# Relaxed Parametric Design with Probabilistic Constraints

Yacov Hel-Or Ari Rappoport Michael Werman

Institute of Computer Science, The Hebrew University, Jerusalem
91904, Israel.
http://www.cs.huji.ac.il/∼arir. arir@cs.huji.ac.il.

**Abstract:** Parametric design is an important modeling paradigm in computer aided design. Relationships (constraints) between the degrees of freedom (DOFs) of the model, instead of the DOFs themselves, are specified, resulting in efficient design modifications and variations. Current parametric modelers require an exact specification of all the constraints involved, which causes over-work on the part of the designer during design iterations. We describe the *relaxed parametric design* modeling paradigm, in which decisions which needlessly limit the freedom of design in later stages are avoided. The designer uses *soft constraints* and specifies the exactness by which they are to be met. As a specific scheme for implementing relaxed parametric design, we present *probabilistic constraints*, where a parametric model is viewed as a stochastic process. Softness of a constraint is represented as the covariance of a suitably distributed random variable. We describe a novel method for expressing the DOFs and the model as a system of probabilistic equations, which is then solved using the Kalman filter, a powerful estimation tool for stochastic systems. An *a priori* covariance matrix associated with a DOF can be used as a guideline to the solver to select a particular solution among multiple solutions.

**Keywords:** Relaxed parametric design, design by least commitment, soft constraints, probabilistic constraints, geometric constraints, Kalman filter.

**Publication:**

1. *Computer-Aided Design*, 26(6):426-434, 1994.
2. Appears in the proceedings of the refereed conference *Second ACM/SIGGRAPH Symposium on Solid Modeling and Applications,* pp. 261-270, ACM Press, Montreal, May 1993.

# 1 Introduction

The design of geometric models is a major activity in various fields, including computer graphics, mechanical computer aided design (MCAD), bio-medical CAD, robotics, and more. A design paradigm should support fast model specification, modification and variation synthesis.

The process of designing geometric models consists of iterations of top-down and bottom-up passes. In each iteration, one of the levels of the design, and its components, are refined and tuned based on experience gained by previous passes. It is highly desirable that the designer not over-work on a particular level or over-specify it, i.e., make decisions unnecessarily limiting the freedom in the design of other levels.

Parametric design [Sutherland63, Borning81, Lin81, PTC87] is a paradigm in which the degrees of freedom (DOFs) of a model are not determined directly by the designer; rather, a set of constraints (relations) is defined over them, and the system automatically computes the degrees of freedom. Parametric design is one of the most important development in MCAD is the last few years and is commercially successful [Porter91]. However, current parametric systems require full and exact specification of the DOF constraints. Under- and over-constrained models are easy to produce, and manual correction of these is time consuming and error prone. Current parametric systems therefore cause over-specification and over-work.

In this paper we suggest a design method we call *relaxed parametric design*. The method provides the designer with the capability of expressing *soft constraints*, constraints which do not have to be met exactly. Soft constraints are used whenever the designer wishes to express a general decision or guideline, avoiding over-specification.

We present the *probabilistic constraints* scheme for implementing the relaxed parametric design paradigm. The mathematical tool with which soft constraints are expressed is probability theory. The rigidity of a constraint is determined by the uncertainty (covariance) of a suitable random variable. The model is viewed as a static stochastic process, and the resulting system of probabilistic equations is solved using the Kalman filter, a powerful estimation tool from the theory of stochastic systems. The model's degrees of freedom constitute the process's state vector, and the constraints are regarded as *measurements* of a function of the state. The solution yields the model with the highest probability under some criterion.

In addition to the advantages in terms of relaxed design, the system is easier to solve since it is less rigid. Additionally, the user can provide guidelines as to which solution to choose among the possible multiple solutions, by using an *a priori* state vector and a covariance matrix associated with it.

Soft constraints on the location of representative points and on geometric dimensions bear a strong similarity to schemes for representing tolerances [Roy91, Juster92]. The tolerance and dimensioning representation problem is in some sense simpler than the general parametric constraints problem, and in some sense more complex. In this paper we do not attempt to address tolerances.

In Section 2 we discuss relaxed design and parametric design and describe the relaxed parametric design paradigm. Section 3 presents the general idea of the probabilistic constraints scheme and reviews the Kalman filter. In Section 4 the solution algorithm is described in detail and Section 5 studies some of its properties. Section 6 describes an implementation of a simple parametric modeler using the algorithm.

## 2 Relaxed Parametric Design

In this section we discuss the parametric design modeling paradigm and its main disadvantage, over-specification. We then describe the relaxed parametric design paradigm, which attempts to solve this disadvantage.

### 2.1 Relaxed Design

A model to be designed can be viewed in numerous levels of abstraction. In the highest level, only the outside behavior (functionality) of the model is considered, while the lowest level specifies the exact design and operation of the smallest sub-parts. Interim levels specify the relationship of modules assembled from sub-modules (components).

