Viewing Graph Solvability in Practice

Federica Arrigoni
Politecnico di Milano (Italy)
federica.arrigoni@polimi.it

Tomas Pajdla
CIIRC - CTU in Prague (Czechia)
pajdla@cvut.cz

Andrea Fusiello
University of Udine (Italy)
andrea.fusiello@uniud.it

Abstract

We present an advance in understanding the projective Structure-from-Motion, focusing in particular on the viewing graph: such a graph has cameras as nodes and fundamental matrices as edges. We propose a practical method for testing finite solvability, i.e., whether a viewing graph induces a finite number of camera configurations. Our formulation uses a significantly smaller number of equations (up to $400\times$) with respect to previous work. As a result, this is the only method in the literature that can be applied to large viewing graphs coming from real datasets, comprising up to $300K$ edges. In addition, we develop the first algorithm for identifying maximal finite-solvable components.

1. Introduction

Structure-from-Motion [13] has been widely explored for its various applications in Computer Vision, with most research focusing on developing robust 3D scene reconstruction algorithms. In this respect, one compelling problem is establishing *if* a set of fundamental matrices uniquely determines a configuration of cameras. Indeed, it is known that fundamental matrices can be uniquely computed from the camera matrices [7], but the converse may not be true when only a *subset* of all possible fundamental matrices is available (see Fig. 1). The *viewing graph* [10] can be analyzed to answer this question. Such a graph has vertices corresponding to cameras, and an edge is present if and only if the fundamental matrix between the two cameras exists.

A *solvable* graph uniquely identifies a configuration of cameras up to a single transformation (projective or Euclidean, depending on camera knowledge). Conversely, an *unsolvable* viewing graph is one for which multiple transformations exist that can be applied to the cameras without changing the fundamental matrices (or essential matrices in the calibrated case). The term "multiple" means any number of solutions – finite or infinite – strictly greater than one. An unsolvable graph is undesirable in practice as it represents an inherently ill-posed problem, hence it is important to test solvability before running a structure-from-motion method [15, 8]. When a graph turns out to be non-solvable, the challenge is to identify its largest solvable component, but this is (at least) as difficult as testing.

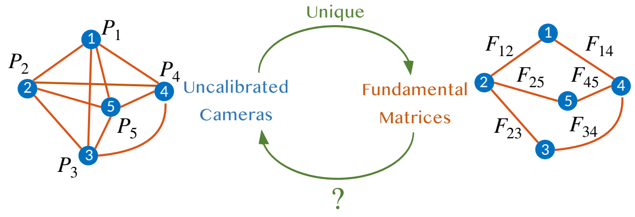

Figure 1. While a set of cameras uniquely determines the fundamental matrices, the opposite may not hold. Fundamental matrices can be conveniently represented as a viewing graph. The knowledge of the cameras is equivalent to a complete graph [10].

Related Work. Previous studies have demonstrated that, in the calibrated case, solvable graphs are those that are parallel rigid [12, 1]. Effective methods for extracting maximal rigid components are available for non-solvable graphs [9, 18]. Hence the calibrated case is considered *solved*.

The uncalibrated case, instead, is less explored and still offers open issues. Early works focused on manually analyzing small graphs [10] or providing sufficient conditions for solvability [14, 16]. Trager et al. [17] showed that – in principle – any viewing graph can be classified into solvable or not solvable via a system of *polynomial equations*, although this is computationally expensive. Arrigoni et al. [2, 3] proposed a novel set of polynomial equations based on cycle consistency, which decided the solvability of minimal graphs up to 90 nodes and non-minimal (dense) graphs with 9 nodes: this sets the state of the art in the uncalibrated case in terms of solvable problem size, but it is still far from the size of typical real datasets (see Tab. 3). Larger graphs have not been processed in [3]: actually, solving polynomial equations requires Gröbner basis computation, whose worst-case complexity is doubly exponential in the number of variables [5]. In addition, the task of computing maximal solvable components has never been addressed so far.

The authors of [17] also introduced the concept of *finite solvability*, a viewing graph that determines a *finite* num-

Table 1. Differences between solvability and finite solvability.

	Definition	Test
Solvability [17, 3]	unique solution	#solutions of polynomial system
Finite Solvability [17]	finite #solutions	rank of linear system

ber of projective configurations of cameras (see Tab. 1 and Fig. 2). Clearly, solvability (i.e., unique solution) implies finite solvability. However, the converse does not hold: examples of graphs corresponding to two solutions, meaning that they are finite solvable but not solvable, are reported in [3]. Finite solvability can be established as a consequence of solvability testing, namely by counting the number of solutions retrieved, as done in [3]. However, this is prohibitive for practical scenarios: even the smallest graphs from our real experiments (see Tab. 3) cannot be decided by [3]. An alternative (and efficient) approach is followed in [17], which derived *linear equations* that can be used to check finite solvability: this is the only available characterization of finite solvability, which constitutes the starting point of our work.

Contribution. In this paper we focus on the computational problem of deciding *finite solvability* in the uncalibrated case. Although this is not equivalent to the sought notion of solvability (see Tab. 1), it is a valuable property that presents two main advantages:

- the cases of graph that are finite solvable but not solvable are really rare (only ten examples were found in [3]), hence – *in practice* – finite solvability is a **very good approximation** of solvability;
- solvability has mainly a theoretical relevance, for its computational complexity is prohibitive. Finite solvability, instead, reduces to testing the rank of a matrix, hence it is **applicable in practice**.

We present a computationally feasible method for checking finite solvability of a viewing graph: with respect to [17] we **significantly reduce** (up to $400\times$ on large real datasets) the number of equations.

Moreover, inspired by the algorithms proposed in [9, 18] for calibrated cameras, we develop the **first procedure** for identifying the maximal finite-solvable components of an unsolvable viewing graph with uncalibrated cameras. This is accomplished by analyzing the null-space of the matrix defining finite solvability equations.

