Scene Reconstruction Based on Constraint System Decomposition Techniques

Paper ID: 559

Abstract

We present a new approach to 3D scene modeling based on geometrical constraints. Contrary to most of the existing methods, our approach allows us to obtain 3D scene models that respect the given constraints exactly. Our system is based on constraints that allow to describe a variety of shapes in a flexible way. We use this system for reconstructing 3D models of buildings from images. The constraint solving is performed by a constraint system decomposition technique called GPDOF. *It is a simple and polynomial-time algorithm that can find a reduced parameterization of a scene, allowing to optimize it while satisfying all the imposed constraints.*

1. Introduction

Reconstruction of accurate and photorealistic 3D models is one of the most challenging tasks in Computer Vision. It often requires dealing with problems which have been an object of research in several communities such as Computer Graphics and Computer-Aided Design.

In this paper, we address the problem of image-based reconstruction of a scene respecting a set of geometrical constraints. Scene reconstruction using only the image information is often an ill-conditioned problem. It is thus important to include additional information in the reconstruction framework. Depending on the application, constraints like coincidence, distance, angle between scene primitives are often known or easy to provide. Incorporating them into the reconstruction system has several advantages: it helps to stabilize the calibration and reconstruction processes and to improve the geometrical quality of the model. It makes it possible to model scenes from small sets of images.

There are two main approaches for imposing scene constraints during model creation. The first one consists in incorporating the constraints into the optimization process [7]. These methods however are often costly and do not guarantee the convergence and the constraint satisfaction in the general case.

The other solution is to construct a set of model parameters, called the **input parameters**, which, combined with the geometric constraints, would completely describe the model. Then, the optimization of the model by (bundle) adjusting it to the images would be performed only on the input parameters. This reduces the number of parameters to be optimized and provides a solution satisfying the imposed constraints perfectly.

This paper presents a new approach in the second category: a set of input parameters can be computed and the imposed constraints can be solved exactly. We use a constraint system decomposition algorithm, called GPDOF, presented in [16]. GPDOF is a simple and fast algorithm that can solve a system of geometric constraints while working at the algebraic level with a simple and general scheme. It determines (if possible) a sequence of so-called *r-methods*: an *r-method* is a routine which computes the coordinates of *output objects* based on the current value of *input object* coordinates, and satisfies the underlying constraints between input and output objects. An example would be an r-method that computes the position of some point located at known distances from three other points. A sequence of such r-methods allows thus to parameterize a scene model with a reduced set of parameters.

Several r-method patterns have been incorporated in the dictionary used by our system. They correspond to ruler-and-compass routines used in geometry or, more generally, to standard theorems of geometry. Our approach works on a *graph* that gives a structural view of the variables and equations defining the scene. First, we automatically add to our graph a set of r-methods that correspond to patterns in the dictionary. Second, GPDOF is applied. It provides the scene parameterization and a sequence of r-methods. Executing this sequence of selected r-methods on the current values of the parameterization, produces a 3D model that satisfies the underlying geometric constraints exactly.

We should highlight that, providing that the system contains no redundant equation, GPDOF can always produce a valid sequence of r-methods if such a sequence exists. Of course, a dictionary has a limited size, but the r-methods inside are tuned to our application and can tackle linear as well as non-linear subsystems of equations.

A first validation of our novel approach has been obtained on a scene made of 119 primitives and 131 geometric constraints, including a few quadratic distance constraints.

After the presentation of related work (Section 2), some background is detailed in Section 3. An overview of the whole process is given in Section 4 and an example illustrates the constraint satisfaction part in Section 5. Section 6 details the constraint solving process. Section 7 shows the results we have obtained on a real scene. Limitations are

discussed in Section 8.

2. Related work

Many works have focused on incorporating geometrical constraints for camera calibration and 3D reconstruction [15, 19, 11, 13, 2]. These works use various types of geometric constraints in order to stabilize the calibration and reconstruction results. A lot of approaches are proposed to deal with the coplanarity and colinearity constraints. However, more complex dependencies like distances and angles are still problematic to be used in the computer vision field.

Some works are based on techniques used in CAD systems or user interface design. The Facade system [6] uses a CAD-like approach to build a scene from complex primitives like cubes, prisms etc., and fit it to the image data. For obtaining a more flexible scene description, some researchers in computer vision [5, 4, 3, 4, 9] proposed to model scenes with simple primitives like points and lines. They design various constraint propagation schemes to search for a parametric description of the scene satisfying the constraints. However, the methods often require costly computations and do not guarantee to provide a solution. No results are shown on satisfaction of constraints more complex than bilinear.

