Interactive Design of Smooth Objects with Probabilistic Point Constraints

ARI RAPPOPORT, YAACOV HEL-OR, and MICHAEL WERMAN
The Hebrew University

Point displacement constraints constitute an attractive technique for interactive design of smooth curves, surfaces, and volumes. The user defines an arbitrary number of "control points" on the object and specifies their desired spatial location, while the system computes the object's degrees of freedom so that the constraints are satisfied. A constraint-based interface gives a feeling of direct manipulation of the object. In this article we introduce *soft constraints*, constraints which do not have to be met exactly. The softness of each constraint serves as a nonisotropic, local *shape parameter* enabling the user to explore the space of objects conforming to the constraints. Additionally, there is a global shape parameter which determines the amount of similarity of the designed object to a rest shape, or equivalently, the rigidity of the rest shape.

We present an algorithm termed *probabilistic point constraints* (*PPC*) for implementing soft constraints. The PPC algorithm views constraints as stochastic measurements of the state of a static system. The softness of a constraint is derived from the *covariance* of the "measurement." The resulting system of probabilistic equations is solved using the *Kalman filter*, a powerful estimation tool in the theory of stochastic systems. We also describe a user interface using *direct-manipulation devices* for specifying and visualizing covariances in 2D and 3D.

The algorithm is suitable for any object represented as a parametric blend of control points, including most spline representations. The covariance of a constraint provides a continuous transition from exact interpolation to controlled approximation of the constraint. The algorithm involves only linear operations and allows real-time interactive direct manipulation of curves and surfaces on current workstations.

Categories and Subject Descriptors: G.1.1 [**Numerical Analysis**]: Interpolation—*spline and piecewise polynomial interpolation;* G.1.2 [**Numerical Analysis**]: Approximation—*least squares approximation; spline and piecewise polynomial approximation;* G.3 [**Mathematics of Computing**]: Probability and Statistics—*statistical computing;* I.3.4 [**Computer Graphics**]: Graphics Utilities—*graphics editors;* I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling—*curve, surface, solid, and object representations; splines;* I.3.6 [**Computer Graphics**]: Methodology and Techniques—*interaction techniques;* J.6 [**Computer Applications**]: Computer-Aided Engineering—*computer-aided design (CAD)*

General Terms: Algorithms, Design

Additional Key Words and Phrases: Interpolation, Kalman filter, probabilistic point constraints, relaxed design, shape parameter, smooth objects, soft constraints, splines

Authors' address: Institute of Computer Science, The Hebrew University, Jerusalem 91904, Israel; email: arir@cs.huji.ac.il.

1. INTRODUCTION

Smooth objects such as parametric curves, surfaces, and volumes are of great importance in geometric modeling and computer graphics. To facilitate the usage of such objects a convenient interface for their specification must be provided. The interface must be suitable for all stages in the design process. The strongest requirements are posed by the conceptual design stage, in which ease of specification and experimentation and rapid system response are critical.

Interactive design is usually an incremental refinement process in which the general shape is specified first and then local refinements are made. During local refinement it is of great advantage if the system provides arbitrarily located *shape parameters* with local effect whose tuning aids the user in exploring the space of possible objects. Additionally, control over the amount of deviation from the current object is favorable.

Internally, smooth parametric objects are usually represented as a linear combination of parametric blending functions. The type of blending functions and their domain determine the type and general properties of the object. The coefficients of the blending functions constitute the object's *degrees of freedom* (*DOF*), and determine the actual shape of the specific object.

1.1 Current Design Techniques

Direct Control over Object DOF. In commercial systems the prevalent method for interactive design of smooth parametric objects is by direct control of the user over the DOF defining the object. For example, B-splines are usually specified by manipulating their control points, which are the coefficients of the B-spline blending functions. In this scheme there is a direct relation between the internal object representation and the interface provided to the user. Interactive design using this metaphor is often cumbersome, since the user has to manipulate a large number of DOF to achieve a desired result. It is difficult to assess which DOF should be manipulated and exactly how to do it, and it is close to impossible to place an arbitrary object point in an arbitrary location. Additionally, the necessary display of the control mesh clutters the screen considerably.

Direct-Manipulation Constraint-Based Interface. It is generally believed that the designer should be provided with a higher-level interface than directly setting the object DOF. Recently, several authors have suggested the use of a constraint-based interface, whereby the user selects points or curves lying on the object and specifies a new desired location. The configuration of the DOF that satisfies the constraints is automatically computed.

Bartels and Beatty [1989] and Fowler [1992] use heuristics as to which DOF should be modified; in Borrel and Bechmann [1991], Borrel and Rappaport [1994], and Hsu et al. [1992] the modified DOF are determined by the underlying mathematical representation of the object; typically every constraint influences only a small number of DOF. In these papers the constraints are satisfied using direct linear methods. Borrowing from regularization techniques developed originally for computer vision [Kass et al. 1988;

Terzopoulos et al. 1987], minimization of an energy functional subject to user-defined constraints is performed in Celniker and Gossard [1991], Fang and Gossard [1992], Moreton and Séquin [1992], and Welch and Witkin [1992]. The energy functional is usually based on physical analogues of resistance to stretching and bending; Moreton and Séquin used rate of change of curvature instead.

The constraint-based scheme has the feel of directly manipulating the designed object. The user is not limited to a fixed set of controls; each point on the object can be used as a "control point." Additionally, the system has more freedom in choosing internal representations, since these are not presented directly to the user. Internal representation can be chosen according to desired mathematical properties rather than suitability for direct interaction. Among the new internal representations suggested are refinement and sums of B-splines [Forsey and Bartels 1988; Welch and Witkin 1992], constraint-centered basis functions [Borrel and Rappaport 1994], quintic Hermite patches [Moreton and Séquin 1992], triangular finite elements [Celniker and Gossard 1991], and modulated superquadrics [Metaxas and Terzopoulos 1992].

Drawbacks of Current Methods. A disadvantage in most current methods is that the user can specify only completely rigid constraints. In many situations what the user actually wants is to specify a general position for an object point rather than an exact position. The fact that constraints are treated as completely rigid constitutes *overspecification*. Instead, it is more attractive to let the user define a *softness* for a constraint which specifies the degree to which the constraint must be satisfied. Such a technique would be in accord with the *relaxed-design* modeling paradigm [Hel-Or et al. 1993], in which the designer is not committed to early design decisions.

In some sense, load forces [Celniker and Gossard 1991] do provide some form of softness, but only in the context of variational modeling. The combination of a triangular finite-element internal representation, a direct-manipulation constraint-based user interface, and functional optimization is attractive for many reasons. However, global optimization has two drawbacks. First, many users claim that it takes away control; they actually like the feeling of controlling every little piece of the object. Second, these methods are currently not suitable for truly interactive design of large objects. Current algorithms for real-time minimization of the energy functional while interactively manipulating constraints do not scale well to large problems [Kallay 1993]. We do not rule out the usage of such techniques for fairing an existing design, nor their future applicability, but currently only the techniques which use direct linear methods [Borrel and Bechmann 1991] provide real-time performance that scales to models with a large number of DOF.

