CONSTRAINTS FOR MODELLING COMPLEX OBJECTS

Claus Brenner

Institute of Cartography and Geoinformatics, University of Hannover, Germany – claus.brenner@ikg.uni-hannover.de

KEY WORDS: Weak primitives, constraint equation systems, constraint graphs, Gröbner bases.

ABSTRACT

When complex scenes are modelled using measured data, such as mass data from laser scanners, the objects generated do not only have to fit the data, but also have to fulfill additional constraints, like incidence, distance, or angle relations. Weak primitives have been proposed as a means to automatically introduce larger numbers of constraints into the modelling process. In this article, several aspects of constraint modelling are discussed, in particular constraint graphs and maximum matchings, polynomial constraints and Gröbner bases, and linearization. The application to the problem of interactively modifying constrained geometries is shown.

1 INTRODUCTION

Constraints play an important role in the modelling of complex scenes, such as three-dimensional city models. If models are built from primitives, constraints have to be provided in order to enforce regularity conditions. For example, walls should be upright and form right angles with other walls, gables should be horizontal and four roof planes should meet in a single point.

In the building reconstruction community, constraints have been proposed, but not used to a larger extend. Regular structures have sometimes been enforced by using constructive solid geometry (CSG). However, this does not solve the problem how constraints can be enforced across primitives. In (Brenner, 2004), this situation is discussed, several existing reconstruction approaches are compared, and *weak primitives* are suggested as a means to model objects.

Although weak primitives would hide constraints in the first place, the ability to remove and add individual constraints would expose the user to constraint modelling. Therefore, this paper reviews some general machinery to handle constraints as well as how this can be applied to interactive modelling.

2 MODELLING USING CONSTRAINTS

A geometric constraint problem consists of a set of geometric primitives and a set of constraints between them. Geometric primitives are e.g. points, lines, circles in two dimensions or points, lines, planes, cylinders, spheres, cones in three dimensions. Constraints may be logical, such as incidence, perpendicularity, parallelism, tangency, etc., or metric such as distance, angle, radius, etc. Finding a solution for a given set of constraint equations is equivalent to finding a real solution to the corresponding algebraic equation system.

Constraint equations are usually written as algebraic equations of the form $f_i(\boldsymbol{x}) = 0$, where $\boldsymbol{x} \in \mathbf{R}^n$ is the parameter vector describing the scene. For example, $\boldsymbol{x}$ can directly contain coefficients describing primitive objects such as points, lines, planes or other surfaces, but can also describe "higher level" parameters such as the transformation parameters of a substructure. Putting all the functions f_i, $1 \leq i \leq m$ together, the problem is to find a parameter vector $\boldsymbol{x}$ which satisfies the implicit (and, usually, nonlinear) constraint equation

$$\boldsymbol{f}(\boldsymbol{x}) = \mathbf{0}\ . \tag{1}$$

This equation can be either *underconstrained* (solution space is more than zero-dimensional), *well constrained* (zero dimensional solution space), or *overconstrained* (no solutions). The distinction can also be made structurally based on the number of unknowns and equations, under-, well-, or overconstrained meaning $m < n$, $m = n$, or $m > n$, respectively. When $m > n$, but still the solution space is nonempty, the system is said to be consistently overconstrained. There is no simple relationship between the two characterizations except that the dimension of the solution space is larger or equal to $n - m$ (over the complex numbers). As opposed to linear systems, a zero dimensional solution space can still involve a large number of individual solutions $\boldsymbol{x}$, exponential in the number of equations. Also, the dimension of the solution space is more involved, for example (Cox et al., 1997), given the two nonlinear equations $f_1 = xz = 0$ and $f_2 = yz = 0$ with $\boldsymbol{x} = (x, y, z)^{\mathbf{T}} \in \mathbf{R}^3$, the solution space is the union of the z-axis with the (x, y)-plane, i.e. the union of a one- and two-dimensional subspace.

When objects are reconstructed using measurements, the task can typically be formulated as

$$\begin{aligned} \|\boldsymbol{g}(\boldsymbol{b}, \boldsymbol{x})\| &\overset{!}{=} \min, \quad \text{subject to} \\ \boldsymbol{f}(\boldsymbol{x}) &= \mathbf{0}, \end{aligned} \tag{2}$$

where $\boldsymbol{g}$ subsumes the (possibly contradictory) constraints imposed by some measurement data $\boldsymbol{b}$, whereas $\boldsymbol{f}$ represents the "hard" constraints imposed by the model. As opposed to the mentioned case in CAD, $\boldsymbol{f}$ will be normally underconstrained (as else the measurements will have no effect on the solution), whereas $\boldsymbol{g}$ will be typically overconstrained (since redundant measurement data is used), which leads to a system which is both locally overconstrained and globally well- or underconstrained. The globally underconstrained case is interesting since it leaves room for interactive manipulation of object parts which are not fixed by either measurements nor model constraints.

As $\boldsymbol{f}$ is usually nonlinear, it cannot be solved analytically in general. However, one can try to *(i)* exploit the special structure of the equations, *(ii)* exploit the type of the equations, which are often polynomial, or *(iii)* linearize the equations and solve them iteratively. In the following section, some well-known techniques are discussed.