The approach presented in this paper overcomes these drawbacks. The method described for constraint satisfaction is complete, fast and can be used to model complex constraints like distances, angles and distance/angle ratios.

3. Preliminaries

3.1. Problem statement

We work with scenes containing a set of geometric constraints $\mathcal{C}$ imposed on a geometric object set Ω in 3D space.

Let us call $\mathbf{M}_j \in \Omega, j \in [1..n]$ the scene points and $\mathbf{m}_{ij}$ their observed projections in cameras $\mathbf{P}_i, i \in [1..m]$. The problem of reconstructing the 3D model which respect the constraints $\mathbf{C}_k$ from $\mathcal{C}$ can be stated in two different ways [5].

1. using a local optimization method:

$$\text{minimize } \hat{C} = \sum_{i=1}^{n}\sum_{j=1}^{m} d(\mathbf{m}_{ij}, \mathbf{P}_i(\mathbf{M}_j)$$

$$\text{subject to constraints } \mathbf{C}_k \qquad (1)$$

   This method requires the use of numerical optimization methods which will incrementally improve the 3D structure at each optimization step. However, the convergence of those methods is not guaranteed in general.

2. using constraints $\mathbf{C}_k$ to reduce the number of scene parameters. Indeed, $\mathbf{C}_k$ can be used to find a set of input parameters and functions μ_j such that $\mu_j(\omega_c)$ yields the position of point $\mathbf{M}_j$. A model constructed this way satisfies all the constraints $\mathbf{C}_k$. Thus the minimization problem can be stated as follows:

$$\text{minimize } \hat{C} = \sum_{i=1}^{n}\sum_{j=1}^{m} d(\mathbf{m}_{ij}, \mathbf{P}_i(\mu_j(\omega_c)); \qquad (2)$$

Note that, in the second approach, the cost function involves only point and camera parameters. Thus, lines and planes are used only to constrain the point positions. Their coordinates however can be included into the input set ω_c.

As stated in [5], the advantage of the second method above is its simplicity (once the parametrization is computed) and the fact that constraints are satisfied at every optimization step. The constraint system decomposition technique presented in this paper follows this principle.

3.2. Scene modelization

The objects implemented currently in our system are stated in Table 1. All of the object coordinates are represented by homogeneous vectors of coordinates. For the details of this representation, especially for the Plücker line representation, see [8].

Our system accepts most of the common linear and non–linear constraints: distance (point-point, point-line, point-plane), incidence (point-line, point-plane and line-plane), parallelism (line-line, line-plane, plane-plane) and orthogonality (line-line, line-plane, plane-plane) constraints. Any other constraint which can be expressed as equations in terms of objects coordinates, like angles, distance and angle ratios can be easily incorporated.

object	dof	representation	internal constraints
point	3	$\mathbf{X}^\intercal = (x, y, z, t)$	$t = 1$
line	4	$\mathbf{L}^\intercal = (m^\intercal_{1\times3}, v^\intercal_{1\times3})$	$m^\intercal v = 0, \|v\| = 1$
plane	3	$\mathbf{A}^\intercal = (n^\intercal_{1\times3}, d)$	$\|n\| = 1$

Table 1: Representation of points, lines and planes in 3D

3.3. Definitions

The geometric constraints in the scene yield a set of equations between the parameters of the objects. The scene can then be modeled by:

- a set E of *equations* generated by geometric constraints from $\mathcal{C}$; the equations are linear or non-linear;
- a set V of *variables* over the reals with a current value each; the variables are the coordinates (or parameters) of objects in Ω.

Our model reconstruction system also requires an input set M of *r-methods*. An *r-method* satisfies a subset E_m of equations in E by calculating values for its *output variables* as a function of the other variables implied in the equations. For example, an r-method can compute the parameters of a line based on the current position of two points coincident to this line.

Definition 1 *An* **r-method** m *in* M *can satisfy a subset of equations* $E_m \subset E$. *Let* V_m *be the set of variables linked to one or several equations in* E_m. V_m *is partitioned into two disjoint subsets: the non–empty subset of* **output variables** V_m^{out} *and the* **input variables** V_m^{in}.

R-method m *is a function defined by* $S_m^{out} = m(\overline{V_m^{in}})$ *such that the equations in* E_m *are satisfied.*

The **execution** *of r-method* m *produces all the solutions* S_m^{out} *to* V_m^{out} *satisfying* E_m. *Method* m *is* **free** *if no variable* v *in* V_m^{out} *is connected to a constraint in* $E - E_m$. *Thus, executing a free method cannot violate other constraints in* $E - E_m$.