Cheng's interproximation scheme [Cheng and Barsky 1991] can in principle be regarded as providing soft rectangular constraints. However, the scheme was only presented for curves, and it is too slow for interactive design.

Shape Parameters. As stated above, a very attractive feature of a design scheme is the provision of shape parameters that let the user navigate inside

the space of objects "close" in some sense to the current object, for the purpose of local refinements. Interactive manipulation of shape parameters suggests to the user a multitude of objects to choose from.

The Beta parameters in Beta-splines [Bartels et al. 1988] and the control point weights in the NURBS representation [Farin 1992] are shape parameters. However, no one has yet formulated these in terms of constraint-based interaction. The stress, shear, and bend coefficients, and the load vectors in physically-based methods [Celniker and Gossard 1991; Welch and Witkin 1992], can be viewed as shape parameters, but their application is currently too time consuming for interactive design. Additionally, these papers have not described any user interface for defining shape parameters in arbitrary localized regions.

In many cases, the importance of staying "close" to the current object during automatic computation of DOF is somehow neglected. Welch and Witkin [1992] suggest defining an arbitrary situation as a "rest shape." However, the issue is mostly taken care of by keeping the set of constraints that the user used in order to define the object and using them in each stage of the computation.

1.2 Soft Constraints on an Existing Object

The Proposed Scheme. In this article we introduce the *soft-constraints* scheme for interactive design of smooth objects. In this scheme, the user starts from an existing object and pulls it by using soft constraints, constraints which are not completely rigid and do not have to be met exactly.

Each constraint possesses a user-determined softness tensor. Softness is nonisotropic; it can be defined differently along different spatial directions. This softness determines the amount by which the new object is similar to the currently existing one locally, in the vicinity of the constraint. Hence, it serves as an arbitrarily located nonisotropic local shape parameter.

Additionally, a global shape parameter is available for determining the global amount of deviation of the new object from the existing one. This shape parameter can be tuned to make the existing object very rigid, necessitating a strong pull to modify, or very soft.

Following are some figures which serve as simple examples for the proposed scheme. Figure 1 shows an existing flat surface (dark, bottom) with three soft constraints that yield the upper, lighter surface. Figure 2 shows the same situation but with a larger softness of the top constraint. The surface area near the constraint is less affected by it and conforms more to the original surface. Figure 3 shows an identical configuration but with a greater weight of the existing surface. The new surface is globally less affected by all of the constraints. Finally, in Figure 4 the designed surface has been made the reference object, and new constraints have been defined on it. Once the designer is satisfied from the overall shape of an object, it can be made a reference object, and a new refinement design cycle commences.

Contribution. The three basic ingredients of the proposed scheme are softness of a constraint, control of similarity to the existing object, and truly interactive performance. No previous method possesses all three ingredients

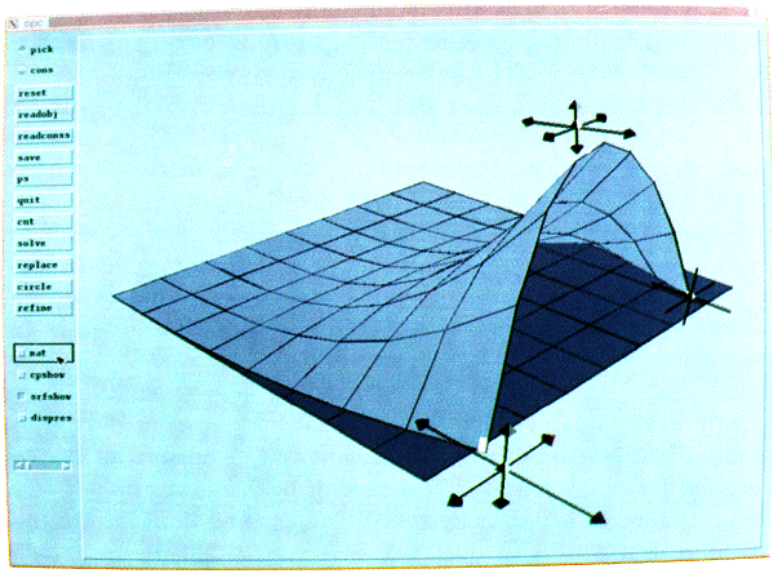

Fig. 1. Three constraints defined on a surface. The one on the right has a zero covariance.

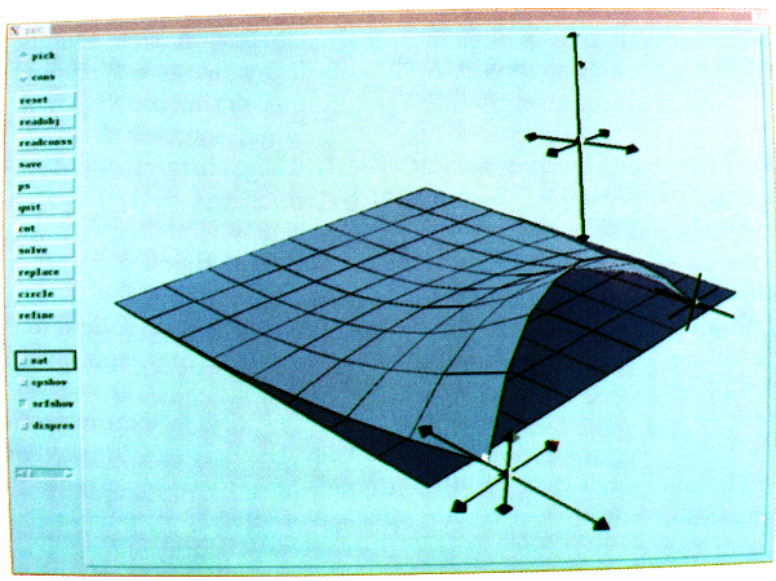

Fig. 2. The same constraints as in Figure 1, but with a much larger covariance of the top constraint. Note how the surface is less attracted to the constraint.

together. Energy minimization methods provide some form of softness and control of similarity, but they take away some of the user's control over the designed object and do not scale well to interactive design of large objects. Direct linear methods, although efficient and sometimes providing control of similarity, do not provide softness.