2.1 Exploiting Structure

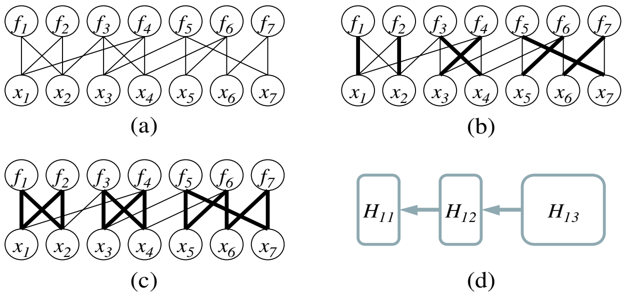
In computer aided design (CAD), a common problem is as follows: given a sketch of an object, annotated with dimensional information, derive an instance of that object. When a drawing is constructed using compass and ruler, a step-by-step procedure is usually applied. That is, objects are added one-by-one as soon as there are enough constraints which allow to place them unambiguously. (It might also happen that complete "subgroups" have to be constructed separately which are assembled only at the end.) Approaches which exploit this are called *constructive* constraint solvers. Their goal is to decompose the original constraint system into small, manageable subsystems, solve them, and recombine the solutions to an overall solution. The decomposition should be optimal in such a way that the size of the largest subsystem is minimized. A *decomposition-recombination (DR) plan* can be made as a preprocessing step before actually starting to solve the equation system. Two major types of DR plans are *constraint shape recognition* (Owen, 1991, Fudos and Hoffmann, 1997) and *generalized maximum matching* (Kramer, 1992, Ait-Aoudia et al., 1993), see (Hoffmann et al., 2001) for an overview. In the following, the basic principles of the latter are sketched.

Suppose, upon closer examination of (1), it turns out that $\boldsymbol{f} = \boldsymbol{0}$ consists of the following equations (Ait-Aoudia et al., 1993):

$$\begin{array}{ll} f_1(x_1,x_2)=0 & f_2(x_1,x_2)=0 \\ f_3(x_2,x_3,x_4)=0 & f_4(x_1,x_3,x_4)=0 \\ f_5(x_3,x_5,x_7)=0 & f_6(x_4,x_5,x_6)=0 \\ f_7(x_6,x_7)=0\ . & \end{array} \tag{3}$$

From this, a *constraint graph* can be derived, which contains a node for each constraint and unknown, and an edge if an unknown appears in a constraint, see Fig. 1(a). This *bipartite* graph G captures structural properties of the equation system. G is said to be *structurally well-constrained* if there are as many constraint nodes as unknown nodes and for no subset of constraints, there are less unknowns than constraints. If $U(\cdot)$ returns the set of unknowns for a given set of equations, e.g. $U(\{f_5, f_6\}) = \{x_5, x_6, x_7\}$, then, in the example graph with $F = \{f_1, \ldots, f_7\}$, $7 = |F| = |U(F)|$, and for any subset $F' \subseteq F$, $|F'| \leq |U(F')|$. If for a well constrained graph the property $|F'| < |U(F')|$ holds for any true subset $F' \subset F$, the graph is said to be *irreducible*. Any well constrained graph is either irreducible or contains an irreducible subgraph. One is interested in irreducible subgraphs, since they correspond to smaller size subproblems which are easier to solve.

The irreducible subgraphs of a graph can be computed from a *perfect matching* (a set of edges where no two distinct edges share a vertex and all vertices are covered, Fig. 1(b)) by finding strongly connected components,



f_1 f_2 f_3 f_4 f_5 f_6 f_7
x_1 x_2 x_3 x_4 x_5 x_6 x_7
(a)
(b)
(c)
H_{11}
H_{12}
H_{13}
(d)

Figure 1: Constraint graph for Eq. (3). (a) Original graph, (b) perfect matching, (c) strongly connected components, (d) collapsed graph H.

Fig. 1(c). By collapsing the irreducible subgraphs to single nodes, and replacing the edges between them by single, directed edges, a new graph H is obtained, which is acyclic, Fig. 1(d). It induces a partial order which can be used to solve the equation system. In the example, H_{11} can be solved first, since it contains two equations f_1, f_2, dependent on two unknowns x_1, x_2. Then, H_{12} can be solved, since there are two equations f_3, f_4, which depend on four unknowns, however, x_1, x_2 are known already, so x_3, x_4 are obtained. Finally, H_{13} can be solved using the values for x_3, x_4.

It has to be noted that since G captures only structural properties, there is not necessarily a one-to-one relationship between G and the original equation system E: G can be overconstrained, while E still has a finite (or infinite) set of solutions (in case of consistent overconstraints); G can be underconstrained but E can still have a finite set of solutions over the reals (consider $x_1^2 + x_2^2 = 0$); or G can be well-constrained but E has an infinite set of solutions (when the Jacobian $\boldsymbol{C}$ of $\boldsymbol{f}$ does not have full rank). If G is not well-constrained, the method above will find subgraphs G_1, G_2, G_3 which are well-, over-, underconstrained, respectively.