Design is an incremental, exploratory activity [Smithers89]. The process of designing a model is composed of iterations between *top-down* and *bottom-up* design stages. A top-down stage starts from a specification of the functionality of a particular component in some level, and breaks the problem into a set of smaller problems, to be solved by the next level. A bottom-up stage assembles various already-designed components into a larger one, having more complex functionality. Usually, each level has to be tuned according to knowledge gained while designing the ones above and below it. The design proceeds in such iterations until the goal is met [Bañares-Alcãntara91].

There are two related potential dangers when using this design paradigm, *over-work* and *over-specification*. Over-work means spending too much work on a level or a component while the iterations between levels are still being performed. Many of the decisions taken when designing a particular level are modified later, when tuning an adjacent level. Over-specification of one level means that while designing it, decisions that needlessly influence the design of other levels are taken. The two dangers are highly related and we will refer to both as over-specification.

The *design by least commitment (DLC)* paradigm [Mäntylä89] avoids over-specification of the design in order to facilitate later process planning. The definition of DLC given in [Mäntylä89] is: 'systematically avoiding making design decisions that unnecessarily limit the freedom of later process planning'. A so-called 'relaxed feature model' is defined in which aspects unimportant from the designer's point of view are left to be determined by the process planner.

This definition can be rephrased to say: 'systematically avoiding making design decisions that needlessly limit the freedom of the design of other levels or components'. We refer to this paradigm as *relaxed design*.

### 2.2 Parametric Design

Parametric design is one of today's strongest trends in commercial solid modeling [PTC87, Porter91]. Its main purpose is achieving flexibility and efficient variation generation, but it can also be viewed as a relaxed design technique.

A *parametric geometric model* consists of

- A collection $O = \{o_1, ..., o_g\}$ of *geometric objects* or *features*,
- A set $D = \{u_1, ..., u_s\}$ of *degrees of freedom (DOFs)* which completely determine the state of the geometric objects in $O$,

- An external *parameter vector* $\lambda$,
- A set $C = \{c_1, ..., c_r\}$ of parameterized *constraints* among the members of $D$, $c_i = c_i(u_{i_1}, ..., u_{i_m}, \lambda)$.

The geometric objects can be simple geometric primitives, such as cubes, spheres, cylinders, cones and torii, or more complex primitives, such as polyhedra and spline surfaces, or assemblies of such objects. Any collection of objects which can be represented in a computer can be completely specified by a finite set of degrees of freedom. It is convenient to choose the DOFs so that they have a geometric interpretation, e.g. points, vectors and angles. We denote by $n$ the total dimensionality of the degrees of freedom. For example, if all $s$ DOFs are represented as $d$-dimensional points, then $n = ds$.

Every constraint $c_i \in C$ specifies a *relation* that must hold between some DOFs $u_{i_1}, ..., u_{i_m}$. Constraints can be of numerous types. The most common types specify constraints through an implicit equation (i.e., $c_i(u_{i_1}, ..., u_{i_m}, \lambda) = 0$) or inequality. Other types involve geometric constructions, e.g. a point has to be on the convex hull of a set of points, or functional optimizations, e.g. minimum strain energy.

Design of a level is done by imposing constraints between its degrees of freedom, instead of fixing them exactly. This has the effect of expressing the model in higher level terms, since functionality is captured rather than one of its particular implementations. The composition and behavior of a level are thus determined by a generally smaller number of parameters, since the relational, constraint-based, expression is higher level than DOF manipulation. The advantage of parametric design is the ability to make fast modifications and produce design variations easily, through modifications of the parameters.

Parametric design can thus be be viewed as a technique for avoiding over-specification of the degrees of freedom. The designer does not make decisions regarding the DOFs themselves; rather, the system can compute the exact DOFs corresponding to a particular choice of parameter vector automatically, using numeric constraint solvers (symbolic solvers are basically still a research topic when dealing with practical parametric systems with many types of constraints [Roy91, Juster92]).

Notwithstanding, parametric design can also cause over-specification. Existing parametric systems require a complete and accurate specification of all the constraints between DOFs. This is due to the fact that computing the DOFs from the constraints is done by solving systems of equations, and solvers encounter problems with under-constrained systems. When confronted with such a system, the typical solver has to fix the under-constrained DOFs so that the system could be expressed and solved. Usually, the only sensible way to do this is to use an *initial guess* for the solution, supplied by the user or taken from the current system state. Consequently, the user can only express exact relations and give an initial guess; it is impossible to give *inexact guidelines* to the solver. This stands in contradiction with the relaxed design paradigm, since the user may be forced to express many more constraints than currently necessary, limiting the design freedom of later stages by over-specification.

Another problem with the requirement for exactly constrained models is the danger of producing *over-constrained* models. It may be difficult for the user to express a set of constraints which leave the model neither under-constrained nor over-constrained. Thus, parametric design requires a much greater effort than is intuitively needed. This phenomenon is being recognized

by many design teams, who found out that a great deal of effort is required in order to benefit from the advantages of parametric design [Porter91].

