Haptic Rendering of Parametric Surfaces Using a Feedback Stabilized Extremal Distance Tracking Algorithm

Volkan Patoglu and R. Brent Gillespie

***Abstract*— A new extremal distance tracking algorithm is presented for convex parametric curves and surfaces undergoing rigid body motion. The geometric extremization problem is differentiated with respect to time to produce a dynamical system that incorporates dependence on both surface shape and rigid body motion. Extremization then takes place by integrating these dynamical equations, but with a feedback controller in place to stabilize the solution. A controller design using feedback linearization is developed that simultaneously accounts for surface shape and motion while asymptotically achieving (and maintaining) the extremal pair. Collision detection then takes place in a framework fully analogous to that used for multibody simulation. Local stability results are extended to provide global stability for body shapes composed of pieced-together convex parametric surface patches using a switching algorithm.**

I. Introduction

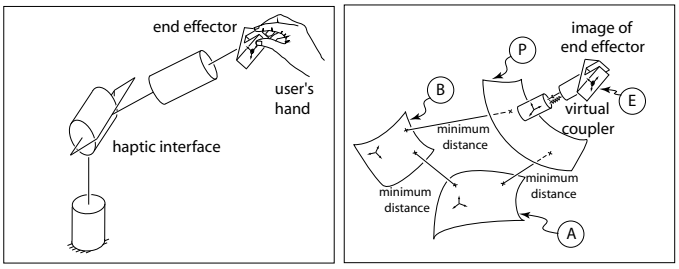

Fig. 1. Schematic representation of haptic rendering.

Haptic rendering is the process by which virtual objects are made apparent to be felt and at the same time made available to be manipulated by a human user. Haptic rendering requires a haptic interface, a computationally mediated virtual environment, and a paradigm according to which the two are linked. Figure 1 presents a schematic view of a haptic interface and the manner in which it is most commonly linked to a virtual environment. On the left portion of the figure, mechanical interaction takes place between a human and the physical haptic interface device, or more specifically, between a fingertip and the device end-effector. In the computational domain depicted on the right, an image of the device end-effector E is connected to a proxy P through what is called the *virtual coupler*. The proxy P in turn interacts with objects such as A and B in the virtual environment. Proxy P might take on the shape of the fingertip or a tool in the user's grasp.

The virtual coupler is depicted as a spring and damper in parallel, which is a model of its most common computational implementation. Rigid bodies in the virtual environment, including P, have both configuration and shape —they interact with one another according to their dynamic and geometric models. Configuration (including orientation and position) is indicated in Figure 1 using reference frames (three mutually orthogonal unit vectors) and reference points fixed in each rigid body, while shape is indicated by a surface patch. Note that the end-effector image E has configuration but no shape. Its interaction with P takes place through the virtual coupler and requires only the configuration of E and P.

Volkan Patoglu and R. Brent Gillespie (corresponding author) are with the Department of Mechanical Engineering, University of Michigan, Ann Arbor, MI 48109, USA. (e-mail: vpatoglu@umich.edu and brentg@umich.edu)

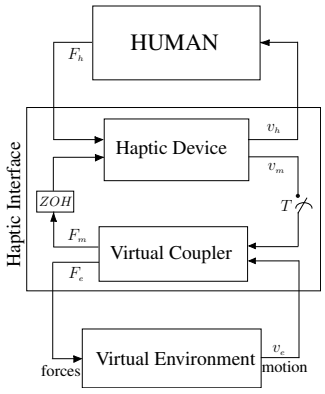

Fig. 2. Block diagram of haptic rendering.

The various components in Figure 1, including the human user, haptic device, virtual coupler, and virtual environment, form a coupled dynamical system whose behavior depends on the force/motion relationship of each component. Figure 2 shows these components interconnected in a block diagram, where the additional indication of causality has been made.

Not apparent in the block diagram is the detail inside the virtual environment block which must be brought to bear to simulate systems with changing contact conditions, including a forward dynamics solver, collision detector, and interaction response algorithm. The focus of this paper is a collision detector that treat bodies whose geometry is modelled using parameterized functions that may collide, rest, slide, or roll on one another. In particular, we treat objects whose boundaries are represented using parametric surface patches. Patches are joined at boundaries that might correspond to discontinuities in curvature or might simply be put in place to effect a decomposition into a set of convex patches.

The core of this paper contains a new approach for tracking the pair of extremal points on a pair of convex parametric surfaces. Features of our approach include its guaranteed stability and seamless integration with the forward dynamics solver and interaction response algorithms. In section II, we will undertake a literature review of collision detectors. In section III, the proposed dynamic model for the tracking algorithm is layed out in detail. Controller design and analysis takes place in section IV. The switching algorithm that handles transitions is presented in section V. In section VI, simulation results are presented. We further discuss our algorithm in Section VII and wrap up in section VIII.

II. Background

To calculate a global solution in a computationally efficient manner, it is very common to handle the collision detection problem in two parts: a *far phase* which involves a coarse global search for potentially interacting surfaces and a *near phase*, which is usually based on a fast local optimization scheme. *Local* operations, if they can be used to improve a proposed extremal distance by some form of gradient descent operation, can be made quite fast. Thus the use of a near phase algorithm contributes to efficiency, especially when the distance problem is called not just once for a particular pair of objects, but at each time step in order to *track* the evolution of the extremal distance over a number of time-steps. Additionally, restriction to *convex* objects or features is often made in the near phase, since in such case the distance problem also becomes convex and admits fast, iterative solution by convex optimization.

1) Far Phase: The far phase is composed of two major steps. First, a global proximity test is performed using hierarchies of bounding volumes or spatial decompositions for each surface patch. The distances between bounding boxes for each pair of surface patches drawn from all pairs of bodies are compared to a threshold distance and patches that are too distant to be contacting are pruned away. Remaining surface patches are declared active. For a full review of bounding volume and space decomposition methods, see [1] and references therein.

In the second step of the far phase, approximate interaction points on active surfaces are computed. For example, if the geometric models are represented by non-uniform rational B-splines (NURBS), control polygons of the active surface models can be used to calculate a first order approximation to the closest points. Bounding box centroids of paired surface patches can be projected onto the polygonal control meshes of each other and parameters can be interpolated from the mesh node values. These approximate projections serve as good initialization points for the near phase of the collision detector.