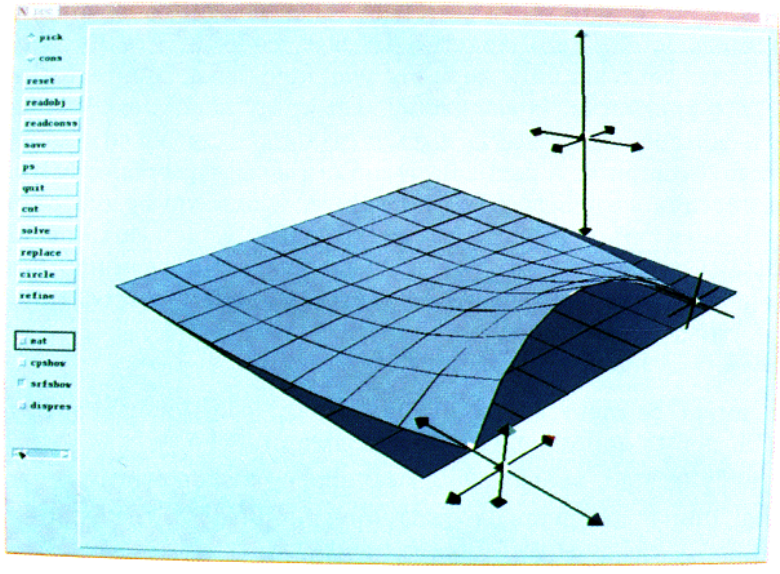

Fig. 3. The same constraints as in Figure 2, but with a smaller initial covariance estimate. Note how the surface is closer to the initial one.

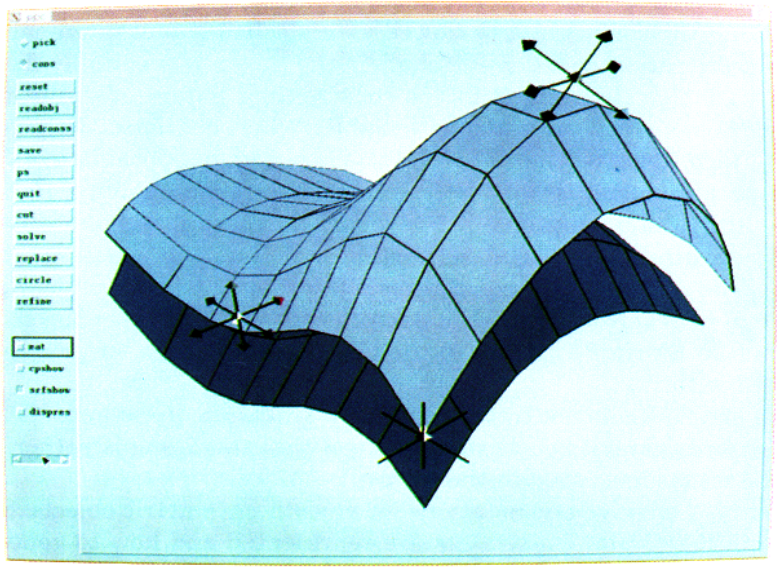

Fig. 4. The designed object has been made the reference object.

Our method bears closest resemblance to weighted least squares. One could think of augmenting the constraints in the direct linear methods with non-isotropic weights, whose relative sizes would serve as softness parameters. However, this scheme does not provide control of similarity to the existing object. Additionally, for the weights to act as softness parameters there would

have to be more constraints than DOF; otherwise all constraints would be exactly interpolated, nullifying the softness concept. Finally, we are not aware of any work actually applying weighted least square to interactive design. This application is not trivial, since any such work would have to provide a suitable user interface. Our Kalman filter-based algorithm (see below) in fact computes a particular kind of a least-squares solution.

The algorithm is applicable to every object which can be represented as a linear combination of parametric blending functions; this includes most spline-based representations. The equations solved by the algorithm are all linear, and no global optimization is performed, enabling real-time interaction on current workstations.

The advantages of our algorithm can be summarized as:

—Provision of softness to constraints and a continuous transition from interpolation to approximation of a constraint.
—Arbitrarily located, nonisotropic, local shape parameters.
—A global shape parameter for controlling similarity to the existing object.
—Real-time performance on current workstations.
—It is suitable for most parametric smooth object representations.

Method of Implementation The scheme is implemented using a novel algorithm, which we term *probabilistic point constraints* (PPC). The process of specifying the constraints is viewed as a stochastic measurement process. The desired location of an object point is the mean of a suitably distributed random variable. The rigidity of the constraint is determined by the uncertainty (covariance) of the random variable. When equipped with a suitable user interface, the covariance can be used as a shape parameter. Moreover, this shape parameter is both local and nonisotropic, enabling fine control. The resulting system of probabilistic equations is expressed and solved using the Kalman filter, a powerful estimation tool from the theory of stochastic systems. The objects' DOF constitutes the process's state vector. Control of the amount of deviation from the currently existing object is achieved through the use of an a priori estimate to the object's DOF and a covariance matrix associated with it.

In this article we do not treat the issue of automatic insertion of DOF when the constraints cannot be satisfied. We feel that this issue is rather orthogonal to the issues discussed in the article.

In Section 2 we give our notations for smooth parametric objects. Section 3 explains in detail how constraints are represented and how to generate and solve the system of probabilistic equations using the PPC algorithm. Section 4 discusses some properties of the algorithm, most notably, the role of the covariance as a shape parameter. Finally, our implementation and user interface issues are discussed in Section 5.

2. REPRESENTATION OF A SMOOTH PARAMETRIC OBJECT

The probabilistic point constraints (PPC) algorithm presented in this article is suitable for any object represented as a linear blend of "control points." The

only strict requirement is linearity in the DOF. This requirement poses no practical limitation, since most spline representations conform to it. In this section we introduce notations regarding object representation which are used in the rest of the article. The object does not have to be parametric, although it is more convenient to describe it as such.

A smooth parametric object is represented by

$$w(u) = \sum_{i=1}^{n} b_i(u) p_i \tag{1}$$

where $u \in U \subset R^d$ is a point in a parameter space; $b_i(u)$ is a scalar-valued *blending function* defined on U; and p_i is a point in R^k. In most applications k is 3 or 2. A point on the object corresponding to a particular parameter value is a linear combination of "control points" whose dimensionality k is that of the space in which the object resides, such that the scalar coefficient of a control point is obtained as the value of a blending function taken at the parameter. We refer to the set of $m = kn$ components of the p_i's as the *degrees of freedom* determining the object. In the context of specification by constraints, we refer to the p_i's as the *internal control points* of the object.

In the case of curves, the dimension d of the parameter space U is one, and the blending functions are simple one-dimensional functions. Common functions are the Bézier and B-spline blending functions, for which the "control points" have a clear geometric meaning [Bartels et al. 1988; Farin 1992]. In the case of surfaces and volumes the blending functions are usually defined as tensor products of two and three one-dimensional functions or as polynomials defined over triangles and tetrahedra, respectively.

Equation 1 can be written as

$$w(u) = (B_1(u) \ldots B_n(u))p = B(u)p \tag{2}$$

where p is an m-dimensional vector concatenating all the degrees of freedom, $p^T = (p_1^T \ldots p_n^T)$; $B_i(u) = b_i(u)I$ is a $k \times k$ identity matrix scaled by the value of a blending function, and $B(u)$ is a $k \times m$ matrix obtained by horizontal concatenation of the $B_i(u)$'s (Figure 5).