Our contributions result in a practical method that can be applied to large/dense viewing graphs from popular structure-from-motion datasets, as demonstrated by our experiments. We significantly outperform [17] in terms of execution times, and extend the size of manageable graphs to the **unprecedented** value of 300K edges.

The paper is organized as follows: Sec. 2 reviews relevant background while Sec. 3 presents our contributions and the derived method; experiments are presented in Sec. 4 and the conclusion is given in Sec. 5.

2. Background

In this section, we review the concepts of solvability and finite solvability, with the latter being the main point of focus of this paper. For more details see [17].

2.1. Solvability and Finite Solvability

Let $P_1, \dots, P_n$ denote n uncalibrated cameras, represented by matrices in $\mathbb{R}^{3\times4}$ of rank 3. The center of camera P_i has coordinates given by $\mathbf{c}_i \in \mathbb{R}^4$, a non-zero element of the kernel of P_i. We represent projective variables using non-homogeneous coordinates with the introduction of proper tricks for the scale ambiguity[1], as in [17]. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an undirected graph with vertex set $\mathcal{V} = \{1, \dots, n\}$ and edge set $\mathcal{E} \subseteq \{1, \dots, n\} \times \{1, \dots, n\}$ representing a *viewing graph* of an uncalibrated structure from motion problem. We denote the number of edges with m and the fundamental matrix of $(i, j) \in \mathcal{E}$ with F_{ij}.

The key question is the following: **given a graph with fundamental matrices on the edges, how many camera configurations exist yielding such fundamental matrices?** Several cases are possible:

1. there exists a *unique* camera configuration;
2. there exist a *finite number* of camera configurations;
3. there exist *infinitely many* camera configurations.

Clearly, uniqueness is intended up to a single projective transformation[2]. Such cases give rise to different notions of solvability, that will be introduced next. More formally, it is possible to define solvability as a property of the pair $(\mathcal{G}, \mathcal{P})$ where $\mathcal{P}$ is a set of cameras and $\mathcal{G}$ a graph that represents the connectivity pattern between the cameras.

Definition 1 ([17])**.** Let $\mathcal{P} = \{P_1, \dots, P_n\}$ be a configuration of cameras and $\mathcal{G}$ be a graph. The pair $(\mathcal{G}, \mathcal{P})$ is called *solvable* if there exists a *unique* (up to a single projective transformation) camera configuration yielding the same fundamental matrices as $\mathcal{P}$.

It can be proved that solvability actually depends on the structure of the *graph* and on the *camera centres* only [17]. In other terms, the remaining parts of the 3×4 camera matrices are not involved. A generic configuration is typically considered to eliminate the reliance on cameras, resulting in

[1]In this paper, we use uppercase letters to denote matrices, lowercase bold letters for vectors, and lowercase letters for scalars.

[2]We alway assume that there exists at least one solution, as happens in a noiseless case. This is a standard practice, followed also by previous work [17, 3]. Solvability in a noisy case has never been considered so far.

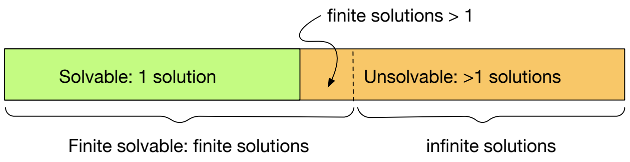

Figure 2. Relations between different notions of solvability.

another concept of solvability, that is a property of a graph by itself. In practice, generic configurations are obtained by sampling at random the coordinates of the centres.

Definition 2 ([17]). A graph $\mathcal{G}$ is called *solvable* if it is solvable for a *generic* configuration of cameras. Otherwise it is called *non solvable*.

Solvability, as defined above, is tantamount to a unique solution. A relaxed notion is that of *finite solvability*, which only requires a finite number of solutions. Note that solvability implies finite solvability, but the converse is not true (see [3] for some counterexamples). Observe also that a non-solvable graph can be either finite solvable or entail infinitely many configurations of cameras. The relations between these concepts are summarized in Fig. 2.

Determining the solvability of a graph requires solving a polynomial system of equations [17, 3]. This is highly demanding from the computational point of view, hence prohibitive for large/dense graphs that appear in practice. The solution set to such polynomial equations is an algebraic variety and, in particular, a smooth algebraic group, as observed in [17]. This property implies that the dimension of such a variety coincides with the dimension of the tangent space at the identity, which can be easily computed. Specifically, since the tangent space is a linear space, computing its dimension reduces to determining the rank of a linear system of equations. Such a dimension reveals if the original polynomial system admits a finite number of solutions or, in other terms, if the graph is finite solvable. However, the exact number of solutions can not be determined in this way, or, in other terms, we can not distinguish between a solvable case (unique solution) and a non-solvable one.

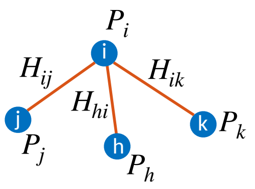

Figure 3. Edges in the viewing graph with a common vertex. Each edge is associated with an unknown 4×4 transformation.

