3D Surface Models by Geometric Constraints Propagation

M. Farenzena, A. Fusiello
Dip. Informatica, University of Verona
Strada Le Grazie 15, I-37134, Verona, Italy
farenzena@sci.univr.it

Abstract

This paper proposes a technique for estimating piecewise planar models of objects from their images and geometric constraints. First, assuming a bounded noise in the localization of 2D points, the position of the 3D point is estimated as a polyhedron containing all the possible solutions of the triangulation. Then, given the topological structure of the 3D points cloud, geometric relationships among facets, such as coplanarity, parallelism, orthogonality, and angle equality, are automatically detected. A subset of them that is sufficient to stabilize the 3D model estimation is selected with a flow-network based algorithm. Finally a feasible instance of the 3D model, i.e. one that satisfies the selected geometric relationships and whose 3D points lie within the associated polyhedral bounds, is computed by solving a Constraint Satisfaction Problem.

1. Introduction

The problem of recovering 3D surface models from images and geometric clues has been widely studied in literature. The proposed methods can be mainly classified as *model-based* and *constraint-based*.

In *model-based methods* [6, 13, 16, 20], the scene is defined as in CAD systems: objects are the assemblage of known primitive shapes. Reconstruction is carried out by fitting a 3D model to image data, thus determining its dimension, its position and orientation. The fact that the scene must be decomposable in primitive shapes is the main limitation of such methods.

Constraint-based methods [1, 3, 7, 8, 10, 21] are more flexible, as they do not rely on a-priori models but use simple primitives like points and lines. Geometric information, such as orthogonality, parallelism, or planarity, is given in the form of constraints on 3D points and reconstruction is obtained as the solution of an optimization process.

In most of previous works, e.g. [1, 8, 10, 20], the constraints detection phase requires the user to provide a geometrical description of the model, which can be very time-consuming. In other cases [3, 7], geometric constraints are detected automatically thanks to prior knowledge about the model to reconstruct.

Besides, the *analysis* of constraints is usually overlooked. In fact, datasets with many points and geometric constraints do not necessarily define a consistent and unique 3D object. Parts of the scene may not be rigidly connected, so that there exist various shapes that verify the geometric constraints and project to identical image points. In addition, constraints may be redundant, making the optimization uselessly harder or even unfeasible. [10] proposes an algebraic method to check whether a configuration of points and constraints leads to a unique reconstruction, but it does not deal with redundancies. As far as we know, a principled analysis of constraints has not been proposed yet.

Geometric constraints may be directly embedded into the minimization of the reprojection error (or bundle adjustment) [3, 7], but this causes a substantial increase of computational costs and both convergence and exact constraint satisfaction are not guaranteed. An alternative is to make the geometric constraints implicit in the parametrization of 3D points [1, 10, 21], so as they are satisfied exactly at every optimization step.

This paper follows the approach of [8] that avoids altogether the non-linear least-squares problem arising in the methods above. It casts the problem as a Constraint Satisfaction (CSP), where the 3D point positions are bounded by 3D boxes and a feasible solution is one that satisfies all selected geometric relationships and whose 3D points lie within the associated bounds. These bounds are there obtained by Interval Analysis (IA) applied to triangulation, which produces axis-aligned boxes that loosely overestimate the result. In this work, instead, the delimitation of the output of triangulation is tighter: each 3D point is bound by a polyhedron, that estimates the result exactly.

Moreover, this work is enriched by the automatic detection of constraints, provided manually in [8], and by the consequent analysis and pruning of these constraints. That permits to verify if a unique solution can be obtained, and at the same time to prune redundancies.

2. Overview of the approach

The approach, summarized in Fig. 1, consist of two stages.

The first stage deals with triangulation, i.e., reconstructing 3D points from their corresponding image points (provided manually in this paper) and known camera matrices (from a Structure-and-Motion pipeline). A statistical optimal solution, under the assumption of Gaussian noise, exists for two [11] and three views [19], but seems to be unfeasible beyond that. Our noise model, instead, is a uniform distribution inside a rectangular region centered around each image point. This enables us to compute the correct solution as a polyhedron that contains all the possible 3D point positions, for any number of views. This polyhedron can be regarded as representing the probability distribution function over the 3D point positions, which is uniform inside the polyhedron, and zero outside.

Evaluating uncertainty is crucial when the results are to be used as input for other processes. Albeit simple in concept, this polyhedral triangulation is a principled and efficient approach for evaluating 3D point positions and the associated uncertainty, and represents the counterpoise of the max-likelihood approach using the Gaussian noise model.