2.2 Polynomial Equations

Constraints between objects can often be expressed using bilinear equations. For example, in the two-dimensional case, with $\boldsymbol{p} = (x_1, y_1)^\mathsf{T}, \boldsymbol{q} = (x_2, y_2)^\mathsf{T} \in \mathbb{R}^2$ denoting points and $\boldsymbol{l} = (a_1, b_1, c_1)^\mathsf{T}, \boldsymbol{m} = (a_2, b_2, c_2)^\mathsf{T}$ representing lines in Hesse normal form $ax + by + c = 0$,

$$\begin{aligned}
a_1^2 + b_1^2 - 1 &= 0 && (4)\\
a_1 x_1 + b_1 y_1 + c_1 &= 0 && (5)\\
a_1 x_1 + b_1 y_1 + c_1 - d &= 0 && (6)\\
(x_1 - x_2)^2 + (y_1 - y_2)^2 - d &= 0 && (7)\\
a_1 a_2 + b_1 b_2 &= 0 && (8)\\
a_1 b_2 - a_2 b_1 &= 0 && (9)\\
a_1 a_2 + b_1 b_2 - \cos\varrho &= 0 && \\
a_1 b_2 - a_2 b_1 - \sin\varrho &= 0 && (10)
\end{aligned}$$

are equations for (4) $\boldsymbol{l}$ having a unit length normal vector, (5) $\boldsymbol{p}$ incident $\boldsymbol{l}$, (6) $\boldsymbol{p}$ having (signed) distance d from $\boldsymbol{l}$, (7) $\boldsymbol{p}$ having Euclidean distance d from $\boldsymbol{q}$, (8) $\boldsymbol{l}$ perpendicular $\boldsymbol{m}$, (9) $\boldsymbol{l}$ parallel $\boldsymbol{m}$, (10) oriented lines $\boldsymbol{l}$ and $\boldsymbol{m}$ enclosing the fixed angle ϱ. (See e.g. (Heuel, 2004) for a more complete list of relationships.)

Thus, $\boldsymbol{f}$ from (1) consists of a set of polynomial equations $f_i(\boldsymbol{x}) = 0$, for $1 \le i \le m$. The set of all solutions $\boldsymbol{x}$ which satisfy all those equations is called the *affine variety* (Cox et al., 1997) $\boldsymbol{V}(f_1, \dots, f_m)$, i.e.

$$\boldsymbol{V}(f_1, \dots, f_m) = \{\boldsymbol{x} \in \mathbf{R}^n \; : \; f_i(\boldsymbol{x}) = 0 \; \forall \, 1 \le i \le m\} \; .$$

The corresponding algebraic object is the *ideal* $\boldsymbol{I}$, which is a subset of all polynomials over a field which satisfies *(i)* $0 \in \boldsymbol{I}$, *(ii)* $f_1, f_2 \in \boldsymbol{I} \Rightarrow f_1 + f_2 \in \boldsymbol{I}$, *(iii)* $f_1 \in \boldsymbol{I}$, h polynomial $\Rightarrow hf \in \boldsymbol{I}$. Given the set of polynomials $f_1, \dots, f_m$, the set

$$\langle f_1, \dots, f_m \rangle := \left\{ \sum_{i=1}^{m} h_i f_i \right\}$$

with h_i being arbitrary polynomials over the field is indeed an ideal, the *ideal generated* by $f_1, \dots, f_m$. Interestingly, it turns out that the variety $\boldsymbol{V}(f_1, \dots, f_m)$ does not depend on the particular basis functions chosen, but rather only on the ideal generated by them, i.e. $\boldsymbol{V}(f_1, \dots, f_m) = \boldsymbol{V}(\langle f_1, \dots, f_m \rangle)$. The consequence of this is that given a set of polynomial equations $f_1, \dots, f_m$, looking for their solution space $\boldsymbol{V}(f_1, \dots, f_m)$, one can replace the original set of polynomials by any other set $h_1, \dots, h_s$ for which $\langle f_1, \dots, f_m \rangle = \langle h_1, \dots, h_s \rangle$, i.e. which spans the same ideal.

A particularly useful set is the *Gröbner* basis of an ideal $\boldsymbol{I}$, a finite set $\{h_1, \dots, h_s\}$ for which

$$\langle \mathrm{LT}(h_1), \dots, \mathrm{LT}(h_s) \rangle = \langle \mathrm{LT}(\boldsymbol{I}) \rangle \; ,$$

where $\mathrm{LT}(h)$ denotes the *leading term* of the polynomial h under a given monomial order and $\langle \mathrm{LT}(\boldsymbol{I}) \rangle$ is the ideal spanned by all leading terms of $\boldsymbol{I}$. The Gröbner basis can be systematically obtained from any given set of polynomials using Buchberger's algorithm. It is the generalization of the familiar row reduction for linear matrices, and in fact it will yield the row reduced form of a matrix if applied to a linear equation system.

Gröbner bases can be used to *solve* polynomial equation systems by successively eliminating variables (with usually a numerical solution required at the end). Unfortunately, not only the number s of polynomials in the Gröbner basis can be very large, but also the order of the polynomials. It has been shown that the construction of a Gröbner basis from polynomials of degree of at most d can involve polynomials of degree proportional to 2^{2^d}, and thus even if a Gröbner basis can be obtained, the polynomials can probably not be solved.

However, Gröbner bases have also the property that the *ideal membership* problem can be solved using polynomial division. That is, the question if a given polynomial f is member of the ideal $\boldsymbol{I}$ can be answered easily by dividing f by the polynomials $h_1, \dots, h_s$ of the Gröbner basis of $\boldsymbol{I}$. If there is no remainder, $f \in \boldsymbol{I}$. For the corresponding varieties, this means that for any solution $\boldsymbol{x} \in \boldsymbol{V}(\boldsymbol{I})$, also $f(\boldsymbol{x}) = 0$ holds.

In the context of constraint modelling, this can be used as follows. Given a partially modelled object involving variables $\boldsymbol{x}$ and constraints $h_1, \dots, h_{s-1}$, suppose the user wants to add another constraint h_s. This additional constraint can *(i)* really add information (which is what usually is intended), *(ii)* be superfluous (since the constraint is actually already enforced by some combination of $h_1, \dots, h_{s-1}$), or *(iii)* can render the system unsolvable. These cases can be distinguished as $h_s \in \boldsymbol{I} = \langle h_1, \dots, h_{s-1} \rangle \Rightarrow$ case *(ii)*, $\langle h_1, \dots, h_s \rangle = \{1\} \Rightarrow$ case *(iii)*, and else case *(i)* holds.