The algorithms used in this paper require a structural view of the entities in the scene. The constraint system and the equation system are respectively represented by a *constraint graph* and an *equation graph* (see Figures 1 and 2).

Definition 2 *A* **constraint graph** *is a bipartite graph where nodes are constraints and objects which are represented by rectangles and circles respectively. Each constraint is connected to its objects.*

An **equation graph** *is a bipartite graph where nodes are equations and variables which are represented by rectangles and circles respectively. Each equation is connected to its variables.*

An **enriched equation graph** (V, E, M) *is an equation graph* (V, E) *enriched with a set* M *of r-methods.*

An equation graph is similar to a matrix and indicates the dependencies between equations and variables in the scene.

4. Overview

The model acquisition process is divided in several steps:

. **Object and constraint definition:** At present, the process is manual[1].

. **Initial camera calibration and model reconstruction:** We use the linear calibration method described in [18] and an unconstrained bundle adjustement to obtain initial values for the camera and model parameters.

[1] It could be semi-automatized but, to obtain satisfying results in a general case, some manual correction is necessary. The user interaction can be limited using automatic point extraction and matching between images. Then RANSAC-based algorithms can be run to test the geometric constraints.

. **Computation of a scene parametrization and a constructive order:** A graph-based process finds a set of input parameters and a way to satisfy all the equations, based on the values of the input parameters. Two steps are performed:

1. *R-method addition phase:* Add *automatically* in the equation graph all the r-methods corresponding to an r-method pattern present in a dictionary.
2. *Planning phase:* Perform `GPDOF` on the enriched equation graph. GPDOF produces two major results:
   - a set of input parameters,
   - a sequence of r-methods (called *plan* or constructive order) to be executed one by one.

. **Model optimization:** Once the input parameters and the constructive order have been computed, the minimization of the expression (2) is performed. We run a bundle adjustment over the input parameters using the Levenberg-Marquadt optimization method. Every time the cost function is computed, the following operations are performed:

1. replace the input parameters by their current value,
2. compute the points $\mathbf{M}_j$ by executing one by one the r-methods in the plan; among the several solutions computed for $\mathbf{M}_j$, select the one that minimizes the cost function (reprojection errors).

5. Example of constraint satisfaction

To illustrate this process, let us take a small example describing a parallelogram in 2D in terms of lines, points, coincident constraints and parallelism constraints (see Figure 1). Of course, the scenes we handle with our tool are in 3D, and this example is just presented for didactic reasons. Figure 1 also shows the bipartite constraint graph containing four points P_a,...,P_d, four lines L_a,...,L_d, eight coincidence constraints C_1,...,C_8 and two parallelism constraints P_1, P_2.

. **R-method addition phase.** It is essentially based on a subgraph matching performed on the constraint graph. When a subgraph matches an entry in the dictionary, the corresponding r-methods are added to the equation graph. For instance, the subgraph made of nodes P_a, C_8, L_a, C_1, P_b matches a pattern in the dictionary: it has added the only r-method which places the line L_a on the points P_a and P_b. This r-method outputs on variables of the line L_a, satisfies the equations of the coincident constraints C_1 and C_8. Its input variables come from the points P_a, P_b. At the end, the equation graph is enriched with 16 r-methods. Only eight of them are depicted in Figure 2 for the sake of clarity. These

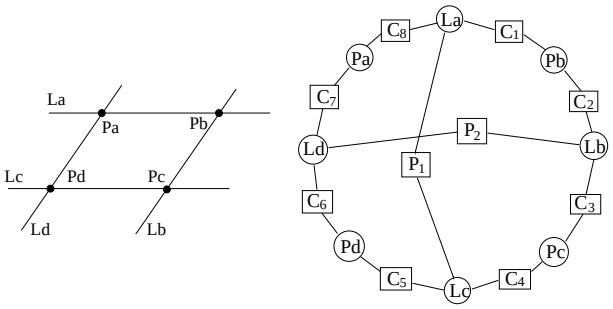


Figure 1: A didactic 2D scene (left) and the corresponding constraint graph (right)

r-methods match one of the three following patterns: line coincident to two points (e.g., r-methods m_1 and m_7); point at the intersection of two known lines (e.g., m_2, m_4, m_6, m_8); line passing through a known point and being parallel to another line (e.g., m_3, m_5).