### 2.3 Relaxed Parametric Design

We can view the constraints that the designer expresses in parametric design as 'guidelines' given to the system so that it can compute exact DOFs. The problem described above can be succinctly summarized by saying that designer guidelines have to be exact. We propose a modeling paradigm in which the designer is given the capability of explicitly expressing *relaxed* constraints. Relaxed (or *soft*) constraints are constraints which do not have to be satisfied exactly, but can be approximated according to some criterion.

The above definition of a parametric model is augmented by:

- A set $S = \{f_1, ..., f_r\}$ of *softness functions*, $f_i = f_i(c_i)$.

A softness function is defined for each constraint, giving the amount of rigidity with which the constraint has to be satisfied. The value of a softness function can be a scalar, a vector, a matrix, or a more complex entity. The semantics of the value of a softness function is given by the constraint solver. This rigidity is in general a multi-dimensional entity, not a scalar, in order to be able to assign a different softness to different characteristics of a constraint (for example, along different spatial directions).

In relaxed parametric design, only the constraints that are required in the design of the current level or component are expressed as completely rigid. All other constraints are expressed as soft constraints with a user-defined certainty. Conflicts with later design decisions can be resolved without re-designing the whole component by assigning a higher rigidity to the new decision. The degree of satisfaction of each constraint can be derived from the estimated covariance matrix.

In addition to the advantage of avoiding over-specification, the relaxed parametric design paradigm possesses advantages related to the solution process itself. A common problem with numerical constraint solvers is their inability to recognize the existence of several solutions; usually the solver converges to the solution closest to the initial guess. The relaxed parametric design scheme can be used to guide the solver to prefer one solution over another, by defining relaxed constraints whose sole purpose is to hint at the preferred solution. These relaxed constraints can be used as persistent guidelines for choosing a solution, if they are stored as a part of the model and not discarded after each solution.

The modeling system described in [Mäntylä90] uses the delta-blue algorithm for constraint satisfaction, which enables the assignment of a 'strength' to a constraint. In theoretical AI various approaches towards representing the degree of belief in a piece of information have been suggested. In the rest of the paper we will present a novel scheme for implementing relaxed parametric design, termed *probabilistic constraints*. Softness functions of constraints are expressed as covariance matrices, and the resulting system is solved using the Kalman filter, an estimation tool for stochastic systems.

# 3 Probabilistic Constraints

In this section we describe a scheme for implementing the relaxed parametric design paradigm, a scheme which we term *probabilistic constraints*. We also review necessary techniques and concepts from the theory of stochastic systems.

## 3.1 General Idea

The general idea in the probabilistic constraints scheme is to treat the relaxed parametric model as a static stochastic system. The general *measurement model* of a static stochastic system is

$$\hat{m} = f(u) + e,$$

where $u$ is the vector of parameters describing the *state* of the system, $f(u)$ is a mathematical function of the state, which can be measured, $\hat{m}$ is the vector of actual measurements, and $e$ is the measurement noise, whose covariance (or *uncertainty*) is assumed known. By convention, actual measurements are denoted with hats, while 'true' measurements (without the noise vector $e$) are written without the hat. The central problem in the theory of such systems is to *estimate* the state vector from the measurements.

In the probabilistic constraints scheme, the system's state is the vector of the model's degrees of freedom (the external parameter vector is considered constant, being set by the user). Constraints are viewed as 'measurements' [Zhang88]. Since the noise in the stochastic model is associated with a measurement, we have a tool of expressing the softness of a constraint. The softness function (measurement uncertainty) is a covariance matrix which represents the prescribed rigidity of the constraints.

A measurement (i.e., a constraint), with a large uncertainty will not be enforced as strictly by the estimator as one with a small amount of uncertainty. Note the nomenclature: we say *estimate* instead of compute, and *uncertainty* (or *covariance*) instead of softness.

## 3.2 Kalman Filter Summary

The main tool which we use is the *Kalman filter*. The Kalman filter is a tool for estimating the state of a stochastic linear system from measurements. Here we briefly describe the static Kalman filter and state some of its properties. For a complete discussion see e.g. [Anderson79].

The static Kalman filter is based on the measurement model $\hat{m} = Hu + e$. The vector $u$ is the state vector. The (not necessarily square) matrix $H$ is a linear operator relating the state to the true measurements. $e$ is the noise vector associated with the measurements, and $\hat{m}$ is a vector of actual measurements.

