Scene Modeling Based on Constraint System Decomposition Techniques

Marta Wilczkowiak[1], Gilles Trombettoni[2], Christophe Jermann[3], Peter Sturm[1], Edmond Boyer[1]

[1] MOVI Project, INRIA Rhône-Alpes, 655 avenue de l'Europe, Montbonnot, 38334 Saint Ismier cedex, France
{Marta.Wilczkowiak, Edmond.Boyer, Peter.Sturm}@inrialpes.fr

[2] COPRIN Project, I3S-INRIA, 2004 route des lucioles, 06902 Sophia Antipolis cedex, B.P. 93, France
trombe@sophia.inria.fr

[3] Artificial Intelligence Laboratory, EPFL CH-1015 Lausanne, Switzerland
jermann@lia.di.epfl.ch

Abstract

We present a new approach to 3D scene modeling based on geometric constraints. Contrary to the existing methods, we can quickly obtain 3D scene models that respect the given constraints exactly. Our system can describe a large variety of linear and non-linear constraints in a flexible way.

To deal with the constraints, we decided to exploit the properties of the GPDOF *algorithm developed in the Constraint Programming community [12]. The approach is based on a dictionary of so-called* r-methods*, based on theorems of geometry, which can solve a subset of geometric constraints in a very efficient way.* GPDOF *is used to find, in polynomial-time, a reduced parameterization of a scene, and to decompose the equation system, induced by constraints, into a sequence of r-methods. We have validated our approach in reconstructing, from images, 3D models of buildings based on linear and quadratic geometric 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 geometric 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. Defining geometric constraints between scene primitives and incorporating them into the reconstruction system helps to stabilize the calibration, improves the quality of the model and limits the number of required images.

Our approach is based on a dictionary of so-called *r-methods*, based on theorems of geometry, which can solve a subset of geometric constraints in a very efficient way. Two graph-based algorithms are proposed to find a set of *input parameters* in a scene (i.e., a reduced parameterization), and to decompose the equation system, induced by constraints, into a sequence of r-methods. The input parameters, combined with the geometric constraints, completely describe the model. When a value is given to the input parameters, there exists a finite set of solutions for the rest of the system satisfying the imposed constraints. Values of input parameters are obtained by a standard model optimization which (bundle) adjusts the model to the images. Then, the r-methods in the computed sequence are executed to produce a model that satisfies all the constraints. In our approach, a set of input parameters can be computed in polynomial-time and the imposed constraints can be solved exactly and quickly. We should highlight that, provided that the system contains no redundant equation, we can always produce a sequence of r-methods if such a sequence exists.

A first validation has been obtained on a scene made of 119 primitives and 137 geometric constraints, including quadratic distance constraints.

Many works have focused on incorporating geometric constraints for camera calibration and 3D reconstruction [11, 17, 8, 9, 1]. 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 coplanarity and collinearity constraints. However, more complex dependencies like distances and angles are still problematic.

Some works are based on techniques used in CAD systems or user interface design. The Facade system [5] uses a CAD-like approach to build a scene from complex primitives like cubes, prisms etc., and fits it to the image data. In [6], these primitives are automatically detected in the images. For obtaining a more flexible scene description, some researchers in computer vision [4, 2, 7] 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 or 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. It is fast and sufficiently flexible to model various types of geometric constraints, including non-linear constraints like distances, angles and distance/angle ratios.

An overview of the whole process is given in Section 2. Section 3 details the constraint solving process and the optimization phase. Section 4 shows the results obtained on a real scene. Limitations are discussed in Section 5.

2. Overview

The problem is to build 3D models from images. It consists in estimating camera and model parameters, such that the projection of the 3D model points conform to the input image points. More formally, using the projective camera model, the image m of a 3D point $\mathbf{M}_j$ can be expressed by $m = \mathbf{P}_i\,\mathbf{M}_j$.
$\mathbf{P}_i = K\,[R\ t]$ is a 3×4 matrix: the 3×4 matrix $[R\ t]$ encapsulates the relative orientation R and the translation t between the camera i and the global coordinate system. The matrix K is the 3×3 calibration matrix containing the intrinsic camera parameters.

Generally, in such systems, the final refinement step, called *bundle adjustement*, tends to minimize a cost function given by the sum of distances between given image points and projections of the model 3D points. These distances (i.e., functions d in the formula below) are called *reprojection errors*.

