GPDOF : un algorithme polynomial et complet pour décomposer les systèmes de contraintes géométriques

Gilles Trombettoni

Projet COPRIN, I3S-INRIA, Université de Nice Sophia Antipolis,
2004 route des lucioles, 06902 Sophia Antipolis cedex, B.P. 93, France
trombe@sophia.inria.fr

Résumé : *Cet article a pour but de présenter l'algorithme* GPDOF *à la communauté française travaillant sur les contraintes géométriques.* GPDOF *sait décomposer un système de contraintes géométriques en une séquence de sous-systèmes solubles très rapidement. L'approche repose sur un dictionnaire de r-méthodes qui correspondent à des théorèmes de géométrie. Appliqué à un système sous-contraint, et à condition que ce système ne contienne pas de contraintes redondantes,* GPDOF *peut trouver, en un temps polynomial et de manière complète, un ensemble de paramètres d'entrée et une séquence de r-méthodes résolvant toutes les équations.*

L'article décrit l'algorithme, montre son application à un problème de reconstruction de scènes en vision où des centaines de contraintes bilinéaires et quadratiques sont résolues en quelques secondes, et discute de ses limites. Une brève comparaison est faite avec d'autres algorithmes de décomposition de systèmes d'équations et avec des résolveurs de contraintes géométriques.

Mots-clés : Contraintes géométriques, décomposition de systèmes d'équations

1 Introduction

GPDOF date de 1997 où il apparaît dans ma thèse [17] et a été publié dans la conférence de programmation par contraintes CP'98 [18]. Depuis la fin 2002, un travail de collaboration avec Marta Wilczkowiak, en thèse à l'INRIA Rhône-Alpes, a permis de l'appliquer à un problème de reconstruction de scènes architecturales. Ces travaux ont été publiés dans la conférence de programmation par contraintes CP'2003 [19] dont l'article est donné en section A (Marta Wilczkowiak en est co-auteur) et aussi dans la conférence de vision ICCV'2003 [22]. Une version plus détaillée apparaitra dans la thèse de Marta Wilczkowiak et vient d'être soumise à une revue spécialisée en vision par ordinateur.

Ces travaux valident l'intérêt de GPDOF pour la décomposition de contraintes géométriques. Les pages qui suivent ont pour but de présenter l'algorithme à la communauté des contraintes géométriques. La section 2 présente rapidement l'algorithme et résume ses différentes propriétés. La section 3 compare GPDOF à d'autres algorithmes de décomposition (d'équations ou de systèmes de contraintes géométriques). La section A reprend l'article en anglais publié à la conférence CP'2003 et résumant l'application de GPDOF au problème de reconstruction de scènes.

2 Description de GPDOF

GPDOF est l'acronyme de "General Propagation of Degrees of Freedom". GPDOF est une généralisation de l'algorithme PDOF conçu par Sutherland dans son système interactif SketchPad [16].

2.1 Entrée

L'algorithme de décomposition de contraintes géométriques basé sur GPDOF prend en entrée :

- Un système d'équations représenté par un graphe de dépendances entre les variables et les équations appelé *graphe d'équations*. Ce système d'équations est généralement sous-

contraint, c.a.d., possède plus de variables que d'équations. Chaque variable a une valeur courante.
- Un dictionnaire de *r-méthodes* correspondant à des méthodes à la règle et au compas ou plus généralement à des théorèmes de géométrie.

Par exemple, une r-méthode calcule les coordonnées d'une droite en fonction de la position courante de deux points incidents à cette droite. Une autre r-méthode calcule les 0, 1 ou 2 positions d'un point P_1 en 3D placé à distances connues de 3 autres points P_2, P_3 et P_4 (intersection de 3 sphères). Une r-méthode est une procédure dont l'exécution satisfait un sous-ensemble des équations du système (comme les 3 distances dans l'exemple ci-dessus) en calculant une ou plusieurs valeurs pour ses *variables de sortie* (e.g., les coordonnées de P_1) en fonction de la valeur courante des autres variables, dites d'entrée, impliquées dans les équations.