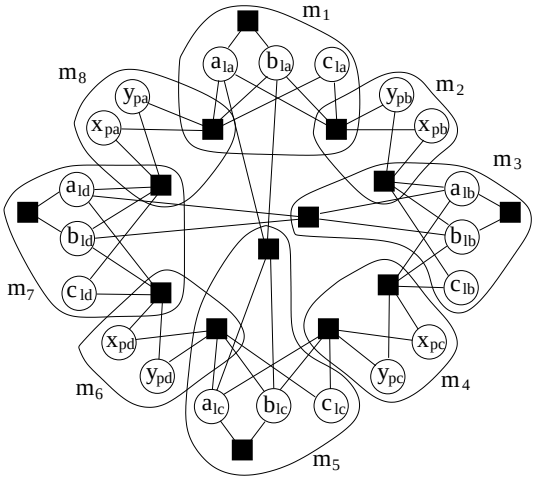


Figure 2: The equation graph of our didactic 2D scene enriched with automatically defined r-methods. An r-method is represented by a hyper-arc including its equations and its output variables.

. **Planning phase.** GPDOF extracts a sequence of r-methods among those in the enriched equation graph. Its objective is to iteratively select a free r-method in this graph. At the beginning, r-methods m_2, m_4, m_6, m_8 are free, so that one of them is selected, e.g., m_4. This selection implies the removal of the equations and the output variables of m_4 from the equation graph (see Fig. 3(a)). This frees r-methods m_3 and m_5 which are selected and removed next (see Fig. 3(b)). The r-methods m_1 and m_7 are then free and can be selected. The process ends since no more constraint remains in the equation graph. The obtained plan is the sequence of r-methods ranked in reverse order as compared to the selection order: (m_1, m_7, m_3, m_5, m_4). The input parameters are the variables in points P_a, P_b and P_d.

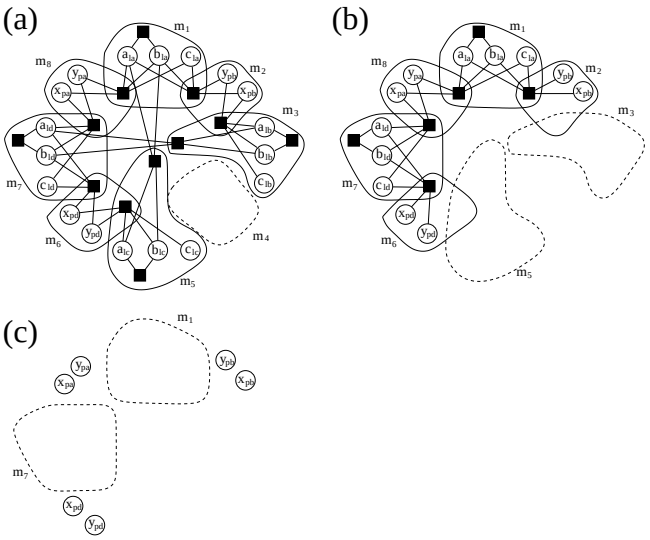


Figure 3: A possible planning phase performed by GPDOF on the didactic scene

. **Execution phase.** The r-methods in the plan are executed in order. Figure 4 shows the corresponding geometric effect. For example, the execution of r-method m_1 places the line L_a on the points P_a and P_b.

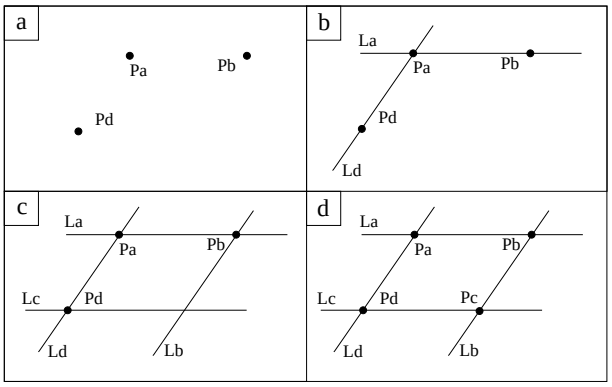


Figure 4: Execution of r-methods in the plan. (a) The input parameters are issued from one iteration of bundle adjustment and are placed first. (b) m_1 and m_7 computes L_a and L_d resp. (c) m_3 and m_5 computes L_b and L_c resp. Finally, m_4 places point P_c.

6. Constraint solving

6.1. Designing r-methods

Since a set of r-methods will be executed for every call of the cost function during the bundle adjustment process, it

is very important that method execution is fast. A lot of r-methods are designed to solve the linear equations due to incidences, parallelisms, orthogonalities. These r-methods are easy to be implemented in an efficient way.