2.2. Linear Equations for Finite Solvability

We now briefly review the linear equations for finite solvability derived in [17], which constitute the starting point of our developments. Building blocks are a collection of matrices $L_1, \dots, L_n$ of dimension 20×16, where L_i depends only on the camera centre $\mathbf{c}_i = [c_1\ c_2\ c_3\ c_4]^\mathsf{T}$:

$$L_i = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\mathbf{c}_4 & 0 & 0 & 0 & \mathbf{c}_3 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\mathbf{c}_4 & 0 & 0 & 0 & \mathbf{c}_3 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -\mathbf{c}_4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{c}_2 & 0 \\
0 & 0 & 0 & 0 & -\mathbf{c}_4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{c}_2 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -\mathbf{c}_3 & 0 & 0 & 0 & \mathbf{c}_2 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -\mathbf{c}_3 & 0 & 0 & 0 & \mathbf{c}_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -\mathbf{c}_4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{c}_1 & 0 \\
0 & -\mathbf{c}_4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{c}_1 & 0 & 0 \\
0 & 0 & 0 & -\mathbf{c}_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{c}_1 & 0 & 0 & 0 & 0 \\
0 & -\mathbf{c}_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{c}_1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -\mathbf{c}_2 & 0 & 0 & 0 & \mathbf{c}_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -\mathbf{c}_2 & 0 & 0 & 0 & \mathbf{c}_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -\mathbf{c}_3 & \mathbf{c}_4 & 0 & 0 & \mathbf{c}_2 & 0 & 0 & 0 & 0 & -\mathbf{c}_2 \\
0 & 0 & 0 & 0 & 0 & -\mathbf{c}_3 & 0 & 0 & 0 & \mathbf{c}_2 & 0 & -\mathbf{c}_4 & 0 & 0 & 0 & \mathbf{c}_3 \\
-\mathbf{c}_4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{c}_4 & 0 & \mathbf{c}_1 & 0 & -\mathbf{c}_3 & 0 \\
0 & 0 & -\mathbf{c}_3 & \mathbf{c}_4 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{c}_1 & 0 & 0 & 0 & 0 & -\mathbf{c}_1 \\
-\mathbf{c}_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{c}_1 & 0 & 0 & -\mathbf{c}_4 & 0 & 0 & 0 & \mathbf{c}_3 \\
0 & 0 & 0 & 0 & 0 & -\mathbf{c}_4 & 0 & 0 & 0 & 0 & \mathbf{c}_4 & 0 & 0 & \mathbf{c}_2 & -\mathbf{c}_3 & 0 \\
0 & -\mathbf{c}_2 & 0 & \mathbf{c}_4 & 0 & \mathbf{c}_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\mathbf{c}_1 \\
-\mathbf{c}_2 & 0 & 0 & 0 & \mathbf{c}_1 & 0 & 0 & -\mathbf{c}_4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{c}_2
\end{bmatrix}.$$

Recall that camera centers are typically sampled at random to comply with the assumption of generic configurations, hence $L_1, \dots, L_n \in \mathbb{R}^{20\times 16}$ are *known* matrices. For each edge in the graph $(i,j) \in \mathcal{E}$ let us consider an *unknown* projective matrix (not necessarily invertible) denoted as $H_{ij} \in \mathbb{R}^{4\times 4}$. Each pair of adjacent edges $(h,i) \in \mathcal{E}$ and $(i,j) \in \mathcal{E}$ – see Fig. 3 – gives rise to a linear equation:

$$L_i \operatorname{vec}(H_{hi}) - L_i \operatorname{vec}(H_{ij}) = \mathbf{0} \tag{1}$$

where vec denotes the vectorization that turns a matrix into a vector. For more details about the mathematical derivation of Eq. (1), please see the supplementary material of [17]. Hence, the equations coming from *all* adjacent edges in the graph can be collected in a homogeneous linear system:

$$S\mathbf{x} = \mathbf{0} \tag{2}$$

where $\mathbf{x} = [\operatorname{vec}(H_{12})^\mathsf{T} \dots \operatorname{vec}(H_{hi})^\mathsf{T} \dots \operatorname{vec}(H_{ij})^\mathsf{T} \dots]^\mathsf{T}$ is the vertical concatenation of all the unknowns. Accordingly, the number of columns in S is equal to $16m$ where m is the number of edges in the graph.

Proposition 1 ([17]). *Let $\mathcal{G}$ be a graph with m edges and let $\mathbf{c}_1, \dots, \mathbf{c}_n \in \mathbb{R}^4$ be n generic camera centres. $\mathcal{G}$ is finite solvable if and only if:*

$$\dim(\operatorname{null}(S)) = 15 + m. \tag{3}$$

The right term in Eq. (3), namely $15 + m$, refers to the fact that each H_{ij} matrix is defined up to scale, and there is a global projective ambiguity inherent to the problem, meaning that a single transformation can be arbitrarily fixed. This suggests that additional equations can be included in order to remove all the ambiguities:

- One matrix can be arbitrarily fixed, e.g., $H_{12} = I_4$ with I_4 being the 4×4 identity matrix.
- The scale of the remaining H_{ij} can be arbitrarily set, e.g., by fixing the sum of its entries to 1. This results in a linear equation for each edge (except the one used to fix the global transformation): $\mathbf{1}_{16}^{\mathsf{T}} \operatorname{vec}(H_{ij}) = 1$, where $\mathbf{1}_{16}$ denotes a vector of ones of length 16.

We denote by $\bar{S}\mathbf{x} = \mathbf{b}$ the resulting (non-homogeneous) linear system obtained by appending the above additional equations to Eq. (2). $\bar{S}$ is called the *solvability matrix.*

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. $\mathcal{G}$ is finite solvable if and only if:*

$$\dim(\operatorname{null}(\bar{S})) = 0. \tag{4}$$

Hence finite solvability can be checked as follows: starting from a graph $\mathcal{G}$, a set of camera centres $\mathbf{c}_1, \dots, \mathbf{c}_n \in \mathbb{R}^4$ is sampled at random and the solvability matrix $\bar{S}$ is constructed according to Eq. (1), with the provision that the additional equations for ambiguity fixing should be included as well; if $\dim(\operatorname{null}(\bar{S})) = 0$ then the graph is finite solvable, otherwise it is non solvable.

3. Practical Finite Solvability

We first clarify the limitations of the previous approach for finite solvability [17], thus motivating the need of a more scalable method (Sec. 3.1). Accordingly, in Sec. 3.2 we introduce a new *reduced* system of equations. Then, we show how to extract maximal components in the case where a graph is not finite solvable (Sec. 3.3). Finally, in Sec. 3.4 we report some implementation details.