Definition 1 *Soit un système d'équations E portant sur un ensemble de variables V possédant chacune une valeur réelle courante.*

Une **r-méthode** *m porte sur un ensemble d'équations $E_m \subset E$, un ensemble non vide de* **variables de sortie** *$V_m^{out} \subset V$ et un ensemble de* **variables d'entrée** *$V_m^{in} \subset V$. ($V_m^{in} \cup V_m^{out}$ constitue l'ensemble des variables impliquées dans une ou plusieurs équations de E_m.)*

*L'***exécution** *de la r-méthode m remplace V_m^{in} par leur valeur $\overline{V_m^{in}}$ et fournit toutes les solutions S_m^{out} pour V_m^{out} satisfaisant E_m.*

La r-méthode m est **libre** si aucune variable v de V_m^{out} n'est impliquée dans une équation de $E \setminus E_m$. Ainsi, exécuter une r-méthode libre ne peut pas violer d'autres équations de $E \setminus E_m$.

Le système de contraintes géométriques et le système d'équations sont représentés resp. par un *graphe de contraintes* et un *graphe d'équations*. La figure 2 montre un exemple de ces graphes pour un système de contraintes géométriques (incidences, parallélismes) portant sur des points et des droites en 2D et qui modélisent un parallélogramme.

Definition 2 *Un* **graphe de contraintes** *est un graphe biparti où les sommets sont les contraintes et les objets (représentés par des rectangles et des cercles resp.). Chaque contrainte est connectée par un arête à ses objets.*
Un **graphe d'équations** *est un graphe biparti (V,E,A) où les sommets sont les équations de E et les variables de V (représentés par des rectangles et des cercles resp.) Chaque équation est connectée à ses variables par une arête de A.*
Un **graphe d'équations enrichi** *(V,E,A,M) est un graphe d'équations (V,E,A) enrichi par un ensemble M de r-méthodes.*

2.2 Sortie

L'algorithme de décomposition de contraintes géométriques basé sur `GPDOF` procède en deux phases :

1. Un premier algorithme utilise le graphe de contraintes et le dictionnaire de r-méthodes pour enrichir le graphe d'équations.
2. Basé sur le graphe d'équations enrichi, `GPDOF` fournit des *paramètres d'entrée* (i.e., un sous-ensemble des variables de V) et une séquence de r-méthodes $(m_1,...,m_i,...,m_r)$ appelé *plan*.

Pour un jeu de valeurs données pour les paramètres d'entrée, l'exécution des r-méthodes du plan en séquence fournit une ou plusieurs solutions satisfaisant toutes les équations du système.

La partie gauche de la figure 3 montre le déroulement de l'algorithme `GPDOF` sur notre exemple de parallélogramme.

2.3 Principe de GPDOF

Jusqu'à ce qu'il ne reste plus d'équations dans le graphe d'équations G (succès) ou qu'il n'y a plus de r-méthode libre (échec)[1] faire :

1. sélectionner une r-méthode *libre* m,
2. enlever de G les équations de m et les variables de sortie de m,
3. créer toutes les *sous-méthodes* des r-méthodes m_i qui partagent des équations ou des variables de sortie avec m.

Un plan s'obtient en inversant l'ordre de sélection : la r-méthode sélectionnée en premier sera exécutée en dernier. Les deux premières étapes ci-dessus définissent l'algorithme standard PDOF (qui n'accepte que des r-méthodes avec une équation et une variable de sortie). Sélectionner itérativement des r-méthodes qui sont libres assurent que le plan ne contient pas de circuit.

En théorie, il peut se produire que tous les plans possibles contiennent au moins deux r-méthodes qui se *recouvrent*, c.a.d., deux r-méthodes qui partagent les mêmes contraintes ou les mêmes variables de sortie[2]. Autrement dit, si la sélection de r-méthodes qui se recouvrent n'était pas autorisée, rien n'assurerait que GPDOF trouve un plan s'il en existe un. Un plan qui contient des r-méthodes se recouvrant signifie que des équations seront résolues plusieurs fois pendant l'exécution du plan.

GPDOF doit ainsi être capable de sélectionner des r-méthodes qui se recouvrent pour garantir la complétude. La notion de *sous-méthode* introduite à l'étape 3 de GPDOF ci-dessus a précisément cet objectif. Brièvement, la notion de sous-méthode explique que, à une itération donnée de GPDOF, une r-méthode partiellement enlevée par l'étape 2 (c.a.d., certaines contraintes de la r-méthode sont enlevées alors que d'autres non), reste candidate pour une prochaine sélection. Cette notion est détaillée dans [18] et la partie droite de la figure 3 montre un exemple.

Ce phénomène, qui peut paraître étrange, n'a que peu d'incidence sur la résolution du système. Il peut de plus être limité par une heuristique qui favorise la sélection de r-méthodes "classiques" plutôt que des sous-méthodes (quand c'est possible). Notons également que la notion de sous-méthode sert seulement au travail structurel de GPDOF (qui est un algorithme de graphe) et que ce sont bien les r-méthodes "entières" correspondant aux sous-méthodes qui seront exécutées par la suite.