There are two ways to incorporate in the system a set $\mathcal{C}$ of geometric constraints between points:

1. Most existing approaches incorporate the (soft) constraints into the local optimization process (the constraint violations are part of the cost function to be minimized):
$$\text{minimize } \hat{C} = \sum_{i=1}^{n}\sum_{j=1}^{m} d(\mathbf{m}_{ij}, \mathbf{P}_i\,\mathbf{M}_j) \quad \text{subject to constraints } \mathbf{C}_k \tag{1}$$
These methods however are often costly. They require the user to rule additional parameters in practice. Furthermore, they guarantee neither the convergence nor the (exact) constraint satisfaction in the general case.

2. Another approach uses the 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 ω_c 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 $\mathbf{C}_k$. The minimization problem can thus be stated as follows:
$$\text{minimize } \hat{C} = \sum_{i=1}^{n}\sum_{j=1}^{m} d(\mathbf{m}_{ij}, \mathbf{P}_i\,\mu_j(\omega_c)) \tag{2}$$

As stated in [4], the advantage of the second method above is that the constraints are satisfied exactly at every optimization step. The approach presented in this paper follows this principle. As shown in Fig. 1, it is divided into three main phases: initialization, constraint planning and optimization.

2.1. Initialization

In addition to 2D images, geometric objects and constraints must be defined. The 3D model is represented by *points*, *lines* and *planes*. They are subject to linear and non–linear constraints such as *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). Other constraints like angles, distance and angle ratios can be easily incorporated.

The cameras are then calibrated using the linear method described in [15]. An initial reconstruction is provided by a quasi-linear approach exploiting projections and geometric constraints [16]. *After this phase, we should highlight that all the variables (camera and model parameters) have an initial value.*

2.2. Constraint planning

Our model reconstruction system requires a set of *r-methods* which allows us to decompose the whole equation system into small subsystems. An r-method (see [12] and Definition 1) is a predefined routine used to solve a set of geometric constraints. An r-method 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 which computes the parameters of a line based on the current positions of two points incident to this line. Another example would be an r-method that computes the positions of some 3D point located at known distances from three other points.

Several r-method patterns have been incorporated in a dictionary used by our system. They correspond to standard theorems of geometry. The constraint planning works with this dictionary and with graphs, detailed in Section 3, which store dependencies between objects, constraints, variables and equations. The process is divided into two steps :

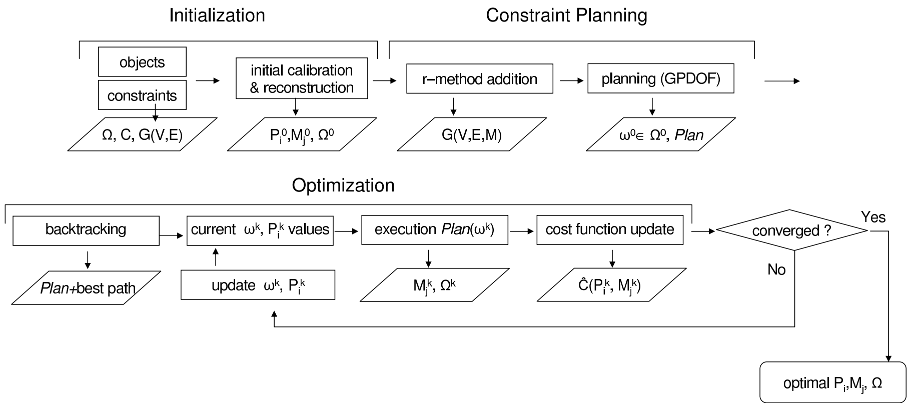


Figure 1. The overview of the model acquisition process. X^k represents the entity X at the k^{th} iteration of the optimization.

1. *R-method addition phase: Automatically* add to the equation graph all the r-methods corresponding to r-method patterns present in the dictionary.
2. *Planning phase:* Perform `GPDOF` [12][1] on the enriched equation graph. `GPDOF` produces a set of input parameters and a sequence of r-methods (called *plan*) to be executed one by one.