It is assumed that the mean vector of the noise is zero and that its covariance matrix $R$ is given:

$$E\{e\} = 0, \qquad E\{ee^T\} = R,$$

and that an *a priori* estimate of the state vector with its associated covariance are known:

$$E\{u\} = \hat{u}_0, \qquad E\{(u - \hat{u}_0)(u - \hat{u}_0)^T\} = \Sigma_0.$$

The Kalman filter equations for estimating $u, \Sigma$ given $\hat{m}$ are:

$$\begin{aligned} K &= \Sigma_0 H^T (H\Sigma_0 H^T + R)^{-1} \\ \hat{u} &= \hat{u}_0 + K(\hat{m} - H\hat{u}_0) \\ \Sigma &= \Sigma_0 - KH\Sigma_0. \end{aligned} \tag{1}$$

The first equation computes the *Kalman gain matrix*, the second produces the new estimate, and the third gives the certainty of the estimate. A 'black box' view of the filter is shown in Figure 1. The equations result in an unbiased estimator of $u$ (i.e., having the same mean) which is optimal in the sense of linear minimal variance (i.e., among all linear estimators it produces the smallest unconditional error covariance matrix, hence yielding the minimum squared error). When the noise $e$ is normally distributed, which is a very reasonable assumption, the estimator is optimal in the maximum likelihood sense (i.e., it is the estimator for which the input measurement vector is the event with maximum probability).

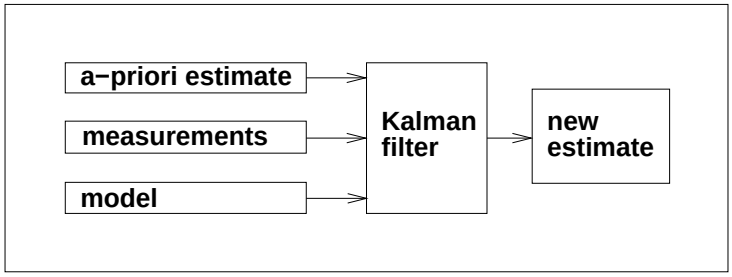


Figure 1: A 'black box' view of the Kalman filter.

When the model is non-linear, the *extended Kalman filter* is used. The extended filter performs a linearization of the model to produce the linear matrix $H$. We use the extended filter to enable non-linear constraints. This linearization process is explained in detail later.

The model and equations described above are actually a degenerate case of the Kalman filter for static system and measurement models. We have chosen to present it as a Kalman filter estimator since we make use of a more general model in other applications of our techniques.

# 4 Expressing and Solving the System

In this section we describe an algorithm for expressing and solving systems of probabilistic constraints, utilizing the Kalman filter. We detail the expression of the previous state estimate, the measurements and their uncertainties.

In this paper we assume that all the constraints can be expressed as implicit mathematical functions with a Taylor expansion around any points. This assumption encompasses almost all types of first order constraints used in current parametric modelers. We do not treat symbolic, non-numeric constraints such as those that can be solved using the algorithm in [Mäntylä90].

### A Priori State Estimate

In our algorithm the *a priori* state estimate $\hat{u}_0$ consists of the current values of the system DOFs. There are two methods for determining the *a priori* uncertainty $\Sigma_0$. The simple method is to use a uniform, very high uncertainty, to avoid clinging to a far-off *a priori* estimate. In this case the covariance matrix is a diagonal matrix with numbers which are large relative to the numbers in which the DOFs are expressed. This method corresponds to the initial guess demanded by exact parametric systems.

A more interesting method enables the user to provide guidelines for choosing among multiple solutions. The *a priori* covariance matrix is defined according to the relative weight assigned to each DOFs guideline. If this covariance matrix is stored with the model, it can be viewed as a persistent guideline for choosing among multiple solutions.

### Measurements and Linear Operator

Recall that the main idea is to represent the constraints as measurements. Linear constraints can be expressed directly. For non-linear constraints, we use a process of *linearization* of the constraints. This process can be viewed as an internal part of the extended Kalman filter.

As an example, take a constraint $c_i$ between two degrees of freedom $u_j, u_k$, $c_i(u_j, u_k, \lambda) = 0$. A first order expansion of $c_i$ to a Taylor's series around any pair of similar DOFs $a, b$ gives

$$c_i(a, b, \lambda) + \left.\frac{\partial c_i}{\partial u_j}\right|_{(a,b)}^{T} (u_j - a) + \left.\frac{\partial c_i}{\partial u_k}\right|_{(a,b)}^{T} (u_k - b) \approx 0,$$

using the Jacobian matrices of $c_i$ with respect to $u_j$ and $u_k$, computed at the 'point' $(a, b)$. This equation can be re-written as a linear equation:

$$m_i = H_i u$$

where $u$ is the $n$-dimensional vector of the components of the $u_i$s, and

$$m_i = -c_i(a, b, \lambda) + \left(\frac{\partial c_i}{\partial u_j}\right)^T a + \left(\frac{\partial c_i}{\partial u_k}\right)^T b$$

$$H_i = (0, \ldots, \left(\frac{\partial c_i}{\partial u_j}\right)^T, 0, \ldots, 0, \left(\frac{\partial c_i}{\partial u_k}\right)^T, 0, \ldots, 0).$$

Note that a single constraint does not necessarily generate a single equation. For example, an exact constraint on the location of a 2D point generates two rows. We denote by $dim(c)$ the number of equations generated by a constraint c, and $w = \sum_{i=1}^{r} dim(c_i)$.