2.4 Calcul des paramètres d'entrée

Rappelons que nous travaillons sur un système sous-contraint, c.a.d. que le nombre de variables excède celui des contraintes. Il faut donc pouvoir choisir un sous-ensemble des variables pour en faire en quelque sorte des constantes. Ces variables sont appelées *paramètres d'entrée*. En pratique, les valeurs de ces paramètres sont fournies par l'utilisateur, une esquisse (dessin) ou un processus antérieur (reconstruction de scènes).

GPDOF calcule le plan en ordre inverse et obtenir les paramètres d'entrée est un effet de bord de cet algorithme. Sans détailler, les paramètres d'entrée se répartissent en deux sous-ensembles disjoints $\mathcal{P}_1$ et $\mathcal{P}_2$: l'ensemble $\mathcal{P}_1$ contient les variables qui ne sont en sortie d'aucune r-méthode du plan (par exemple, les 6 coordonnées des points P_a, P_b et P_d pour le plan de la figure 3 (gauche) ; l'ensemble $\mathcal{P}_2$ vient en fait du phénomène de recouvrement des r-méthodes (sous-méthodes). En pratique, l'ensemble $\mathcal{P}_2$ semble petit.

Quand aucune sous-méthode n'est sélectionnée dans le plan, l'ensemble $\mathcal{P}_2$ est vide. Plus précisément, on peut montrer que plus GPDOF sélectionne de sous-méthodes, plus important sera l'ensemble $\mathcal{P}_2$.

1. On obtient alors un plan incomplet qui résout un sous-ensemble des équations seulement.
2. C'est en fait généralement le cas en pratique dès que le système devient suffisamment important.

2.5 Propriétés, attraits et limites de GPDOF

On a montré dans [18] que GPDOF est polynomial en fonction des paramètres du graphe d'équations enrichi : le nombre de variables, le nombre d'équations et le nombre maximum de r-méthodes par équation. Notre application en reconstruction de scènes a montré que GPDOF est quasi-linéaire en pratique et que son temps d'exécution est négligeable pour des systèmes de quelques centaines d'équations. Le temps de l'enrichisssement du graphe d'équations (préalable à l'application de GPDOF) est de quelques secondes quand les r-méthodes du dictionnaire ne portent que sur 1 à 5 contraintes géométriques (c.a.d. ne modifient qu'un seul objet géométrique).

On peut également montrer que, sous réserve que *le système d'équations ne contient pas de contraintes redondantes (c.a.d. contient seulement des équations indépendantes)*, alors GPDOF est complet : GPDOF peut nécessairement trouver un plan s'il en existe un [18].

Il faut donc que l'utilisateur crée des systèmes sans équations dépendantes (ce qui est irréaliste en pratique) ou qu'une procédure automatique traite le système et enlève ces redondances. Des procédures simples ont été ajoutées à l'outil de reconstruction de scènes pour enlever des redondances triviales (par exemple : incidence d'un point P à une droite D, incidence de D à un plan Pl et incidence de P à Pl...). Des approches plus robustes pourraient s'inspirer de travaux récents en CAO [14, 13].

Les équations redondantes peuvent empêcher en fait GPDOF de trouver une r-méthode libre et l'analyse du graphe d'équations au moment où GPDOF échoue peut alors nous permettre de localiser le sous-système dépendant.

L'autre difficulté tient aux singularités qui peuvent se produire pendant l'exécution d'une r-méthode. Par exemple, si une r-méthode cherche à placer un plan incident à 3 points donnés alors que ces 3 points sont quasiment alignés. La thèse de Marta Wilczkowiak détaille des moyens pratiques pour (a) travailler sur la modélisation des r-méthodes et éviter qu'elles ne soient singulières (b) ne pas sélectionner de telles r-méthodes pour la construction du plan. Il faut souligner que l'aspect local des r-méthodes présente l'avantage de localiser très rapidement une telle singularité et d'en cerner les causes.

3 Comparaison de GPDOF avec d'autres algorithmes de décomposition

Il existe de nombreux algorithmes de décomposition de contraintes géométriques qui ont été utilisés en CAO, en dessin à l'aide d'esquisse, en vision, etc. Certains travaillent au niveau géométrique (en utilisant les degrés de liberté des contraintes ou des objets par exemple), d'autres au niveau équationnel. Cette section donne une rapide comparaison de ces algorithmes avec GPDOF.