For example, consider 3 lines $\boldsymbol{l}_1, \boldsymbol{l}_2, \boldsymbol{l}_3$ in two dimensional Euclidean space, where $\boldsymbol{l}_1 \perp \boldsymbol{l}_2$ and $\boldsymbol{l}_2 \perp \boldsymbol{l}_3$. Clearly, then, $\boldsymbol{l}_1 \| \boldsymbol{l}_3$, so adding this constraint would have no effect. Using equations (4) and (8), the algebraic formulation of the constraints is

$$\begin{array}{ll} f_1 = a_1^2 + b_1^2 - 1 = 0 & f_4 = a_1 a_2 + b_1 b_2 = 0 \\ f_2 = a_2^2 + b_2^2 - 1 = 0 & f_5 = a_2 a_3 + b_2 b_3 = 0 \\ f_3 = a_3^2 + b_3^2 - 1 = 0 & \end{array} \qquad (11)$$

from which the Gröbner basis $B = \{h_1, \dots, h_{11}\} = \{1 - a_1^2 - b_3^2, a_1 a_2 + b_1 b_2, a_1 b_1 - a_3 b_3, -a_3 b_1 + a_1 b_3, -a_2^2 + b_3^2, a_2 a_3 + b_2 b_3, -a_2 b_2 - a_3 b_3, -a_2 - a_3 b_2 b_3 + a_2 b_3^2, -1 + a_3^2 + b_3^2, -b_1^2 + b_3^2, 1 - b_2^2 - b_3^2\}$ is obtained (using lexicographic monomial order). Now, according to Eq. (9) $\boldsymbol{l}_1 \| \boldsymbol{l}_3$ corresponds to $f_6 = a_1 b_3 - a_3 b_1 = 0$, which can be subdivided by the polynomials of B, i.e. $f_6 \in \langle f_1, \dots, f_5 \rangle = \langle h_1, \dots, h_{11} \rangle$ — in fact, one sees that $f_6 = h_4 \in B$. Thus, adding $\boldsymbol{l}_1 \| \boldsymbol{l}_3$ does not change the solution set. However, if we add to Eqs. (11) the inconsistent constraint $\boldsymbol{l}_1 \perp \boldsymbol{l}_3$, i.e. $f_7 = a_1 a_3 + b_1 b_3 = 0$, the Gröbner basis will reveal that $\langle f_1, \dots, f_5, f_7 \rangle = \langle 1 \rangle$, i.e. there remains no feasible solution $\boldsymbol{x}$.

Altogether, this approach can be used to identify redundant as well as conflicting constraints. However, the caveat is that computing a Gröbner base can take substantial time and space, so that its computation is not feasible in an interactive environment. The question here is if the special nature of the equations involved, such as (4)–(10), can somehow lead to lower time and space bounds.

2.3 Linearization of Equations

The linearization of (2) results in an equation system of the form

$$\begin{array}{rcll} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\| & \overset{!}{=} & \min, & (12) \\ \text{subject to} \quad \boldsymbol{C}\boldsymbol{x} & = & \boldsymbol{d}, & (13) \end{array}$$

where $\boldsymbol{A}$ and $\boldsymbol{C}$ are the design and constraint matrices, respectively (a weight matrix is omitted for clarity). If $\|\cdot\|$ in (12) is the L_2 norm, this can be solved in numerous ways (Förstner, 1995, Lawson and Hanson, 1995), *(i)* by weighting

$$\tilde{\boldsymbol{A}} = \begin{bmatrix} \boldsymbol{A} \\ \varepsilon \boldsymbol{C} \end{bmatrix}, \quad \tilde{\boldsymbol{b}} = \begin{bmatrix} \boldsymbol{b} \\ \varepsilon \boldsymbol{d} \end{bmatrix}$$

with some "large" ε, and minimizing $\|\tilde{\boldsymbol{A}}\boldsymbol{x} - \tilde{\boldsymbol{b}}\|$ using the normal equations

$$\tilde{\boldsymbol{A}}^{\mathrm{T}} \tilde{\boldsymbol{A}} \boldsymbol{x} = \tilde{\boldsymbol{A}}^{\mathrm{T}} \tilde{\boldsymbol{b}} \; , \qquad (14)$$

(ii) by introducing Lagrange multipliers $\boldsymbol{k}$ and solving

$$\begin{bmatrix} \boldsymbol{A}^{\mathrm{T}}\boldsymbol{A} & \boldsymbol{C}^{\mathrm{T}} \\ \boldsymbol{C} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{k} \end{bmatrix} = \begin{bmatrix} \boldsymbol{A}^{\mathrm{T}}\boldsymbol{b} \\ \boldsymbol{d} \end{bmatrix} \qquad (15)$$

or *(iii)* by an explicit parametrization of the null space of $\boldsymbol{C}$: since all solutions to (13) can be parameterized as $\boldsymbol{x} = \boldsymbol{C}^{+}\boldsymbol{d} + \boldsymbol{N}\boldsymbol{y}$, where the columns of $\boldsymbol{N}$ span the null space of $\boldsymbol{C}$ and $\boldsymbol{C}^{+}$ is the Moore-Penrose pseudoinverse, instead of minimizing (12) over all $\boldsymbol{x}$, one can minimize $\|\boldsymbol{A}(\boldsymbol{C}^{+}\boldsymbol{d} + \boldsymbol{N}\boldsymbol{y}) - \boldsymbol{b}\|$ with respect to $\boldsymbol{y}$, which yields (Lawson and Hanson, 1995)

$$\boldsymbol{x} = \boldsymbol{C}^{+}\boldsymbol{d} + (\boldsymbol{A}(\boldsymbol{I}_n - \boldsymbol{C}^{+}\boldsymbol{C}))^{+}(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{C}^{+}\boldsymbol{d}) \ . \qquad (16)$$