Designing efficient r-methods for solving non-linear equations is generally not trivial. Also, numerical methods cannot be used if one wants to guarantee finding an existing solution[2]. That is why we made symbolic manipulation of the equations until "simple" routines have been obtained. An r-method execution procedure is divided in a sequence of fast atomic steps: evaluations of polynomial terms and solving of equations of the form: $a\,x^2 + b\,x + c = 0$.

Our r-methods have as many equations as outputs. They compute a finite set of solutions for the output variables: the dimension of the variety of the solutions is 0. This allows us to combine the partial solutions computed by the different r-methods in the plan without losing any solution.

The current dictionary contains all the r-methods that solve the constraints by computing the parameters of one object. About 60 r-method patterns have been defined.

6.2. Automatic addition of r-methods

The subgraph matching is based on two algorithms.

Exploring all connected subgraphs of size at most k

It builds all the connected subgraphs of size less than a small value k. This value is the maximum number of nodes (objects+constraints) implied in any r-method of the dictionary (e.g., 7 in our tool). Starting from a single node, the subgraphs are built by incrementally adding a neighbor node to the current connected subgraph until the size k is reached.

This depth first search algorithm is a simplification of the algorithmic scheme presented in [1]. Its time-complexity is $O(N \times d \times k^4)$, where N is the number of connected subgraphs of size k or less and d is the maximum degree of nodes in the graph. In practice, the complexity of this algorithm is highly dominated by N.

Subgraph recognition

Every subgraph found with the process above must be compared with the subgraph patterns in our dictionary. However, the problem of deciding whether two graphs are *isomorphic* is still an open problem for which no polynomial algorithm is known [10]. Therefore the dictionary is implemented as a hash table, which eliminates most of the subgraphs not in the dictionary. A final comparison is made by a standard combinatorial process.

[2]R-methods must be able to compute **all** the partial solutions of the involved equations in order to combine them together in a combinatorial way.

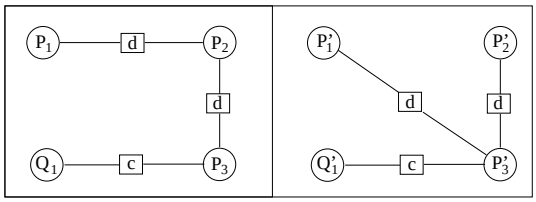


Figure 5: Two subgraphs with the same entry in the hash table. *Right:* a subgraph pattern in our dictionary made of 3 points P'_1, P'_2, P'_3, a plane Q'_1, two distance constraints and one coincidence. *Left:* a "bad" subgraph with no corresponding r-method.

6.3. The GPDOF algorithm

GPDOF is a generalization of local propagation algorithms used to solve multiway dataflow constraints [14, 17]. It works on an enriched equation graph [16]. It finds a sequence of r-methods to be executed for satisfying all the equations. GPDOF runs the three following steps until no more equation remains in the graph G (success) or no more free r-method is available (failure)[3]:

1. select a free r-method m,
2. remove from G the equations satisfied by m and remove the output variables of m,
3. create all the *submethods* of a r-method m_i that share equations or output variables with m.

A plan can be obtained by reversing the selection order: the first selected r-method will be executed last.

The first two steps define the standard PDOF algorithm [14] on which GPDOF is based[4]. Selecting free r-methods iteratively ensures that no loop is created between the selected r-methods.

It turns out that there is no guarantee that PDOF finds a valid plan when applied to problems with general r-methods. In our 2D scene example, selecting first m_6, m_4, m_8 and m_2 makes it impossible to select the other defined r-methods. This highlights the notion of *submethod* which renders the PDOF scheme complete in the general case. Roughly, the notion of submethod explains that a partially removed r-method remains available for a future selection. The notion is examplified here and the reader will refer to [16] to get a more detailed information. Let us take again the enriched equation graph in Fig. 2. GPDOF may select first m_6 which is free. The third step of GPDOF then creates the submethod m'_5 of m_5 and the submethod m'_7 of m_7 (see Fig. 6). The process continues and selects m_4, m'_5, m_1, m'_2, m'_3, and finally m'_7.

[3]In case of failure, one obtains a incomplete plan which can solve only a subpart of the equations.

[4]PDOF accepts only r-methods solving one equation.