The second part of the paper focuses on obtaining a complete surface model. Given a set of reconstructed 3D points, represented by the polyhedra provided by the above 3D triangulation, and the connectivity of the points into triangular facets (provided manually), the method consists in a three-steps automatic process. First the geometric relationships, such as coplanarity, parallelism, etc. are detected; then a set of minimal relationships that allow a unique reconstruction is selected using the structural rigidity analysis; finally, a feasible instance of the 3D model, i.e. one that satisfies all selected geometric relationships and whose 3D points lie within the associated polyhedral bounds, is computed using a constrained optimization technique.

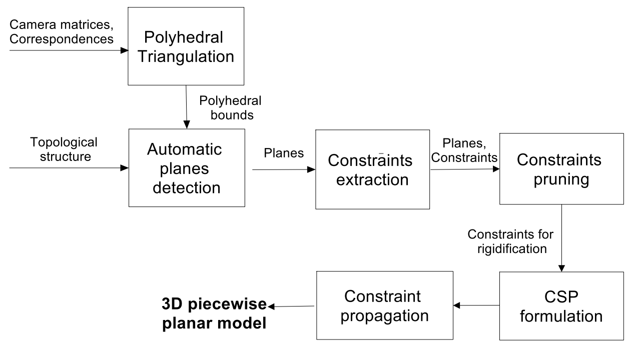


Figure 1. Overview of the proposed method. External inputs are: camera matrices, 2D point correspondences, 3D points connectivity.

3. Polyhedral triangulation

Once camera matrices are known, the first and most important stage of model reconstruction consists in recovering the coordinates of points in 3D space given their images in two or more views. It is usually assumed that the camera matrices are known exactly, or at least with greater accuracy than point localization. In the absence of noise, i.e. when correspondences are perfectly detected, the problem is trivial, involving only finding the intersection of rays in the space. If data are perturbed, however, the rays corresponding to back-projections of image points do not intersect, and obtaining the 3D coordinates of the reconstructed points becomes far from trivial, as witnessed by the renewed interest aroused by this issue [15, 19].

As in [8], the proposed method refrains from searching for *one* optimal solution and computes instead a *set* of possible solutions (defined in terms of errors affecting the image points) that contains the error-free solution. This permits to bound the exact solution in the 3D space for any number of views and to estimate, at the same time, the uncertainty of the result.

Let P^i, $i = 1, \dots, n$ be a sequence of n known cameras and m^i be the image of some unknown point M in 3D space, both expressed in homogeneous coordinates. It is assumed that the localization error is bounded by a rectangular region $\mathcal{B}_i$ centered around each image point (one can imagine a uniform distribution inside $\mathcal{B}_i$). Each region $\mathcal{B}_i$ bounds the possible locus of the 3D point inside a semi-infinite pyramid $\mathcal{Q}_i$ with its apex in the camera center (see Fig. 2). The solution set is defined as the polyhedron formed by the intersection of the n semi-infinite pyramids generated by the intervals $\mathcal{B}_1, \dots, \mathcal{B}_n$. Analytically, this region is defined as the following set:

$$\begin{aligned}\mathcal{D} &= \mathcal{Q}_1 \cap \mathcal{Q}_2 \dots \cap \mathcal{Q}_n = \\ &= \{\mathrm{M} \colon \exists \mathrm{m}^i \in \mathcal{B}_i, i = 1 \dots n \text{ s.t. } \forall i \colon \mathrm{m}^i \simeq P^i \mathrm{M}\}. \quad (1)\end{aligned}$$

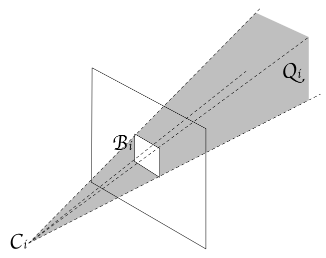


Figure 2. The semi-infinite pyramid $\mathcal{Q}_i$ is defined from the camera centre C_i and the bound $\mathcal{B}_i$.

Instead of approximating $\mathcal{D}$ using Interval Analysis as in [8], it is computed precisely using Computational Geometry techniques.

The semi-infinite pyramid $\mathcal{Q}_i$ can be written as the intersection of the four negative half-spaces $\mathcal{H}_1^i, \mathcal{H}_2^i, \mathcal{H}_3^i, \mathcal{H}_4^i$ defined by its supporting planes. Thus, the solution set D can be expressed as the intersection of $4n$ negative half-spaces:

$$\mathcal{D} = \bigcap_{\substack{i=1\ldots n \\ \ell=1\ldots 4}} \mathcal{H}_\ell^i \tag{2}$$

The vertices and the faces of $\mathcal{D}$ can be enumerated in $O(n \log n)$ time, being n the number of cameras [18].

As an example, Fig. 3 shows the polyhedral triangulation result obtained from seven calibrated images of a Lego object. Thirty-eight points are manually matched in the sequence and a uniform error in the 2D point location bounded by a 5-pixel wide square is assumed. The mean volume of the polyhedra is $(0.3cm)^3$, with respect to a volume of $(18.5cm)^3$ of the object.