The linear operator $H_i$ is a $dim(c_i) \times n$ matrix full of zeros except at the columns corresponding to the DOFs $u_j, u_k$. The values $(a, b)$ around which linearization is performed are initially identical to the *a priori* estimate (see below).

This linearization process can be done for all the constraints $C = (c_1, \ldots, c_r)$, regardless of the number of DOFs that they constrain, yielding the linear system $m = Hu$. The vector $m$

and the matrix $H$ are formed by concatenating the vectors $m_i$ and the matrices $H_i$ vertically. The matrix $H$ is therefore of dimensions $w \times n$, and the vector $m$ is of dimension $w$.

For example, a constraint $c_i(u_j, u_k, d_i)$ stating that the distance between two 2D points should be $d_i$ can be expressed as $\| u_j - u_k \|^2 - d_i^2 = 0$. Linearization around points $a, b$ yields

$$\| a - b \|^2 - d_i^2 = 2(a-b)^T u_j - 2(a-b)^T u_k$$

The appropriate entries in the row vector $H$ are $2(a_x - b_x), 2(a_y - b_y), -2(a_x - b_x)$ and $-2(a_y - b_y)$. The measurement $m_i$ is $\| a - b \|^2 + d_i^2$.

**Uncertainty**

Constraints are viewed as measurements; measurement uncertainty is where we represent the softness of a constraint. A constraint $c_i$ generates a square block $R_i$ on the diagonal of the covariance matrix $R$. An interface is needed for transforming user-intents into suitable covariance matrices.

The equation $m_i = H_i u$ resulting from linearization of the constraint $c_i$ becomes a stochastic equation $\hat{m_i} = H_i u + e_i$ after incorporation of the noise vector $e_i$ associated with the constraint. The Kalman filter does not assume knowledge of $e_i$, but of its covariance matrix $R_i$. A completely rigid constraint has a zero covariance matrix (corresponding to 'zero uncertainty'). A non-rigid constraint has a non-zero covariance matrix. Denote by $\beta_i$ the sub-set of the parameter vector $\lambda$ on which softness is defined in constraint $c_i$ (for example, the distance in the 'distance between two points' constraint). To obtain the covariance matrix $R_i$, linearization is done around $\hat{\beta}_i$ in addition to the participating DOFs. This results in

$$m_i = H_i u + \left.\frac{\partial c_i}{\partial \beta_i}\right|_{\hat{\beta}_i}^T (\hat{\beta}_i - \beta_i).$$

We assume that translation of user specifications regarding the softness of $\beta_i$ to a covariance matrix $R_i'$ of the noise vector $\hat{\beta}_i - \beta_i$ is available. The desired block covariance matrix is given by $R_i = \frac{\partial c_i}{\partial \beta} R_i' \frac{\partial c_i}{\partial \beta}^T$.

We model the error distribution as a multi-dimensional normal random variable with mean zero. The standard deviation of each dimension corresponds to the rigidity of the constraint along that dimension. For example, assume that the softness is defined for a constraint on the location of a 2D point. For better intuition, we can think of the covariance matrix as defining a local coordinate system centered at the desired location of the point, rotated by any desired angle $\alpha$ and scaled by any desired amount along its local axes. The scale factors along the axes are the standard deviations of the random variable. In our implementation (see below) we visualize such a covariance as a rectangle centered at the desired location (Figure 2). The covariance matrix $R$ is obtained by $R = Rot_\alpha^T Sca Rot_\alpha$, where $Sca$ is a diagonal matrix having the standard deviation along the axis $i$ in the entry $(i, i)$ and $Rot_\alpha$ is a rotation by the angle of rotation $\alpha$.

Note that the scale factors on the diagonal are not to be interpreted as defining a 'tolerance' region beyond which the probability of the constraint is zero. The probability of a randomly distributed event is non-zero everywhere. The scale factors only denote the area whose distance from the mean is the standard deviation.

**Local Iterative Kalman Filter**

The Kalman equations (1), when given the input described above, supply a new estimate for the state vector. If all the constraints are linear the filter estimate is the weighted least square solution, hence when the system is solvable the correct solution is immediately obtained. However, non-linear constraints are abundant (e.g. distances). In this case, the exactness of the solution depends upon the points around which linearization was performed. If these are far away from a correct solution, the estimate generated by the Kalman filter is not accurate.

To improve the Kalman estimate we use the *local iterative Kalman filter*, in which the whole process is iterated by using the points estimated by a previous stage as linearization points in the current stage. This is continued until convergence is detected (using standard methods) or until reaching a pre-defined limit for the number of iterations.

It is important to emphasize that this process is not equivalent to a dynamic measurement model. The previous estimate is only used for finding better linearization points; it does *not* replace the *a priori* estimate in the next iteration, since this would mean that a single measurement (constraint) is counted more than once: once in the measurement $m$ and operator $H$ of the present stage, and also in the estimate $u$ from the previous stage. Counting measurements more than once increases their certainty, and in the limit (performing the process an infinite number of times) we would treat *all* constraints as rigid constraints. The process performs multiple iterations of a single time step Kalman filter.