3. PROBABILISTIC POINT CONTRAINTS (PPC)

In the previous section we saw that an object is represented internally using a vector p of DOF. The user does not manipulate p directly. Rather, the user grabs a point on an existing object and drags it to a desired location, specifying its rigidity and the desired amount of similarity to the current object. The system automatically computes a vector p that satisfies the constraints. In this section we describe how constraints are represented and present the PPC solution algorithm.

3.1 Representation of Constraints

The central idea is the probabilistic constraints scheme is to treat the point constraints as stochastic measurements of the state of a static system. The

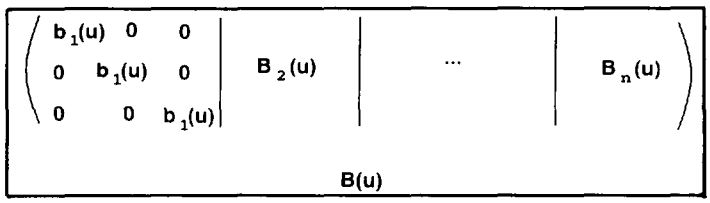

Fig. 5. The structure of the matrix $B(u)$ defining a parametric object, in the case when $k = 3$.

general stochastic-measurement model of a static system is

$$\hat{q} = f(x) + e, \tag{3}$$

where x is the vector of parameters describing the *state* of the system; $f(x)$ is a function of the state which can be measured; $\hat{q}$ is the vector of actual measurements; and e is the measurement *noise*, whose covariance (or *uncertainty*) is assumed known [Anderson et al. 1979; Schweppe 1973]. The central problem in the theory of such systems is to *estimate* the state vector x from the measurement vector q. By convention, actual measurements and estimates are denoted with hats ($\hat{q}$) while "true" measurements (without the noise vector e) are written without the hat. We will omit the hats for clarity since there are no true measurements in our model. In a general stochastic system both the model and the measurement process can be time varying (dynamic). We present only the static version for simplicity of notation.

A probabilistic point constraint c_i is represented as a tuple $c_i = (q_i, u_i, R_i)$ which specifies a desired spatial location q_i with a covariance matrix R_i for a parameter vector u_i. R_i is a $k \times k$ matrix where k is the dimension of q_i. We refer to the q_i's as *external control points* of the object. Naturally, the user is not expected to deal directly with covariance matrices. A possible user interface for specifying the softness of a constraint is given in Section 5.

The r constraints must satisfy a system of equations

$$q_i = B(u_i)p + e_i \quad i = 1 \ldots r, \tag{4}$$

which can be written as

$$q = Hp + e \tag{5}$$

where q is a kr-dimensional vector which concatenates all desired locations $q^T = (q_1^T \ldots q_r^T)$; H is a $kr \times kn$ matrix obtained by vertical concatenation of the matrices $B(u_1) \ldots B(u_r)$; and e is a kr-dimensional vector concatenating the vectors e_i, $e^T = (e_1^T \ldots e_r^T)$.

Equation (5) has the same form as Eq. (3) where the system's state is the vector p of the object's degrees of freedom; the state function being measured is the linear operator H, and the measurements are the desired locations of the constraints. The constraint location is the position with highest probability for the specified object point, but other locations have a nonzero probability. The covariance of a constraint determines principal directions for the softness of a constraint and its magnitude along each direction.

As will be explained below, we do not need to have explicit knowledge of the vector e, only of its covariance matrix R. R is a $kr \times kr$ matrix which is all zeros except at r blocks on the main diagonal, of size $k \times k$ each. Each such block is the covariance matrix R_i of the noise e_i associated with constraint c_i. The upper row in Figure 8 shows the matrix R (left) and the corresponding matrix H (right) generated by three constraints. Zero entries in the matrices are shown in white. One of the constraints has a zero covariance, which is reflected as a white block of R. Figure 8 is fully discussed in Section 5.3.

We say that an internal control point p_i is *affected* by a constraint c_j if $B_i(u_j) \neq 0$. We will use this term in Section 4.

3.2 Kalman Filter Solution

The main tool which we use is the *Kalman filter*, which estimates the state of a stochastic linear system from measurements. Here we briefly describe the Kalman filter for static systems and measurement models, and state some of its properties. For a complete discussion see Anderson and Moore [1979] and Schweppe [1973], for example.

The static Kalman filter is suited for a measurement model $q = Hp + e$, i.e., a special case of Eq. (3) with a linear operator H relating the state vector p and the measurements q. The filter needs to be given the covariance matrix of the noise (E denotes the expectation operator)

$$R = E\{ee^T\}. \tag{6}$$

Also provided are an a priori estimate of the state vector with its associated covariance

$$E\{p\} = p_0, \quad E\left\{(p - p_0)(p - p_0)^T\right\} = \Sigma_0. \tag{7}$$

For interactive design there are three main alternatives for the a priori estimates: the current state vector, a fixed reference object, and a very large uncertainty. These are elaborated upon in Section 4.4. A "black-box" view of the filter is shown in Figure 6.

The Algorithm Given an a priori estimate (p_0, Σ_0), measurements (q, R), and measurement model H, the Kalman filter first computes the *Kalman gain matrix* K

$$K = \Sigma_0 H^T (H\Sigma_0 H^T + R)^{-1}. \tag{8}$$

When the matrix $H\Sigma_0 H^T + R$ is singular, its pseudoinverse [Boullion and Odell 1971] is used instead of the inverse. The gain matrix is then used to produce an estimate p to the state vector

$$p = p_0 + K(q - Hp_0) \tag{9}$$

and, if needed, an estimate Σ to the uncertainty of p

$$\Sigma = \Sigma_0 - KH\Sigma_0. \tag{10}$$

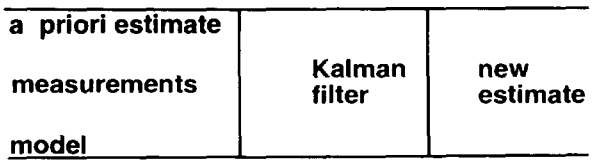

Fig. 6. A black-box view of the static Kalman filter.

The process as described is a one-step process, not an iterative one, since we treat all measurements at once. An alternative is to fuse the constraints one by one, by using a matrix H that corresponds to a single constraint, estimating p and Σ, and using these as a priori estimates for the next constraint. The latter option can be more efficient when there is a large number of constraints (see Section 4.5). The Kalman filter has the following statistical properties

—It is an unbiased estimator of p (i.e., the estimator has the same mean as p).

—It is optimal in the sense of linear minimal variance (i.e., among all linear estimators of a certain form it produces the smallest unconditional error covariance matrix, hence yielding the minimum squared error).

—When the noise e is normally distributed 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).

Note that we can neglect the a priori estimate and treat all constraints as completely rigid by setting $e = p_0 = 0$, $\Sigma_0 = I$. In this case, and if H and HH^T are square and nonsingular, the filter estimates p to be $p = H^{-1}q$, which is the obvious way of satisfying the constraints. In this degenerate case the PPC algorithm behaves similar to the algorithms in Borrell and Bechmann [1991], Borell and Rappoport [1994], and Hsu et al. [1992].