2) Near Phase: After initialization with the approximations calculated in the far phase, the near phase employs an algorithm to iteratively improve the extremal distance between each active pair of surface patches. Most previous work in collision detection has concentrated on computing the distance between convex polyhedral objects. Polyhedral models have been extensively studied because they are simple to handle, their resolution is sufficient for many applications, and they are almost always compatible with 3-D modelling or CAD systems. We first review the near phase algorithms for polyhedra, then discuss algorithms that compute the distance between objects bounded by continuous surfaces.

A. Polyhedral Models

State of the art algorithms for computing the distance between convex polyhedra are based on the algorithm by Gilbert, Johnson and Keerthi (GJK) [2] and the algorithm of Lin and Canny [3]. The GJK algorithm makes use of Minkowski difference and (simplex-based) convex optimization techniques to calculate the minimum distance. It is an iterative algorithm that generates a sequence of ever improving intermediate steps *within the polyhedra* to converge to the true solution. The algorithm of Lin and Canny makes use of Voronoi regions and temporal/spatial coherence between successive queries to navigate *along the boundaries* of the polyhedra in the direction of decreasing distance. The V-Clip algorithm by Mirtich [4] is reminiscent of the Lin and Canny closest features algorithm but makes several improvements. H-Collide by Gregory et.al. [5] is a specialized collision detection framework for haptic interaction which makes use of several earlier algorithms to detect collisions between polygonal surfaces at interactive rates.

B. Nonpolyhedral Models

Most of the available closest point algorithms for nonpolyhedral models address the problem indirectly —by converting the problem into a polyhedral one with the use of adaptively refined meshes. Another indirect approach proposed by Adachi [6] and Stewart [7] uses intermediate tangent representations.

Although these indirect methods can be successfully implemented for some applications, there also exist cases for which they are not sufficient. Intermediate representations fail to approximate surfaces with high curvature, and polyhedral approximations to complex models can grow very large in the number of polygons.

Less literature exists on direct methods for nonpolyhedral models. Gilbert et.al. extended their algorithm to general convex objects in [8]. In a related paper [9], Turnbull modifies the GJK algorithm to handle convex shapes defined using NURBS. Similarly, in [10] Lin and Manocha present an algorithm for curved models composed of spline or algebraic surfaces by extending their earlier algorithm for polyhedra.

Also worth noting are the subdivision techniques implemented by Duff [11] and Von Herzen [12]. Snyder [13] improves these subdivision methods by modelling the collision detection between time dependent surfaces as a constrained minimization problem. The problem is then solved using interval Newton methods.

In the field of computer graphics, Kriezis [14], Barnhill [15] and Bajaj [16] propose to model parametric surface intersections using differential equations whose solution may be interpreted as a tracing/marching method. Although the goal of these approaches is only to calculate surface intersections, the manner in which the problem is modelled and appropriate points are traced is closely related to our extremal pair tracking algorithm.

Our contribution is a near-phase algorithm designed specifically for parametric surface patches. It is a tracking algorithm like the near-phase algorithms designed for polyhedral models and thus depends on convexity and acquires efficiency. Unlike those near-phase models, however, our algorithm in its basic form is designed to operate on features rather than bodies. (The extremal distance problem on the previously identified extremal features is quite trivial for polyhedra, but not for nonpolyhedral bodies.) To extend our algorithm to handle bodies (composed of surface patches), we employ a switching algorithm that handles *transitions* when tracked pairs encounter surface boundaries. That switching algorithm also uses Voronoi regions and thus is closely related to the Lin-Canny algorithm. Thus the switching algorithm extends a local solution to become global for convex bodies. Finally, the inclusion of a coarse global search (far phase) may be used to extend our algorithm to cover nonconvex bodies and further to determine whether an extremum is a maximum or minimum solution.

Thompson et.al. also contribute a tracking type closest point algorithm for non-polyhedral models. Their near-phase algorithm is based on Newton's method. Extensions to this work include [17], which handles a moving surface and [18], which makes use of higher order derivatives and tangent plane projections at singularities. Finally in [19] this approach is generalized to surface-to-surface interactions and combined with the "velocity formulation", which keeps track of the exact extremal distance during contact and penetration as surfaces move, given exact initial conditions.

Our algorithm is based on a dynamic formulation of the motion of the extremal points and the dependence of point motion on both surface motion and surface shape. To continually solve the relationship between point motion and surface shape and motion, a feedback control problem is formulated and solved with the design of a stabilizing controller. The controller output is the motion of each of the extremal points, and may be used to update the parameter values that locate the points themselves. The speeds along the tangent curves are produced by the controller as functions of the surface motions and surface shapes. These speeds may be integrated to arrive at the closest points, where integration is the essential process of "maintenance".

In our algorithm, we differentiate the essentially geometric problem to form the "differential kinematics" of the closest points, and then we integrate these differential kinematics, but with a feedback loop in place to ensure that the integration is robust and stable. As a result, collision detection takes place in a framework fully analogous to that used for the simulation of the dynamical response. In fact, as will be shown, collision detection and forward dynamics solution can take place through the solution of a single set of coupled differential equations. Similar feedback stabilization techniques will be familiar to roboticists. In particular, to solve for the inverse kinematics of robot manipulators, it is customary to integrate the differential kinematics in a feedback loop to avoid drift and numerical disturbances [20].

III. Modeling

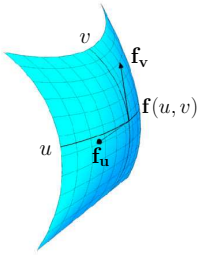

Fig. 3. A parametric surface

Let there exist a parametric representation for the surface shown in Figure 3. Note that all surfaces described by algebraic implicit equations have parametric representations. We use $\mathbf{f}$ to denote a position vector to a point on the surface. And we use $\mathbf{f}(u,v)$ to refer to the mapping from $\Re^2$ to $\Re^3$ that generates the Cartesian coordinates $[x(u,v)\ y(u,v)\ z(u,v)]^T$ from the independent parameters u and v.

