To appear, IEEE Transactions on Robotics

Feedback Stabilized Minimum Distance Maintenance for Convex Parametric Surfaces

Volkan Patoglu and R. Brent Gillespie

***Abstract*— A new minimum distance tracking algorithm is presented for moving convex bodies represented using tiled-together parametric surface patches. The algorithm is formulated by differentiating the geometric minimization problem with respect to time. This produces a hybrid dynamical system that incorporates dependence on rigid body motion, surface shape, and surface boundary interconnectedness. The minimum distance between a pair of previously identified closest features is found by feedback stabilizing the dynamical equations and numerically solving the resulting closed loop system equations. Maintenance of the minimum distance and the associated closest points during motion is achieved through the action of a feedforward controller and a switching algorithm. The feedforward controller simultaneously accounts for surface shape and motion while the switching controller triggers updates to the extremal feature pair when extremal points on one body cross between Voronoi regions of the other body. The algorithm may be implemented within the same framework used for multibody simulation since the minimum distance tracking algorithm itself follows as the simulation of a hybrid dynamical system. In contrast to previously available minimum distance determination algorithms, attractive properties of the new algorithm include a means of determining the highest gain K that maintains stability under a given discretization scheme and a large and easily characterized basin of attraction of the stabilized closest points. These properties may be used to achieve higher computational efficiency. Simulation results are presented for various planar and spatial systems composed of a body and point or composed of two bodies.**

***Index Terms*— Minimum Distance Tracking, Collision Detection, Closest Point Determination, Parametric Surfaces, Haptic Rendering.**

I. Introduction

COLLISION detection is an essential technology in many applications, including virtual environments, computer animation, and robot simulation. A collision detector finds the points in time at which geometric objects first make contact with one another and might be called upon to report the minimum distance between a set of objects and to produce the corresponding pair of closest points. Desirable properties of a collision detector include computational efficiency, ease of implementation, and broad applicability. Computational speed is critical, as a collision detector generally shares computational resources with dynamic simulation, collision response, and graphic or haptic rendering algorithms. Computational efficiency is at a particular premium in virtual environments with haptic rendering, where collisions must be found and the entire dynamic response computed in real-time at rates in excess of 1 kHz.

To increase computational efficiency, it is very common to handle the collision detection problem in two parts: a *broad phase* which involves a coarse global search for potentially interacting surfaces and a *narrow phase*, which is initialized by the board phase and is usually based on a fast local optimization scheme [1]. *Local* operations, if they can be used to improve a proposed solution by some form of gradient descent operation, can be made quite fast. Thus the use of a narrow 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 minimum distance during simulation. Additionally, restriction to *convex* objects or features is often made in the narrow phase, since in such case the distance problem also becomes convex and admits fast, iterative solution by convex optimization. In addition to the collision detectors based on broad and narrow phases, there exist single-phase algorithms, reviewed for example in [2], but not covered here. Among these H-Collide by Gregory et.al. [3], [4] is a specialized framework for haptic interaction and is based on an extension of OBBTree methods. For an overview of the state of the art in collision detection algorithms categorized with respect to their geometric representations, see [5]. Our interest in this paper lies primarily in narrow phase algorithms for parametric surfaces.

Most narrow phase collision detection algorithms available to date are applicable only to polyhedral objects or polyhedral approximations of continuously defined objects, called polygonal meshes. State of the art algorithms for convex polyhedra are based on the algorithm by Gilbert, Johnson and Keerthi (GJK) [6] [7] and the algorithm of Lin and Canny [8]. The GJK algorithm makes use of Minkowski difference and simplex-based convex optimization techniques to generate a sequence of ever improving intermediate steps *within the polyhedra* to converge to the minimum distance 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. Extensions to the Lin-Canny algorithm include V-Clip by Mirtich [9] and SWIFT by Ehmann et.al. [10], [11], which is based on Voronoi marching.

Less well developed are collision detection algorithms that operate directly on objects modeled with parametric surfaces. Parametric surfaces are used in Computer Aided Design (CAD) and Computer Aided Engineering (CAE) tools, where the surface representation must provide smoothness and continuity independent of the resolution of a particular rendering. One important type of parametric surface is the non-uniform rational B-spline (NURBS), which has the advantages of compact representation, high order continuity, and easily computable surface derivatives and normals [12]. Although polygonal meshes and other indirect methods [13], [14] can

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)

be successfully used to convert a parametric model into a polyhedral model for some applications, there also exist cases in which the polyhedral approximations grow very large in the number of polygons or the intermediate representations fail to approximate surfaces with high curvature.