4. PROPERTIES OF THE PPC ALGORITHM

In this section we discuss a few properties of the PPC algorithm. The most important of these is the fact that varying the covariance of a constraint yields a continuous transition from exact interpolation to a controlled approximation; thus the covariance can serve as a shape parameter. We also discuss the influence of the a priori estimate, the situations in which the algorithm fails, and its complexity.

4.1 Existence of a Solution and Filter Failure

When the covariance matrix of a constraint is not completely zero, the constraint has a nonzero probability of being anywhere in space; hence every estimation of the DOF produces "a solution." The only situation in which a constraint may not be satisfied is when its covariance is zero, i.e., when exact interpolation is required. Section 4.2 shows that when the system is not

overconstrained the filter achieves exact interpolation for a zero covariance. Here we show that the situations in which the filter fails correspond to overconstrained systems.

By filter failure we mean singularity of the matrix $H\Sigma_0 H^T + R$, whose inverse is required during the computation of the Kalman gain matrix (Eq. (8)). Denote $B = H\Sigma_0 H^T$. Let us first assume that $R = 0$, i.e., exact interpolation is required for all the constraints. In this case we are interested in the singularity of B.

Recall that for every $n \times m$ matrix A we always have $Rank(A) \leq \min(n, m)$, that for every two matrices A, B we have $Rank\ (AB) \leq \min(Rank(A), Rank(B))$, and that a square matrix is singular if and only if its rank is smaller than its dimensions. In our case,

$$Rank(B) = Rank(H\Sigma_0 H^T) \leq \min(Rank(H), Rank(\Sigma_0)) \leq \min(kn, kr).$$

Suppose that the number r of constraints is greater than the number n of internal control points, in which case $Rank(B) < kr$. B is $kr \times kr$; hence it is singular.

We have shown that a necessary condition for B to be nonsingular is that the number of constraints is not greater than the number of internal control points. This condition is intuitive, since in overconstrained systems it is in general impossible to satisfy all the constraints.

In fact, by repeating the above analysis for submatrices of B, we can phrase a more precise necessary condition: for every subset of the constraints, their number should not exceed the number of control points affected by them. This condition is more restricted than the previous one, since the total number of constraints may be smaller than the total number of control points but greater than the number of affected control points.

Another obvious necessary condition is that there are no two constraints defined for the same parameter value, since this would cause identical rows in the matrix H and reduce its rank.

So far we have dealt with the more difficult case, $R = 0$. When $R \neq 0$ the situation improves, since R is block diagonal, which raises the chances that its addition to another matrix B will result in a nonsingular matrix.

When $B + R$ is singular we use its pseudoinverse instead of the inverse. The statistical properties of the filter do not hold anymore, but the effect is nonetheless what the user expects. The pseudoinverse computes a least-square solution, which is the type of solution produced by the Kalman filter. Additionally, practical experience with the application of the pseudoinverse for interactive design [Bechmann and Dubreuil 1992; Borrell and Rappoport 1994; Hsu et al. 1992] shows that it conforms to user's intuition.

4.2 Influence of the Covariance of a Constraint

The main novelty in the PPC algorithm is the usage of covariances. The covariance matrix associated with a constraint determines its rigidity, i.e., the amount by which the constraint is satisfied, and enables a continuous

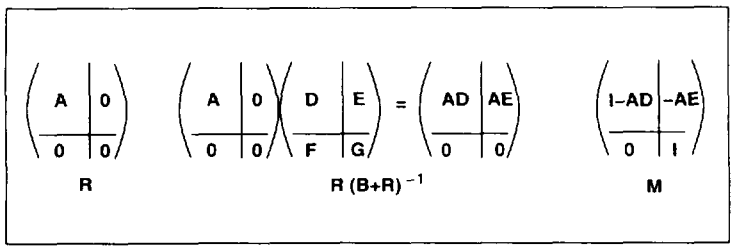

Fig. 7. The matrices R, $R(B + R)^{-1}$, and M.

transition from exact interpolation to a controlled approximation. In the following we prove these statements.

Denote $M = HK = B(B + R)^{-1} = H\Sigma_0 H^T(H\Sigma_0 H^T + R)^{-1}$. Assume that the matrix $B = H\Sigma_0 H^T$ is nonsingular, i.e., the system is not overconstrained. Without loss of generality, assume that R is arranged such that the diagonal blocks corresponding to constraints with nonzero covariances come before those with zero covariances. Denote the nonzero block of R by A (Figure 7.) We have $I = (B + R)(B + R)^{-1} = B(B + R)^{-1} + R(B + R)^{-1} = M + R(B + R)^{-1}$, hence

$$M = HK = I - R(B + R)^{-1}.$$

Each zero $k \times k$ block on the diagonal of R generated from a zero covariance matrix R_i of a constraint induces a corresponding $k \times k$ identity block on the diagonal of M; the other entries in the rows of this block are zero. We refer to such rows as an "identity row block" (Figure 7, right).

The new internal control points given by the Kalman filter are $p = p_0 + K(q - Hp_0)$ (Eq. (9)). Premultiplying each side of the equation by H, we obtain

$$Hp = Hp_0 + M(q - Hp_0).$$

For a vector x in the dimensions of q, denote by x' the part of x corresponding to constraints with zero covariance. For each identity row block of M the terms $(Hp_0)'$ and $M(Hp_0)'$ cancel each other, and we get $(Hp)' = q'$. Conversely, $(Hp)' = q'$ is obtained only when the corresponding rows in M constitute an identity row block. In other words, a constraint is satisfied if and only if its covariance is zero.

Note that when a $k \times k$ block in R approaches zero, the corresponding rows in M approach an identity row block. The smaller the entries in a constraint's covariance matrix, the farther the constraint gets from the initial estimate p_0 and the closer it gets to exact interpolation. It is now clear that the covariance can serve as a shape parameter to explore the range of objects lying "between" the initial estimate and the exact interpolant object.

Figure 8 shows the matrices R (top left), H (top right), M(bottom left), and $I - M$ (bottom right) resulting from three constraints. The last constraint has a zero covariance, and we see a corresponding zero block in R and $I - M$ and

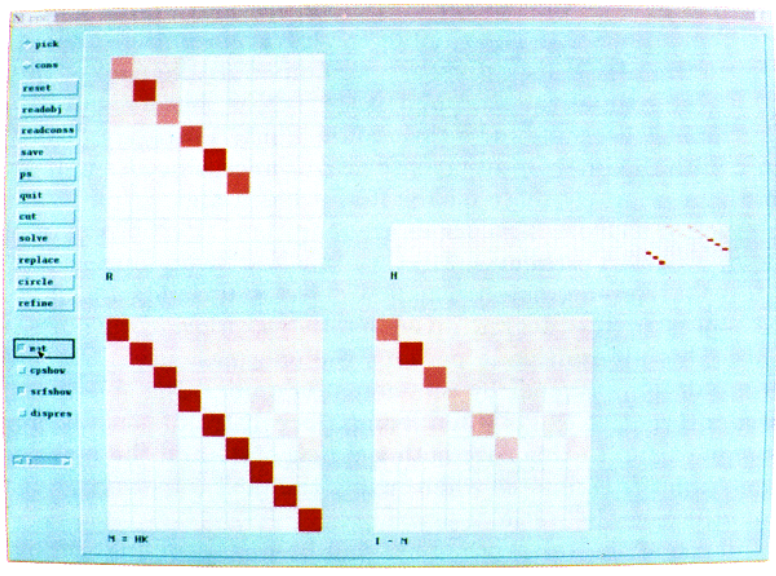

Fig. 8. The matrices R (upper left), H (upper right), M(lower left), and $I - M$ (lower right) resulting from the situation in Figure 1.

an identity row block in M. Of the other two constraints, one of them has a larger covariance, which shows as more saturated color in R. We see that the corresponding entries in $I - M$ have a larger absolute value.