The following development relies on the existence of surface continuity through at least two differentiations. We also require that both surfaces be strictly convex. However, the method can be generalized to piecewise continuous surfaces and non-convex surfaces using an appropriate switching method, as mentioned above.

Let $\mathbf{f_u}(u,v)$ and $\mathbf{f_v}(u,v)$ denote the first partial derivatives with respect to u and v of the parametric surface at the point $\mathbf{f}(u,v)$. Similarly, let $\mathbf{f_{uu}}(u,v)$ denote the partial derivative with respect to u of $\mathbf{f_u}$ and so on. Note that the first partials are tangent to the isoparametric curves of u and v respectively.

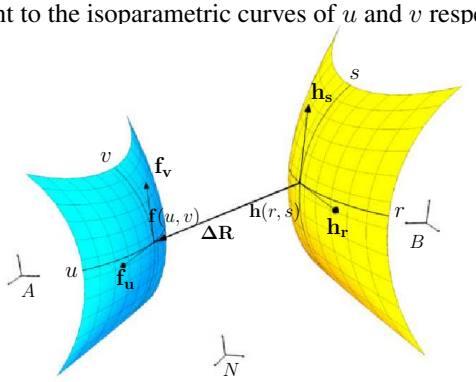

Fig. 4. Two parametric surfaces with a position vector $\mathbf{\Delta R}$ between two arbitrary points

Two parametric surfaces are plotted in Figure 4 with their corresponding isoparametric curves. On these surfaces, two arbitrary points $\mathbf{f}(u,v)$ and $\mathbf{h}(r,s)$ and surface tangents evaluated at these points are shown using notation similar to that used in Figure 3. $\mathbf{\Delta R}(u,v,r,s)$ is a vector between these arbitrary points.

Note that when the difference vector $\mathbf{\Delta R}$ is normal to *both* surfaces, the requirements of the extremal distance are satisfied. In such case the values, denoted $u^\star, v^\star, r^\star$ and $s^\star$, of the parameters u, v, r, and s locate the extremal pair $\mathbf{f}(u^\star, v^\star)$ and $\mathbf{h}(r^\star, s^\star)$. The extremal distance is then equal to the Euclidian norm of $\mathbf{\Delta R}$.

We define scalars Ψ^u, Ψ^v as the projections of the difference vector $\mathbf{\Delta R}$ onto the tangents $\mathbf{f_u}$ and $\mathbf{f_v}$ of surface $\mathbf{f}$; and similarly we define Ψ^r, and Ψ^s as the projections of the difference vector $\mathbf{\Delta R}$ onto the tangents $\mathbf{h_s}$ and $\mathbf{h_r}$ of surface $\mathbf{h}$ as follows.

$$\Psi^u \triangleq \mathbf{\Delta R} \cdot \mathbf{f_u} \tag{1}$$

$$\Psi^v \triangleq \mathbf{\Delta R} \cdot \mathbf{f_v} \tag{2}$$

$$\Psi^r \triangleq \mathbf{\Delta R} \cdot \mathbf{h_r} \tag{3}$$

$$\Psi^s \triangleq \mathbf{\Delta R} \cdot \mathbf{h_s} \tag{4}$$

When the projection errors are all zero, the conditions for the extremal pair are met: the difference vector $\mathbf{\Delta R}$ is perpendicular to both surfaces at $\mathbf{f}$ and $\mathbf{h}$.

Note that it is possible to define the extremal distance condition by an alternate set of equations as presented in [19]. This alternative formulation makes use of surface tangents of one surface and normals of both surfaces. Although we use the set (1) - (4) in our further derivation in this paper, very similar results can be achieved using the alternate set.

Given the set of equations (1) - (4), one way to find the extremal pair is to search for the solution $u^\star$,$v^\star$,$r^\star$,$s^\star$ that minimizes the projection errors using a gradient descent algorithm. This procedure would require the computation of a Jacobian for use in Newton Iteration. This approach is undertaken in [19].

In the present work, rather than taking the Jacobian of the system of equations (1) through (4) with respect to the independent parameters u, v, r and s, we differentiate them with

$$M = \begin{bmatrix} \mathbf{f_u}\cdot\mathbf{f_u}+\mathbf{\Delta R}\cdot\mathbf{f_{uu}} & \mathbf{f_v}\cdot\mathbf{f_u}+\mathbf{\Delta R}\cdot\mathbf{f_{uv}} & -\mathbf{h_r}\cdot\mathbf{f_u} & -\mathbf{h_s}\cdot\mathbf{f_u} \\ \mathbf{f_u}\cdot\mathbf{f_v}+\mathbf{\Delta R}\cdot\mathbf{f_{uv}} & \mathbf{f_v}\cdot\mathbf{f_v}+\mathbf{\Delta R}\cdot\mathbf{f_{vv}} & -\mathbf{h_r}\cdot\mathbf{f_v} & -\mathbf{h_s}\cdot\mathbf{f_v} \\ \mathbf{f_u}\cdot\mathbf{h_r} & \mathbf{f_v}\cdot\mathbf{h_r} & -\mathbf{h_r}\cdot\mathbf{h_r}+\mathbf{\Delta R}\cdot\mathbf{h_{rr}} & -\mathbf{h_s}\cdot\mathbf{h_r}+\mathbf{\Delta R}\cdot\mathbf{h_{rs}} \\ \mathbf{f_u}\cdot\mathbf{h_s} & \mathbf{f_v}\cdot\mathbf{h_s} & -\mathbf{h_r}\cdot\mathbf{h_s}+\mathbf{\Delta R}\cdot\mathbf{h_{rs}} & -\mathbf{h_s}\cdot\mathbf{h_s}+\mathbf{\Delta R}\cdot\mathbf{h_{ss}} \end{bmatrix}, \quad b = \begin{bmatrix} -({}^{\mathbf{A}}\boldsymbol{\omega}^{\mathbf{B}}\times\mathbf{h})\cdot\mathbf{f_u} \\ -({}^{\mathbf{A}}\boldsymbol{\omega}^{\mathbf{B}}\times\mathbf{h})\cdot\mathbf{f_v} \\ ({}^{\mathbf{B}}\boldsymbol{\omega}^{\mathbf{A}}\times\mathbf{f})\cdot\mathbf{h_r} \\ ({}^{\mathbf{B}}\boldsymbol{\omega}^{\mathbf{A}}\times\mathbf{f})\cdot\mathbf{h_s} \end{bmatrix}$$

respect to time. The differentiation operation causes the motion of the surfaces and the time rates of change of the parameters $du/dt, dv/dt, dr/dt$ and ds/dt, called the parametric velocities, to show up in the expression for the projection error derivatives.