# 5 Properties of the Algorithm

In this section we study some properties of the Kalman filter probabilistic constraints algorithm: its treatment of under- and over-constrained systems, convergence and complexity.

## 5.1 Under-Constrained Systems

A designer given a probabilistic constraints interface can produce under-constrained systems if not required to supply an uncertainty to all the constraints. The user interface can either require this, use a default uncertainty, or both, assuming that constraints that were not defined to be exact are soft. Even in the first option, specifying an uncertainty is much easier than specifying an exact constraint, and it conforms to the relaxed design paradigm when the designer does not care at the moment about the exact nature of the degree of freedom. In any case, it is easy to prevent under-constrained systems altogether.

## 5.2 Over-Constrained Systems

Recall that the Kalman filter equations use the inverse of a matrix $B = H\Sigma H^T + R$. The filter fails when this matrix is singular. In the following we characterize one case when this matrix is singular.

Suppose that all the constraints are rigid, so that the matrix $R$ is zero, and that the *a priori* uncertainty matrix is normalized to be the identity matrix. In this case the Kalman filter attempts to compute the inverse of $HH^T$. $H$ is a $w \times n$ matrix.

Suppose first that $n < w$, in which case $Rank(H) \leq n$, therefore also $Rank(HH^T) \leq n < w$. But $HH^T$ is a $w \times w$ matrix, therefore it is singular. In words, there is no solution when the system is over-constrained (more equations than unknowns). On the other hand, suppose that $w < n$ and that the system is over-constrained in the sense that for almost every choice of linearization points some rows of $H$ are linearly dependent on other rows. In this case $HH^T$ is singular, since $Rank(H) < w$ hence $Rank(HH^T) < w$.

We can summarize the above by saying that when all constraints are rigid, $B$ is singular if and only if the system is over-constrained.

The case when there is no inverse to $B$ can be easily identified in the course of the solution process. When this happens, we can try additional linearization points, since singularity may be a result of a special configuration of these points. When this fails, we can add a non-zero matrix $R$ giving equal uncertainty to all the constraints, resulting in a compromise between conflicting constraints. We also can naturally use the pseudo-inverse of $B$ or attempt to remove dependent rows. However, the optimality properties of the filter would not necessarily hold.

### 5.3 Convergence

Although it has many attractive properties, we should not forget that the Kalman filter is designed for use in linear systems. When there are non-linear constraints, as is the case in all interesting parametric modelers, we are not guaranteed to find a solution. There is no magic here: our algorithm does not guarantee solution of general non-linear constraint systems. It is unique in that it provides a general framework for dealing with probabilistic constraints and their solution. In our implementation convergence was generally satisfactory. Problems occurred mainly when dealing with highly over-constrained models with totally rigid constraints.

### 5.4 Complexity

Each iteration of the Kalman filter involves matrix inversion. Practical algorithms for matrix inversion take $O(n^3)$ time. There are algorithms which are asymptotically more efficient, but it is not clear that they are better in practice [Press88]. The complexity of all other steps (linearization, matrix multiplications) is dominated by that of matrix inversion, hence we can assume that $O(n^3)$ is the complexity of one Kalman iteration. The number of iterations is generally much smaller than $n$ and can be regarded as constant relative to $n$. Consequently, $O(n^3)$ is also the complexity of the whole algorithm. Improved algorithms for inversion of sparse matrices will improve this cubic complexity. It is worth mentioning that there exist hardware implementations of the Kalman filter [Anderson79].

## 6 Results

A simple two-dimensional parametric modeler was implemented to test the correctness of the algorithm. the modeler is described in detail in [Rappoport93]. It is implemented in C and runs under Unix, using SGI GL or X-Windows for graphics and the Motif user interface toolkit. In our implementation all the model DOFs are represented as points. In most cases of interest other degrees of freedom can be represented in terms of points or vectors. For example, an angle

can be represented by three points. This representation simplifies the treatment of the set $D$, since all its members are of a single type.

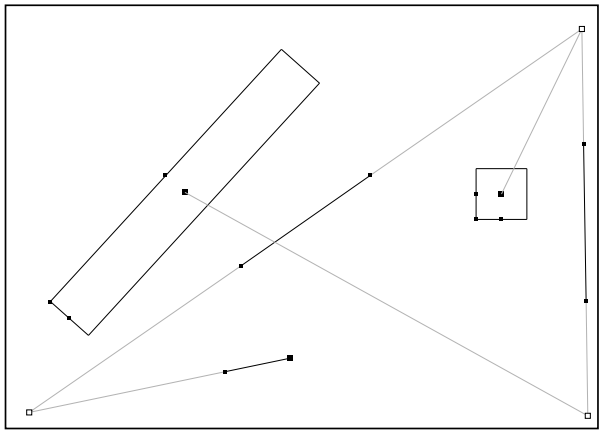

Figure 2: A simple model.