A problem arises with (14), since it enforces the constraint equations (13) only to a certain degree, dependent on ε. Constraint equations might compete against observation equations — actually, with mass data from laser scanning, there can be many thousands “against” a single constraint equation. Consequently, ε has to be set very high, however running the risk that $\tilde{\boldsymbol{A}}$ gets poorly conditioned. One can think of situations where the requirement to exactly fulfill the constraints (13) can be relaxed. For example, it might be not so important that a right angle is exact to 8 decimal places. However, $\boldsymbol{C}$ may also contain incidence relations which must be satisfied in order for subsequent CAD operations, such as Boolean union or intersection, to succeed.

Care must also be taken considering the rank of the matrices. Problem (12,13) has only a unique solution if $[\boldsymbol{A}^{\mathbf{T}}, \boldsymbol{C}^{\mathbf{T}}]$ has full rank. If not, solutions based on normal equations and inversion (14), (15) are not suitable, but pseudoinverse (16) or SVD solutions are. In the context of a constraint modeler, a newly placed object will usually be underconstrained, and in fact it will usually stay underconstrained as long as it is intended to be modified interactively.

3 INTERACTIVE MODIFICATION OF CONSTRAINED OBJECTS

Assume that a scene contains instantiations of a number of objects which fulfill all given constraints. For example, placing a weak primitive in the scene would automatically instantiate the unknowns in such a way that the associated constraints are fulfilled. It would then add the unknowns to the overall parameter vector $\boldsymbol{x}$ and the associated constraints to the constraint set. If additional constraints have been added, methods from Sect. 2.2 or 2.3 can be used to obtain an instantiation which fulfills all constraints.

Now suppose that the user picks some (part of an) object O in order to modify it interactively. For example, he could pick a point of a polygon in order to drag it to another position. Now it can happen that *(1)* O is completely free with respect to the selected modify operation and no other objects are involved when it is modified, *(2)* O is free, however it is linked by constraints to other objects, which have to be modified as well as a result of modifying O, *(3)* O is partially or fully constrained, i.e. the user's wish to modify O can be only fulfilled partially or cannot be fulfilled at all.

In case (1) there is no problem. However, in case (2) the question is which other objects are modified as result of modifying O, i.e. how the change of O “propagates”. Finally, in case (3), as the situation of objects and constraints might be quite complex, the user would want details why his attempt to interactively modify the scene fails. That is, the system should be able to explain the reason for the failure in such a manner that the user can take appropriate action. These aspects are dealt with in the next two subsections.

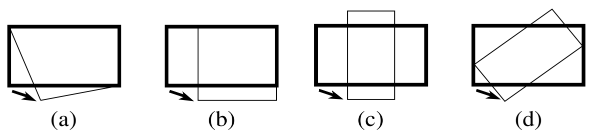


Figure 2: Different user interface behavior when a point of a polygon (a) or a rectangle (b)–(d) is changed.

3.1 Propagation of Changes

If a scene fulfils all constraints, the linearized constraints $\boldsymbol{C}\boldsymbol{x} = \boldsymbol{d}$ hold (Sect. 2.3). Now suppose the user picks some object O in order to modify it, i.e. he wants to apply a change $\triangle\boldsymbol{x}$ such that still all constraints are fulfilled, $\boldsymbol{C}(\boldsymbol{x} + \triangle\boldsymbol{x}) = \boldsymbol{d}$. Then, $\boldsymbol{C}\triangle\boldsymbol{x} = \boldsymbol{0}$, i.e. $\triangle\boldsymbol{x}$ is in the null space of $\boldsymbol{C}$. If the null space of $\boldsymbol{C}$ has dimension k, and $\boldsymbol{N}$ is the $n \times k$ matrix containing k column vectors spanning the null space of $\boldsymbol{C}$ (i.e., $\boldsymbol{C}\boldsymbol{N} = \boldsymbol{0}$), $\triangle\boldsymbol{x} = \boldsymbol{N}\boldsymbol{y}$ must hold, for a suitable $\boldsymbol{y}$. Without loss of generality, one can assume that the parameters which are directly modified by the user are the first j elements of the parameter vector $\boldsymbol{x}$. That is, by the interactive modification, the user attempts to set $\triangle\boldsymbol{x} = (\triangle x_1, \ldots, \triangle x_j, 0, \ldots, 0)^{\mathbf{T}}$. However, this vector might not be in the null space of $\boldsymbol{C}$. In general, equation

$$\begin{bmatrix} \triangle x_1 \\ \vdots \\ \triangle x_j \\ 0 \\ \vdots \\ 0 \end{bmatrix} + \begin{bmatrix} v_1 \\ \vdots \\ v_j \\ v_{j+1} \\ \vdots \\ v_n \end{bmatrix} = \boldsymbol{N}\boldsymbol{y} \qquad (17)$$

holds where the user wants to set $\triangle x_1, \ldots, \triangle x_j$ arbitrarily while $v_1 = \ldots = v_j = 0$. The different cases are: *(1)* The null space of C is $\{\mathbf{0}\}$. Then, no $\triangle x_i \neq 0$. The user cannot modify the picked object at all, since it is fully constrained. *(2)* The dimension of the null space is > 0, however the first j rows of $\boldsymbol{N}$ have rank $< j$. Then, $v_1 = \ldots = v_j = 0$ cannot be achieved. The user can modify the picked object, however not with all degrees of freedom. *(3)* The first j rows of $\boldsymbol{N}$ have full rank, movement is not constrained. If all of the $v_{j+1}, \ldots, v_n$ are zero, no other objects are affected, but this will happen only when there are no constraints or the constraints have zero gradient in the direction of the parameter modification.