3.1. Drawback of Previous Approach

Previous formulation [17] considers an equation of the form (1) for each pair of adjacent edges, as reviewed in Sec. 2.2. In other terms, for a fixed vertex, **all possible combinations** of pairs of adjacent edges are considered. If d_i denotes the degree of vertex i, that is the number of incident edges, then the number of such combinations is

$$d_i(d_i - 1)/2. \tag{5}$$

Hence, the number of equations produced by a single node is **quadratic** in the degree of the node. For instance, a node with degree 100 will contribute with 4950 equations. This is a severe limitation, as managing equations coming from all the nodes in a graph can be prohibitive in practice. See Fig. 7 for typical degrees of real datasets.

3.2. Reduced Formulation

We show here how to substantially reduce the number of equations, thus alleviating the aforementioned drawback. Let us start with a simple observation related to the building block of the solvability matrix.

Lemma 1. *For any generic centre $\mathbf{c}_i \in \mathbb{R}^4$, the 20×16 matrix L_i has only 11 independent rows. In particular, the following 11×16 sub-matrix of L_i, obtained by selecting 11 rows out of 20:*

$$K_i = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\mathbf{c}_4 & 0 & 0 & 0 & \mathbf{c}_3 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\mathbf{c}_4 & 0 & 0 & 0 & \mathbf{c}_3 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -\mathbf{c}_4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{c}_2 & 0 \\
0 & 0 & 0 & 0 & -\mathbf{c}_4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{c}_2 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -\mathbf{c}_3 & 0 & 0 & 0 & \mathbf{c}_2 & 0 & 0 & 0 & 0 \\
0 & 0 & -\mathbf{c}_4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{c}_1 & 0 \\
0 & -\mathbf{c}_4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{c}_1 & 0 & 0 \\
0 & 0 & 0 & -\mathbf{c}_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{c}_1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -\mathbf{c}_3 & \mathbf{c}_4 & 0 & 0 & \mathbf{c}_2 & 0 & 0 & 0 & 0 & -\mathbf{c}_2 \\
0 & 0 & 0 & 0 & 0 & -\mathbf{c}_3 & 0 & 0 & 0 & \mathbf{c}_2 & 0 & -\mathbf{c}_4 & 0 & 0 & 0 & \mathbf{c}_3 \\
-\mathbf{c}_4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{c}_4 & 0 & \mathbf{c}_1 & 0 & -\mathbf{c}_3 & 0
\end{bmatrix}$$

has full rank.

Proof. First of all one can show that the rank of K_i is 11 by symbolic computation (e.g., in MATLAB). Then, again by symbolic computation, it can be proved that any other row of L_i is a linear combination of the rows of K_i. □

Thanks to Lemma 1, we can replace L_i with K_i in Eq. (1), obtaining:

$$K_i \operatorname{vec}(H_{hi}) - K_i \operatorname{vec}(H_{ij}) = \mathbf{0}. \tag{6}$$

This corresponds to **reducing by 45%** the number of rows in the solvability matrix with respect to [17]. More importantly, such a gain is not optimal as the number of equations involved can be further reduced: the following result proves that, for a given vertex i, we do not need to consider all possible pairs of adjacent edges, but only a minimal number of $d_i - 1$ equations. In other terms, the number of equations produced by a single node becomes **linear** in the degree of the node (whereas in [17] it was quadratic, see Eq. (5)).

Proposition 2. *Let $\mathcal{G}$ be a graph and $\mathbf{c}_1, \dots, \mathbf{c}_n \in \mathbb{R}^4$ be n generic camera centres. For each vertex i, a minimal set of $d_i - 1$ equations of the form (6) is chosen by selecting one edge as reference, and pairing it with each of the remaining edges that are incident to node i. Let us collect equations coming from all nodes in the graph in a unique homogeneous linear system:*

$$R\mathbf{x} = \mathbf{0} \tag{7}$$

where $\mathbf{x} = [\operatorname{vec}(H_{12})^{\mathsf{T}} \dots \operatorname{vec}(H_{hi})^{\mathsf{T}} \dots \operatorname{vec}(H_{ij})^{\mathsf{T}} \dots]^{\mathsf{T}}$ is the vertical concatenation of all the unknowns, as before. Then, Eq. (7) is equivalent to (2).

Proof. We restrict our attention to a single vertex i and its associated set of incident edges (see Fig. 3). In particular we consider three edges (i, j), (i, h) and (i, k) where (i, j) is set as reference, and assume that equations of the form (6) holds true for the pairs (i, j)-(i, h) and (i, j)-(i, k), namely:

$$\begin{aligned} K_i \operatorname{vec}(H_{ih}) - K_i \operatorname{vec}(H_{ij}) &= \mathbf{0} \\ K_i \operatorname{vec}(H_{ij}) - K_i \operatorname{vec}(H_{ik}) &= \mathbf{0}. \end{aligned} \tag{8}$$

Then

$$K_i\,\mathrm{vec}(H_{ih}) - K_i\,\mathrm{vec}(H_{ik}) =$$
$$\underbrace{K_i\,\mathrm{vec}(H_{ih}) - K_i\,\mathrm{vec}(H_{ij})}_{\mathbf{0}} + \underbrace{K_i\,\mathrm{vec}(H_{ij}) - K_i\,\mathrm{vec}(H_{ik})}_{\mathbf{0}} \tag{9}$$

hence equations associated with the pair (i,h)-(i,k) are dependent from equations associated with the other pairs of edges. A key fact for proving this result is that all the involved edges share a common vertex i, hence all the equations share the same building block K_i. Thus, equations from (2) are redundant and depend on those from (7): this allows to use only a subset of equations. □

As done before, we append to (7) the $15 + m$ equations aimed at fixing all the ambiguities and obtain a new non-homogeneous linear system $\bar{R}\mathbf{x} = \mathbf{b}$, where $\bar{R}$ is called the *reduced solvability matrix*. Combining Prop. 1-2 we get:

Corollary 2. *Let $\mathcal{G}$ be a graph and $\mathbf{c}_1, \dots, \mathbf{c}_n \in \mathbb{R}^4$ be n generic camera centres. $\mathcal{G}$ is finite solvable if and only if:*

$$\dim(\mathrm{null}(\bar{R})) = 0. \tag{10}$$

To summarize, we have shown that we can substantially reduce the number of equations needed for checking finite solvability, resulting in a reduced solvability matrix. Table 2 summarizes our key contributions and the main differences with respect to [17]. See also Tab. 3 for the number of equations employed by the two formulations on real datasets.