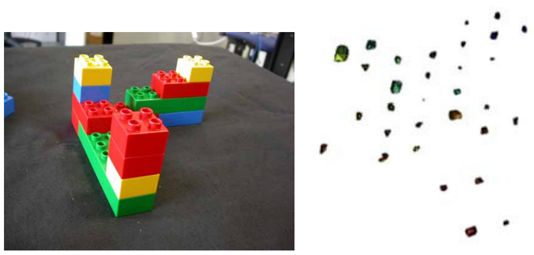


Figure 3. One of the seven images of the *Lego* sequence (left) and the polyhedral triangulation result (right). The small polyhedra bound the corners of the object.

4. Constraints detection

From the polyhedral triangulation a bounded estimation of the position of reconstructed 3D points is obtained. Any random choice of 3D points inside the bound is an approximation of the exact 3D reconstruction. Considering one of these approximations, its points are connected (manually) into a triangular mesh, obtaining a piecewise planar surface model. This section describes how geometric constraints such as coplanarity, parallelism, orthogonality and angles equality are automatically detected on the approximate model.

4.1. Planes detection

Planes in the model are extracted using a Mean Shift clustering procedure [4] on the triangular facets. The proposed technique is composed by a two-step, hierarchical strategy.

1. First facets are clustered according to their normal, thereby grouping together (approximately) coplanar and (approximately) parallel facets.
2. Then, within each group, the clustering is refined by taking into account also the distance to the origin of the plane containing the facet. In this way facets belonging to parallel planes are separated.

We adopted in both cases the uniform kernel, i.e., a multidimensional unite sphere, with bandwidth automatically selected as described in [5].

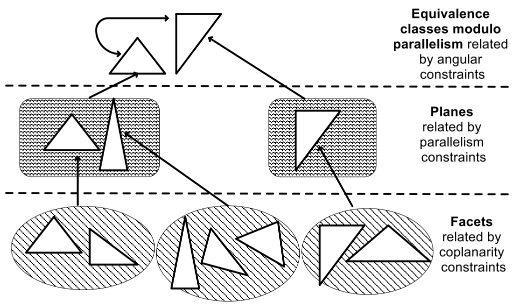
Please note that the process clusters together facets belonging to the same plane, regardless of their distance.

4.2. Constraints extraction

Geometric constraints involving planes can now be automatically inferred.

Facets belonging to the same group after the second clustering step are related by coplanarity constraints. Each plane is identified by one reference facet. As to parallelism constraints, if two different reference facets belong to the same group after the first clustering step, their respective planes are parallel. Moreover, angular constraints are deduced from *grouping heuristics*: if two or more planes nearly satisfy a constraint then they are forced to satisfy it. Orthogonality is checked for every pair of reference facets: whenever two of them are found to be approximately orthogonal (within 5 degrees), then they are linked by an orthogonality constraint. Likewise, equality of angles is checked for every quartet of reference facets.

The constraints form a hierarchy (Fig. 4): at the bottom level there are facets, grouped into planes by coplanarity constraints, then planes, grouped into equivalence classes modulo parallelism, and, finally, these equivalence classes related by angular constraints.


Equivalence classes modulo parallelism related by angular constraints
Planes related by parallelism constraints
Facets related by coplanarity constraints
Figure 4. The hierarchy induced by constraints detection.

At the highest level the position of the planes does not matter, as only the orientation is considered. This is consistent with the fact that the 3D points position is not determined by the polyhedral triangulation.

Carrying on with the *Lego* model, the 38 polyhedra are manually connected into triangular facets. Then, 15 planes are automatically extracted by the algorithm, as depicted in Fig. 5. This clustering process implies 12 parallelism constraints after the first clustering step (middle level of the hierarchy) and 53 coplanarity constraints after the second step (bottom level of the hierarchy). At the higher level of the constraints hierarchy three equivalence classes modulo parallelism are found, related be three orthogonality constraints.

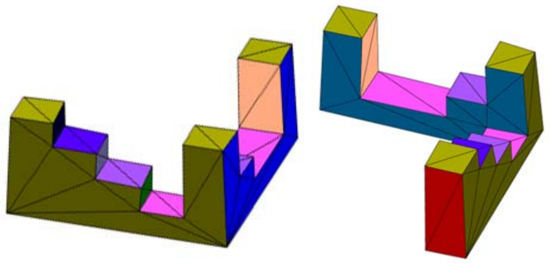


Figure 5. Automatic extraction of planes for the *Lego* model. Each plane is identified by a different colour.

5. Constraints Analysis

In this section we will discuss how the angular constraints, which – in general – are redundant, can be pruned while maintaining their capacity of stabilizing the estimation of the 3D model.