Note that one must effectively freeze time (and consequently the motion of the bodies) while using a gradient descent algorithm to find the extremal pair. In contrast, taking the time derivative of equations (1) through (4) produces a dynamic expression where the time rates of change of the projection errors are expressed in terms of the motion and shape of the surfaces.

It is worth mentioning that although we use vector expressions throughout the paper, one needs to express each vector consistently in a single reference frame before interpreting the operations as matrix operations. Where dot products and cross products appear, we use boldface notation to indicate operations which may be performed in a basis-independent fashion. Once suitably expressed in a reference frame, standard matrix operations may be used, and we indicate this using normal typeface. Note also that since the right hand sides of equations (1) through (4) are basis-independent vector expressions, it is important to specify a frame in which differentiation is to be performed. We choose to express the vectors in the first two equations, (1) and (2), in the body frame A and the vectors in last two equations, (3) and (4), in the body frame B (see Figure 4). This choice results in simpler matrix expressions.

Consider the case where each surface is attached to a rigid body in motion. In Figure 4 these bodies are named A and B. Assume that the configuration of bodies A and B with respect to a reference frame N is known. Then motion of these bodies with respect to the reference frame N will be specified by the vectors ${}^{\mathbf{N}}\boldsymbol{\omega}^{\mathbf{A}}$, ${}^{\mathbf{N}}\boldsymbol{\omega}^{\mathbf{B}}$, ${}^{\mathbf{N}}\mathbf{v}^{\mathbf{A_o}}$ and ${}^{\mathbf{N}}\mathbf{v}^{\mathbf{B_o}}$.

Using the notation ${}^{A}\frac{d}{dt}(.)$ to indicate differentiation in reference frame A, and noting that, for any vector $\boldsymbol{\beta}$, ${}^{A}\frac{d}{dt}(\boldsymbol{\beta}) = {}^{B}\frac{d}{dt}(\boldsymbol{\beta}) + {}^{\mathbf{A}}\boldsymbol{\omega}^{\mathbf{B}}\times\boldsymbol{\beta}$, we take time derivatives of Equations (1)-(4) as follows

$$\dot{\Psi}^u = {}^{A}\frac{d}{dt}\left[(\mathbf{f}-\mathbf{h})\cdot\mathbf{f_u}\right] \quad (5)$$

$$\dot{\Psi}^v = {}^{A}\frac{d}{dt}\left[(\mathbf{f}-\mathbf{h})\cdot\mathbf{f_v}\right] \quad (6)$$

$$\dot{\Psi}^r = {}^{B}\frac{d}{dt}\left[(\mathbf{f}-\mathbf{h})\cdot\mathbf{h_r}\right] \quad (7)$$

$$\dot{\Psi}^s = {}^{B}\frac{d}{dt}\left[(\mathbf{f}-\mathbf{h})\cdot\mathbf{h_s}\right] \quad (8)$$

and rearrange into matrix form to produce the following differential equations for the projection errors

$$\dot{\Psi} = M\,U + b \quad (9)$$

$$\dot{x} = U \quad (10)$$

$$y = x \quad (11)$$

where $\Psi = \begin{bmatrix} \Psi^u \\ \Psi^v \\ \Psi^r \\ \Psi^s \end{bmatrix}$, $U = \begin{bmatrix} \frac{du}{dt} \\ \frac{dv}{dt} \\ \frac{dr}{dt} \\ \frac{ds}{dt} \end{bmatrix}$

and M and b are shown at the top of the page.

In this state space realization the state variables Ψ are taken to be the projection errors. The inputs U are time derivatives of surface parameters whereas the system outputs are denoted by y. The desired outputs from the algorithm are the estimates $x = [u, v, r, s]^T$ of the parametric values of extremal points on each surface, u^*, v^*, r^* and s^*, at every instant of time. These estimates can be calculated as a by-product of the control effort that regulates the projection errors to zero. Details of this procedure are shown in the next section.

IV. Control

Equations (9)-(11) define a nonlinear dynamic model to maintain the extremal pair on two surfaces undergoing rigid body motion. It characterizes the projection error derivatives in state space form and formulates the extremal distance problem as a standard nonlinear control problem.

The control input vector U is composed of time derivatives of surface parameters, i.e. the elements of U are speeds along the tangent curves. The objective of the controller is to continually update these speeds to regulate the projection errors to zero, i.e. to maintain the extremal pair on the surfaces.

With the model (9)-(11) in hand, the extremal pair on the surfaces can be dynamically tracked making use of a control loop with exact feedback linearization. Exact feedback linearization is feasible since the plant is implemented in the computer and at any instant of time the specific values of M and b are exactly known. Note that feedback linearization is fundamentally different than Jacobian linearization in that feedback linearization is achieved by exact state transformation and feedback, rather than by linear approximations of the dynamics for a small range of operation [21].

First, in order to feedback linearize the model, an inner feedback loop is designed. Assuming the matrix M is not singular in the range of operation, we define the control input vector U in terms of a new input vector μ as

$$U = M^{-1}\left(\mu - b\right) \quad (12)$$

and apply this control input to (9). Then the nonlinear model is algebraically transformed into an equivalent linear model

$$\dot{\Psi} = \mu \quad (13)$$

Second, an outer loop linear controller is used to impose the desired linear dynamics to equation (13). In this paper, a full state linear feedback

$$\mu = -K\,\Psi \quad (14)$$

is utilized to stabilize the closed loop dynamics and to achieve desired performance: to keep projection errors small. However, it is possible to synthesize different outer loop controllers to satisfy various design objectives.