Table 2. Differences between [17] and our formulation.

	Trager et al. [17]	Our formulation
Building block	20×16 L_i	11×16 K_i
#Equations for vertex i	$d_i(d_i-1)/2$	$d_i - 1$

3.3. Finite Solvable Components

Previous section has derived a reduced solvability matrix whose null-space reveals whether the input graph is finite solvable or not. In the latter case the following relevant question arises: **Can we restrict our attention to the largest sub-graph that is finite solvable?** Actually, the answer is yes, as will be shown next.

More formally, a *finite-solvable component* is a subgraph $\mathcal{G}' = (\mathcal{V}', \mathcal{E}')$ such that $\mathcal{V}' \subseteq \mathcal{V}$, $\mathcal{E}' \subseteq \mathcal{E}$ and $\mathcal{G}'$ is finite-solvable. It is called *maximal* if it is not a subset of any other component. The following result states that each edge is included in *exactly one* maximal finite-solvable component. A similar result holds for the calibrated case [9].

Proposition 3. *Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a graph. Then the set of edges of all maximal finite-solvable components induces a partition of the original edge set $\mathcal{E}$.*

Proof. A single edge is solvable (hence finite solvable) since the fundamental matrix uniquely determines a pair of cameras (up to a projective transformation) [7]. In other terms, each edge must be in *at least* one maximal finite-solvable component. In order to show that an edge belongs to *at most* one maximal finite-solvable component, we suppose by contradiction that an edge (i,j) is in two components: since they are both maximal, it is not possible that one component is contained in the other one; since each component is finite solvable, the system obtained after fixing the global ambiguity ($\bar{R}\mathbf{x} = \mathbf{b}$) has a unique solution within each component (see Corollary 2), therefore we end up with two different solutions for the transformation H_{ij} associated with the common edge, which contradicts the uniqueness of the solution. □

Hence the task is to partition the edges of a (non-solvable) viewing graph into maximal finite-solvable components. This section presents the first algorithm that accomplishes such a task[3]. Since we are considering a non-solvable case here, the condition in Eq. (10) is violated, meaning that the reduced solvability matrix $\bar{R}$ has a **non-trivial null-space**. Let d denote the (unknown) dimension of such a null-space and let N be a $16m \times d$ matrix containing a basis for the null-space as columns. N has a particular structure that reveals the membership of each edge to a specific component, as demonstrated by the following result.

Proposition 4. *Let $\mathcal{G}$ be a graph, let $\mathbf{c}_1, \dots, \mathbf{c}_n \in \mathbb{R}^4$ be n generic camera centres and let N denote a basis for null-space of the reduced solvability matrix $\bar{R}$. Then:*

1. *the rows of N corresponding to the edge used to fix the global ambiguity are zero;*
2. *two edges are in the same component $\iff$ the corresponding rows in N are equal.*

Proof. Since we are dealing with a non-solvable case, the system $\bar{R}\mathbf{x} = \mathbf{b}$ (that defines finite solvability) has multiple solutions. In particular, all solutions are of the form:

$$\mathbf{x} = \tilde{\mathbf{x}} + N\mathbf{w} \tag{11}$$

where $\tilde{\mathbf{x}}$ denotes a particular solution satisfying $\bar{R}\tilde{\mathbf{x}} = \mathbf{b}$, and $N\mathbf{w}$ represents a linear combination of the columns of N for any $\mathbf{w} \in \mathbb{R}^d$. Recall that $\bar{R}N = 0$ and $\mathbf{x} = [\mathrm{vec}(H_{ij})^\mathsf{T} \dots \mathrm{vec}(H_{lk})^\mathsf{T}]^\mathsf{T}$. It is convenient to express Eq. (11) edge by edge:

$$\begin{bmatrix} \mathrm{vec}(H_{ij}) \\ \dots \\ \mathrm{vec}(H_{lk}) \end{bmatrix} = \begin{bmatrix} \mathrm{vec}(\tilde{H}_{ij}) \\ \dots \\ \mathrm{vec}(\tilde{H}_{lk}) \end{bmatrix} + \begin{bmatrix} N_{ij}\mathbf{w} \\ \dots \\ N_{lk}\mathbf{w} \end{bmatrix} \tag{12}$$

[3] For simplicity of exposition, hereafter we will sometimes drop the term "maximal" when referring to a component, for it is taken for granted.

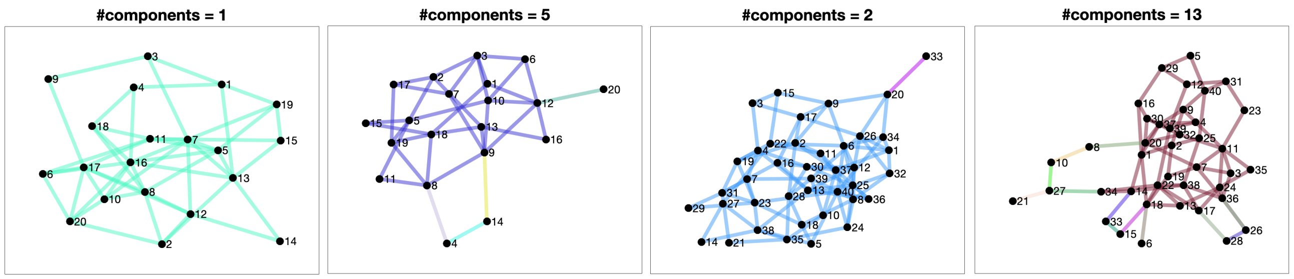

Figure 4. Maximal finite-solvable components extracted with our algorithm on synthetic viewing graphs. Each component is color-coded. Note that the cases with #components>1 resemble typical topologies of unsolvable graphs (see [17, 3]).