Quoique très efficaces, certains algorithmes ne sont pas vraiment comparables à GPDOF dans la mesure où ils sont limités à un ensemble restreint de contraintes ou d'objets géométriques. Dans cette catégorie, on peut classer la méthode de Owen [11] (points et droites en 2D soumis à des contraintes d'angle et de distance) ou la méthode de Sunde [15] étendue par Verroust dans [20] (points et segments en 2D soumis à des contraintes d'angle et de distance).

Plusieurs algorithmes sont basés sur la rigidification récursive du système de contraintes (détecter des sous-systèmes rigides et utiliser cette propriété pour calculer les positions relatives des différents composants...). On peut citer notamment les travaux de Kramer [9] et de Hoffmann [5, 7] dans ce domaine (voir aussi [8]). Ces algorithmes ne sont pas adaptés au traitement de systèmes sous-contraints (ayant plus de 6 degrés de liberté en fait). En effet, ils ne savent pas décomposer les sous-parties qui ne sont pas rigides, ni calculer les paramètres d'entrée.

Certains algorithmes sont réactifs, c.a.d., sélectionnent un sous-système à résoudre et le résolvent à la volée. Il est montré dans [17] que l'on ne peut pas garantir l'obtention d'un plan, s'il en existe

un, sans rendre l'algorithme réactif exponentiel (ce qui n'est pas le cas de `GPDOF`). L'algorithme de Bondyfalat et al. [2] a été appliqué également au problème de reconstruction de scènes et est effectivement exponentiel dans le pire des cas (quoique des heuristiques pertinentes limitent souvent l'explosion combinatoire).

Il existe d'autres algorithmes de planification (capables de calculer une séquence de sous-systèmes à résoudre) qui fonctionnent au niveau du système d'équations et non pas au niveau géométrique. Parmi eux, un seul algorithme présente un intérêt réel par rapport à `GPDOF` et est détaillé ci-dessous. Il s'agit du célèbre algorithme de graphe du *couplage maximum* (en anglais : Maximum-Matching) appliqué au graphe d'équations. Cette approche a été retenue (a) par des chercheurs de la communauté de *propagation locale* pour les systèmes interactifs [6], (b) pour décomposer de très grands systèmes d'équations linéaires [4, 12], et enfin (c) pour résoudre des contraintes géométriques [10].

Nous reprenons l'exemple du parallélogramme pour souligner les difficultés liées à cet algorithme.

Un *couplage maximum* d'un graphe biparti est un sous-ensemble d'arêtes de cardinalité maximum tel que toute paire d'arêtes du couplage ne partage pas de sommet. Brièvement, appliquer un algorithme de couplage maximum à un graphe d'équations fournit :

- une séquence de sous-systèmes d'équations (une arête du couplage indique une équation et une variable traitées dans le sous-système correspondant) ;
- un ensemble de paramètres d'entrée (les variables qui ne sont pas incidentes à une arête du couplage).

Cet algorithme est également quasi-linéaire en pratique et n'a en outre pas besoin d'un dictionnaire de r-méthodes.

En revanche, il est encore plus sensible que `GPDOF` à des contraintes redondantes introduites dans le système (car il s'agit d'un algorithme de graphe pur). Il peut même produire des sous-systèmes géométriquement incorrects[3], ce qui n'est pas le cas de `GPDOF` dont la sémantique des r-méthodes est clairement définie. De plus, le couplage maximum peut induire des sous-systèmes de taille arbitraire pour lesquels aucun algorithme (rapide) n'est connu. La figure 1 montre que le couplage maximum pourrait mettre toutes les contraintes de notre exemple dans un seul sous-système.