The modeler enables the definition of constraint graphs using the following constraint types: location of a point, co-linearity of three points, the angle between three points, the distance between two points, the distance between a point and a fixed location, and constraining a point to lie on a fixed line. The user interface uses the concept of the direct manipulation device (Dmd) [Emmerik93, Rappoport93] to achieve intuitive and easy-to-use interaction. The interface currently enbales defining softness only on point locations. Linearization equations were derived using the symbolic mathematical package Maple from the University of Waterloo. It was demonstrated that the algorithm is capable of computing solutions to non-trivial models.

The following figures were created using this software. In the figures, model points are shown hollow, fixed points used by constraints are drawn full, distance constraints are visualized by line segments connecting the constrained points, and 'three point co-linearity' constraints are shown by a fixed length line segment connected on two sides and its middle to the points (in the figures showing the situtation after the solution it is hard to distinguish them from distance constraints). Softness is visualized by rectangles, as explained in Section 4. Connections between the visual appearance of constraints and the points they constrain are shown by gray lines. The gray lines also give a visual impression regarding the amount in which the constraints are satisfied.

Figure 2 shows a simple model with three points. The lower-left one is constrained to be on a certain distance from a fixed point and on a different distance from the upper-right point. The distance between the upper-right and lower-right points is constrained. In addition, there are two soft constraints on the locations of the two right points. Figure 3 shows the model after the solution. All required distances are satisfied. We can see that the upper point is farther away from its desired soft location than the middle point from its soft location. This stems from the fact that the standard deviation of the second distribution (visualized by the lengths of the

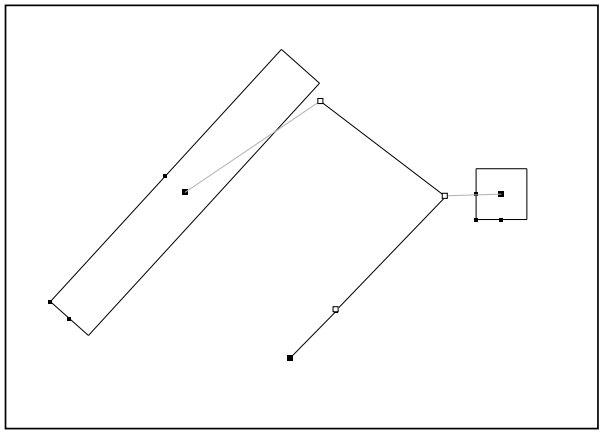

Figure 3: The model after solution.

rectangles' sides) is much smaller.

Figures 4 and 5 show a more complex model, involving 10 points. The points are circularly connected by 10 distance constraints, one of them is distance-constrained from a fixed point, and there are three 'three point co-linear' constraints. In Figure 4 the points were located rather randomly on purpose, to show that the solver can cope with a bad initial guess. Note how the soft point-location constraints enable choosing among the multitude of solutions. On further stages of the design they can be made more rigid, to finalize the guideline for a point's location.

# 7 Discussion

We described *relaxed parametric design*, which uses *soft constraints*, constraints which do not have to be satisfied exactly. We presented the *probabilistic constraints* scheme, which views the model as a stochastic system and constraints as measurements, representing their rigidity by covariance matrices. An estimate of the model's DOFs is computed using the Kalman filter.

The relaxed parametric modeling paradigm is suitable for most geometric design processes in which the design iterates between various levels and components. The designer is not forced to over-work on one level and to make decisions which may limit the freedom of design of other levels. The numerical system of equations which has to be solved in order to compute the degrees of freedom of the parametric model is more relaxed, hence is easier to solve. There are no problems of under-constrained systems, and the user can provide guidelines for choosing between multiple solutions.

Our implementation is 2D and uses a limited (albeit very useful) set of constraints. We plan to extend our implementation to 3D with more complex soft constraints types (e.g., constraining a point to lie on a spline curve or surface). The density function of a constraint can be extended beyond the normal distribution. In addition, ways of integrating our numerical constraints with

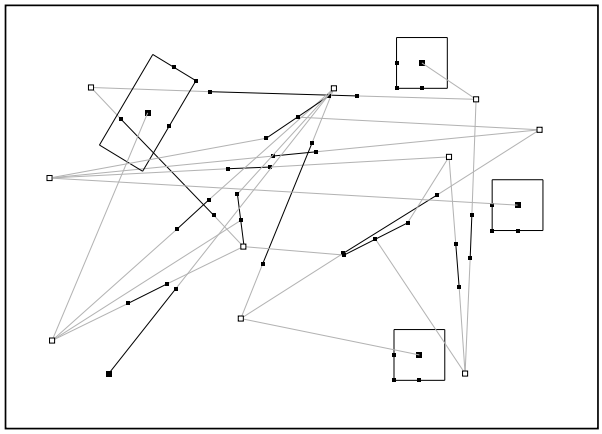

Figure 4: A more complex model.

symbolic and discrete constraints should be investigated.

The algorithm as described in this paper is not very efficient. We are currently developing an efficient incremental algorithm which fuses the constraints one at a time instead of all at the same time.