The concept of rigidity (or *constriction*) for geometric systems, has been studied in several scientific fields like Computational Geometry and Structural Topology, with application mainly in Computer-Aided Design (CAD). We are applying here the notion of *structural rigidity* to systems of planes (modulo parallelism) in order to remove redundant constraints while keeping the "rigidity" of the system. Some definitions, taken from [14], are in order here to introduce notation and concepts.

Definition 1 (Geometric Constraint System) *A Geometric Constraint System (GCS) is a pair $S = (O, C)$, where O is a set of geometric objects (represented by some variables), and C is a set of constraints.*

Our geometric objects are equivalence classes of planes modulo parallelism. They are identified by their direction (the normal vector). The constraints are orthogonality and angle equality.

Definition 2 *Let $S = (O, C)$ be a GCS. A solution to S is an evaluation θ_O of the variables in O such that every predicate in C is true. The set of solutions to S is denoted by $Sol(S)$.*

Definition 3 (Constriction) [1] *A GCS S is well-constrained if $Sol(S)$ is finite, over-constrained if $Sol(S) = \emptyset$ and under-constrained if $Sol(S)$ is infinite.*

[1]In fact, this is the definition of *global* [2] or *generic* [12] constriction.

In practice, a GCS can be under-constrained, but its solutions be identical modulo a geometric transformation (e.g., translation, rotation). Constriction modulo direct isometries (also called *rigidity*) is the type of constriction usually sought in CAD. In our case, translations are factorized out by the parallelism equivalence, hence only rotations are left. As a consequence, we consider constriction modulo orthogonal transformations.

Constriction depends on the number of solutions, but computing all of them is intractable. Hence, approximate characterizations that can be checked in polynomial time are frequently used. A characterization known as *structural constriction* is based on the *degrees of freedom* abstraction of the geometric constraints and objects.

Definition 4 *The number of degrees of freedom (DOFs) of a geometric object is the number of independent parameters used to represent it. The number of DOFs of a geometric constraint is the number of independent equations needed to represent it.*

In the following, we denote by $\mathrm{dof}(\cdot)$ the number of DOFs of an object or a constraint.

In our case, geometric objects have 2 DOF, because normals are unit vectors, and angle constraints have 1 DOF.

Definition 5 (Structural G-constriction) *Let $S = (O, C)$ be a GCS. Let G be an invariance group of dimension $\mathcal{D}$. The system S is structurally G-over-constrained if there exists a subsystem $S' = (O', C')$ of S such that $\sum_{x \in O'} \mathrm{dof}(x) - \sum_{c \in C'} \mathrm{dof}(c) < \mathcal{D}$.*

The system S is structurally G-well-constrained if it is not structurally G-over-constrained and $\sum_{x \in O} \mathrm{dof}(x) - \sum_{c \in C} \mathrm{dof}(c) = \mathcal{D}$.

The system S is structurally G-under-constrained if it is not structurally G-over-constrained and $\sum_{x \in O} \mathrm{dof}(x) - \sum_{c \in C'} \mathrm{dof}(c) > \mathcal{D}$.

In our case, structural constriction modulo orthogonal transformations can be checked using $\mathcal{D} = 3$.

Definition 6 (Constraint graph) *Let $S = (O, C)$ be a GCS. Its constraint graph, denoted by $G_S = (V, E)$, is a bipartite undirected graph where $V = O \cup C$ (every object in S and every constraint in C is a vertex in G_S) and an edge connects each constraint to each entity it constrains.*

Hoffmann et al. in [12] introduced the DENSE algorithm that checks structural constriction in polynomial time, considering a flow-network derived from the bipartite constraint graph. The source is linked to each constraint, and each object is linked to the sink. The capacities correspond to the DOFs of the constraints (edges from the source to constraints) and to the DOFs of the objects (edges from objects to the sink). Edges from constraints to objects have infinite capacity. A maximum flow in this network represents an optimal distribution of the constraints DOFs onto the objects DOFs. To identify over-rigid subsystems, the method adds an additional $\mathcal{D}$ capacity to one constraint at a time. If a maximum flow distribution cannot saturate all the edges from the sink to the constraints, this means that some constraints DOFs cannot be absorbed by the objects. Thus there exists a subsystem with less DOFs than $\mathcal{D}$, and the GCS is over-constrained (Fig. 6). Please note that, being structural constriction an abstraction, a GCS is deemed over-constrained as soon a redundant constraints are detected, regardless of the fact that they are consistent or not.

We exploit over-rigidity in order to detect redundancies. DENSE returns the over-constrained subsystem S' if the system is over-constrained, or an empty set otherwise. This S' is the subsystem induced by the objects traversed during the last search for an augmenting path, in the max flow computation. Constraints binding the objects in S' can be removed until the system itself becomes structurally well-constrained. This is implemented in the PRUNE procedure:

Algorithm 1 PRUNE

```
Input $S = (O, C)$: GCS
Output $S_o = (O, C_o)$: GCS such that $C_o \subseteq C$ and
$S_o$ is structurally well-constrained
    $S' = (O', C') \leftarrow$ DENSE(S)
    if isEmpty($S'$)
        Return $S$
    else
        select $c \in C'$
        $S \leftarrow S(O, C \backslash \{c\})$
        PRUNE(S)
    end
```

The selection of c is random, provided that its removal do not leave any object node with less inbounding arcs than the object's DOFs in the constraint graph. In that case the object would become under-constrained.