Exponential stability of the overall closed loop system is guaranteed since there are no internal dynamics associated with this input-output linearization. This observation follows from the fact that the relative degree of the system is the same as

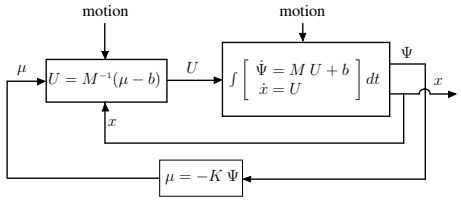

Fig. 5. Control block diagram showing an inner feedback linearization loop and an outer linear control loop

its order and input-output linearization leads to input-state linearization [21].

Figure 5 shows the block diagram of the completed controller design. Recognizable here is the inner loop that renders the system linear from input μ to output x. The outer loop achieves desired dynamics of the linear system via full state feedback with matrix gain K.

Furthermore, the desired outputs, i.e. the parametric values of the extremal pair, are continually maintained using the control input vector U. This is simply achieved by integrating the input vector U with initial conditions extracted from the starting points. In practice, the state vector Ψ is augmented with the input vector U to perform all integrations in a single operation.

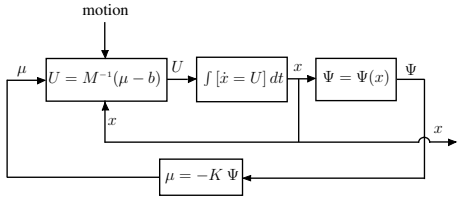

Fig. 6. Control block diagram showing an alternative implementation

In fact, even a simpler implementation is possible. Figure 6 demonstrates this equivalent case. Since the projection errors Ψ can be directly calculated through equations (1) to (4), the derived dynamic model can be replaced by this set of nonlinear equations. Note that although the derived dynamic model is replaced, the controller design remains unchanged.

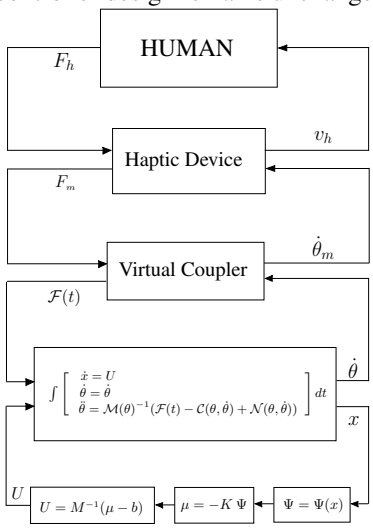

Fig. 7. Combining extremal pair tracking with dynamics

It is also possible to combine the extremal pair tracking algorithm with dynamics as discussed in section I. One such case is shown in Figure 7. Here, motion of the bodies is calculated simultaneously with the maintenance of the extremal pair between them. In this figure, the equations of motion for the bodies are defined by a second-order differential equation where θ represents the set of configuration variables. The inertia matrix $\mathcal{M}(\theta)$ and the Coriolis matrix $\mathcal{C}(\theta, \dot{\theta})$ summarize inertial properties of the bodies. $\mathcal{F}(t)$ denotes external control forces acting on the bodies while $\mathcal{N}(\theta, \dot{\theta})$ includes all other frictional and gravitational forces. As shown in Figure 7, all integration (update and maintenance) operations are combined in a single integration operation.

Finally, it is important to mention that the algorithm need not be initialized with the exact extremal points. Any initial point within the region of attraction of the designed nonlinear controller (which is the entire convex surface) will converge to the extremal pair since the controller is exponentially stable. Moreover the convergence rate can be adjusted by tuning the controller gain K.

V. Hybrid Systems

We have adopted the mathematical language of hybrid dynamical systems to describe collision detection between bodies composed of parametric surface patches. Hybrid systems contain both discrete and continuous state variables and exhibit both discrete and continuous state dynamics. In certain hybrid systems, continuous and discrete dynamics not only coexist, but *interact* such that changes in the continuous and discrete dynamics occur in response to continuous and/or discrete state variables. In the case of collision detection between parametric bodies, however, the coupling between the continuous and discrete dynamics is somewhat limited. When tracking the extremal distance between a body and a point, the continuous state the describes the motion of the extremal point on the body is decoupled from the discrete state that specifies the relevant feature on the body. In particular, the switching times do not depend on the continuous states. On the other hand, when tracking the extremal distance between two bodies (either in a plane or in space) the switching times do in fact depend on certain portions of the continuous states.

In the following, let us develop a hybrid mathematical model to describe the most general case: extremal distance tracking between two bodies. Thereafter, we will show how certain arguments may be removed from the argument lists for the various functions to describe the case of extremal distance tracking between a body and a point.

Consider a system described by a state space $S = \bigcup_{i=1}^{n} S_i$, where the state x within each mode S_i evolves according to the differential equation

$$\dot{\mathbf{x}}(t) = \mathbf{F}_i(\mathbf{x}(t), \theta(t)), \tag{15}$$

where $\theta(t)$ is an exogenous input representing the specified motion of the bodies. Note that the state vector x has the same dimension in all modes, a property that can be accommodated by setting unused elements to zero. Associated with each mode S_i is a set of transitions to other modes. And associated with each transition in that set is a switching function f and a resetting function ϕ. Let us index the members of the set of transitions (and likewise the set of f's and set of ϕ's) with a superscript j and indicate the size of each set with m_i, which in general is different for each mode. A subscript i indicates association with the ith mode. Members f_i^j of the set of switching functions J_i trigger transitions out of mode S_i :

$$J_i = \left\{ f_i^j(\mathbf{x}(t), \theta(t)) = 0, \quad (j = 1, ..., m_i) \right\} \tag{16}$$

The time t^* that, together with the state $\mathbf{x}^- = \mathbf{x}(t^*)$ and specified motion $\theta(t^*)$, produces a zero of transition function f_i^j is called a *switching instant*; it triggers the associated transition. Once a transition is triggered, the associated member ϕ_i^j of the set of resetting functions Φ_i is executed to relate the initial state values $\mathbf{x}^+$ in the new mode to the final state values $\mathbf{x}^-$ in the last mode.

$$\Phi_i = \left\{ \mathbf{x}^+ = \phi_i^j(\mathbf{x}^-, t^*), \quad (j = 1, ..., m_i) \right\} \tag{17}$$

A special case of the resetting functions is the set of initial conditions for the initial mode S_1.