2.3. Model optimization

The optimization process is based on a standard numerical algorithm and minimizes the reprojection errors. In our approach, it only adjusts the input parameters. Every time the cost function is computed (inside the numerical algorithm), the r-methods in the plan are executed, producing a new value for the other variables such that all the constraints in the model are satisfied.

3. Constraint planning and model optimization

After some definitions required for a full understanding of the presented techniques, we give some details on the constraint planning and on the optimization process: designing r-method patterns, automatically adding r-methods to the equation graph, computing a set of input parameters and a sequence of r-methods, executing the r-methods in the plan. To illustrate this part, we take a small example describing a parallelogram in 2D in terms of lines, points, incidence constraints and parallelism constraints (see Figure 2). Of course, the scenes we handle with our tool are in 3D, and this example is just presented for didactic reasons. Figure 2 also shows the bipartite constraint graph containing four points P_a,...,P_d, four lines L_a,...,L_d, eight incidence constraints C_1,...,C_8 and two parallelism constraints P_1, P_2.

3.1. 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:

[1] `GPDOF` stands for General Propagation of Degrees of Freedom.

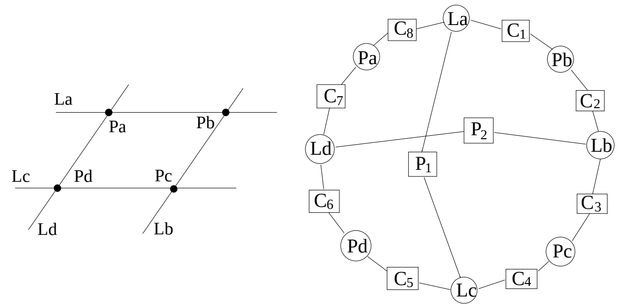


Figure 2. A didactic example of a 2D scene (left) and the corresponding constraint graph (right).

- a set V of *variables* over the reals with a current value each; the variables are the coordinates (or parameters) of geometric objects;
- a set E of *equations* generated by geometric constraints; the equations are linear or non-linear.

Our model reconstruction system also requires an input set M of *r-methods*. An *r-method* is a routine executed to satisfy 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.

Definition 1 *An* **r-method** m *in* M *is a function over a set of equations* $E_m \subset E$*, a non–empty set of* **output variables** $V_m^{out} \subset V$*, and a set of* **input variables** $V_m^{in} \subset V$*. (*$V_m^{in} \cup V_m^{out}$ *forms the set of variables involved in one or several equations in* E_m*.)*

The r-method m *replaces* V_m^{in} *by their values* $\overline{V_m^{in}}$ *and yields all the solutions* S_m^{out} *to* V_m^{out} *satisfying* E_m*.*

The r-method m is **free** if no variable v in V_m^{out} is involved in a constraint in $E \setminus E_m$. Thus, executing a free method cannot violate other equations in $E \setminus E_m$.

The algorithms used in this paper require a structural view of the entities in the scene. The geometric constraint system and the equation system are respectively represented by a *constraint graph* and an *equation graph* (see Figures 2 and 3). An equation graph is similar to a matrix and indicates the dependencies between equations and variables in the scene.

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.*

3.2. Design of r-methods

The current dictionary of our tool contains 60 r-method patterns. It includes all the r-methods that solve constraints by computing the (output) parameters of *one* object.

R-methods have as many equations as outputs, so that they compute a finite set of solutions for the output variables. (The dimension of the variety of the solutions is 0.) In addition, an r-method must be able to compute *all* the partial solutions of the involved equations. Indeed, this allows the backtracking phase to combine the partial solutions, computed by the different r-methods in the plan, without losing any solution (see Section 3.5). Note that local numerical methods cannot be used if one wants to guarantee to find an existing solution.

For obtaining fast routines able to find all the partial solutions, we made symbolic manipulations of the equations involved in r-methods. This is straightforward for linear equations, but generally not trivial for non-linear algebraic equation systems. In that case, 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: $a\,x^2 + b\,x + c = 0$.

3.3. Automatic addition of r-methods

This phase is essentially based on a simple subgraph isomorphism algorithm (i.e., subgraph matching) performed on the constraint graph. This identifies all the subgraphs in the constraint graph corresponding to an r-method in our dictionary. A found r-method is then added to the equation graph. For example, when this algorithm is applied to the constraint graph of Fig. 2, the equation graph is enriched with 16 r-methods. Only eight of them are depicted in Figure 3 for the sake of clarity. These r-methods match one of the three following patterns: line incident 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 parallel to another line (e.g., m_3, m_5). The subgraph made of nodes P_a, C_8, L_a, C_1, P_b matches a pattern in the dictionary and creates the r-method m_1. This phase is detailed in [13].