In case (3), any $\boldsymbol{y}$ which fulfills the first j rows of Eq. (17) can be used, so the question is according to what criterion $\boldsymbol{y}$ is chosen. It turns out that this question is less a mathematical one but has rather more to do with what we expect from a user interface. Consider a 2-D drawing program. If a “polygon primitive” is placed in the scene and a single polygon vertex is selected and dragged, one would usually expect that only the vertex itself and the incident edges are actually changed, see Fig. 2(a). On the other hand, if the vertex of a “rectangle primitive” is moved, it is not so clear what a user would expect. For example, it would be sen-

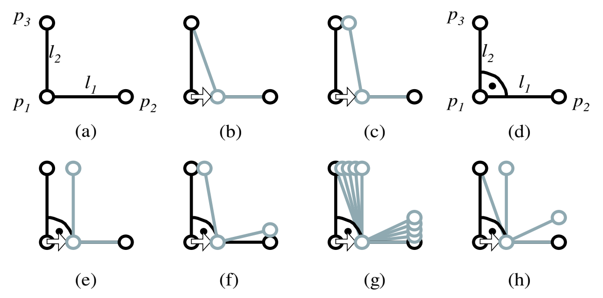


Figure 3: Behavior of a simple constraint system under different optimization criteria (see text).

sible that the diagonally opposite vertex and the rotation stays fixed while the width and height is adjusted (b). Or, the midpoint of the rectangle and the rotation stays fixed, while width and height are adjusted (c). Or, rotation and scale are changed (d). That is, a drawing program usually has interactive modification operators which implement a certain behavior. Such an approach, however, is not feasible for arbitrary objects and constraints.

In order to select $\boldsymbol{y}$ in this situation, a systematic approach could be that there is "not much change" among the objects which are not directly modified. Interpreting this in the least squares sense, the task is to minimize

$$\sum_{i=j+1}^{n} (v_i)^2 \text{ , subject to} \tag{18}$$

$$(\triangle x_1, \ldots, \triangle x_j, v_{j+1}, \ldots, v_n)^{\mathbf{T}} = \boldsymbol{N}\boldsymbol{y} \ . \tag{19}$$

To see what happens, consider the simple example in Fig. 3(a). There are three points $\boldsymbol{p}_1 = (0,0)^{\mathbf{T}}$, $\boldsymbol{p}_2 = (1,0)^{\mathbf{T}}$, $\boldsymbol{p}_3 = (0,1)^{\mathbf{T}}$, linked by two lines $\boldsymbol{l}_1 = (0,1,0)^{\mathbf{T}}$, $\boldsymbol{l}_2 = (1,0,0)^{\mathbf{T}}$, so that the unknown vector is $\boldsymbol{x} = (x_1, y_1, x_2, y_2, x_3, y_3, a_1, b_1, c_1, a_2, b_2, c_2)^{\mathbf{T}}$. Two normal vector constraints of type (4) for $\boldsymbol{l}_1$ and $\boldsymbol{l}_2$, and for incidence constraints $\boldsymbol{p}_1 \in \boldsymbol{l}_1$, $\boldsymbol{p}_2 \in \boldsymbol{l}_1$, $\boldsymbol{p}_1 \in \boldsymbol{l}_2$, $\boldsymbol{p}_3 \in \boldsymbol{l}_2$ of type (5) are used, which yields the linearized constraint matrix

$$\boldsymbol{C} = \begin{bmatrix} 0&0&0&0&0&0&0&2&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&2&0&0 \\ 0&1&0&0&0&0&0&0&1&0&0&0 \\ 0&0&0&1&0&0&1&0&1&0&0&0 \\ 0&0&0&0&1&0&0&0&0&0&1&1 \\ 1&0&0&0&0&0&0&0&0&0&0&1 \end{bmatrix} ,$$

and the matrix $\boldsymbol{N}$ is obtained (in this particularly simple form, by column reduction) as

$$\boldsymbol{N} = \begin{bmatrix} 1&0&0&0&0&0 \\ 0&1&0&0&0&0 \\ 0&0&1&0&0&0 \\ 0&0&0&1&0&0 \\ 0&0&0&0&1&0 \\ 0&0&0&0&0&1 \\ 0&1&0&-1&0&0 \\ 0&0&0&0&0&0 \\ 0&-1&0&0&0&0 \\ 0&0&0&0&0&0 \\ 1&0&0&0&-1&0 \\ -1&0&0&0&0&0 \end{bmatrix} . \tag{20}$$