4.3 Estimated Covariance

The Kalman filter gives an estimate $\Sigma = \Sigma_0 - KH\Sigma_0$ to the covariance of the estimated state vector p (Eq. 10). Premultiplying both side by H we obtain

$$H\Sigma = (I - M)H\Sigma_0.$$

Again, for every zero constraint covariance we have a corresponding identity row block of M and a corresponding zero block of $H\Sigma$. The equation is true for every H; hence the corresponding block of Σ must be zero. In other words, when a constraint is supposed to be satisfied its estimated uncertainty is zero, which is what we would expect. The norm of a Σ block corresponding to a constraint is a measure of the desired degree of rigidity of the constraint.

Naturally, if an estimated covariance is not needed there is no reason to compute Σ at all. The estimated covariance is useful for automatic refinement schemes [Welch and Witkin 1992] in which a measure for the desired degree of satisfaction of a constraint is needed. In addition to this, the estimated Σ can be given as an a priori estimate to the next solution stage.

4.4 A Priori State Estimate

The Kalman filter equations (8), (9), and (10) utilize a priori estimates of the state vector and its covariance. The new estimate is directly influenced by the

a priori estimates. The first factor on the left of the Kalman gain matrix is the a priori covariance Σ_0. The smaller this covariance, the smaller the gain matrix. Since the new state estimate is computed as the sum of the a priori state and a vector times the gain matrix, it is clear that a small gain matrix will cause the new estimate to be close to the a priori estimate. This behavior is in accord with the interpretation of the covariance, since a small covariance matrix of the a priori estimate means that it has a high certainty.

For interactive shape design, there are four main alternatives in providing the necessary a priori estimates. First, we can supply an a priori state vector defining a fixed reference object. The new object will "follow the shape" of the reference object to the degree given by the a priori covariance and the constraints. This alternative is perhaps the one most useful for interactive design, since design proceeds by refinement of an existing object.

A related alternative exists when fusing the constraints one by one instead of at the same time. In this case both the new state and the new covariance are used as a priori estimates when fusing the next constraint, but the first constraint still uses the initial a priori.

The influence of the reference object can be made nonuniform by using a more complex a priori covariance matrix. The matrix can be composed of diagonal blocks, each one corresponding to an internal control point, and can have different values on the diagonal of each block whose relative sizes weight the influence of the internal control points of the reference object. The matrix can even be arbitrary, thereby modifying the preferred direction of influence.

A third alternative is to set the a priori estimate to be the current state while the user interactively modifies the location of constraints, and initialize the a priori covariance each time. The object smoothly follows the constraints as they are manipulated. In this mode an almost exact interpolation is achieved, since the initial estimate and the new object are very close to each other. These two methods can also be used in physically based design [Welch and Witkin 1992].

Finally, we can force the system to practically neglect the a priori estimate by using a very large matrix Σ_0, corresponding to a high uncertainty. By a very large matrix we mean a diagonal matrix with entries that are large relative to the numbers in which the other covariance matrices are expressed.

4.5 Complexity

Each iteration of the Kalman filter involves matrix inversion and several matrix multiplications. Practical algorithms for computing the inverse and the pseudoinverse of an $h \times h$ matrix take $O(h^3)$ time. There are algorithms which are asymptotically more efficient, but it is not clear that they are better in practice [Press et al. 1988].

The dimensions of the matrix that is being inverted are $kr \times kr$. The largest dimensions of multiplied matrices are those involving the state vector p, $kn \times kn$. Only when there are more constraints than internal control points, the complexity of all steps is dominated by that of matrix inversion. Consequently, $O(r^3)$ is the worst-case complexity of the algorithm. Improved

algorithms for inversion of sparse matrices can improve this worst-case behavior.

Practically, however, the algorithm's performance is much better. In many cases r can be counted as a constant, and the complexity is only $O(n^2)$. When the blending functions that define the smooth object (cf. Eq. (1)) possess finite local support, as with B-splines and finite elements, each constraint will affect only a small number of internal control points. We can exclude from the estimated state vector p all those control points that are not affected, since they are guaranteed not to be modified. This includes all internal control points whose regions of support in parameter space do not contain the parameter value of any constraint. This exclusion can significantly improve the performance of the algorithm, since in most cases design is done incrementally by local refinement, so at each manipulation stage the constraints will influence only a small subset of the internal control points.

Finally, performance can also be improved by using hardware implementations of the Kalman filter [Anderson and Moore 1979].

5. IMPLEMENTATION

A prototype system for experimenting with probabilistic point constraints was implemented in C under Unix, on Sun and Silicon Graphics workstations using the X/Motif user interface toolkit, X and GL for 2D graphics, and GL for 3D graphics. The system enables interactive design of curves and surfaces and visualization of the matrices used during the solution. The internal representation chosen for smooth objects is the B-spline representation, because B-splines are very common in computer graphics and geometric modeling. Our emphasis was on experimenting with the probabilistic constraints methods more than on investigating internal representations. It may well be that B-splines are not the best internal representation for constraint-based interfaces. Some of our work on internal representations is described in Borrel and Rappoport [1994].

5.1 Covariance as a Local Coordinate System

The main novelty of the system from the user's point of view is the ability to define a *softness* for each constraint. Internally, softness is represented as a covariance matrix. A conceptual model for intuitive reasoning about covariance matrices is needed.

The system uses a normally distributed, zero-mean multidimensional noise vector e. The standard deviation along each principal dimension corresponds to the rigidity of the constraint along that dimension. The normal distribution yields a softness which is symmetric along its principal axes. The standard deviations along the directions can be different, yielding a nonisotropic softness. This is advantageous since it enables the user to use the covariance for pulling and pushing the object in a certain arbitrary direction.

We can think of the covariance matrix as defining a local coordinate system (Lcs) centered at the desired location of the point, rotated according to the desired principle directions, and scaled by any desired amount along its local

axes. The scale factor along an axis could be, for example, the variance of the random variable on that axis.