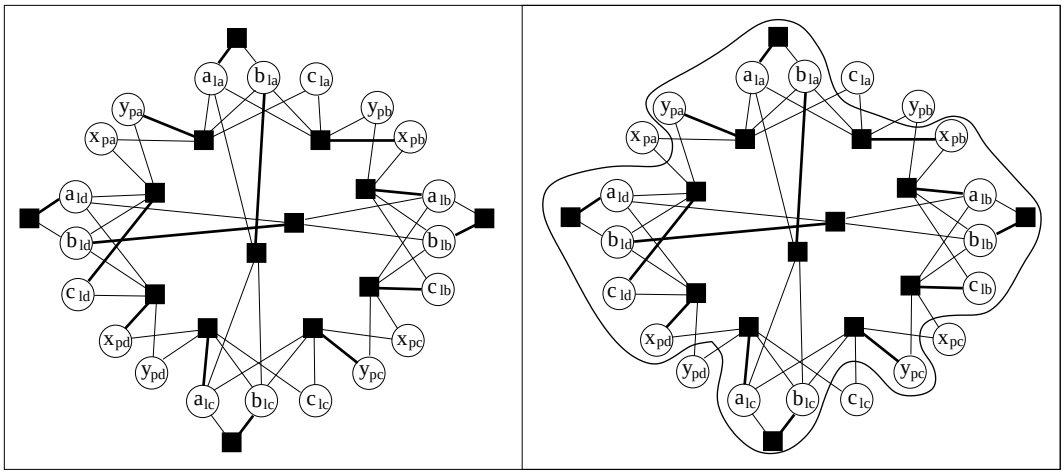


FIG. 1 – *Application du* `couplage maximum` *à notre exemple.* **A gauche :** *Les arêtes du couplage sont en gras (le couplage est maximum puisque toutes les contraintes sont couplées).* **A droite :** *Tout le système est placé dans une seule composante fortement connexe et aucune méthode générale de résolution ne permet de résoudre rapidement de tels systèmes d'équations bilinéaires.*

3. Un mauvais choix de "variables d'entrée" peut conduire à un système d'équations contradictoires (voir [18]).

4 Conclusion

L'algorithme GPDOF est un algorithme très prometteur pour la résolution rapide de systèmes de contraintes géométriques (linéaires ou non-linéaires). L'algorithme est en effet très rapide et garantit de trouver une séquence de r-méthodes s'il en existe une. Les r-méthodes sont des procédures câblées qui permettent de résoudre un sous-système d'équations très rapidement. GPDOF se compare favorablement à d'autres algorithmes de décomposition de systèmes de contraintes géométriques et à l'algorithme de couplage maximum.

Deux conditions sont nécessaires pour appliquer GPDOF en pratique. Il faut d'abord construire un dictionnaire de r-méthodes adaptées aux types d'objets et de contraintes que l'on veut admettre dans notre système. Il faut ensuite prévoir des procédures capables d'enlever un maximum de contraintes redondantes que l'utilisateur pourrait introduire malencontreusement dans le système.

A Application de GPDOF à la reconstruction de scènes

Title : Scene Reconstruction Based on Constraints: Details on the Equation System Decomposition

Auteurs : Gilles Trombettoni et Marta Wilczkowiak

E-mail : Marta.Wilczkowiak@inrialpes.fr

Résumé : *W*e present a new approach to 3D scene modeling based on geometrical constraints. Contrary to most of the existing methods, we obtain 3D scene models that respect the given constraints *exactly*. Our tool can describe a large variety of linear and non-linear constraints in a flexible way.

Our approach is based on a dictionary of so-called *r-methods*, based on theorems of geometry, which can solve a subset of geometrical constraints in a very efficient way. Two fast and complete graph-based algorithms are proposed to find a reduced parameterization of a scene, and to decompose the equation system in a sequence of r-methods.

B Introduction

Reconstruction of accurate and photorealistic 3D models is one of the most challenging tasks in Computer Vision. In this paper, we address the problem of image-based reconstruction of a scene respecting a set of geometrical constraints. Defining geometrical 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.

A common approach consists in incorporating the constraints into the optimization process. These methods however are often costly. Furthermore, they guarantee neither the convergence nor the (exact) constraint satisfaction.

Our model acquisition approach is detailed in [22]. It is divided into three main phases: initialization, constraint planning and optimization.

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**, **incidence**, **parallelism** and **orthogonality**.

An initial reconstruction is provided by a quasi-linear approach exploiting projections and geometrical constraints [21]. After this phase, all the variables (camera and model parameters)

have an initial value.

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 [18] is a predefined routine used to solve a subset 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. For example, an r-method computes the parameters of a line based on the current position 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 is divided into two steps:

1. *R-method addition phase:* Add *automatically* in the equation graph all the r-methods corresponding to r-method patterns present in the dictionary.
2. *Planning phase:* Perform `GPDOF` [18][4] 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. Input parameters are a subset of the variables describing the scene such that, when a value is given to them, there exists a finite set of solutions for the system satisfying the constraints.

Model optimization

The optimization process[5] 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 are satisfied. The detailed process can be found in [22].