Now suppose the user picks point $\boldsymbol{p}_1$ and wants to drag it along the x direction, see Fig. 3(b). One can immediately deduce that any move of $\boldsymbol{p}_1$ is possible since the first two rows of $\boldsymbol{N}$ have rank 2 — in fact, any of the points can be moved freely since the first 6 rows have full rank. However, what would be usually expected is that when only $\boldsymbol{p}_1$ is moved, $\boldsymbol{p}_2$ and $\boldsymbol{p}_3$ stay fixed. Unfortunately, minimizing (18) with constraint $\triangle x_1 = 1$ and $\triangle y_1 = 0$ yields an optimum of $\sqrt{3/2}$ at $\boldsymbol{y} = (1,0,0,0,1/2,0)^{\mathbf{T}}$, with $\boldsymbol{N}\boldsymbol{y} = (1,0,0,0,1/2,0,0,0,0,0,1/2,-1)^{\mathbf{T}}$. The corresponding situation is shown in Fig. 3(c). Both $\boldsymbol{p}_1$ and $\boldsymbol{p}_3$ will move, together with $\boldsymbol{l}_2$. The reason for this behavior is that Eq. (18) not only minimizes the change of $\boldsymbol{p}_2$ and $\boldsymbol{p}_3$, but also that of $\boldsymbol{l}_1$ and $\boldsymbol{l}_2$. This suppresses the turn of $\boldsymbol{l}_2$ and forces $\boldsymbol{p}_3$ to move. One can resolve this by regarding points as "primary" and lines as "secondary" variables, and restricting the minimization to primary variables. Then, only $v_3^2 + v_4^2 + v_5^2 + v_6^2$ is minimized, and one obtains the expected result of figure 3(b).

However, consider the slightly more complex scene of Fig. 3(d), where the equation for $\boldsymbol{l}_1 \perp \boldsymbol{l}_2$ has been added. $\boldsymbol{C}$ is now a 7×12 matrix of rank 7, while $\boldsymbol{N}$ is 12×5. Again, one can see from $\boldsymbol{N}$ that $\boldsymbol{p}_1$ can be dragged freely. However, the least squares solution, even when constrained to only minimizing the movement of the point coordinates yields a minimum of $1/\sqrt{2}$ and $\boldsymbol{N}\boldsymbol{y} = (1,0,0,1/2,1/2,0,-1/2,0,0,0,1/2,-1)$. That is, instead of the situation in Fig. 3(e), all 3 points will move, as depicted in Fig. 3(f).

In general, due to the nature of least squares, the minimization of (18) will tend to change many variables by a small amount rather than a few variables by a large amount. In an attempt to affect fewer variables, one could instead minimize the L_1 norm, i.e. $\sum_{i=j+1}^{n} |v_i|$. In the example (again with only the point movement $|v_3| + |v_4| + |v_5| + |v_6|$ minimized), a minimum of 1 is obtained. However, any $\boldsymbol{y} = (1,0,0,\lambda,0)$ with $0 \le \lambda \le 1$ yields this minimum, with $\boldsymbol{N}\boldsymbol{y} = (1,0,0,\lambda,1-\lambda,0,-\lambda,0,0,0,\lambda,-1)$ so there is no unique solution. For $0 < \lambda < 1$ both $\boldsymbol{p}_2$ and $\boldsymbol{p}_3$ are affected, see Fig. 3(g).

From a computational perspective, it would be desirable that as few as possible variables are affected by a user interaction, although it is not clear if this leads sometimes to unusual behavior from an user interface viewpoint. That is, we want as few as possible $v_i \neq 0$, the function to minimize is

$$\sum_{i=j+1}^{n} \psi(v_i) \tag{21}$$

subject to (19), with $\psi(x) = \mathrm{sgn}|x|$, i.e. (21) counts the number of nonzero v_i. In the example, this leads to two solutions, see Fig. 3(h). Numeric optimization of (21) is problematic due to the discrete nature of ψ. One can however see that the minimization problem is equivalent to a standard problem in computational geometry. If the first j rows of (19) are solved for $\boldsymbol{y}$, and $v_{j+1}, \ldots, v_n$ are set to zero, one obtains an equation system of the form

$$\begin{bmatrix} \mu_{j+1} \\ \vdots \\ \mu_n \end{bmatrix} = \boldsymbol{M}\boldsymbol{\nu} \tag{22}$$

with $\boldsymbol{M}$ being $(n-j) \times (k-j)$ and $\boldsymbol{M}$ and μ_i fixed. The solution sought for is the $\boldsymbol{\nu}$ for which the largest number of rows of the equation system (22) holds.

Since each of the rows in (22) defines a hyperplane in $(k-j)$ dimensional space, this is equivalent to finding a point $\boldsymbol{\nu}$ in $(k-j)$ dimensional space which is incident to

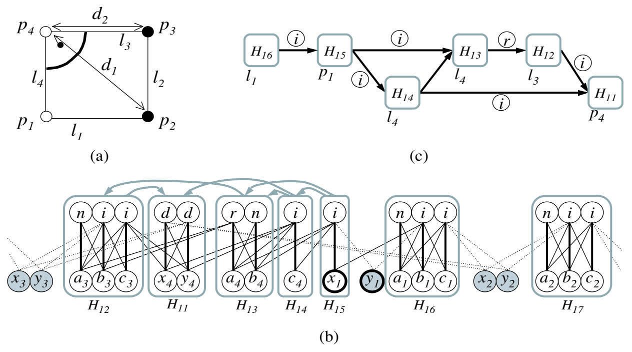


Figure 4: Introspection example. (a) Original scene, (b) constraint graph with maximum matching, (c) collapsed graph H. n, i, d, r are normal, incidence, distance and rectangularity constraints, respectively.

the largest number of hyperplanes of a given set of $(n-j)$ planes. This is called the *exact fitting problem*. (Guibas et al., 1996) have shown that while the problem is $\Omega(n^d)$ for n hyperplanes in d dimensional space when the number of hyperplanes meeting in $\boldsymbol{\nu}$ is fixed, it can be solved in *linear* time when the number of planes meeting in $\boldsymbol{\nu}$ is a *fixed fraction* of n. This results in a total complexity of $\mathbf{O}(\min\{(n^d/m^{d-1})\log(n/m),\ n^d\})$, with m being the number of planes incident in $\boldsymbol{\nu}$. When constraint equations involve spatially close objects only, most of the row equations in (22) are satisfied and the linear time bound holds.