To particularize the description of a hybrid system contained in Eqs. (15) through (17) for the case of collision detection between a body and a point, one may remove the argument $\mathbf{x}$ from each switching function in (16). The switching functions depend only on the specified motion of the body and point, not on the motion of the extremal point on the body. In fact, for collision detection between two bodies, the switching between features on one body depends only on the motion of the extremal point on the other body and the motion, not the motion of the extremal point on its own surface.

Evaluation of the hybrid system can be viewed as a sequence of subproblems, each characterized by a continuous evolution in a mode terminated by an event (i.e. zero crossing of a switching function), and then evaluation of the resetting functions to initialize the new mode.

A transition from one discrete state to another one is triggered when a discrete state (a feature) encounters a transition condition (i.e. an active feature goes out of the Voronoi region of the other active feature). These discrete transitions are handled in a manner similar to the Lin and Canny algorithm. Note that whenever a transition occurs, not only the discrete state changes, but also the continuous system model changes and the continuous states are updated according to the reset relations Φ. It is convenient to represent the discrete dynamics of the hybrid system using an automaton as in Figure 9.

To analyze convergence to the extremal pair, first consider the case when both objects are stationary. In section IV it was shown that, once initialized with the correct pair of features (but not necessarily with the correct extremal pair within the features), our algorithm accounts for the initialization errors and disturbances, exponentially converging to the extremal pair. This exponential convergence within the feature is guaranteed since the basin of attraction of the controlled system spans all the feature itself. Therefore (for the stationary case) in order to achieve global asymptotic stability, it would be sufficient to introduce a far phase that locates the correct pair of features to initialize the near algorithm.

Now, consider the case when objects are in motion. Introduction of motion converts the problem into a tracking problem. In our method, within each feature, the motion of the objects are exactly accounted for by the feedforward term of the controller. However, some extra effort is required to handle transitions between features that are due to the motion. In order to be able to track the extremal distance through features, the switching algorithm is put in place. The switching algorithm decides when and how to handle transitions between features according to Voronoi criteria. Therefore, with the installment of the switching algorithm, our method simultaneously tracks the extremal pair and the correct set of features containing them.

Even under motion, convergence to the extremal pair is guaranteed since the motion is handled in a feedforward manner and the transitions are handled in such a way that each time a transition is triggered the extremal distance is guaranteed not to increase.

VI. Simulation Results

We developed computer simulations to check the dynamic formulation and the control algorithm discussed in the previous sections. Our simulations are implemented in Matlab/Simulink and sample results are presented below.

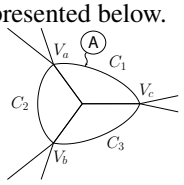

Fig. 8. A convex planar body A composed of curves C_1, C_2, C_3 joined at vertices V_a, V_b, V_c. Lines indicate boundaries between interior and exterior Voronoi regions.

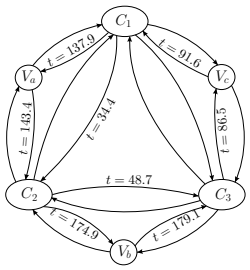

Fig. 9. The automaton used to govern the motion of the extremal point on body A of Figure 8. Switching times for the simulation shown in Figure 10 are also indicated.

Figure 8 shows a convex planar body A formed by joining three convex curves C_1, C_2, and C_3 at vertices V_a, V_b, and V_c. Also shown in Figure 8 are the six lines that bound the external Voronoi regions of A and the three lines, called *medial axes* that bound the internal Voronoi regions of A. The medial axes are each the locus of points equidistant to two curves of A. The automaton that describes the hybrid dynamics of an extremal point lying on the boundary of A is shown in Figure 9. The three large ellipses in Figure 9 each describe a mode in which the extremal point lies on a particular curve. The continuous dynamics within a particular mode govern the motion of the extremal point so long as it lies on the correspondingly labelled curve. The three smaller circles each describe the (trivial) dynamics of the extremal point while it lies on the correspondingly labelled vertex. The arrows indicate the transition functions that switch between modes. The two outer circular loops indicate transitions from one external Voronoi region to another, while the two inner triangular loops (not involving vertices) indicate transitions across medial axes from one internal Voronoi region to another.

Figure10 shows a simulation that produces the tracking behavior of an extremal point on A as a point P (the other member of the extremal pair) traces out a pre-specified spiral path that begins inside A and ends outside A. In all, 100 snapshots of the simulation are shown, or 100 extremal distances connecting the extremal pair. To begin, the extremal point is initialized with the wrong value and the algorithm quickly converges to the correct answer as indicated by the dotted lines and sweeping arrow. Thereafter, one end the extremal distance is either perpendicular to a curve of A or anchored on a vertex of A.

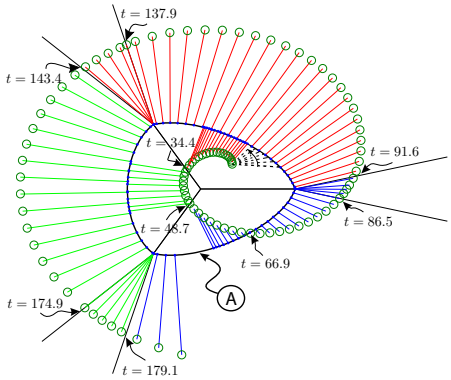

Fig. 10. The extremal distance is shown between A and a point tracing out a spiral that begins inside and ends outside A. Simulated transition times are indicated.

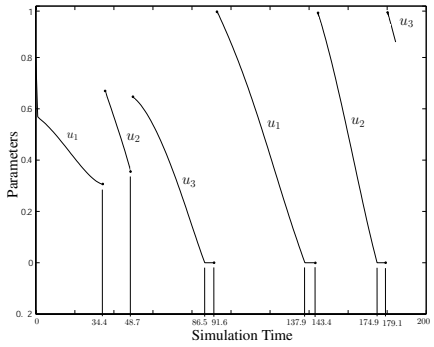

Fig. 11. The trajectory and sequence of parameters u_1, u_2, u_3 on curves C_1, C_2, C_3 for the simulation shown in Figure 10. Transition times are also indicated.