Inequality constraints can in principle be dealt with using extra variables. It has to be demonstrated that this technique actually works in the context of probabilistic constraints.

Some of our techniques have already been applied to interactive design of smooth objects [Rappoport92] and pose estimation [Hel-Or92a, Hel-Or92b]. The probabilistic constraints scheme can be used for many other applications. Two which immediately come to mind are the representation of tolerances in mechanical CAD and constrained key-frame animation.

# References


[Anderson79] Anderson, B.D.O., Moore, J.B., Optimal Filtering, Prentice-Hall, NJ, 1979.

[Bañares-Alcãntara91] Bañares-Alcãntara, R., Representing the engineering design process: two hypotheses, *Computer-Aided Design*, 23(9):595-603, 1991.

[Borning81] Borning, A.H., The programming language aspects of ThingLab, a constraint-oriented simulation laboratory, *ACM Trans. on Prog. Lang. Sys.*, 3(4):353-387, 1981.

[Emmerik93] Emmerik, M.J.G.M. van, Rappoport, A., Rossignac, J., Simplifying interactive design of solid models: a hypertext approach. *The Visual Computer*, 9:239-254, 1993.

[Hel-Or92a] Hel-Or, Y., Werman, M., Absolute orientation from uncertain data: a unified approach, in: proceedings, *Computer Vision and Pattern Recognition (CVPR)*, pp. 77-82, 1992.

[Hel-Or92b] Hel-Or, Y., Werman, M., Pose estimation of articulated and constrained models, Technical Report, Institute of Computer Science, The Hebrew University of Jerusalem, September 1992.

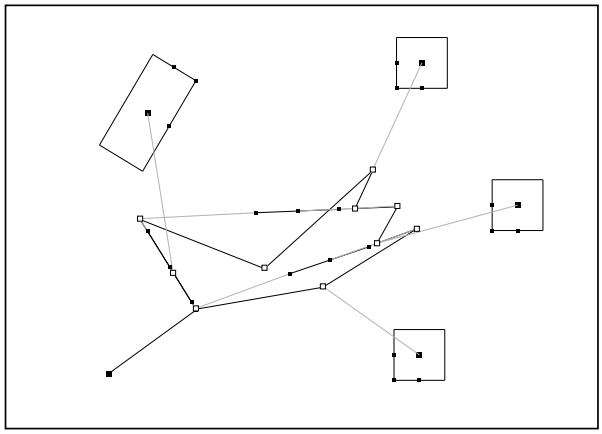

Figure 5: The model after solution.


[Juster92] Juster, N.P., Modelling and representation of dimensions and tolerances: a survey, *Computer-Aided Design*, 24(1):3-17, 1992.

[Lin81] Lin, V.C., Gossard, D.C., Light, R.A., Variational geometry in computer aided-design, *Computer Graphics*, 15(3):117-126, 1981 (Siggraph '81).

[Mäntylä89] Mäntylä, M., Opas, J., Puhakka, J., Generative process planning of prismatic parts by feature relaxation, in Ravani, B. (ed), *Advances in Design Automation – 1989, Volume One, Computer-Aided and Computational Design,* ASME, New York, 1989, pp. 49-60.

[Mäntylä90] Mäntylä, M., A modeling system for top-down design of assembled products, *IBM J. Res. & Dev.* 34(5):636-659, 1990.

[Porter91] Porter, S., Solid modeling: winds of change, *Computer Graphics World*, February 1991, PennWell.

[Press88] Press, W.H., Flannery, B.P., Teukolsky, S.A., Vetterling, W.T., Numerical Recipes in C, Cambridge University Press, 1988.

[PTC87] Pro/ENGINEER: Concepts and Capabilities, Parametric Technology Corporation, Waltham, MA, 1987.

[Rappoport92] Rappoport A., Hel-Or, Y., Werman, M,. Interactive design of smooth objects using probabilistic point constraints, submitted, December 1992.

[Rappoport93] Direct manipulation devices for the design of geometric constraint networks. *Computer Graphics International '93*, Lausanne, June 1993. Published in: Magnenat-Thalmann, N., Thalmann, D. (eds), Communicating with Virtual Worlds, pp. 294-305, Springer, 1993.

[Roy91] Roy, U., Liu, C.R., Woo, T.C., Review of dimensioning and tolerancing: representation and processing, *Computer-Aided Design*, 23(7):466-483, 1991.

[Smithers89] Smithers, T., AI-based design versus geometry-based design, or why design cannot be supported by geometry alone, *Computer-Aided Design*, 21(3):141-150, 1989.

[Sutherland63] Sutherland, I.E., Sketchpad: a man-machine graphical communication system, PhD thesis, MIT, 1963.

[Zhang88] Zhang, Z., Faugeras, O.D., Ayache, N., Analysis of a sequence of stereo scenes containing multiple moving objects using rigidity constraints, *Proc. 2nd Int'l. Conf. on Computer Vision*, 1988, pp. 177-186.