Let S be the diagonal matrix such that $S_{i,i}$ is the standard deviation along axis i. Let L be the rotation matrix that brings the global coordinate system to the desired local coordinate system of the constraint. The covariance matrix R is obtained by $R = L^T S L$, a similarity transform that modifies the eigenvectors of S to coincide with the desired principal directions.

5.2 Direct-Manipulation Device User Interface

An interface is needed for transforming user's intents into local coordinate systems, which in turn will be converted to covariance matrices. To visualize and specify Lcs's we use direct-manipulation devices (Dmds). A Dmd is an abstract data type augmented with a visual representation and with methods for interactive specification of the operations defined for the abstract data type [Rappoport 1993]. Our naming convention is to call a Dmd whose data type is Name by DmdName.

We experimented with two DmdLcs's, one in 2D and one in 3D. In 2D, the Lcs is visualized as a rectangle centered at the Lcs's location, scaled and oriented appropriately. In 3D, it is visualized as a 3D cross. Interaction is mostly done using the left mouse button. Translation, Scale, and Rotate handles are provided to interactively support these operations. In 3D, the translation handle operates like the well-known 3D cursor of Nielson and Olsen [1986]. Scale and Rotate are overloaded into the same handle, and the distinction between them is done using gestures. The parameters to the operations are computed such that the handles follow the mouse. The middle mouse button is used to toggle between degenerate transformation (e.g., zero scale) and the previous Lcs. The 3D DmdLcs and other Dmds are described in detail in Emmerik et al. [1993].

Note that the rectangle is 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 nonzero everywhere, unless a zero covariance is used (in which case the rectangle degenerates into a point anyway). The rectangle only serves for visualization and direct manipulation of the covariance. The choice of the standard deviations as the rectangle scale factors is completely arbitrary; any consistent choice would be as suitable. Because of this possible misunderstanding, we concluded that a cross (either 2D or 3D) is more suitable than a rectangle or a box for interacting with covariances.

5.3 Examples

We describe some example images created using our prototype implementation. Figure 1 shows three constraints on a flat reference surface. The reference surface is shown in a darker color than the color of the designed surface. The user can turn off the display of the reference object if desired. The constraint on the right has a zero covariance, which is visualized by not showing the scale and rotate handles on the Dmd. The white spots on the

designed surface are the preimages of the constraints. We can see that the surface exactly interpolates the constraint with the zero covariance.

Figure 2 shows the same situation, but after increasing the covariance of the top constraint along one of its axes. The surface is less attracted by the constraint. Figure 3 shows the same constraint, with a decreased initial covariance estimate, making the designed surface more similar to the reference surface. A value of zero for the initial covariance freezes the designed surface to be identical to the initial one. In Figure 4 the designed surface has been made the reference object, and new constraints have been defined on it.

Figure 8 visualizes some matrices resulting from the situation in Figure 1. Zero entries are shown in white, while the absolute value of nonzero entries is encoded into saturation of the color. Larger values are shown in a deeper saturation. The matrix R of the covariances of the constraints is shown on the top left. R is a 9×9 matrix since there are three constraints, each lying in 3D. The diagonal block corresponding to the zero covariance constraint is white. The upper block is not diagonal since the Dmd of the uppermost constraint was scaled nonisotropically and rotated.

The matrix H is 9×48, since there are three 3D constraints (defining the number of rows) and 16 3D internal control points (defining the number of columns). Note the block structure of the matrix (cf. Figure 5). Most of the entries are zero (white) because two of the constraints were defined on extreme parameter values, for which the surface basis functions are zero. The matrices M and $I - M$ are 9×9, like R. Note that the zero block in R results in an identity row block in M, which causes a zero block in $I - M$. This zero block guarantees that the constraint will be exactly interpolated since the system is not overconstrained.

Figure 9 shows a 2D face. The small squares on the curve serve for visualization of some of the B-spline knots, to convey a feeling for the amount and location of the DOF available. Overall, there are hundreds of DOF. The reference object for the face is a circle. The face was designed using about 20 constraints, but only one of them is shown, connected to the mouth. In Figure 10 the covariance of this constraint along one of its axes was increased, pushing the mouth toward the reference circle. Interactive modifications of the orientation and size of the rectangle produce many subtly differing mouth shapes. The designer "plays" with the rectangle to let the system suggest many shapes similar to the designed one, among which the most satisfying can be chosen.

6. CONCLUSION

In this article we presented soft constraints for interactive design of smooth objects. We described the probabilistic point constraints (PPC) algorithm for implementing and solving soft constraints. Each constraint has an associated covariance matrix, which can serve as a local, nonisotropic shape parameter. The covariance provides a means for continuously moving from exact interpolation to approximation of a constraint. The algorithm is linear, enabling real-time interactive design. We demonstrated the algorithm for curves and

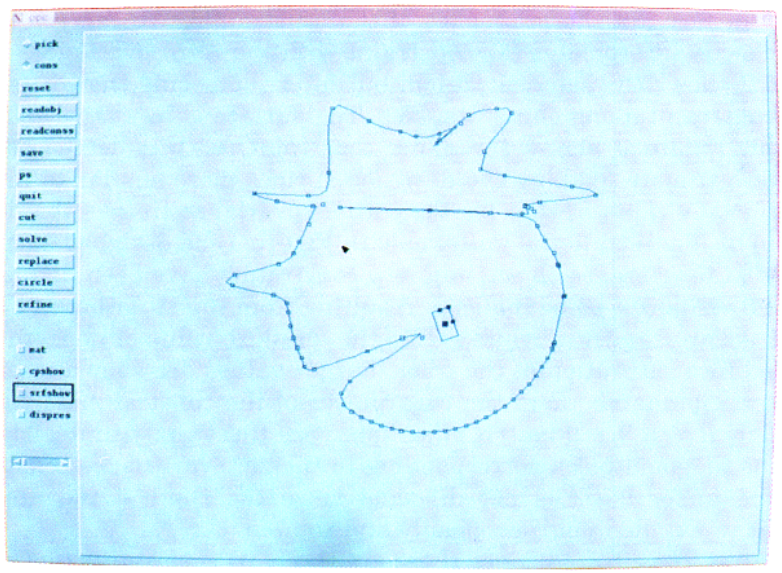

Fig. 9. A 2D face designed with probabilistic constraints. The reference object is a circle. Only a single constraint is shown, on the mouth.