The times at which transitions between modes occur are shown both in Figure 10 and on the automaton in Figure 9. For example, at $t = 34.4$ seconds, P crosses the medial axis and tracking on C_1 jumps to tracking on C_2. At $t = 66.9$ seconds, P crosses the boundary of A and the definition of extremal changes from *maximum penetration depth* to *minimum distance*, but tracking continues on C_3. At $t = 86.5$ seconds, the extremal point anchors on V_c and at $t = 91.6$ seconds it continues on C_1 again. The transition times are also indicated for this particular simulation in Figure 11, which is a plot of the curve parameters u_i versus time. All three curve parameters u_i, $(i = 1, 2, 3)$ can be varied from 0 to 1 to trace out body A while the particular sequence and trajectory of values shown in Figure 11 pertains to the evolution of the extremal point shown in Figure 10. Also visible in Figure 11 is the fast convergence of startup error at the very beginning of the simulation.

Figure 12 shows twelve irregularly spaced, labelled snapshots from the simulation of the extremal pair tracking problem for two convex planar bodies. During the simulation body A remains fixed and B undergoes a motion in which it spins around its own center while its center traces out an inward spiral centered on A. The active curves or vertices, which correspond to similarly labelled modes in the automata not shown, are also indicated in Figure 12. Figure 13 shows the curve parameters u_i of A and r_i of B. The sequencing and evolution of both curve parameters u_i and r_i that locate the extremal points on A and B can be seen. The snapshot times are also indicated in Figure 13 as are the transition times.

Figure 14 shows the extremal distance between a spatial body C and a point that traces out a helix. Body C is formed by joining three parametric surfaces at their intersecting curves and

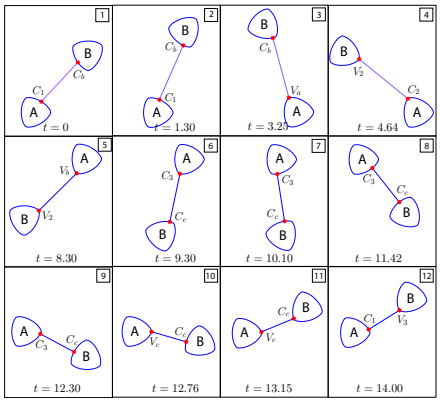

Fig. 12. The extremal distance between planar bodies A and B as A remains fixed and B spins and traces a circle around A. Twelve snapshots are shown, taken at irregular intervals. The active curves or vertices of A and B are indicated.

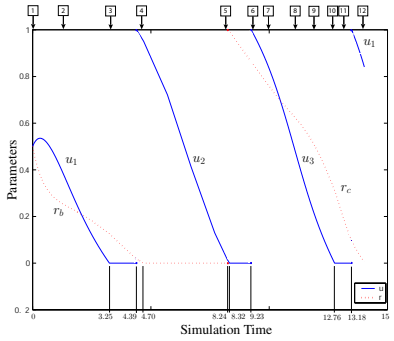

Fig. 13. The trajectory and sequence of parameters u_1, u_2, u_3 on curves C_1, C_2, C_3 and r_a, r_b, r_c on curves C_a, C_b, C_c for the simulation shown in Figure 12. Transition times are indicated.

vertices. The surfaces as labelled $S1, S2$, and $S3$, the curves C_a, C_b, and C_c and the vertices V_α and V_β. The sequence and trajectory of the surface parameters u_i and v_i, taken pairwise with $(i = 1, 2, 3)$ to locate the extremal point on C are shown in Figure 15. Periods of time in which the extremal point lies on the bounding curves can be recognized.

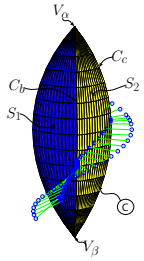

Fig. 14. The extremal distance is shown between a spatial body C and a point tracing out a helix. Body C is composed of 3 parametric surfaces (only S_1 and S_2 are visible) joined at 3 curves (only C_a and C_b are visible) and 2 vertices (V_α and V_β.)

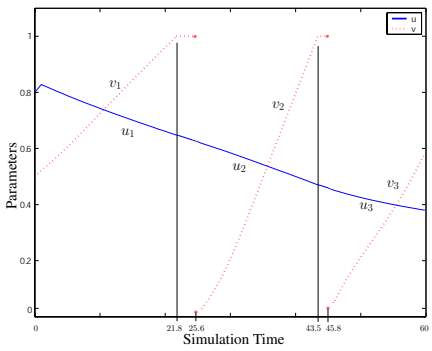

Fig. 15. Changes in curve parameters during tracking.

VII. Discussion

The control theoretic approach to extremal point tracking that we have described in this paper possesses several significant advantages. First of all, it is very general —in the sense that various feedback control designs and discretization choices exercised within our approach yield various algorithms, some of which have previously appeared in the literature.

A second significant advantage of our approach is that local exponential stability can be guaranteed within each feature. This stability guarantee holds no matter how fast the bodies might be moving or how sharp their curvature might be, since these effects are accounted for in the controller design. Specifically, the motion (known exactly through the solution of the forward dynamics) is essentially fed forward using the term b and the dependence on curvature is accounted for with the use of M^{-1} (see Eq. (12)) in the production of the linearized system dynamics. The stability properties of closed loop system then follow from linear controls theory. Furthermore, the convergence rate can be chosen arbitrarily fast for the tracking problem as posed in continuous time.

Since the controller and its associated dynamics will be implemented in discrete time, the important question is the preservation of the stability properties through discretization. Since our system equations are in fact linear, stability would be preserved through discretization using the the trapezoidal (bilinear) rule or Backward Euler integration scheme, even for convergence rates set arbitrarily fast using aggressive gain values K. But since such implicit methods are not an option for real-time haptic rendering, we must consider preservation of stability through discretization using explicit methods. As it turns out, there exist standard techniques whereby the convergence rate (determined by gain K) and step size may be traded off against one another while maintaining stability. In [22] standard discrete time controller design techniques are utilized to calculate proper controller gains given an explicit integration method and fixed integration step size. With these techniques, stability of the algorithm is guaranteed even after discretization.