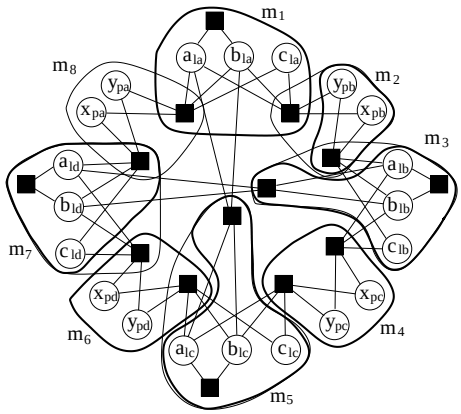


Figure 6: Creation of submethods by GPDOF. Selected r-methods (m_4, m_6) and submethods (m'_2, m'_3, m'_5, m'_7) are represented by thick hyper-arcs.

The work of GPDOF is purely structural and can viewed as the task of selecting and transforming defined hyper-arcs. One can prove that a submethod m' of m created after the removal of a free r-method can still be selected later. Indeed, the set of solutions of m also solve the equations of m'.

The notion of submethod can then be forgotten in the execution phase and the initially defined r-methods can be applied. In our example, the sequence (m_7, m_3, m_2, m_1, m_5, m_4, m_6) is executed. This implies that constraints are solved several times by different r-methods.

In practice, the time complexity of GPDOF is quasi-linear. Its worst-case time complexity is O($n \times dc \times dv \times m \times (g \times dc + g^2)$) (see [16]). n is the number of equations, m is the maximum number of r-methods per equation, dc and dv are the maximum degrees of respectively an equation and a variable in the equation graph, and g is the maximum number of equations and output variables involved in an r-method. This complexity is a polynomial function of the input parameters: $dv \leq n$; $g \leq n$; $dc \leq |V|$; $m \leq |M|$.

6.4. Determining the input parameters

The input parameters given to the bundle adjustment simply consist of the variables which are output by no r-method in the plan, e.g., the 6 coordinates of points P_a, P_b, P_d in Fig. 2 or the 2 coordinates of point P_a in Fig. 6. Even if, of course, some other variables serve as input of the execution process (e.g., the coordinates of P_b are input variables for m_3), they cannot be given to the bundle adjustment. Indeed, their current value is read at the beginning but modified again when executing subsequent r-methods in the plan.

This surprising phenomenon is due to the fact that equations may be solved several times during the process, so that more variable values are modified by r-method execution.

6.5. Execution phase

At first, the input parameters are considered constant, that is, they are replaced by their value in subsequent equation systems solved by r-methods. Then, the r-methods in the sequence are executed one by one.

When an r-method solves non-linear constraints, it generally produces several solutions for its output variables. Therefore, several total solutions are generally obtained at the end. A combinatorial optimization process is performed to combine all the partial solution given by the r-methods in the plan. It selects the solution which minimizes the reprojection errors. The backtracking process used for this task is standard and will not be detailed further.

7. Results

We have used our approach to build a model of a church. A set of five images were used, together with architectural plans from which several distance measurements were extracted. Overall, 131 constraints (112 coincidence, 15 parallelism and 4 distance constraints) were used to constrain 119 objects (91 points, 20 lines and 8 planes). Our r-method dictionary contains 60 r-methods.

. Performance tests. The time required for the r-method addition and the planning phase was 2 min 40s on a Pentium IV 2 Ghz. The optimization phase (bundle + plan execution) required between 1 and 5 min. The most time-consuming step in the first part is the exploration of all connected subgraphs of size at most k. As mentioned in Section 6.2, its time complexity is highly dominated by the number of connected subgraphs. This number strongly depends on the size k of the largest subgraph. Performance test results are shown in Table 2.

k	# connected subgraphs	time	# subgraphs
6	2.7×10^5	18s	3.2×10^{11}
7	1.55×10^6	2 min 27s	1.1×10^{13}
8	8.14×10^6	16 min 15s	3.5×10^{14}
9	4×10^7	104 min 44s	9.4×10^{15}

Table 2: Exploration of all the connected subgraphs of size at most k in our scene. The results highlight two interesting points. Limiting the search for connected subgraphs is crucial and leads to gain 7 orders of magnitude for $k = 7$. However, our approach cannot be used in pratice for patterns of size more than 10.

Execution time of the subgraph matching phase is about 7% of the overall time. The execution time of GPDOF is negligible.

We have also tested the performance of implemented r-methods. R-methods solving both linear and non-linear constraints need 3.5×10^{-5}s (average on 1000000 tries).

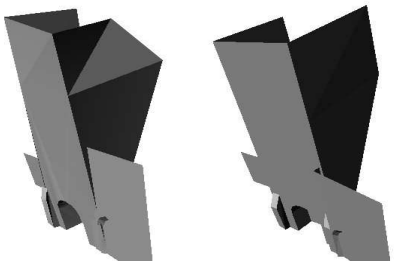


Figure 9: The model before and after the optimization.