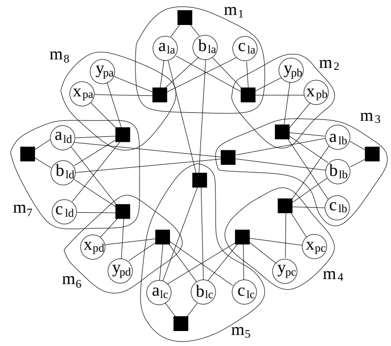


Figure 3. The equation graph of our didactic 2D scene enriched with automatically defined r-methods. Equations are represented by black rectangles and an r-method is represented by a hyper-arc including its equations and its output variables.

3.4. Computation of a sequence of r-methods

`GPDOF` [12] is a generalization of local propagation algorithms used to solve multiway dataflow constraints [10, 14]. It works on an enriched equation graph. It aims at selecting a sequence of r-methods to be executed for satisfying all the equations. `GPDOF` is an extension of the `PDOF` schema [10] (`PDOF` accepts only r-methods solving one equation). In short, `GPDOF` runs the two following steps until no more equation remains in the graph G (success) or no more free r-method is available (failure):

1. select a free r-method m (see Def. 1),
2. remove from G the equations satisfied by m; remove from G the output variables of m.

A plan can be obtained by reversing the selection order: the first selected r-method will be executed last. Iteratively selecting free r-methods ensures that no loop is created between the selected r-methods. Fig. 4 shows an example.

Note that, in case of failure, one obtains an incomplete plan which can solve only a subpart of the equations. In this case, more input parameters are (bundle) adjusted and not all the geometric constraints in the scene are taken into account.

Properties

In [12], it is proven that `GPDOF` guarantees to compute a sequence of r-methods (if one such sequence exists). In addition, `GPDOF` solves this combinatorial problem in polynomial-time. The complexity is a polynomial function of the number of equations, the maximum number of r-methods per equation and the maximum number of equations involved in an r-method [12]. In practice, it is quasi-linear.

Determination of input parameters

Obtaining the input parameters is a side-effect of `GPDOF`. The input parameters given to the bundle adjustment simply consist of the variables which are output of no r-method in

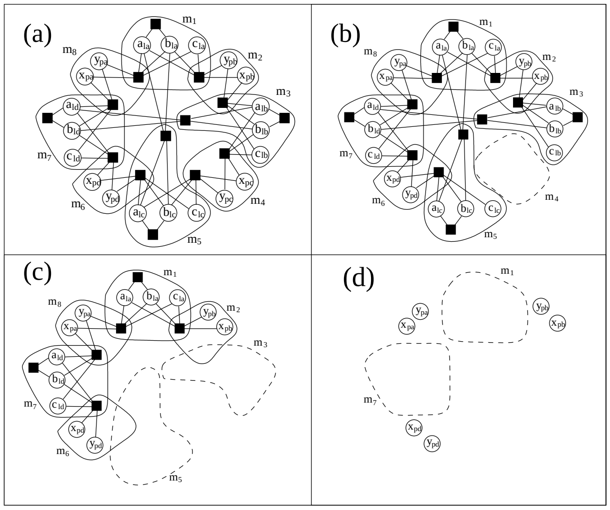


Figure 4. A planning phase performed by GPDOF on the didactic scene. **(a)** At the beginning, r-methods m_2, m_4, m_6, m_8 are free, so that one of them is selected, e.g., m_4. **(b)** This selection implies the removal of the equations and the output variables of m_4 from the equation graph. **(c)** This frees r-methods m_3 and m_5 which are selected and removed next in any order. **(d)** 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 $(m_1, m_7, m_3, m_5, m_4)$.

the plan. This yields the 6 coordinates of points P_a, P_b, P_d for the plan illustrated in Fig. 4.

3.5. Execution phase and model optimization

The model optimization requires several executions of the plan (the sequence of r-methods). Let us first explain how a plan is executed once.