Methods directly applicable to parametric models include extensions of the GJK algorithm to general convex objects by Gilbert et.al. [15] and to convex objects modeled using NURBS by Turnbull [16]. Similarly, the Lin-Canny algorithm for polyhedra has been extended to models composed of spline or algebraic surfaces by Lin and Manocha [17], [18]. Also worth noting are the subdivision techniques implemented by Duff [19] and Von Herzen [20] and further extended by Snyder [21]. In these algorithms, collision detection is modeled as a constrained minimization problem and solved using interval Newton methods.

In the field of CAD and geometric modeling, Kriezis et.al. [22], Barnhill et.al. [23] and Bajaj et.al. [24] 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 modeled and points are traced is closely related to our proposed collision detection algorithm.

Thompson et.al. also contribute a tracking type closest point algorithm for non-polyhedral models. Their narrow-phase algorithm is based on Newton's method. After initializing the algorithm with the closest point, it maintains or "tracks" the closest point on the body surface. In [25], it is called a "tracking" algorithm. Extensions to this work include [26], which handles a moving surface and [27], which makes use of higher order derivatives and tangent plane projections at singularities. Finally in [28] 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.

In this paper, we present an efficient algorithm suitable for collision detection between objects modeled with parametric surfaces. Our algorithm makes direct use of the parametric surface representations and thus eliminates the need for polygonal meshes. Its efficiency derives from its full use of spatial and temporal coherence. It maintains two points on a pair of convex parametric surface patches, taking into account the motion of the bodies to which those surface patches belong. The motion of each closest point is found as a function of the motion of both bodies and the shape of both surfaces.

Our algorithm is based on the formulation of a control problem which is then solved with the design of a feedback stabilizing controller. We differentiate the geometric minimization problem to form the differential kinematics, then find and track the minimum distance by driving to zero the projections of a candidate minimum distance vector onto the surface tangents at the pair of candidate closest points (witness points). The witness points are moved using a control signal comprising a feedforward term that embodies the effects of object motion and object shape and a feedback term containing the projection errors. Thanks to the feedback term in the controller, our algorithm has a large basin of attraction, that is, it does not require exact initial conditions as required by other tracking methods.

We have presented a feedback stabilized minimum distance tracking algorithm for general convex parametric surfaces but not surface patches in [29]. In this paper, we extend those results into a hybrid dynamical control system so that we can handle parametric convex surfaces built from tiled-together convex surface pathes. Consideration of parametric surfaces by themselves is not quite sufficient, since within a parametric modeling environment, objects are generally modeled using collections of tiled-together surface *patches*. Each surface patch is bordered by curve segments formed at the intersection of two adjacent surface patches and each curve segment is bordered in turn by points formed at the intersection of three or more adjacent surface patches. These surface patches, curve segments, and points of parametric objects may be termed *faces*, *edges*, and *vertices*, respectively or generally: *features*. They correspond to planar surface patches, line segments, and points of polyhedral objects. For a parametric object formed using tiled-together surface patches, a Voronoi diagram also exists [30] [31]. Note that determination of Voronoi regions for curved objects can be computationally expensive; however, Voronoi diagrams are generally computed numerically before the simulation is initiated. Consequently, this step does not affect the real time performance of Voronoi based algorithms.

In this paper we treat bodies described by a collection of tiled together surface patches. We restrict the surface patches to be convex and for now, we also require the bodies to be convex [1]. That is, the convex surface patches shall be oriented and joined together at their boundaries in such a way that a line joining any two points in the interior of the compact body will be wholly contained in that body. While defining convexity for a body is straightforward even when it is a tiled body, defining convexity for a surface patch requires special consideration. Like the definition for a convex space curve [32], we have defined a convex surface patch as *any* patch cut from a compact convex body. Convexity of the surface patch depends on convexity of the body from which it is cut, and not on the curvature of the bounding curves that lie in the surface.

