
 & = (I_4 + \mathbf{c}_1 \mathbf{u}_{412}^\top)(I_4 + \mathbf{c}_2 \mathbf{u}_{124}^\top)((1 + \mathbf{c}_4^\top \mathbf{u}_{142})I_4 - \mathbf{c}_4 \mathbf{u}_{142}^\top) \tag{39}
 \end{aligned}$$

Finally, the auxiliary equations given in (25) become

$$\begin{aligned}
 & z_{412} \det(I_4 + \mathbf{c}_1 \mathbf{u}_{412}^\top) + 1 = 0 \\
 & z_{123} \det(I_4 + \mathbf{c}_2 \mathbf{u}_{123}^\top) + 1 = 0 \\
 & z_{234} \det(I_4 + \mathbf{c}_3 \mathbf{u}_{234}^\top) + 1 = 0 \\
 & z_{341} \det(I_4 + \mathbf{c}_4 \mathbf{u}_{341}^\top) + 1 = 0 \\
 & z_{124} \det(I_4 + \mathbf{c}_2 \mathbf{u}_{124}^\top) + 1 = 0 \\
 & z_{423} \det(I_4 + \mathbf{c}_2 \mathbf{u}_{423}^\top) + 1 = 0 \\
 & z_{142} \det(I_4 + \mathbf{c}_4 \mathbf{u}_{142}^\top) + 1 = 0 \\
 & z_{243} \det(I_4 + \mathbf{c}_4 \mathbf{u}_{243}^\top) + 1 = 0 \tag{40}
 \end{aligned}$$

where $z_{412}, z_{123}, z_{234}, z_{341}, z_{124}, z_{423}, z_{142}, z_{243} \in \mathbb{R}$ are unknown.

The graph in Figure 15 is solvable according to [9] and our algorithm returns exactly one solution. Such conclusion can also be easily deduced from Figure 14, which reports the generators of the Gröbner basis associated with the polynomial system in Equation (39) and (40). Note that each generator has exactly one solution, being a linear equation in one variable. In particular, we get $z_0 = z_1 = \dots = z_7 = -1$, $b_0 = b_1 = b_2 =$

$b_3 = 1$ and $u_0 = u_1 = \dots = u_{31} = 0$, as expected for a solvable case.

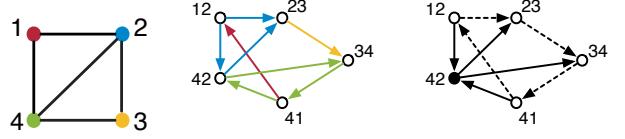


Fig. 15: Solvable viewing graph with 4 vertices (left) and corresponding line graph (middle), where edges are oriented arbitrarily. Please note that a vertex of the original graph (e.g., vertex 2) can appear multiple times as an edge of the line graph, as clarified by colors. On the right a spanning tree is reported, where the root is coloured in black and non-tree edges are drawn with dashed arrows.

8 CONCLUSIONS AND FUTURE WORK

We have studied the solvability of viewing graphs, i.e. whether they uniquely determine projective cameras, and have made several important advances in the theory and practical use of viewing graphs. Based on [16], we proposed a new characterization involving fewer unknowns by exploiting the 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 [16], and proved that *finite solvability does not imply solvability*, thus answering an open question. Moreover, we formally derived a connection with the calibrated case (parallel rigidity), which gives a new necessary condition for solvability.

We were able to process minimal graphs with up to 90 vertices, which establishes the state of the art in the *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 handle is a matter of designing clever solvers and exploiting computing power: we are working to push this limit forward. For example, we plan to investigate numerical algebraic geometry (e.g., [44]), which gives good grounds to expect to make the computation tractable for large-scale scenarios.

In this paper, we have considered the concept of solvability given in Definition 2, which is based solely on the topology of the viewing graph. The use of additional information (e.g., points) would give rise to a different notion of solvability [45], which would be interesting to explore in the future. Besides being of theoretical interest, the solvability problem has a practical impact since reconstruction methods such as [11], [12] 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.

ACKNOWLEDGMENTS

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 European Union’s Horizon 2020 Research and Innovation Programme under the project SPRING (Grant Agreement No. 871245).