Contribution

Many works have focused on incorporating geometrical constraints for camera calibration and 3D reconstruction including [3, 2]. The reader will refer to [22] for more details on the existing approaches which often require costly computations or do not guarantee to provide a solution. The approach presented in this paper overcomes these drawbacks. It is complete, fast and can be used to model non-linear constraints like distances, angles and distance/angle ratios.

This paper focus on the constraint planning process (Section C) and shows experimental results in Section D.

C Constraint Planning

This section details the algorithms necessary for the constraint planning.

C.1 Automatic addition of r-methods

The automatic addition of r-methods is essentially based on a simple subgraph isomorphism algorithm performed on the constraint graph. When a subgraph matches an entry in the dictio-

4. `GPDOF` stands for General Propagation of Degrees of Freedom.
5. based on a standard numerical algorithm and minimizing the *reprojection errors*

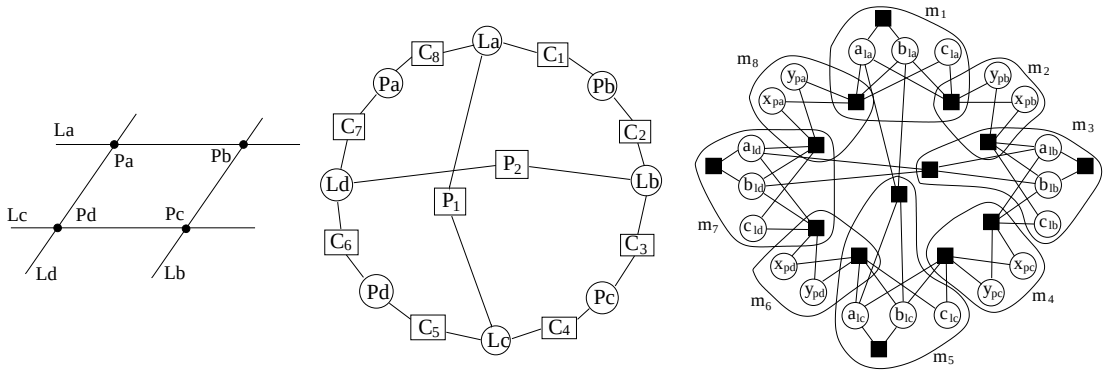


FIG. 2 – **Left:** *A didactic 2D scene describing a parallelogram in terms of lines, points, incidence constraints and parallelism constraints.* **Center:** *The corresponding constraint graph. It contains 4 points P_a,...,P_d, 4 lines L_a,...,L_d, 8 incidence constraints C_1,...,C_8 and 2 parallelism constraints P_1, P_2.* **Right:** *The enriched equation graph after automatic addition of r-methods. Equations are represented by rectangles and variables by circles. An r-method is represented by a hyper-arc including equations and output variables. Only 8 of the 16 r-methods are depicted 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 (m_2, m_4, m_6, m_8); line passing through a known point and parallel to another line (m_3, m_5).*

nary, the corresponding r-methods are added to the equation graph. Two steps are performed:

1. The first step explores all the connected subgraphs with size at most a small value k equal to the maximum number of nodes (objects+constraints) implied in any r-method of the dictionary, e.g., 7 in our tool. Starting from every 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]. The key idea allowing the algorithm to explore a *tree* of subgraphs is to consider at each step only a specific subset of selected neighbors, depending on a unique numbering of the nodes [1].
   In practice, the time complexity of this algorithm is linear in the actual number of connected subgraphs of size less than k (which is $O(n^k)$). It is acceptable for small values of k and sparse graphs.
2. For every found subgraph, a second procedure compares it with the subgraph patterns in our dictionary implemented as a hash table, which eliminates most of the subgraphs. A final comparison is made by a combinatorial process[6] inspired by the solving process of CSPs (BT). In short, objects in the subgraph are reordered to be matched with objects in a subgraph pattern. If the subgraph matches, the corresponding r-methods are added to the equation graph.

C.2 The GPDOF algorithm

GPDOF [18] works on an enriched equation graph. It computes a sequence of r-methods to be executed for satisfying all the equations. GPDOF solves this combinatorial problem in polynomial-time and is quasi-linear in practice. It performs the three following steps until no more equation remains in the equation graph G (success) or no more free r-method is available (failure)[7]:

1. select a **free** r-method m[8],

6. Deciding whether two graphs are *isomorphic* is still an open problem.
7. In this case, one obtains an incomplete plan which solves only a subpart of the equations (geometric constraints) and more parameters are adjusted by optimization.
8. Output variables of a free r-method appear in no "external" equations.