In the *Lego* model example, the constraints analysis correctly reports that the constraint system is already structurally rigid. Consequently, no pruning occurs.

6. Finding a feasible solution

Finally, a feasible instance of the 3D model, i.e. one that satisfies all selected geometric constraints and whose 3D points lie within the associated polyhedral bounds, is computed. This is formalized in the following Constraint Satisfaction Problem (CSP):

$$\begin{aligned} \text{find} \quad & \mathrm{X} \\ \text{subject to} \quad & \mathrm{X}_L \leq \mathrm{X} \leq \mathrm{X}_U \\ & \mathrm{c}_L \leq c(X) \leq \mathrm{c}_U \end{aligned} \qquad (3)$$

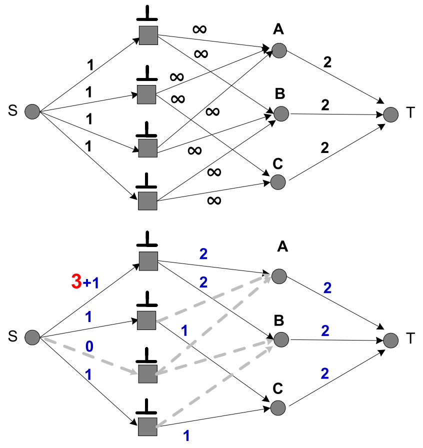


Figure 6. Example of constraint network (top) and an example of flow distribution, when the first constraint edge is overloaded (bottom). The GCS is over-constrained, because not all the arcs from the sink to the constraints are saturated. Removing the first or the third constraint makes the system structurally rigid. $\perp$ indicates the orthogonality constraint.

where X is the variables vector, i.e. the 3D points of the model; X_L and X_U delimit the domain of each variable, and they derive from polyhedral triangulation; $c(\mathrm{X})$ are algebraic equations containing the non linear constraints on X, with bounds c_L and c_U.

The geometric constraints must then be translated into constraints among points and formalized as algebraic equations.

For each equivalence class modulo parallelism a plane is chosen as the reference one. Angular constraints are applied among the reference planes. These are then linked to each other plane in the same equivalence class by a parallelism constraint. Each constraint among planes (i.e. parallelism and angular constraints) is translated into a constraint on the normals of the reference facets, as shown in Table 1. The normal vector, in turn, is a function of the three vertices of the facet, in Cartesian coordinates. In order to simplify the complexity of the algebraic equations, in orthogonality and parallelism constraints the normal vector obtained from the points is not normalized.

Constraint	*Algebraic equation*
Orthogonal(f_1, f_2)	$\mathrm{n}_1 \cdot \mathrm{n}_2 = 0$
SameAngle(f_1, f_2, f_3, f_4)	$(\mathrm{n}_1 \cdot \mathrm{n}_2) - (\mathrm{n}_3 \cdot \mathrm{n}_4) = 0$
Parallel(f_1, f_2)	$(\mathrm{n}_1 \times \mathrm{n}_2) = 0$

Table 1. Translation of constraints among facets $\{f_i\}_{i=1,\ldots,n}$ of the model into algebraic equations among points.

The reference facets are linked to all the other facets belonging to the same plane by coplanarity constraints. Let $C = \{\mathrm{M}_1, \mathrm{M}_2, \ldots, \mathrm{M}_n\}$ be the vertices of a group of coplanar facets, then all these points must lie on the same plane. This can be translated into a set of overlapping coplanarity constraints among four points at a time:

$$\begin{aligned}&\mathrm{Coplanar}(\mathrm{M}_1, \mathrm{M}_2, \mathrm{M}_3, \mathrm{M}_4)\wedge\\&\mathrm{Coplanar}(\mathrm{M}_2, \mathrm{M}_3, \mathrm{M}_4, \mathrm{M}_5)\wedge\dots\\&\mathrm{Coplanar}(\mathrm{M}_{n-3}, \mathrm{M}_{n-2}, \mathrm{M}_{n-1}, \mathrm{M}_n)\end{aligned}\qquad(4)$$

where $\mathrm{Coplanar}(\mathrm{M}_1, \mathrm{M}_2, \mathrm{M}_3, \mathrm{M}_4)$ is equivalent to:

$$[(\mathrm{M}_1 - \mathrm{M}_2) \times (\mathrm{M}_1 - \mathrm{M}_3)]^{\mathsf{T}} \cdot (\mathrm{M}_1 - \mathrm{M}_4) = 0. \qquad (5)$$

Once all constraints are translated into algebraic equations, a solution can be found using a constraint solver. In our case we use SNOPT [17], a general-purpose system for solving optimization problems involving many variables and constraints. It is suitable for large-scale linear and quadratic programming and for linearly constrained optimization, as well as for general nonlinear programs.