REFERENCES

- [1] N. Snavely, S. M. Seitz, and R. Szeliski, “Photo tourism: exploring photo collections in 3D,” in *ACM SIGGRAPH*, 2006. [1](#)
- [2] ——, “Modeling the world from internet photo collections,” *International Journal of Computer Vision*, vol. 80, no. 2, pp. 189–210, 2008. [1](#)
- [3] J. L. Schönberger and J.-M. Frahm, “Structure-from-motion revisited,” in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. [1](#)
- [4] I. Rocco, M. Cimpoi, R. Arandjelović, A. Torii, T. Pajdla, and J. Sivic, “Neighbourhood consensus networks,” in *Neural Information Processing Systems (NeurIPS)*, 2018. [1](#)
- [5] D. Nistér, O. Naroditsky, and J. Bergen, “Visual odometry,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2004, pp. 652–659. [1](#)
- [6] 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](#)
- [7] T. Sattler, B. Leibe, and L. Kobbelt, “Efficient & effective prioritized matching for large-scale image-based localization,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 9, pp. 1744–1756, 2017. [1](#)
- [8] 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](#)
- [9] N. Levi and M. Werman, “The viewing graph,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2003, pp. 518 – 522. [1, 2, 3, 4, 12](#)
- [10] O. Ozysil, V. Voroninski, R. Basri, and A. Singer, “A survey of structure from motion,” *Acta Numerica*, vol. 26, pp. 305 – 364, 2017. [1, 2](#)
- [11] 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*, 2017, pp. 2413–2421. [1, 2, 13](#)
- [12] 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*, 2019, pp. 3259–3267. [1, 2, 13](#)
- [13] O. Ozysil and A. Singer, “Robust camera location estimation by convex programming,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 2674 – 2683. [1, 3](#)
- [14] 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, 4](#)
- [15] F. Arrigoni and A. Fusielo, “Bearing-based network localizability: A unifying view,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 9, pp. 2049 – 2069, 2019. [1, 2, 3, 4](#)
- [16] M. Trager, B. Osserman, and J. Ponce, “On the solvability of viewing graphs,” in *Proceedings of the European Conference on Computer Vision*, 2018, pp. 335–350. [2, 3, 4, 7, 8, 9, 10, 11, 12](#)
- [17] 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*, 2011, p. 38–44. [2](#)
- [18] A. Rudi, M. Pizzoli, and F. Pirri, “Linear solvability in the viewing graph,” in *Proceedings of the Asian Conference on Computer Vision*, 2011, pp. 369–381. [2, 3](#)
- [19] M. Trager, M. Hebert, and J. Ponce, “The joint image handbook,” in *Proceedings of the International Conference on Computer Vision*, 2015, pp. 909–917. [2, 3, 9, 10](#)
- [20] T. Eren, W. Whiteley, and P. N. Belhumeur, “A theoretical analysis of the conditions for unambiguous node localization in sensor networks,” Columbia University, Computer Science Department, Tech. Rep. CU-CS-O32-M, 2004. [2, 4](#)
- [21] F. Arrigoni, A. Fusielo, E. Ricci, and T. Pajdla, “Viewing graph solvability via cycle consistency,” in *Proceedings of the International Conference on Computer Vision*, 2021. [2](#)
- [22] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, 2004. [2](#)
- [23] W. Whiteley, “Matroids from discrete geometry,” in *Matroid Theory*, ser. AMS Contemporary Mathematics. American Mathematical Society, 1997, pp. 171–313. [3](#)
- [24] F. Arrigoni, A. Fusielo, and B. Rossi, “On computing the translations norm in the epipolar graph,” in *Proceedings of the International Conference on 3D Vision (3DV)*, 2015, pp. 300–308. [4](#)
- [25] T. Eren, W. Whiteley, and P. N. Belhumeur, “Using angle of arrival (bearing) information in network localization,” in *Proceedings of the IEEE Conference on Decision and Control*, 2006, pp. 4676 – 4681. [4](#)
- [26] M. Michelini and H. Mayer, “Structure from motion for complex image sets,” *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 166, pp. 140–152, 2020. [5](#)
- [27] F. Harary, *Graph Theory*. Addison-Wesley, 1972. [5](#)
- [28] A. Singer, “Angular synchronization by eigenvectors and semidefinite programming,” *Applied and Computational Harmonic Analysis*, vol. 30, no. 1, pp. 20 – 36, 2011. [5](#)
- [29] V. M. Govindu, “Lie-algebraic averaging for globally consistent motion estimation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2004, pp. 684–691. [5](#)
- [30] 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. [5](#)
- [31] F. Arrigoni and A. Fusielo, “Synchronization problems in computer vision with closed-form solutions,” *International Journal of Computer Vision*, vol. 128, p. 26–52, 2020. [5, 6](#)
- [32] D. Cvetkovic, P. Rowlinson, and S. Simic, *Spectral Generalizations of Line Graphs: On Graphs with Least Eigenvalue -2*, ser. London Mathematical Society Lecture Note Series. Cambridge University Press, 2004. [5](#)
- [33] 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*, vol. 3, no. 4, pp. 199–243, 2009. [6, 8](#)
- [34] L. J. Guibas, Q. Huang, and Z. Liang, “A condition number for joint optimization of cycle-consistent networks,” in *Neural Information Processing Systems*, vol. 32, 2019, pp. 1007–1017. [6, 7](#)
- [35] S. Guillemot, “FPT algorithms for path-traversal and cycle-traversal problems,” *Discrete Optimization*, vol. 8, no. 1, pp. 61 – 71, 2011. [6](#)
- [36] T. W. Dubé, “The structure of polynomial ideals and Gröbner bases,” *SIAM Journal on Computing*, vol. 19, no. 4, pp. 750 – 773, 1990. [8, 9, 11](#)
- [37] D. Lazard, “Gröbner bases, Gaussian elimination and resolution of systems of algebraic equations,” in *Computer Algebra*, J. A. van Hulzen, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1983, pp. 146–156. [8](#)
- [38] E. A. Arnold, “Modular algorithms for computing Gröbner bases,” *Journal of Symbolic Computation*, vol. 35, no. 4, pp. 403 – 419, 2003. [8](#)