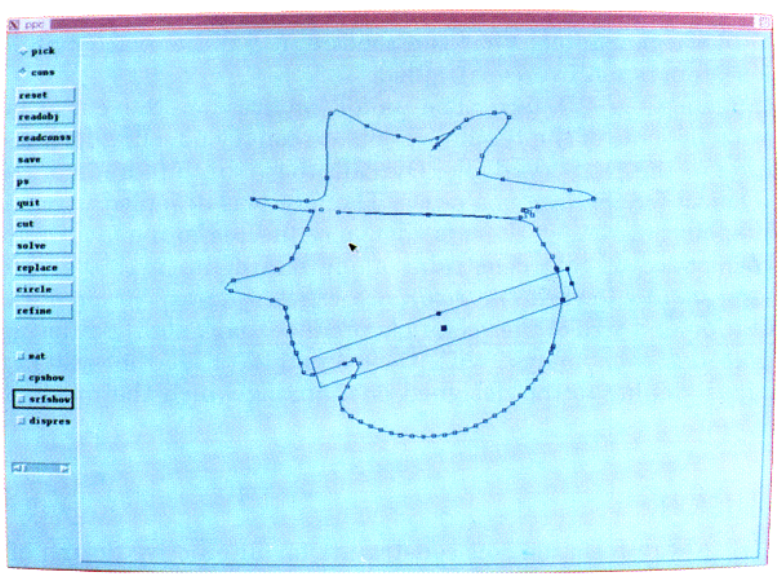

Fig. 10. The face from Figure 8, with a different magnitude of the covariance of the mouth constraint along one of its axes.

surfaces; it could easily be used for smooth volumes (commonly called "free-form deformations"), by modifying the picking and display mechanisms.

We used a scaled identity matrix as the covariance of the current object, providing a global parameter for the amount of similarity between the current and new objects. Currently this parameter is set through a slider widget. Improved user interface techniques for specifying nonuniform similarity controls should be pursued.

Probabilistic constraints are applicable for every object represented as a linear blend of "control points." However, there may be representation which are more suitable than others. In the future we plan to extend our research on internal representations [Borrel and Rappoport 1994] to probabilistic constraints.

Curve constraints are very useful as design operations. We plan to investigate probabilistic curve constraints, perhaps by sampling many points on the curve and fusing them as constraints one by one. Another direction for future research is how to provide the effects of minimization of energy functionals in the context of probabilistic constraints.

Finally, we have already applied the probabilistic-constraints idea to relaxed parametric design [Hel-Or et al. 1993], mostly for mechanical CAD applications. The present application is very different (the type of object designed, the representation of constraints, linearity vs. nonlinearity of constraints, and more). There are many more possible applications of the concept in geometric modeling and computer graphics.

ACKNOWLEDGMENTS

I thank Dani Lischinski and the reviewers for their comments, and Eytan Krein and Dani for help in producing the color images.

REFERENCES

ANDERSON, B. D. O., AND MOORE, J. B. 1979. *Optimal Filtering*. Prentice-Hall, Englewood Cliffs, N.J.

BARTELS, R. AND BEATTY, C. J. 1989. A technique for the direct manipulation of spline curves. In *Graphics Interface '89*, 33–39.

BARTELS, R., BEATTY, J. AND BARSKY, B. 1988. *An Introduction to Splines for Use In Computer Graphics and Geometric Modelling*. Morgan-Kaufmann, Palo Alto, Calif.

BECHMANN, D. AND DUBREUIL, N. 1992 Animation through space and time based on a space deformation model. *Eurographics Workshop on Animation and Simulation*.

BORREL, P. AND BECHMANN, D. 1991. Deformation of n-dimensional objects. *Int. J. Comput. Geom. Appl. 1*, 4. Also in *ACM Symposium on Solid Modeling* ACM, New York, 1991, 351–370.

BORREL, P. AND RAPPOPORT, A. 1994. Simple constrained deformations for geometric modeling and interactive design. *ACM Trans. Graph. 13*, 2 (Apr.).

BOULLION, T. L. AND ODELL, P. L. 1971. *Generalized Inverse Matrices*. Wiley, New York.

CELNIKER, G. AND GOSSARD, D. 1991. Deformable curve and surface finite elements for free form shape design. *Comput. Graph. 25*, 4, 257–266.

CHENG, F. AND BARSKY, B.A. 1991. Interproximation: Interpolation and approximation using cubic spline curves. *Comput. Aided Des. 23*, 10, 700–706.

EMMERICK VAN, M. J. G. M., RAPPOPORT, A. AND ROSSIGNAC, J. 1993. Simplifying interactive design of solid models: A hypertext approach. *Vis. Comput. 9*, 5, 239–254.

FANG, L. AND GOSSARD, D. C. 1992. Reconstruction of smooth parametric surfaces from unorganized data points. In *Curves and Surfaces in Computer Vision and Graphics III*, vol. 1830. SPIE, 226–236.

FARIN, G. 1992. *Curves and Surfaces for Computer Aided Geometric Design*. 3rd ed. Academic Press, New York.

FORSEY, D. R. AND BARTELS, R. H. 1988. Hierarchical B-spline refinement, *Comput. Graph. 22*, 4, 205–212.

FOWLER, B. 1992. Geometric manipulation of tensor product surfaces. In *ACM Symposium on Interactive 3D Graphics*. ACM Press, New York, 101–108.

HEL-OR, Y., RAPPOPORT, A. AND WERMAN, M. 1993. Relaxed parametric design with probabilistic constraints. *Proceedings of the 2nd ACM/SIGGRAPH Symposium on Solid Modeling and Applications*. ACM Press, New York, 261–270. To be published in *Comput. Aided Des.*

HSU, W. M., HUGHES, J. F. AND KAUFMAN, H. 1992. Direct manipulation of free-form deformations. *Comput. Graph. 26*, 2, 177–184.

KALLAY, M. 1993. Constrained optimization in surface design. In *Modeling in Computer Graphics*. Springer-Verlag, New York, 85–94.

KASS, M., WITKIN, A. AND TERZOPOULOS, D. 1988. Snakes: active contour models. *Intl. J. of Computer Vision*. 321–331.

METAXAS, D. AND TERZOPOULOS, D. 1992. Dynamic deformation of solid primitives with constraints. *Comput. Graph. 26*, 2, 309–312.

MORETON, H. AND SÉQUIN, C. 1992. Functional optimization for fair surface design. *Comput. Graph. 26*, 2, 167–176.

NIELSON, G. M. AND OLSEN, D. R. 1986. Direct manipulation techniques for 3D objects using 2D locator devices. In *ACM Symposium on Interactive 3D Graphics*. ACM Press, New York, 175–182.

PRESS, W. H., FLANNERY, B. P., TEUKOLSKY, S. A. AND VETTERLING, W. T. 1988. *Numerical Recipes in C*. Cambridge University Press, Cambridge, Mass.

RAPPOPORT, A. 1993. Direct manipulation devices for the design of geometric constraint networks. In *Communicating with Virtual Worlds. Proceedings of Computer Graphics International '93*. Springer, New York, 294–305.

SCHWEPPE, F. C. 1973. *Uncertain Dynamic Systems*. Prentice-Hall, Englewood Cliffs, N.J.

TERZOPOULOS, D., WITKIN, A. AND KASS, M. 1987. Symmetry seeking models and 3D object reconstruction. *Int. J. Comput. Vis. 1*, 3, 211–221.

WELCH, W. AND WITKIN, A. 1992. Variational surface modeling. *Comput. Graph. 26*, 2, 157–166.

Received April 1993;; revised October 1993; accepted December 1993.