Now, if each of the two surface patches in question are both convex and the closest points lie within the Voronoi regions of each opposing surface, then there exists a unique minimum distance whose endpoints (closest points) lie within the surface patches [9] [8]. Under our algorithm, this unique minimum solution enjoys a large basin of attraction. Thus large initialization errors are tolerated. A local guarantee follows from the asymptotic stability of the system formed by wrapping the minimum distance tracking problem with a stabilizing feedback controller. Stability of this dynamical control system may be preserved through discretization, even when explicit integration routines are used. Once convergence to the unique minimum distance between convex surfaces is achieved, the algorithm tracks the minimum distance making primary use of the feedforward term.

[1]Later, after a means of tracking the *furthest* points on two intersecting convex surface patches has been developed, the restriction of convexity on the tiled-together bodies can be lifted. These topics will be addressed in future papers.

If not only all surface patches comprising an object are convex, but the objects taken pairwise into consideration are also convex (each volume forms a convex set of points,) then the uniqueness of the minimum distance can be extended to the object pairs. If in addition any two surface normals taken from a single surface patch never span more than 180 degrees, then the Voronoi diagram can be used to further advantage to create efficient algorithms to find the unique minimum distance between object pairs. The 180 degree restriction is equivalent to requiring that the Gauss map of the surface patch can be fit into an open hemisphere [33]. This stipulates that even the sphere and ellipse are to be tiled into at least two surface patches. Virtually without modification, the Lin-Canny algorithm, based on the Voronoi diagrams, may be applied to convex parametric objects to find the closest features. Computational efficiency follows from the incorporation of the Lin-Canny closest feature algorithm since now pairs of whole objects may be considered rather than all possible pairs of surface patches. In this paper, we present a discrete switching algorithm based on Lin-Canny that extends the applicability of our narrow phase to whole convex objects. We show how the discrete switching controller acting on features and the continuous controller acting on witness points within surface patches may be combined into a hybrid dynamical system to effectively extend the basin of attraction of the unique minimum distance.

The body of the paper begins in section II with the presentation of the proposed dynamic model. Controller design and analysis takes place in section III. The switching algorithm which generalizes the method to a collision detector that may be used on objects composed of convex surface patches will be presented in section IV. In section V, simulation results are presented and finally, section VI concludes the paper.

II. Modeling

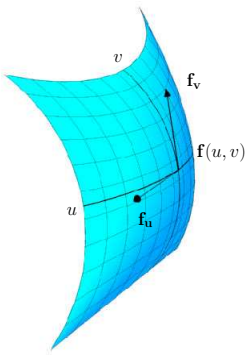

Fig. 1. A parametric surface patch showing isometric curves for parameters u and v and surface tangents at point $\mathbf{f}(u,v)$

Let there exist a parametric representation for the surface patch shown in Figure 1. Use $\mathbf{f}$ to denote a position vector to a point on the surface patch. And 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 patch continuity through at least two differentiations. We also require that both surface patches be convex.

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 patch 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.

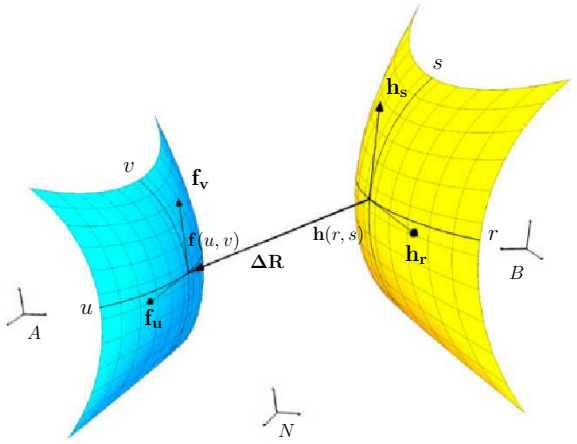

Fig. 2. Two parametric surface patches, $\mathbf{f}$ belonging to body A and $\mathbf{h}$ belonging to body B, with a position vector $\mathbf{\Delta R}$ between two arbitrary points

Two parametric surface patches are plotted in Figure 2 with their corresponding isoparametric curves. On these patches, 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 1. $\mathbf{\Delta R}(u,v,r,s)$ is a vector between these arbitrary points.