where N has been partitioned into blocks of size $16 \times d$. Indeed, observe that a given edge does not correspond to a single row in N but it corresponds to 16 rows in N: in general, these 16 rows will not be the same but they will be equal (one by one) to other 16 rows of all other edges in the same component (meaning that the first row will be equal to the first, the second row to the second, and so on), as will be shown now. **1)** Suppose that edge (i, j) is the one used to fix the ambiguity and let us consider edge (l, k). The fact that all the ambiguities have been fixed in this component, it is equivalent to say that there is no freedom in the possible solutions. In other terms: edge (l, k) lies in the same component as edge $(i, j) \iff N_{lk}\mathbf{w} = 0$ for all $\mathbf{w} \in \mathbb{R}^d \iff N_{lk} = 0$. **2)** Let us consider two edges (i, j) and (l, k) and assume that they do not belong to the same component as the edge used to fix the global projective ambiguity, otherwise see the previous point. This means that the global ambiguity has *not* been fixed but there are degrees of freedom corresponding to a global translation of the solution. In other terms: edges (i, j) and (l, k) are in the same component $\iff \text{vec}(\widetilde{H}_{ij}) = \text{vec}(\widetilde{H}_{ij}) + \mathbf{q}$ and $\text{vec}(\widetilde{H}_{lk}) = \text{vec}(\widetilde{H}_{lk}) + \mathbf{q} \iff N_{ij}\mathbf{w} = \mathbf{q}$ and $N_{lk}\mathbf{w} = \mathbf{q}$ for all $\mathbf{w} \in \mathbb{R}^d \iff N_{ij} = N_{lk}$. $\square$

According to the above result, we can identify maximal finite-solvable components by grouping together those edges with equal rows in the null-space of the reduced solvability matrix. Some examples are given in Fig. 4.

3.4. Algorithm

Our method is summarized in Fig. 5. Testing finite-solvability requires to determine whether $\dim(\text{null}(\bar{R})) = 0$ or not (Corollary 2). This is accomplished numerically by computing the least eigenvalue of $\bar{R}^T\bar{R}$ (i.e., the smallest singular value of $\bar{R}$) with the Matlab function `eigs` and comparing it with a small threshold, e.g. $1e{-}10$. Although such a threshold might be a critical choice, in our experiments we always found a significant gap in the singular values of $\bar{R}$ when the rank drops (see Fig. 6).

Computing the finite-solvable components requires extracting a basis for the null space of $\bar{R}$ (Prop. 4). This is

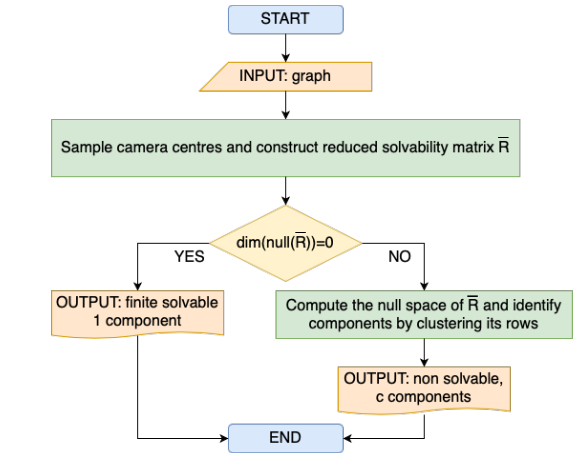
START
INPUT: graph
Sample camera centres and construct reduced solvability matrix R̄
dim(null(R̄))=0
YES
NO
OUTPUT: finite solvable 1 component
Compute the null space of R̄ and identify components by clustering its rows
OUTPUT: non solvable, c components
END

Figure 5. Proposed algorithm for testing finite solvability and extracting maximal finite-solvable components.

not a straightforward task from the numerical point of view, as the matrix may have arbitrarily small non-zero singular values. The full singular value decomposition (SVD) is generally the most reliable choice, but it is computationally prohibitive for large matrices. Cheaper (but less reliable) alternatives to SVD are based on QR or LU rank-revealing factorizations. In particular, we found that those based on LU (such as [6]) perform best in our case. The authors of [6] propose an efficient and reliable method[4] for computing an orthonormal basis of the null space of a large sparse rectangular matrix (usually with more rows than columns), whose dimension is unknown a priori. This is particularly suitable for matrices with a small-dimensional null space. The main computational component is a sparse LU factorization with partial pivoting, which is significantly cheaper than the QR factorization customarily used for dense matrices.

After retrieving a basis N for $\text{null}(\bar{R})$ we are required to cluster blocks of 16 rows of N. One could represent these blocks as vectors of $\mathbb{R}^{16d}$. However, in our implementation we digest them as vectors of $\mathbb{R}^d$ by summing the absolute values of the rows. Since the number of clusters is not

[4]The implementation of this MATLAB function called `nulls` is available at http://www.cs.tau.ac.il/~stoledo/Tools/nulls.m

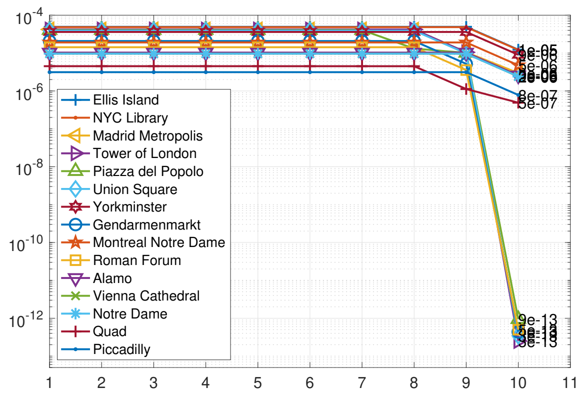

Figure 6. Smallest ten singular values of the reduced solvability matrix for real viewing graphs [19, 4]. Note the drop of the last singular value by several orders of magnitude for unsolvable graphs.

known beforehand, we implemented a simple greedy strategy that starts with the first row and groups together those that differ from it by less than a threshold. It then proceeds with the first of the remaining rows and continues in this manner. Every cluster corresponds to one finite-solvable component. Note that, in general, the number c of components is not equal to the dimension d of the null-space.