2. remove from G the equations and 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 above define the standard PDOF local propagation algorithm [16] on which GPDOF is based (PDOF accepts only r-methods solving *one* equation.) Selecting iteratively free r-methods ensures that no loop is created in the plan.

It turns out that, when r-methods can solve several equations, there is no guarantee that PDOF finds a plan, even if one exists. This highlights the notion of *submethod* which renders GPDOF complete. In short, the notion of submethod explains that a partially removed r-method remains available for a future selection. The reader will refer to [18] to get a more detailed information. Figure 3 shows an example.

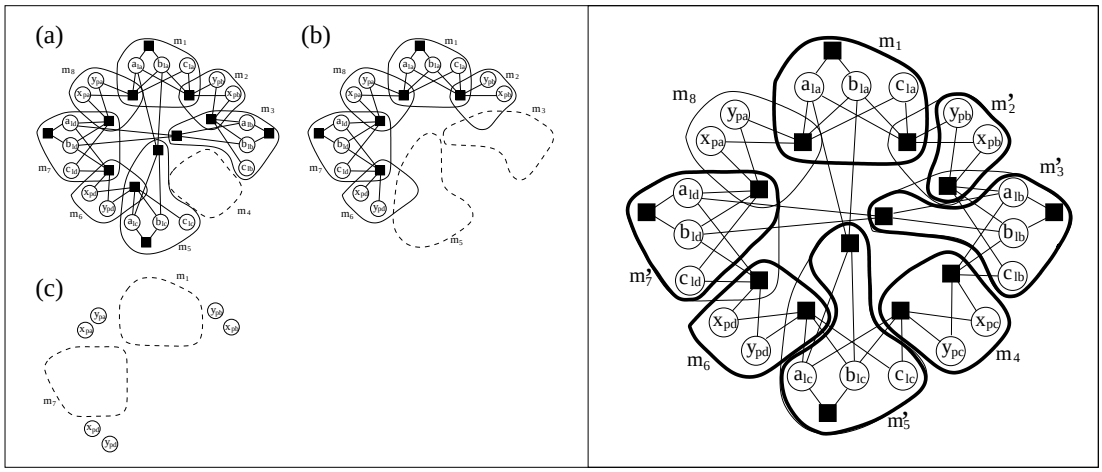


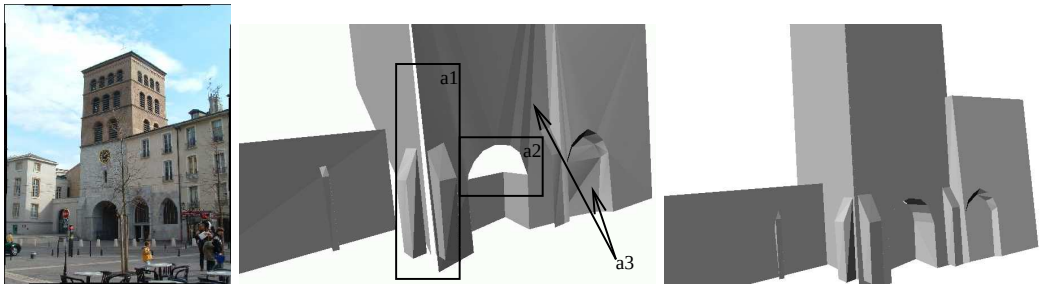
FIG. 3 – *Two possible planning phases performed by GPDOF on the didactic scene.* **Left:** *At the beginning, r-methods m_2, m_4, m_6, m_8 are free, so that one of them is selected, e.g., m_4.* **(a)** *This selection implies the removal of the equations and the output variables of m_4 from the equation graph.* **(b)** *This frees r-methods m_3 and m_5 which are selected and removed next in any order.* **(c)** *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).* **Right:** GPDOF *may also 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. The process continues and selects m_4, m'_5, m_1, m'_2, m'_3, and finally m'_7. Selected r-methods (m_1, m_4, m_6) and submethods (m'_2, m'_3, m'_5, m'_7) are represented by thick hyper-arcs.*

C.3 Determining the input parameters

The input parameters modified by the numerical optimization simply consist of the variables which are output by no r-method in the plan. This yields the 6 coordinates of points P_a, P_b, P_d for the plan illustrated in Fig. 3-left- or the 2 coordinates of point P_a for the plan illustrated in Fig. 3-right-. Due to the selection of submethods, the values of variables in a second set of parameters are read a first time (recall that every variable has an initial value) and computed later by r-methods, e.g., the coordinates of P_b are in this set (Fig. 3-right-). The other variables are only modified by r-method execution. This subtlety cannot be explained here due to a lack of space.