Note that when the difference vector $\mathbf{\Delta R}$ is normal to *both* convex surface patches, the requirements of the unique minimum distance solution 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 closest pair $\mathbf{f}(u^\star, v^\star)$ and $\mathbf{h}(r^\star, s^\star)$. The minimum distance is then equal to the Euclidian norm of $\mathbf{\Delta R}$.

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 patch $\mathbf{f}$; and similarly define Ψ^r, Ψ^s as the projections of the difference vector $\mathbf{\Delta R}$ onto the tangents $\mathbf{h_s}$ and $\mathbf{h_r}$ of surface patch $\mathbf{h}$ 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 closest pair are met: the difference vector $\mathbf{\Delta R}$ is perpendicular to both convex surface patches at $\mathbf{f}$ and $\mathbf{h}$. Note that it is possible to define the minimum distance condition by an alternate set of equations as presented in [28]. This alternative formulation makes use of surface tangents of one surface patch and normals of both surface patches. Although we use the set (1) - (4) in our further derivation in this paper, very similar results can be achieved using the alternate set.

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

Given the set of equations (1) - (4), one way to find the closest 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 is the approach undertaken in [28]. In the present work, rather than obtaining 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 respect to time. The differentiation operation causes the motion of the surface patches and the time rates of change of the parameters $du/dt, dv/dt, dr/dt$ and ds/dt, called the parametric velocities, to appear 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 closest 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 shape *and* motion of the surface patches.

Although we use vector expressions throughout the paper, we are careful 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 equations (1) and (2) in the body frame A and the vectors in (3) and (4) in the body frame B (see Figure 2). This choice results in simpler matrix expressions.

Consider the case where each surface patch is attached to a rigid body in motion. In Figure 2 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 the relative motion of these bodies with respect each other will be specified by the vector ${}^{\mathbf{A}}\boldsymbol{\omega}^{\mathbf{B}}$.

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

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

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

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

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

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

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

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

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

where

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

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

In this state space realization, the projection errors Ψ are taken to be the state variables. The inputs w are time derivatives of surface parameters whereas the system outputs denoted by y are the estimates $x = [u, v, r, s]^T$ of the parametric values u^*, v^*, r^* and s^* that locate the closest points on each surface patch 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.

III. Control

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

The control input vector w is composed of time derivatives of surface parameters, i.e. the elements of w are speeds along the parametric curves. The objective of the controller is to continually update these speeds to regulate the projection errors to zero while staying within the patches, that is, to maintain the closest pair on the surface patches.

With the model (9)-(11) in hand, the closest 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 over a small range of operation [34]. Since feedback linearization takes the higher order terms into account, it results in a large basin of attraction for the controller.

First, in order to feedback linearize the model, an inner feedback loop is designed. Assuming the matrix M is positive definite in the range of operation, we define the control input vector w in terms of a new input vector μ as

$$w = M^{-1}\,(\mu - b) \tag{12}$$

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

$$\dot{\Psi} = \mu \tag{13}$$

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

$$\mu = -K\,\Psi \tag{14}$$

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

Asymptotic stability of the overall closed loop system is guaranteed. This observation follows from the facts that there are no internal dynamics associated with this input-output linearization and positive definiteness of M results in monotonic convergence of the surface parameters. Note that the matrix M has a structured form as the first fundamental matrix plus minimum distance times the second fundamental matrix [35] when evaluated at the closest points. Positive definiteness of M is guaranteed for a neighborhood of the closest points if both of the surface patches are convex.

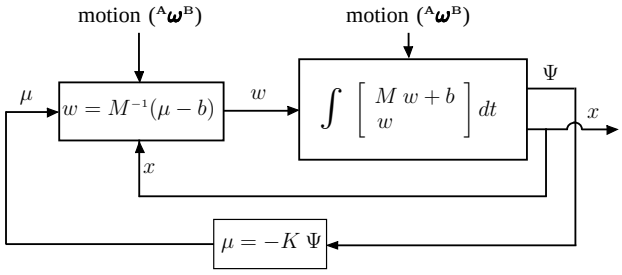

Fig. 3. Control block diagram showing an inner feedback linearization loop and an outer linear control loop

Figure 3 shows the block diagram of the completed controller design. Recognizable here is the inner loop that renders the projection error equations linear from input μ to output x. The outer loop achieves the desired dynamics via full state feedback with gain K.