Moreover, with proper design of the 'far' phase and the switching algorithm for the collision detector as discussed in Section V, global stability of the extremal tracking controller can be achieved, even for geometric models made of several surface patches pieced together.

Finally, whenever one of the pair of objects under consideration is just a point, our algorithm is capable of tracking the minimum distance when the objects are disjoint and maximum penetration depth when they are interpenetrated. If none of the objects of interest are points, then our algorithm can track the minimum distance between them when they are disjoint. However, since the interpenetration case is a nonconvex problem, the algorithm can only return a penetration flag when the objects are interpenetrated. One possible method to utilize our algorithm to track the maximum interpenetration distance for two interpenetrated objects is to simultaneously track the extremal distance between all interpenetrating features.

VIII. Conclusions

We are interested in pursuing a combined simulation/collision detection approach since it results in an algorithm that is easy to implement and that makes maximum use of all the data available to track the extremal points. The proposed algorithm is suitable for both dynamic simulation and haptic rendering due to the continuous availability of surface normals and penetration distances that are necessary to calculate the collision response and/or the haptic feedback.

Our algorithm treats the extremal point problem for objects modelled using parametric curves and surfaces in a direct manner, without resorting to polyhedral approximations. Thus it serves the needs of CAD/CAM and virtual environment systems that require smoothness independent of rendering. Additionally, our algorithm is suited to real-time implementation. Finally, our algorithm features convenient tuning of convergence properties through design of the feedback gain K and enjoys immunity to start-up errors.

Acknowledgments

The authors gratefully acknowledge the support of the National Science Foundation under award number IIS:PECASE #0093290, and fruitful discussions with Elmer Gilbert.

References

[1] J. T. Klosowski, M. Held, J. S. B. Mitchell, H. Sowizral, and K. Zikan, "Efficient collision detection using bounding volume hierarchies of k-DOPs," *IEEE Tran. on Visual and Comp. Graph.*, vol. 4, pp. 21–36, 1998.

[2] E. G. Gilbert, D. W. Johnson, and S. S. Keerthi, "Fast procedure for computing the distance between convex objects in three-dimensional space," *IEEE J. Robotics and Automation*, vol. 4, pp. 193–203, 1988.

[3] M. C. Lin and J. F. Canny, "A fast algorithm for incremental distance calculation," in *IEEE Int. Conf. on Robotics and Automation*, vol. 2, pp. 1008–1014, 1991.

[4] B. Mirtich, "V-Clip: Fast and robust polyhedral collision detection," *ACM Trans. on Graphics*, vol. 17, no. 3, pp. 177–208, 1998.

[5] A. Gregory, M. C. Lin, S. Gottschalk, and R. Taylor, "A framework for fast and accurate collision detection for haptic interaction," in *Proc. of IEEE Virtual Reality*, vol. 4, pp. 38–45, 1999.

[6] Y. Adachi, T. Kumano, and K. Ogino, "Intermediate representation for stiff virtual objects," in *Proc. Virt. Reality Int. Symp.*, pp. 203–210, 1995.

[7] P. Stewart, Y. Chen, and P. Buttolo, "CAD data representations for haptic virtual prototyping," in *Proc. of ASME Des. Eng. Tech. Conf.*, 1997.

[8] E. G. Gilbert and C. P. Foo, "Computing the distance between general convex objects in three-dimensional space," *IEEE Trans. on Robotics and Automation*, vol. 6, no. 1, pp. 53–61, 1990.

[9] C. Turnbull and S. Cameron, "Computing distances between NURBS-defined convex objects," in *Proc. IEEE Int. Conf. on Robotics and Automation*, vol. 4, pp. 3685–3690, 1998.

[10] M. C. Lin and D. Manocha, *Interference Detection Between Curved Objects for Comp. Animation*. Springer-Verlag, 1993.

[11] T. Duff, "Interval arithmetic and recursive subdivision for implicit functions and constructive solid geometry," *ACM Comp. Graphics*, vol. 26, no. 2, pp. 131–138, 1992.

[12] B. Von Herzen, A. H. Barr, and H. R. Zatz, "Geometric collisions for time-dependent parametric surfaces," in *ACM Comp. Graphics*, vol. 24, pp. 39–48, 1990.

[13] J. M. Snyder, A. R. Woodbury, K. Fleischer, and A. H. Barr, "Interval methods for multi-point collision detection between time-dependent curved surfaces," in *Proc. of ACM SIGGRAPH*, pp. 321–334, 1993.

[14] G. A. Kriezis, N. M. Patrikalakis, and F. E. Wolter, "Topological and differential equation methods for surface intersections," *Computer-Aided Des.*, vol. 24, no. 1, pp. 41–51, 1992.

[15] R. E. Barnhill and S. N. Kersey, "A marching method for parametric surface/surface intersection," *Comp. Aided Geo. Des.*, pp. 257–280, 1990.

[16] C. L. Bajaj, C. M. Hoffmann, R. E. Lynch, and J. E. H. Hopcroft, "Tracing surface intersections," *Comp. Aided Geo. Des.*, vol. 5, pp. 285–307, 1988.

[17] T. V. Thompson II, D. D. Nelson, E. Cohen, and J. Hollerbach, "Maneuverable NURBS models within a haptic virtual environment," in *Proc. of ASME Int. Mech. Eng. Congress and Exposition*, vol. 61, pp. 37–44, 1997.

[18] D. E. Johnson and E. Cohen, "An improved method for haptic tracing of a sculptured surface," in *Proc. of ASME Int. Mech. Eng. Congress and Exposition*, vol. 64, pp. 243–248, 1998.

[19] D. D. Nelson, D. E. Johnson, and E. Cohen, "Haptic rendering of surface-to-surface sculpted model interaction," in *Proc. ASME Dynamic Systems and Control Division*, vol. 67, pp. 101–108, 1999.

[20] L. Sciavicco and B. Siciliano, *Modelling and control of robot manipulators*. Springer, 2000.

[21] J. E. Slotine and W. Li, *Applied Nonlinear Control*. Prentice Hall, 1991.

[22] S. T. Lin and J. N. Huang, "Stabilization of Baumgarte's method using the Runge-Kutta approach," *J. Mech. Des.*, vol. 124, pp. 633–641, 2002.