As to the *Lego* model, the formalization of the problem into algebraic equations yields 93 non linear constraints. The results are summarized in Tab. 2 and Fig. 7. As the reader can notice, the whole pipeline described throughout the paper leads to an accurate 3D model. The fact that the constraints are not exactly satisfied is due to the optimizer, that stops when it deems the solution cannot improved further.

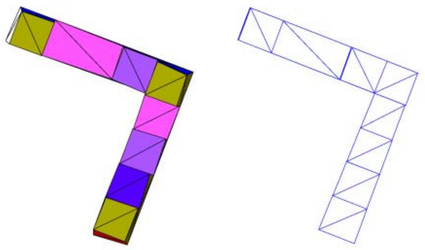


Figure 7. A top view of the *Lego* model before (left) and after (right) constraints propagation.

	$\perp$	$\square$	$\parallel$
Before	$[0.81, 2.67]^\circ$	$[0, 0.0001]^\circ$	$[0.05, 6.60]^\circ$
After	$[0.61, 2.62]^\circ$	$[0, 0]^\circ$	$[0.05, 4.13]^\circ$

Table 2. [Min,Max] deviation from the constraints in the *Lego* model before and after propagation (in degrees). Legend: $\perp$ is orthogonality, $\square$ is coplanarity and $\parallel$ is parallelism.

7. Experimental results

Comparison between IA-based triangulation [8] and polyhedral triangulation was performed on synthetic data consisting of 50 points randomly scattered in a sphere of unit radius, centered at the origin, and generating cameras placed at random positions, at a mean distance from the centre of 2.5 units with a standard deviation of 0.25. The image points were perturbed with a 1-pixel wide uniform distribution, and the same width was used to bound (with a square) the perturbed image points. The number of views was varied, and the corresponding volumes are reported in Tab. 3. Each entry is the mean of 50 independent trials. It is clear that the polyhedral triangulation outperforms the IA approach, both in terms of accuracy and of stability.

# views	*IA-boxes*	*Polyhedral*
2	26.35	3.68e-07
3	3.25e-04	1.01e-07
4	7.87e-04	5.76e-08
5	3.17e-05	3.88e-08

Table 3. Synthetic triangulation comparison experiment. Average volumes of the boxes obtained by IA-based triangulation compared with those of the polyhedra.

Constraints detection and analysis was tested on the synthetic models shown in Fig. 8. The 3D points were replaced by boxes to simulate the output of polyhedral triangulation. The size of the box varied from 0.5% to 3.0% of a "size gauge" computed as the median over the model's points of the farthest point distance. For each value, 20 perturbed models were generated using a uniform random distribution inside the boxes. Planes and constraints were automatically detected by our algorithm, the constraints were cut down using PRUNE, and a feasible solution of the resulting CSP was found using SNOPT.

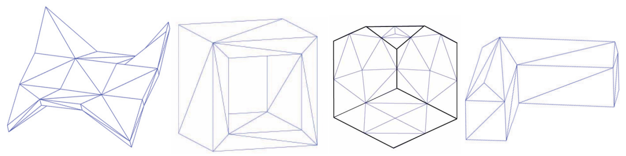


Figure 8. The four objects used for the synthetic experiments, here referred as (from the left) *Test*, *Boxhole*, *Cutcube* and *House*.

The number of planes correctly extracted is 22 for *Test*, 10 for *Boxhole*, 7 for *Cutcube*, and 11 for *House*. Tab. 4 shows the constraints extracted, before and after the structural rigidity analysis. Please note that at each step PRUNE chooses randomly which constraint to eliminate, so the final set of constraints varies from time to time; here, the most common case is reported. The solution of the CSP took about 20 s for *Cutcube* and *Boxhole*, 50 s for *House* and 90 s for *Test*.

The whole method was tested on real images, the *Pozzoveggiani* and *Tribuna* datasets. The *Pozzoveggiani* set is composed by 16 calibrated images of a church (Fig. 9). Polyhedral triangulation was performed, assuming a uniform error in the 2D point location bounded by a 7-pixel

	Automatic constraints				Pruning		
	$\perp$	$\bowtie$	$\square$	$\|$	$\perp$	$\bowtie$	Total
Test	22	61	60	8	9	16	93
Boxhole	12	1	24	4	9	0	37
Cutcube	0	3	40	5	0	3	48
House	9	6	25	4	7	4	40
Pozzoveggiani	59	2043	100	5	1	58	164
Tribuna	29	799	478	34	2	51	565

Table 4. Number of constraints automatically detected and number of remaining constraints after the structural rigidity analysis for the 3D models. The rightmost column reports the total number final of constraints. Legend: $\perp$, $\square$ and $\|$ as defined in Tab. 2, $\bowtie$ is angle equality.