Furthermore, the desired outputs, i.e. the parametric values of the witness points, are continually maintained using the control input vector w. This is achieved by integrating the input vector w with initial conditions extracted from the starting points. To guarantee that the surface parameters remain within the surface patch boundaries, a *saturated* version of the control input vector is utilized when a witness point lies on the boundary of a surface patch. This is implemented simply by setting the components of the input vector that are infeasible (point out of the surface patch) to zero. Note that, as will be presented in the next section, switching from one surface patch to another is dictated by the Voronoi diagram of the current surface patch and the witness point on the opposing surface patch and is decoupled from the continuous dynamics of the witness point on the current patch. Therefore, we require the surface parameters remain within the surface patch boundaries until a discrete transition is triggered.

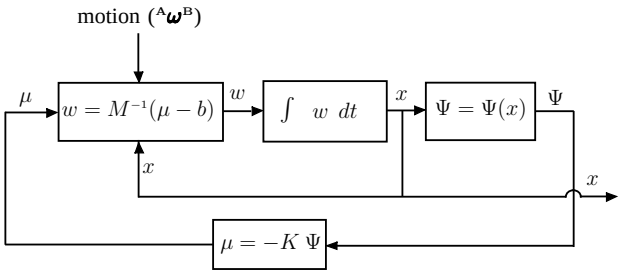

Fig. 4. Control block diagram showing an alternative implementation

In practice, the state vector Ψ is augmented with the input vector w to perform all integrations in a single operation. In fact, even a simpler implementation is possible as demonstrated in Figure 4. Since the projection errors Ψ can be directly calculated through equations (1) to (4), the differentiated kinematic model can be replaced by this set of nonlinear equations. Note that although the differentiated kinematic model is replaced, the controller design remains unchanged.

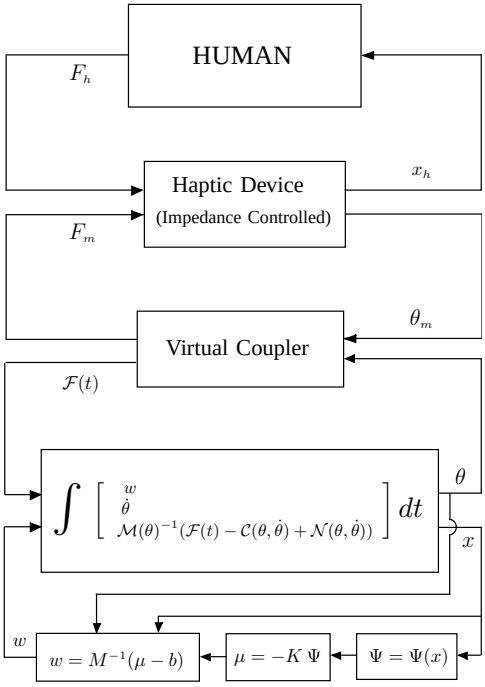

Fig. 5. Combining minimum distance tracking with dynamics

It is also possible to combine the closest pair tracking algorithm with the forward dynamics solution. One such case is shown in Figure 5 where a haptic rendering scheme is presented in block diagram form [36]. The components of Figure 5 include the human user, the physical haptic interface device, the computationally implemented virtual coupler and finally the virtual environment. The virtual environment includes both the forward dynamics solver and the minimum distance maintenance algorithm. Here, motion of the bodies is calculated simultaneously with the maintenance of the minimum distance

in a single operator. 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.

Finally, it is important to mention that the algorithm need not be initialized with the exact closest points. Any initial point within the region of attraction of the designed nonlinear controller (a neighborhood of the closest points where M is positive definite) will converge to the closest pair since the controller is asymptotically stable. Moreover the convergence rate can be adjusted by tuning the controller gain K as demonstrated in section V and further discussed in section VI.