4. Experiments

We demonstrate the effectiveness of our method by applying it to several examples and showing that it can be profitably used to check the finite solvability of a viewing graph and, in non-solvable cases, extract the maximal finite-solvable components. Our algorithm was implemented in MATLAB and the code is publicly available[5]. Experiments were performed on a MacMini M1 (2020) with 16Gb RAM.

Both small-scale and large-scale viewing graphs from real structure-from-motion datasets are used. In particular, we consider the Cornell Arts Quad dataset [4], 14 sequences from the 1DSfM benchmark [19], and 14 sequences from [11]. For each sequence, we utilize only the associated graph, discarding other information (e.g., point correspondences and fundamental matrices) since we only need the graph topology to check finite solvability. In addition, we reduce our analysis to the largest biconnected component, since it is cheap to compute and being biconnected is a necessary condition for finite solvability [17]. Table 3 reports the features of each sequence, including the number of nodes, the number of edges and the percentage of edges (with respect to the complete graph). See also Fig. 7 for information on the degrees, which play a key role in determining complexity. We compare our approach to the method by Trager et al. [17], which is the only characterization of finite solvability available in the literature. Since the code is not available online, we used our implementation, which

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

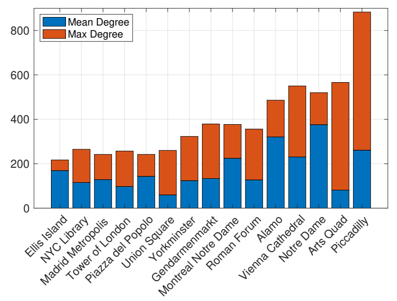

Figure 7. Average and max degree for real viewing graphs [19, 4]. For all the datasets, the min degree is equal to two.

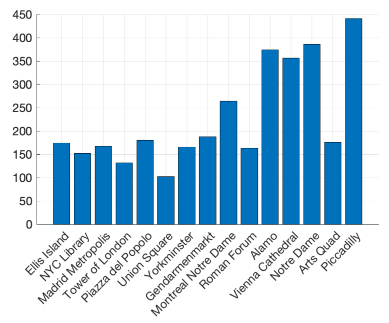

Figure 8. Ratio between the number of equations employed by Trager et al. [17] and those from our formulation, for the largest real viewing graphs [19, 4]. On "Piccadilly" the improvement given by our method even surpasses $400\times$.

shares the same routines and tricks as our method: the two pieces of code construct the solvability matrix differently, but use the same testing procedure. Results are given in Tab. 3, which reports the output of finite-solvability testing, the execution times and the number of equations for the competing methods. The time spent in building the solvability matrix dominates the testing time, but this can be improved by a more careful use of MATLAB indexing for sparse matrices, that we plan to investigate in the future.

Results show that there are only 5 graphs that are not finite-solvable, hence we conjecture that unsolvability is a rare situation in practice. The method by Trager et al. [17] can manage only the ten smallest sequences but fails to handle larger cases. Our algorithm, instead, successfully manages all the considered datasets, including the largest one (Piccadilly) with more than 300K edges. Our method is significantly faster than [17]: for example, the latter runs in about 27 minutes on "Tsar Nikolai I" whereas ours takes less than 2 seconds. Table 3 also reports the number of equations employed by the two formulations: the reduction given by our approach even surpasses 400x (Piccadilly), as can be appreciated from Fig. 8. For the unsolvable cases, we applied our algorithm for extracting maximal components. Our approach found 4 components for all such sequences: we verified that the largest one is indeed finite solvable and

Table 3. Results of our experiments on real datasets [19, 4, 11]. "T build" is the time spent building the solvability matrix (Trager et al. [17]) or the reduced solvability matrix (our method), "T test" is the time for finite solvability test and "T comp" represents the time required for computing the components (only on non-solvable cases). The ↑ symbol means that the computation either led "out of memory" error or crashed. Times are in seconds. Solvable in the table means finite-solvable. Datasets are ordered by the number of edges.