Figure 9. Three of the 16 images of the *Pozzoveggiani* set.

wide square (Fig. 10). The mean volume of the resulting polyhedra is $(13cm)^3$, with respect to a volume of $(16.88m)^3$. Then, starting from the approximate solution obtained by randomly choosing one point inside each polyhedron, and connecting them manually, 36 planes were automatically extracted. Results from constraints detection and analysis are outlined in Tab. 4. As the reader can notice, in this case constraints pruning is essential to simplify the problem. The CSP solver (SNOPT) produced, after less than one minute, the result shown in Fig. 11, with the errors reported in Tab. 5.

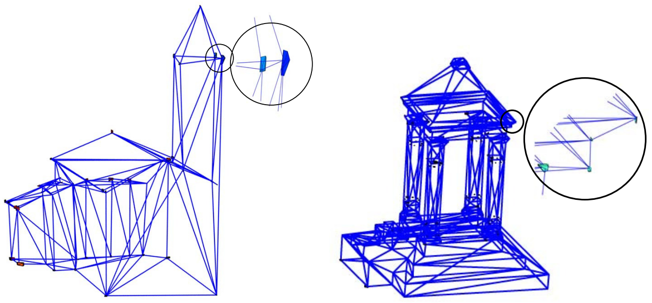


Figure 10. Polyhedral triangulation for *Pozzoveggiani* (left) and *Tribuna* (right). The magnifying glass highlights the polyhedra.

The *Tribuna* set consists of 10 calibrated images of an apse (Fig. 12).

Polyhedral triangulation was performed, assuming a uniform error in the 2D point location bounded by a 2-pixel wide square (Fig. 10). The mean volume of the resulting polyhedra is $(2cm)^3$, with respect to a volume of $(4.05m)^3$. Then, starting from the approximate solution obtained by

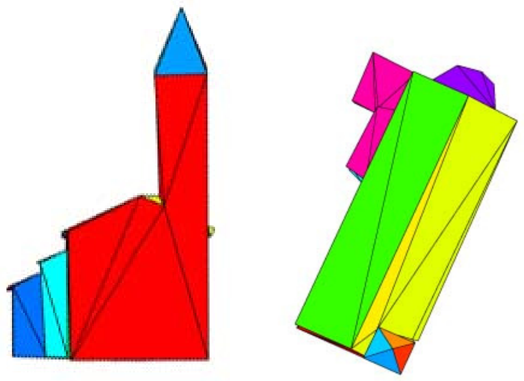


Figure 11. Final geometric reconstruction of *Pozzoveggiani* after the constraints propagation. Each plane is identified by a different colour.

		$\perp$	$\bowtie$	$\square$	$\|$
Pozzoveggiani	Before	$[2.05, 2.05]^\circ$	$[0, 0.04]^\circ$	$[0, 0.19]^\circ$	$[1.2, 6.8]^\circ$
	After	$[1.73, 1.73]^\circ$	$[0, 0.03]^\circ$	$[0, 0.13]^\circ$	$[1.2, 4.7]^\circ$
Tribuna	Before	$[0.03, 3.3]^\circ$	$[0, 0.03]^\circ$	$[0, 0.15]^\circ$	$[0.9, 10.4]^\circ$
	After	$[0.03, 0.3]^\circ$	$[0, 0.17]^\circ$	$[0, 0]^\circ$	$[0.9, 7.9]^\circ$

Table 5. [Min,Max] deviation from the constraints in the initial and final model (in degrees).

Figure 12. Three of the 10 images of the *Tribuna* set.

randomly selecting one point inside each polyhedron, and connecting them manually, 62 planes were automatically extracted. Constraints detection and analysis results are summarized in Tab. 4. The CSP solver (SNOPT) produced, after a few minutes, the result shown in Fig. 13, with the errors reported in Tab. 5.

8. Conclusions and future work

In this paper we presented a new approach to constrained surface modeling from many calibrated views. We demonstrated how polyhedral triangulation and a suitable constraint analysis and propagation can be used to obtain an accurate geometric model of a scene.