The input parameters are first replaced by their current value in subsequent equation systems solved by r-methods[2]. Then, the r-methods in the sequence are executed one by one. Fig. 5 shows an example.

Note that, when an r-method solves non-linear equations, it generally produces several solutions for its output variables. For instance, a 3D point located at known distances from three other points can have two different positions (The intersection of 3 spheres gives two points.) This implies that several total solutions are generally obtained at the end of a plan execution process.

Therefore the model optimization is divided into two main steps (see Fig. 1).

First, a branch and bound (backtracking) process executes the plan *once* to compute all the possible solutions; the best path is stored: for every r-method computing several (partial) solutions, we store the solution "index" which

[2]At the beginning, the values of input parameters are set by the initial reconstruction step. Later, these values are modified by the bundle adjustment optimization process.

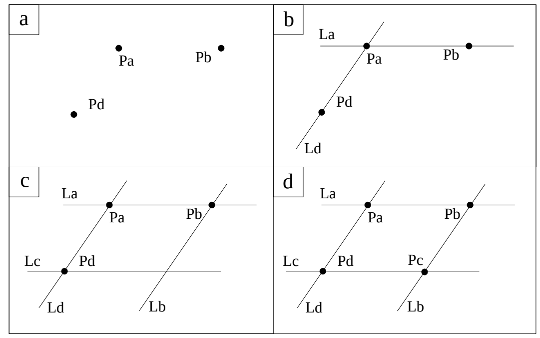


Figure 5. Execution of r-methods in the plan given in Fig. 4. **(a)** The input parameters are issued from one iteration of bundle adjustment: the corresponding points are placed first. **(b)** m_1 and m_7 compute L_a and L_d resp. **(c)** m_3 and m_5 compute L_b and L_c resp. **(d)** Finally, m_4 places point P_c.

leads to a total solution minimizing the cost function (reprojection errors). The backtracking algorithm used for this task is standard, and more sophisticated algorithms could be used instead [3].

Second, the minimization of the expression (2) is performed: a standard *bundle adjustment* is interleaved with the plan execution. Our bundle adjustment uses the Levenberg-Marquardt optimization method and adjusts only the input parameters. More precisely:

1. Every time new values are computed for the input parameters by a Levenberg-Marquardt step, the r-methods in the plan are executed, following the best path previously stored during the branch and bound.
2. Then, the cost function is updated taking into account the reprojection errors of all the variables (the input parameters and the other variables).
3. This process is re-iterated (goto 1).

3.6. Consequences of the use of GPDOF

This section presents the consequences of the additional work performed by GPDOF, as compared to PDOF. This work has been hidden until now for the sake of clarity.

Because the r-methods solve several equations, GPDOF may compute a plan whose r-methods solve a same equation several times or compute a variable several times. GPDOF is able to favor plans with no "overlap" of r-methods, but overlaps are sometimes required to guarantee that GPDOF finds a plan (if such a plan exists) [12].

For example, on the didactic scene, another sequence that could be computed by GPDOF is $(m_7, m_3, m_2, m_1, m_5, m_4, m_6)$. Note that several equations are solved twice in this plan, e.g., the incidence constraint between point P_b and line L_a. The main consequence for our model reconstruction tool is that the input parameters are now divided into two subsets. The first (classical) subset contains the variables which are ouptut of no r-method in the plan, e.g., the coordinates of point P_a. The values of these variables are only modified by optimization. A second subset comes from this overlap phenomenon. In the example, it contains the parameters of points P_b, P_c, P_d and lines L_a, L_b. The values of variables in this second subset are first issued from an optimization step, but are later modified again by an r-method execution. For instance, the P_b coordinates are considered input variables when executing r-method m_3 (to place line L_b), but are modified again by the execution of m_2. The values of the other variables, e.g., the coordinates of L_c, L_d, are modified only by r-method execution.

This sequence has been chosen to highlight this subtlety, but this phenomenon can be easily limited by simple heuristics. In the real model reconstruction presented below, the first subset of input parameters contains 171 variables, and the second subset only includes 10 variables. (The 246 remaining variables are computed by the plan execution.)