. **Reconstruction results.** The scene description contained 425 equations and 237 scene variables. The number of input parameters is 181.

2143 r-methods have been added automatically on the scene equation graph. The r-method plan was built of 107 methods and the number of possible solutions was 16 (due to 4 distance constraints). Our backtracking system chooses the solution giving the smallest reprojection error. The computation time for one r-method plan execution and the induced update of all system parameters was 55ms.

Figure 7–(a) shows one of the five images used. The results obtained through the unconstrained bundle adjustment are presented in Figures 7–(b) and 7–(c). The model suffers from several artifacts: colinearity and coplanarity are not respected for several points, causing an unpleasant visual aspect. Moreover, without imposing constraints, some of the points that are important to model the overall structure, cannot be reconstructed: for example the points inside the main gate of the church. The major artifacts are marked out on Figure 7–(b).

By imposing appropriate constraints, we overcame these problems. Figure 8 shows the model produced using our method. We show how the parts of the model mentioned above were corrected, leading to a visually correct model.

Figure 9 highlights the importance of the optimization phase. Sometimes, model creation from just the initial values of the input parameters and a single execution of the r-methods is not satisfactory. This is due to bad initial values for the input parameters. Thus, geometrical constraints in one part of the model can cause deformations in other (unconstrained) parts. This happened at the top right corner of the tower. Of course, the definition of additional constraints would overcome this problem. However, in the presented scene the initially introduced constraints were sufficient to maintain the correct geometry of the model.

8. Discussion

We have presented a solution to the problem of 3D scene modeling under geometric constraints, based on techniques for constraint system decomposition. The proposed method is original and efficient. Our system has been validated on a model containing 119 primitives and 131 geometric constraints, some of them being quadratic. The obtained results are geometrically correct and fit well the images.

Due to the complexity of the problem, there are still many challenging issues to be solved.

This paper presents the first attempt to automatically add r-methods in a constraint system. However, this is the most costly phase in our process, due to the big number of connected subgraphs. We believe that more sophisticated combinatorial techniques can radically improve the performance of this phase.

Redundant constraints can prevent GPDOF to find a free r-method. Moreover, it is not acceptable to rely on the user to remove redundant constraints manually. Dealing with constraint redundancy has been the subject of research in the CAD community for a long time and it is still open in the general case. However, we hope that special r-methods based on simple geometric theorems can be used in a preprocessing step to remove a lot of occurring redundancies.

It is possible that, during the execution of an r-method, a numerical *singularity* occurs (for example if a plane is being built from 3 almost colinear points). Our current solution makes use of the ad-hoc aspect of an r-method: a condition on the initial values of input variables is checked and well-conditioned solutions are favored.

R-methods have several advantages. They are fast and take into account the geometrical properties of the associated subgraph. Also, they yield all the solutions to the treated equation system. However, the behavior of the whole process is very dependent on the defined r-method dictionary. We think about mixing GPDOF with other constraint graph decomposition methods [12] to improve the generality of the method.

References

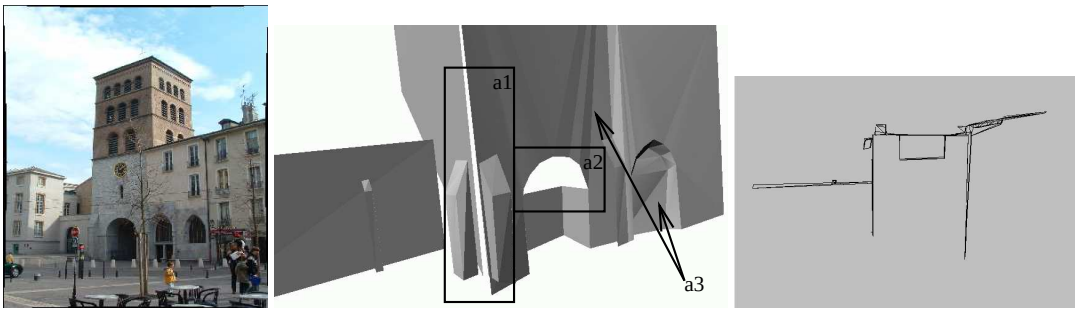
[1] David Avis and Komei Fukuda. Reverse Search Enumeration. *Discrete Applied Math*, 6:21–46, 1996.

[2] A. Bartoli and P. Sturm . *Constrained Structure and Motion From N Views of a Piecewise Planar Scene*. International Symposium on Virtual and Augmented Architecture, pp. 195-206, June 2001.