Given the 3D points connectivity information, our method automatically detect planes in the model using Mean Shift clustering, and geometric angular constraints among such planes, using grouping techniques. The con-

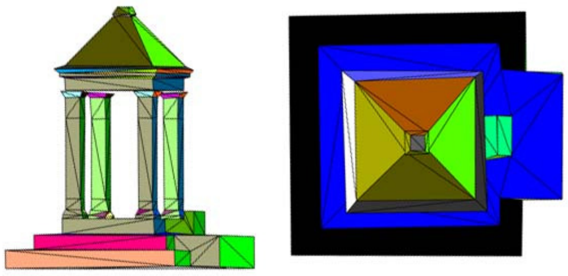


Figure 13. Final geometric reconstruction of *Tribuna* after the constraints propagation. Each plane is identified by a different colour.

straints are then processed in order to eliminate redundancies using structural rigidity analysis. In the end, the final 3D model is obtained by solving a CSP. Experiments show the effectiveness and the accuracy of the approach.

Future work will aim at removing the need for manually entering points and connectivity, thereby making the system fully automatic. Preliminary results in this direction are reported in [9].

Acknowledgments

We are grateful to P. Sturm for valuable comments and to L. Corazza for contributing to the implementation of the software. Some of the 3D models shown in Fig. 4 are courtesy of Mountaz Hascoët[2]. The SNOPT solver is available inside TOMLAB.

References

[1] A. Bartoli and P. Sturm. Constrained structure and motion from multiple uncalibrated views of a piecewise planar scene. *International Journal of Computer Vision*, 52(1):45–64, 2003.

[2] B. N. C. Jermann and G. Trombettoni. A new structural rigidity for geometric constraint systems. In *Fifth International Workshop on Automated Deduction in Geometry*, Linz (Hagenberg), 2002.

[3] H. Cantzler, R. B. Fisher, and M. Devy. Improving architectural 3D reconstruction by plane and edge constraining. In *British Machine Vision Conference*, pages 43–52, Cardiff (UK), 2002.

[4] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, 2002.

[5] M. Cristani, U. Castellani, and V. Murino. Adaptive feature integration for segmentation of 3D data by unsupervised density estimation. In *Proceedings of the International Conference on Pattern Recognition*, volume 4, pages 21–24, August 2006.

[6] P. E. Debevec, C. J. Taylor, and J. Malik. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. In H. Rushmeier, editor, *SIGGRAPH: International Conference on Computer Graphics and Interactive Techniques*, pages 11–20, New Orleans, Louisiana, August 1996.

[7] A. R. Dick, P. H. S. Torr, S. J. Ruffle, and R. Cipolla. Combining single view recognition and multiple view stereo for architectural scenes. In *Proceedings of the International Conference on Computer Vision*, volume 1, page 268, 2001.

[8] M. Farenzena, A. Fusiello, and A. Dovier. Reconstruction with interval constraints propagation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1185–1190, 2006.

[9] M. Farenzena, A. Fusiello, R. Gherardi, and R. Toldo. Towards fully automated architectural sketching. Submitted to 3DPVT 2008.

[10] E. Grossman and J. Santos-Victor. Least-square 3D reconstruction from one or more views and geometric clues. *Computer Vision and Image Understanding*, 99:151–174, 2005.

[11] R. I. Hartley and P. Sturm. Triangulation. *Computer Vision and Image Understanding*, 68(2):146–157, November 1997.

[12] C. M. Hoffmann, A. Lomonosov, and M. Sitharam. Finding solvable subsets of constraint graphs. In *Constraint Programming*, pages 463–477, 1997.

[13] D. Jelinek and C. J. Taylor. Reconstruction of linearly parametrized models from single images with camera of known focal length. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(7):767–774, 2001.

[14] C. Jermann, G. Trombettoni, B. Neveu, and P. Mathis. Decomposition of geometric constraint systems: a survey. *Internation Journal of Computational Geometry and Applications*, 16(5-6):379–414, 2006.

[15] F. Kahl. Multiple view geometry and the l_∞-norm. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 510–517, Beijing, China, 2005.

[16] D. Lowe. Fitting parameterized three-dimensional models to images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(5):441–450, May 1991.

[17] W. M. P. E. Gill and M. A. Sauders. SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM J. Optimization*, 12:979–1006, 2002.

[18] F. P. Preparata and M. I. Shamos. *Computational Geometry. An Introduction*, chapter 2, pages 72–77. Springer-Verlag, first edition, 1985.

[19] H. Stewenius, F. Schaffalitzky, and D. Nister. How hard is 3-view triagulation really? In *Proceedings of the International Conference on Computer Vision*, pages 510–517, Beijing, China, 2005.

[20] M. Wilczkowiak, P. Sturm, and E. Boyer. Using geometric constraints through parallelepipeds for calibration and 3D modeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(2):194–207, 2005.

[21] M. Wilczkowiak, G. Trombettoni, C. Jermann, P. Sturm, and E. Boyer. Scene modelling based on constraint system decomposition techniques. In *Proceedings of the International Conference on Computer Vision*, volume II, pages 1004–1010. IEEE, IEEE, October 2003.

[2]http://www.lirmm.fr/ mountaz/Ens/DessTni/OpenGL/Exemples/tutors/data/