4. Results

We have used our approach to build a model of a church. A set of five images have been used, together with architectural plans from which several distance measurements have been extracted. Overall, 137 constraints (112 incidence, 15 parallelism and 10 distance constraints) are used to constrain 119 objects (91 points, 20 lines and 8 planes). This corresponds to 251 equations and 427 variables. Our r-method dictionary contains 60 r-methods. The most complex r-methods solve 3 geometric constraints (6 equations) and imply 4 geometric objects (1 as output and 3 as input).

Performance tests

The time for the initialization phase (see Fig. 1) is 12 s on a `Pentium IV 2 Ghz`. It is dominated by the quasi-linear reconstruction. The time for the planning phase is 2 min 40 s. The most time-consuming step in this phase is the automatic addition of r-methods. The execution time of `GPDOF` is negligible. The optimization phase requires 3 minutes and executes the plan of r-methods 1100 times (due to numerical differentiation). Solving an r-method is very fast and needs 35μs (average over 1000000 tries). The computation time for the r-method plan execution is 55 ms.

Reconstruction results

2213 r-methods have been added automatically to the equation graph. The r-method plan given by `GPDOF` is built of 107 r-methods. Our backtracking mechanism chooses the solution giving the smallest reprojection error.

Figure 6–(a) shows one of the five images used. The results obtained through the unconstrained bundle adjustment are presented in Figures 6–(b) and 6–(c). The model suffers from several artifacts: collinearity 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 6–(b). By imposing appropriate constraints, we have overcome these problems. Figure 7 shows the model produced using our method. We show how the parts of the model mentioned above have been corrected, leading to a visually correct model. Several artifacts are corrected after several plan executions, which highlights the interest of our optimization phase and of our fast plan execution (due to r-methods).

Note that a numerical *singularity* may occur during the execution of an r-method (for example if a plane is being built from 3 almost collinear points). Our solution makes use of the local aspect of an r-method: a precondition on the values of input variables is checked and well-conditioned solutions are favored.

5. 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: input parameters are extracted in polynomial-time and a sequence of fast r-methods (which take into account geometrical properties) can build a model that satisfies the constraints exactly.

Our system has been validated on a model containing 119 primitives and 137 geometric constraints, 10 of them being quadratic. The obtained results are geometrically correct and fit well the images. We intend to validate our approach on larger models, and compare it with concurrent methods where constraints are introduced as soft constraints in the cost function to be optimized (see Section 2).

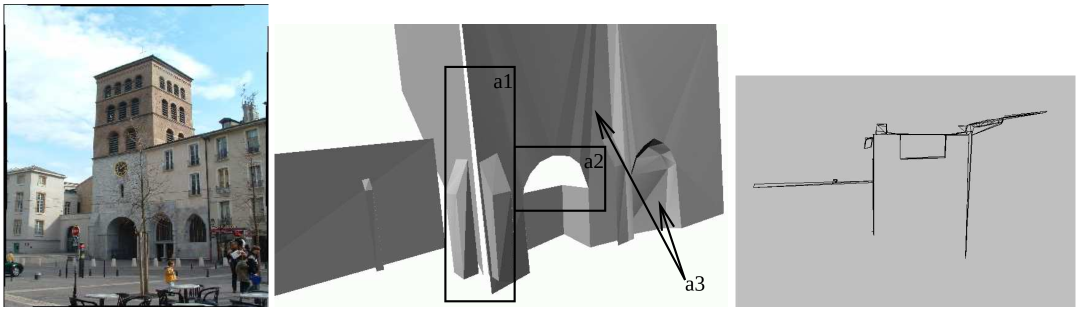
Due to the complexity of the problem, several challenging issues must still be tackled.

The automatic addition of r-methods is a costly phase in our process. We believe that more sophisticated combinatorial techniques can radically improve the performance of this phase.

Redundant constraints involve non-independent equations. They 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, in addition to standard approaches, special r-methods can be used in a preprocessing step to remove a lot of occurring redundancies.