- [39] I. Shafarevich and K. Hirsch, *Basic algebraic geometry*. Springer, 1994, vol. 1. 8
- [40] D. R. Grayson and M. E. Stillman, “Macaulay2, a software system for research in algebraic geometry,” Available at <https://math.uiuc.edu/Macaulay2/>. 8
- [41] 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)*. Springer-Verlag, 2011, pp. 524–535. 9, 10
- [42] 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*, 2011, pp. 3001–3008. 9, 10
- [43] K. Wilson and N. Snavely, “Robust global translations with 1DSfM,” in *Proceedings of the European Conference on Computer Vision*, 2014, pp. 61–75. 9, 10
- [44] 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*, vol. 39, no. 3, pp. 1421–1446, 2018. 13
- [45] B. Nasihatkon, R. Hartley, and J. Trumpf, “A generalized projective reconstruction theorem and depth constraints for projective factorization,” *International Journal of Computer Vision*, vol. 115, pp. 87 – 115, 2015. 13



Elisa Ricci received the PhD degree from the University of Perugia in 2008. She is an associate professor at the University of Trento and a researcher at Fondazione Bruno Kessler. She has since been a post-doctoral researcher at Idiap, Martigny, and Fondazione Bruno Kessler, Trento. She was also a visiting researcher at the University of Bristol. Her research interests are mainly in the areas of computer vision and machine learning. She is a member of the IEEE and an ELLIS fellow.



Tomas Pajdla (Member, IEEE) received the M.Sc. and Ph.D. degrees from the Czech Technical University, in Prague. He works in geometry and algebra of computer vision and robotics, with emphasis on non-classical cameras, 3D reconstruction, and industrial vision. He contributed to the epipolar geometry of panoramic cameras, non-central camera models, generalized epipolar geometries, and to developing solvers for minimal problems in structure from motion.



Federica Arrigoni received her MS degree in Mathematics from the University of Milan (Italy) in 2013, and the PhD degree in Industrial and Information Engineering from the University of Udine (Italy) in 2018. From 2018 to 2020 she was a postdoctoral researcher with the Czech Technical University in Prague. Then, she was an assistant professor with the University of Trento (Italy), from 2020 to 2022. She is currently a tenure-track assistant professor with the Politecnico di Milano (Italy). Her research focuses on geometric problems in 3D computer vision.



Andrea Fusielo received the Laurea (M.S.) degree in computer science from the University of Udine, Udine, Italy, and the Dottorato di Ricerca (Ph.D.) degree in computer engineering from the University of Trieste, in 1994 and 1999, respectively. He was a Research Fellow with Heriot-Watt University, Edinburgh, in 1999. From 2001 to 2011, he was with the Department of Computer Science, University of Verona. As an Associate Professor in 2012 he joined the DPIA at the University of Udine, where he teaches computer vision and computer science basics. His current research interests include computer vision, image analysis and 3-D model acquisition.



Romeo Rizzi received his Ph.D. by the Department of Mathematics of Padova University, Italy, in 1997. He held researcher positions at centers like CWI (Amsterdam, Netherlands), BRICS (Aarhus, Denmark) and FBK (Trento, Italy). In 2001, he became Assistant Professor by the University of Trento. In 2005, he became Associate Professor by the University of Udine. Currently, he is full professor in Operations Research at the University of Verona, Italy. His main interests are in Combinatorial Optimization and Algorithms. He is an Area Editor of 4OR and acts as a Reviewer for the American Mathematical Society.