Dataset				Trager et al. [17]				Our Method					
Name	#nodes	density	#edges	#equations	T build	T test	Solvable	#equations	T build	T test	Solvable	T comp	#comp
Gustav Vasa	18	72 %	110	25920	0.24	0.29	YES	2222	0.02	0.07	YES	-	1
Dino 319	36	37 %	230	58200	0.58	0.47	YES	4664	0.02	0.06	YES	-	1
Dino 4983	36	37 %	231	58760	0.55	0.48	YES	4686	0.02	0.05	YES	-	1
Folke Filbyter	40	32 %	250	63840	0.67	0.66	YES	5060	0.02	0.05	YES	-	1
Jonas Ahls	40	41 %	321	98340	1.5	1.4	YES	6622	0.03	0.07	YES	-	1
Park Gate	34	94 %	529	324920	14	6.8	YES	11264	0.06	0.11	YES	-	1
Toronto University	77	33 %	974	500460	31	9.6	YES	20581	0.15	0.19	YES	-	1
Sphinx	70	55 %	1330	1034720	140	50	YES	28490	0.27	0.33	YES	-	1
Cherub	65	64 %	1332	1075800	160	70	YES	28589	0.29	0.29	YES	-	1
Tsar Nikolai I	98	52 %	2486	2512260	1032	579	YES	53614	0.87	0.57	YES	-	1
Skansen Kronan	131	88 %	7490	17528900	↑	-	-	163339	7.8	2.6	YES	-	1
Alcatraz Courtyard	133	92 %	8058	19596800	↑	-	-	175813	9.1	2.9	YES	-	1
Buddah Tooth	162	73 %	9546	23605780	↑	-	-	208230	12	3.6	YES	-	1
Pumpkin	195	65 %	12276	33703640	↑	-	-	267927	21	5.0	YES	-	1
Ellis Island	240	71 %	20290	77011780	↑	-	-	443740	64	12	YES	-	1
NYC Library	358	32 %	20662	68645240	↑	-	-	450626	64	10	YES	-	1
Madrid Metropolis	370	35 %	23755	86652300	↑	-	-	518540	86	13	YES	-	1
Tower of London	489	20 %	23844	68287460	↑	-	-	519189	86	12	NO	7.6	4
Piazza del Popolo	345	42 %	24701	97222440	↑	-	-	539627	89	16	NO	5.9	4
Union Square	853	7 %	25478	56296820	↑	-	-	551133	100	11	NO	15	4
Yorkminster	448	28 %	27719	100582600	↑	-	-	604890	112	17	YES	-	1
Gendarmenmarkt	722	18 %	48124	197518320	↑	-	-	1050786	361	38	NO	15	4
Montreal N. Dame	467	48 %	52417	303423420	↑	-	-	1148037	415	47	YES	-	1
Roman Forum	1102	12 %	70153	249322400	↑	-	-	1531244	799	62	NO	65	4
Alamo	606	53 %	97184	797160760	↑	-	-	2131382	2186	154	YES	-	1
Vienna Cathedral	898	26 %	103530	807533920	↑	-	-	2267782	2405	166	YES	-	1
Notre Dame	553	68 %	103932	879763460	↑	-	-	2280421	2465	171	YES	-	1
Arts Quad	5460	1 %	221929	867626110	↑	-	-	4942476	10568	442	YES	-	1
Piccadilly	2446	11 %	319195	3082312560	↑	-	-	6995384	24640	1430	YES	-	1

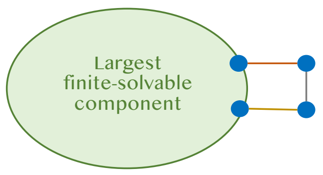

Figure 9. Simplified representation of the non-solvable graphs from Tab. 3: the largest finite-solvable components (green) contains all except three edges (explicitly drawn). Such edges have a "square" topology, which is a well-known non solvable case (see [17]); each edge forms a component consisting of just itself.

inspected the remaining ones, which all turned out to have the same structure depicted in Fig. 9, which is indeed a typical case of a non solvable graph. See also Fig. 4 for additional visualizations of synthetic graphs.

We also attempted to test the solvability (see Tab. 1) using the approach from [3] but the Grobner basis solver, implemented in Macaulay2[6], crashed after building the polynomial system even in the smallest dataset ("Gustav Vasa"). It is known that Grobner basis cannot cope with many unknowns, as it generates huge memory requirements (exponential in the number of variables [5]) to store intermediate results. This further demonstrates the practical importance of the notion of finite solvability, explored in this paper.

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

5. Conclusion

This paper has provided theoretical and practical insights into the problem of viewing graph solvability of Structure-from-Motion problems. We presented a new formulation of finite solvability, that comprises a significant smaller number of equations (up to $400\times$) with respect to previous work, resulting in a practical method for testing finite solvability. We also developed the first algorithm for identifying the maximal finite-solvable components of an unsolvable graph, therefore fulfilling an open issue in the uncalibrated case. Our approach – for the first time in the literature – can manage large/dense viewing graphs from real datasets (up to 300K edges) on an ordinary desktop computer.

Acknowledgements. This paper is supported by PNRR-PE-AI FAIR project funded by the NextGeneration EU program and by EU H2020 No. 871245 SPRING project.

References

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

[2] Federica Arrigoni, Andrea Fusiello, Elisa Ricci, and Tomas Pajdla. Viewing graph solvability via cycle consistency. In *Proceedings of the International Conference on Computer Vision*, 2021.

[3] Federica Arrigoni, Andrea Fusiello, Romeo Rizzi, Elisa Ricci, and Tomas Pajdla. Revisiting viewing graph solvability: an effective approach based on cycle consistency. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–14, 2022.

[4] David Crandall, Andrew Owens, Noah Snavely, and Daniel 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.

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

[6] Craig Gotsman and Sivan Toledo. On the computation of null spaces of sparse rectangular matrices. *SIAM Journal on Matrix Analysis and Applications*, 30(2):445–463, 2008.

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

[8] 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.

[9] Ryan Kennedy, Kostas Daniilidis, Oleg Naroditsky, and Camillo J. Taylor. Identifying maximal rigid components in bearing-based localization. In *Proceedings of the International Conference on Intelligent Robots and Systems*, pages 194 – 201, 2012.

[10] Noam Levi and Michael Werman. The viewing graph. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 518 – 522, 2003.

[11] 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.

[12] 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.

[13] Onur Ozyesil, Vladislav Voroninski, Ronen Basri, and Amit Singer. A survey of structure from motion. *Acta Numerica*, 26:305 – 364, 2017.

[14] Alessandro Rudi, Matia Pizzoli, and Fiora Pirri. Linear solvability in the viewing graph. In *Proceedings of the Asian Conference on Computer Vision*, pages 369–381, 2011.

[15] 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.

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

[17] Matthew Trager, Brian Osserman, and Jean Ponce. On the solvability of viewing graphs. In *Proceedings of the European Conference on Computer Vision*, pages 335–350, 2018.

[18] 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.

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