[3] Pierre-Louis Bazin. *A parametric scene reduction algorithm from geometric relations*, Vision Geometry IX, SPIE 2000.

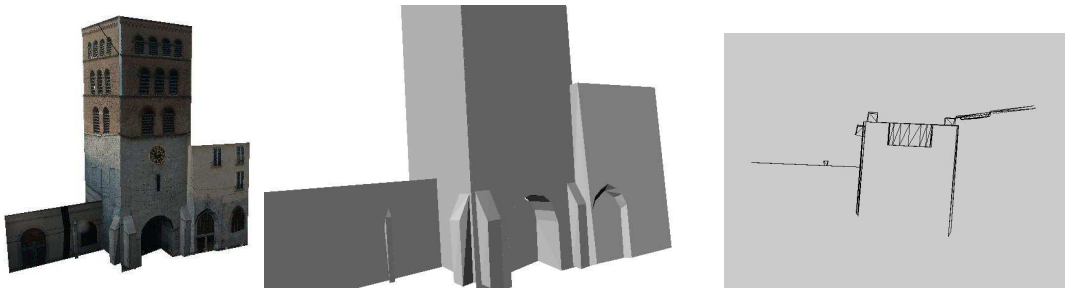
[4] D. Bondyfalat and D. and S.Bougnoux. *Imposing Euclidean Constraints during Self-Calibration Processes*, 3D Structure from Multiple Images of Large-Scale Environments, Springer-Verlag, 1998.

[5] D. Bondyfalat, B. Mourrain and T.Papadopoulo. *An Application of Automatic Theorem Proving in Computer Vision*, 2nd International Workshop on Automated Deduction in Geometry, Springer-Verlag, 1999.



(a) (b) (c)

Figure 7: (a)–one of five photos used for the reconstruction; (b)–some artifacts of the unconstrained model; ($a1$)-the colinearity is not respected; ($a2$)- some points cannot be reconstructed, ($a3$)-the coplanarity of the points is not preserved; (c)– the orthographic view of the model obtained through the underconstrained bundle adjustment; not all expected parallelism constraints are respected.



(a) (b) (c)

Figure 8: (a)–the overview of the model;(b)–the details of the optimized model;(c)– the orthographic view of the optimized model.

[6] P.E. Debevec, C.J. Taylor, and J. Malik. *Modeling and Rendering Architecture from Photographs: A Hybrid Geometry- and Image-Based Approach*, SIGGRAPH, 1996 .

[7] P. E. Gill, W. Murray and M. H. Wright. *Practical Optimization*, Academic Press, 1981.

[8] R.I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*, Cambridge University Press, 2000.

[9] O. Lhomme P. Kuzo P. Mace. *Desargues : a constraint-based system for 3D projective geometry* Workshop on Geometric Constraint Solving and Applications, 1997.

[10] Christos H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[11] P. Poulin, M. Ouimet and M.-C. Frasson. *Interactively Modeling with Photogrammetry*, Eurographics Workshop on Rendering, pp. 93-104, 1998.

[12] Alex Pothen and Jun Chin-Fan. Computing the block triangular form of a sparse matrix. *ACM Transactions on Mathematical Software*, 16(4):303–324, 1990.

[13] P.F. Sturm and S.J. Maybank. *A Method for Interactive 3D Reconstruction of Piecewise Planar Objects from Single Images*, BMVC, pp. 265-274, 1999.

[14] Ivan Sutherland. *Sketchpad: A Man-Machine Graphical Communication System*. PhD thesis, Department of Electrical Engineering, MIT, 1963.

[15] R. Szeliski, P. H. S. Torr. *Geometrically Constrained Structure from Motion: Points on Planes*. SMILE Workshop, pp. 221-237, 1998.

[16] Gilles Trombettoni. A Polynomial Time Local Propagation Algorithm for General Dataflow Constraint Problems. In *Proc. Constraint Programming CP'98, LNCS 1520 (Springer Verlag)*, pages 432–446, 1998.

[17] Bradley Vander Zanden. An incremental algorithm for satisfying hierarchies of multi-way, dataflow constraints. *ACM Transactions on Programming Languages and Systems*, 18(1):30–72, January 1996.

[18] M. Wilczkowiak, E. Boyer and P. Sturm. *3D Modeling Using Geometric Constraints: A Parallelepiped Based Approach*, ECCV, Vol IV, pp. 221-237, 2002.

[19] C. Zeller. *Calibration Projective Affine et Euclidienne en Vision par Ordinateur*. PhD thesis, École Polytechnique, 1996.