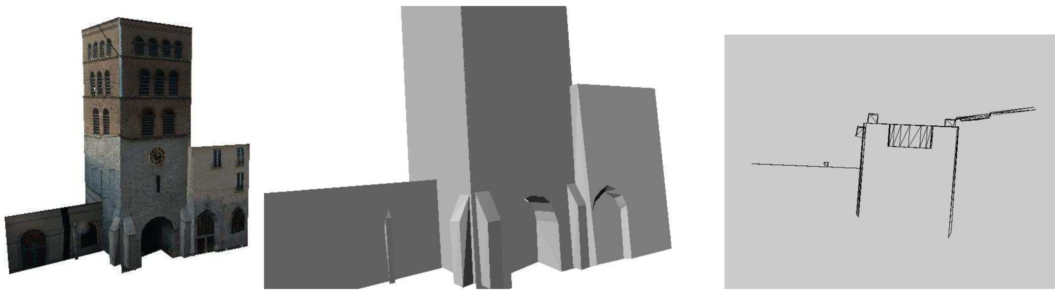
Acknowledgments

Thanks to D. Chancel for the architectural drawings. Thanks to D. Daney, B. Neveu for useful discussions and helpful comments on this paper.



(a) (b) (c)

Figure 6. (a) One of the five photos used for the reconstruction; (b) Some artifacts of the unconstrained model; ($a1$) The collinearity 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 (from bottom) obtained through the underconstrained bundle adjustment; not all expected parallelism constraints are respected.



(a) (b) (c)

Figure 7. (a) The overview of the constrained model. (b) The details of the optimized model. (c) The orthographic view of the optimized model.

References

[1] A. Bartoli and P. Sturm. Constrained structure and motion from N views of a piecewise planar scene. In *Proc. Int. Symposium on Virtual and Augmented Architecture, VAA'01, Dublin, Ireland*, pages 195–206, 2001.

[2] P.-L. Bazin. A parametric scene reduction algorithm from geometric relations. In *Proceedings of Vision Geometry IX, SPIE's 45th annual meeting, San Diego*, June 2000.

[3] C. Bliek, B. Neveu, and G. Trombettoni. Using Graph Decomposition for Solving Continuous CSPs. In *Int. Conf. on Constraint Programming, CP'98*, volume 1520 of *LNCS*, pages 102–116, 1998.

[4] D. Bondyfalat, B. Mourrain, and T. Papadopoulo. An application of automatic theorem proving in computer vision. In *Automated Deduction in Geometry*, pages 207–231, 1998.

[5] P. Debevec, C. Taylor, and J. Malik. Modeling and rendering architecture from photographs: a hybrid geometry-and image-based approach. In SIGGRAPH'96, 1996.

[6] A. Dick, P. H. S. Torr, S. Ruffle, and R. Cipolla. Combining single view recognition and multiple view stereo for architectural scenes. In *ICCV'01*, pages 268–274, 2001.

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

[8] P. Poulin, M. Ouimet, and M.-C. Frasson. Interactively modeling with photogrammetry. In *Eurographics Workshop on Rendering*, pages 93–104, 1998.

[9] P. Sturm and S. Maybank. A method for interactive 3D reconstruction of piecewise planar objects from single images. In *BMVC'99*, pages 265–274, 1999.

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

[11] R. Szeliski and P. Torr. Geometrically constrained structure from motion : Points on planes. In *3D Structure from Multiple Images of Large-scale Environments SMILE'98*. Springer Verlag, June 1998.

[12] G. Trombettoni. A Polynomial Time Local Propagation Algorithm for General Dataflow Constraint Problems. In *Int. Conf. on Constraint Programming CP'98, LNCS 1520*, pages 432–446, 1998.

[13] G. Trombettoni and M. Wilczkowiak. Scene reconstruction based on constraints: Details on the equation system decomposition. In *Int. Conf. on Constraint Programming, CP'03, LNCS*, 2003.

[14] B. Vander Zanden. An incremental algorithm for satisfying hierarchies of multi-way, dataflow constraints. *ACM Trans. on Prog. Languages and Systems*, 18(1):30–72, 1996.

[15] M. Wilczkowiak, E. Boyer, and P. Sturm. 3D modelling using geometric constraints: A parallelepiped based approach. In *ECCV'02, LNCS 2353*, pages 221–236, 2002.

[16] M. Wilczkowiak, P. Sturm, and E. Boyer. The analysis of ambiguous solutions in linear systems and its application to computer vision. In *BMVC'03*, 2003.

[17] C. Zeller. *Calibration projective, affi ne et euclidienne en vision par ordinateur et application à la perception tridimensionnelle*. Thèse de doctorat, École Polytechnique, 1996.