To summarize this section, interactively modifying an element of a scene may influence other elements as well. How they actually are changed depends on the minimization criterion applied. Least squares may not be the appropriate solution, since it tends to modify many objects by a small amount. Minimizing the L_1 norm yields non-unique results, which however may be closer to the expected interaction. Finally, minimizing the number of changed elements also gives non-unique results, leads to the well-known exact fitting problem, and yields the largest reduction in number of equations for a subsequent iterative estimation.

3.2 Introspection

If an object O in the scene cannot be moved as expected, it is partially or fully constrained. The reduction of the degrees of freedom of O results from some combination of constraints. It is clear that unconnected components of the constraint graph cannot influence O. However, it is desirable to have a more fine-grained result in order to assist the user.

The graph decomposition discussed in Sect. 2.1 can be used to derive additional information. Since the graph of collapsed nodes, H, gives an order in which the equations can be solved, any variable solved for can also be "traced back". Consider the example in Fig. 4(a). There are four points and four lines, with $\boldsymbol{p}_2$ and $\boldsymbol{p}_3$ fixed, as well as fixed distances $\boldsymbol{p}_2$–$\boldsymbol{p}_4$ and $\boldsymbol{p}_3$–$\boldsymbol{p}_4$ and a right angle $\boldsymbol{l}_3 \perp \boldsymbol{l}_4$. As a result, $\boldsymbol{p}_1$ cannot be moved freely. Figure 4(b) shows the corresponding bipartite graph with a maximum matching and irreducible subgraphs (grey boxes). Collapsing the subgraphs, Fig. 4(c) is obtained, which nicely explains why $\boldsymbol{p}_1$ is constrained (reading it according to the partial order from left to right): H_{15} ($\boldsymbol{p}_1$) is fixed since it is incident to H_{14} and H_{13} ($\boldsymbol{l}_4$). $\boldsymbol{l}_4$ in turn is fixed since it is going through $\boldsymbol{p}_4$ and it has a right angle with $\boldsymbol{l}_3$. $\boldsymbol{l}_3$ in turn is fixed since it is also going through $\boldsymbol{p}_4$. Thus, by tracing back the graph H, the system can derive the reasoning why $\boldsymbol{p}_1$ is fixed in one direction and output this to a user in text form.

4 CONCLUSION

In this article, some machinery for constraint modelling has been presented. Constraint graphs and maximum matchings have been discussed as a means to investigate the structure of equation systems. Gröbner bases were introduced for the special case of polynomial equations. Finally, the solution of linear constraint equations, especially using a basis of the null space, has been shown.

As examples of how these techniques can be used, constraint graphs have been applied to the introspection problem, Gröbner bases for identifying conflicting or redundant constraints, and linearized equations for the propagation of changes.

ACKNOWLEDGEMENTS

I thank Marc van Kreveld for pointing me to the solution (Guibas et al., 1996) of the exact fitting problem and Hans-Christian Graf v. Bothmer as well as the first anonymous reviewer for their valuable comments. This work has been funded by the VolkswagenStiftung, Germany.

REFERENCES

Ait-Aoudia, S., Jegou, R. and Michelucci, D., 1993. Reduction of constraint systems. In: Compugraphics, pp. 83–92.

Brenner, C., 2004. Modelling 3D objects using weak primitives. In: International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Vol. XXXV, Istanbul, 2004.

Cox, D., Little, J. and O'Shea, D., 1997. Ideals, Varieties, and Algorithms. Springer Verlag New York.

Förstner, W., 1995. Mid-level vision processes for automatic building extraction. In: A. Grün, O. Kübler and P. Agouris (eds), Automatic Extraction of Man-Made Objects from Aerial and Space Images, Ascona Workshop 1995, Birkhäuser, Basel.

Fudos, I. and Hoffmann, C. M., 1997. A graph-constructive approach to solving systems of geometric constraints. ACM Trans. on Graphics 16(2), pp. 179–216.

Guibas, L. J., Overmars, M. H. and Robert, J.-M., 1996. The exact fitting problem for points. Comput. Geom. Theory Appl. 6, pp. 215–230.

Heuel, S., 2004. Uncertain Projective Geometry : Statistical Reasoning for Polyhedral Object Reconstruction. Springer LNCS.

Hoffmann, C. M., Lomonosov, A. and Sitharam, M., 2001. Decomposition plans for geometric constraint systems, part I: Performance measures for CAD. J. Symbolic Computation 31, pp. 367–408.

Kramer, G., 1992. Solving Geometric Constraint Systems. MIT Press, Cambridge, MA.

Lawson, C. L. and Hanson, R. J., 1995. Solving Least Squares Problems. SIAM Society for Industrial and Applied Mathematics.

Owen, J. C., 1991. Algebraic solution for geometry from dimensional constraints. In: ACM Symposium Found. of Solid Modeling, Austin, TX, ACM.