D Results

We have used our approach to build a model of a church (see Figure 4). Five images architectural plans (distances) were used. The scene includes 137 constraints (including 10 distances), 251 equations, 119 objects, 427 variables. 2213 r-methods have been added automatically. The constraint planning requires a few seconds. The execution time of `GPDOF` is negligible. The plan was built of 107 r-methods and is executed in 55 ms.



(a) (b) (c)

FIG. 4 – *(a)–One of the five photos used for the reconstruction; (b)–Some artifacts of the unconstrained model. (c)–The constrained model after optimization corrects the artifacts.*

N.B. : depuis, une version étendue de cette scène a été construite comprenant 444 équations et des temps d'exécution en proportion...

E Acknowlegments

Thanks to D. Chancel for the architectural drawings. Thanks to E. Boyer, D. Daney, C. Jermann, B. Neveu and P. Sturm for useful discussions.

Références

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

[2] Didier Bondyfalat, Bernard Mourrain, and Theodore Papadopoulo. An application of automatic theorem proving in computer vision. In *Automated Deduction in Geometry*, pages 207–231, 1999.

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

[4] A.L. Dulmage and N.S. Mendelsohn. Covering of bipartite graphs. *Canad. J. Math.*, 10:517–534, 1958.

[5] I. Fudos and C.M. Hoffmann. A graph-constructive approach to solving systems of geometric constraints. *ACM Transactions on Graphics*, 16(2):179–216, 1997.

[6] Michel Gangnet and Burton Rosenberg. Constraint programming and graph algorithms. In *Second International Symposium on Artificial Intelligence and Mathematics*, 1992.

[7] C.M. Hoffmann, A. Lomonosov, and M. Sitharam. Finding solvable subsets of constraint graphs. In *Principles and Practice of Constraint Programming CP'97*, pages 463–477, 1997.

[8] C. Jermann, B. Neveu, and G. Trombettoni. Algorithms for Identifying Rigid Subsystems in Geometric Constraint Systems. In *Proc.of IJCAI'03, International Joint Conference on Artificial Intelligence*, pages 233–238, 2003.

[9] Glenn Kramer. *Solving Geometric Constraint Systems.* MIT Press, 1992.

[10] Hervé Lamure and Dominique Michelucci. Qualitative study of geometric constraints. In Beat Brüderlin and Dieter Roller, editors, *Workshop on Geometric Constraint Solving and Applications*, pages 134–145, Technical University of Ilmenau, Germany, 1997.

[11] J. Owen. Algebraic solution for geometry from dimensional constraints. In *Proceedings of the 1st ACM Symposium on Solid Modeling and CAD/CAM Applications*, pages 397–407. ACM Press, 1991.

[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] A. Sosnov. *Modélisation Géométrique par Séparation de Contraintes*. Thèse de doctorat, Université de Nantes, 2003.

[14] A. Sosnov and P. Macé. Rapid algebraic resolution of 3d geometric contraints and control of their consistency. In *Fourth International Workshop on Automated Deduction in Geometry (ADG'02)*, 2002.

[15] G. Sunde. Specification of shapes by dimensions and other geometric constraints. In *IFIP WG 5.2 Geometric Modeling*, 1986.

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

[17] Gilles Trombettoni. *Algorithmes de maintien de solution par propagation locale pour les systèmes de contraintes*. Thèse de doctorat, Université de Nice–Sophia Antipolis, 1997.

[18] 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.

[19] Gilles Trombettoni and Marta Wilczkowiak. Scene Reconstruction based on Constraints: Details on the Equation System Decomposition. In *Proc. International Conference on Constraint Programming, CP'03*, volume 2833 of *LNCS*, pages 956–961, Kinsale,Ireland, 2003. Springer.

[20] A. Verroust, F. Schonek, and D. Roller. Rule oriented method for parametrized computer aided design. *Computer Aided Design*, 24(6):531–540, 1992.

[21] M. Wilczkowiak, P. Sturm, and E. Boyer. The analysis of ambiguous solutions in linear systems and its application to computer vision. In *Proceedings of the 14th British Machine Vision Conference, Norwich, England*, September 2003.

[22] M. Wilczkowiak, G. Trombettoni, C. Jermann, P. Sturm, and E. Boyer. Scene reconstruction based on constraint decomposition techniques. In *Proceedings of the 9th International Conference on Computer Vision*, 2003.