IV. Switching Algorithm and Hybrid System Formulation

To handle changing external features and thereby extend the local stability to cover a much larger basin of attraction, we have adopted the mathematical language of hybrid dynamical systems, in particular the notation in [37], to describe collision detection between bodies composed of tiled together parametric surface patches. Hybrid systems contain both discrete and continuous state variables and exhibit both discrete and continuous state dynamics. In certain hybrid systems, discrete and continuous dynamics not only coexist, but *interact* such that changes in the discrete and continuous dynamics occur in response to discrete and/or continuous state variables. In the case of collision detection between parametric bodies, however, the coupling between the discrete and continuous dynamics is limited. When tracking the minimum distance between a body and a point, the continuous state that describes the motion of the closest point on the body is decoupled from the discrete state that specifies the relevant feature on the body. That is, the switching times do not depend on the continuous states. On the other hand, when tracking the minimum 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: minimum distance tracking between two bodies. Thereafter, we will show how certain variables may be removed from the argument lists for the various functions to describe the case of minimum 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 f's and ϕ's) with a superscript j and indicate the number of transitions 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 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 the witness point on the opposing body, not on the motion of the witness 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 closest point on the other body and the body motion, not the motion of the closest 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-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.

The coordinated action of the discrete and continuous dynamics in the hybrid system ensures that witness points initialized anywhere on two convex bodies composed of tiled convex surface patches will converge to the unique closest points. The discrete switching algorithm finds the appropriate closest features and then the continuous controller handles final convergence to the closest points. Once initialization errors are suppressed, the hybrid system continues to track the closest points as the bodies move and new portions of the bodies' shape are brought into play. The feedforward term of the controller accounts for the motion and shape of the surface patches as the feedback term rejects disturbances and continually drives the witness points toward the closest points. When witness points cross Voronoi boundaries of the opposing surface patch, discrete switching among features occurs as

dictated by the hybrid system dynamics. Between convex bodies, the inter-feature distance cannot increase at the switching instants [9], so convergence properties are maintained.

To extend our algorithm to handle non-convex bodies composed of convex surface patches, it is only necessary to modify the switching algorithm. Extensions to handle interpenetration are also relatively straightforward, which we are exploring in current and future research.

V. Simulation Results

We have developed a suite of computer simulations to check the dynamic formulation and the hybrid control algorithm discussed in the previous sections. Our simulations are implemented in Matlab/Simulink and sample results are presented below. Not presented here, but worth mentioning, are demonstrations on a real-time operating system that features haptic display. See [38] for a description of our hardware.

A. Convex Surface Patch and a Point

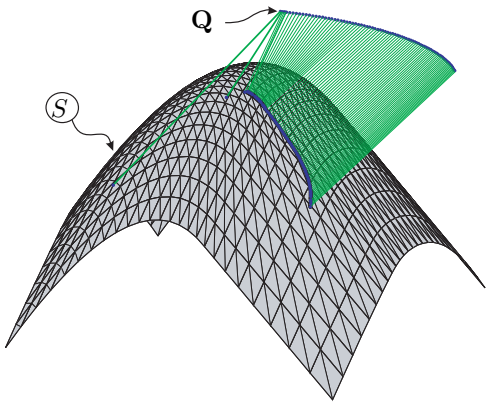

Fig. 6. The initial convergence of a witness point to the closest point on surface patch S is followed by smooth tracking as point $\mathbf{Q}$ moves according to a pre-specified trajectory.

Figure 6 shows a simulation that produces the tracking behavior of a witness point on the convex surface patch S as a point $\mathbf{Q}$ traces out a pre-specified path.[2] Lines connect the pair of witness points in each of the snapshots shown. At startup, the witness point on S has a large initialization error. One can observe that the algorithm quickly converges to the correct answer as indicated by the first three snapshots.

Figure 7 demonstrates asymptotic convergence of the two normalized projection errors Ψ^u and Ψ^v, where each projection error is normalized using the length of the difference vector $\mathbf{\Delta R}$ and the appropriate tangent vector. From this figure, it is evident that the algorithm quickly converges from a poor initialization value to the closest points and then maintains the closest points as the surfaces move.

The inset in Figure 7 shows how the convergence rate may be adjusted by tuning the linear feedback gain K. In the simulation shown, a very non-aggressive feedback gain of $K = 100$ is used with a fixed-step integrator running at 1 kHz and it takes 75 steps to recover from the initialization

[2]For haptic rendering a forward dynamics solver would produce the motion. Here we are only testing the minimum distance tracking algorithm.

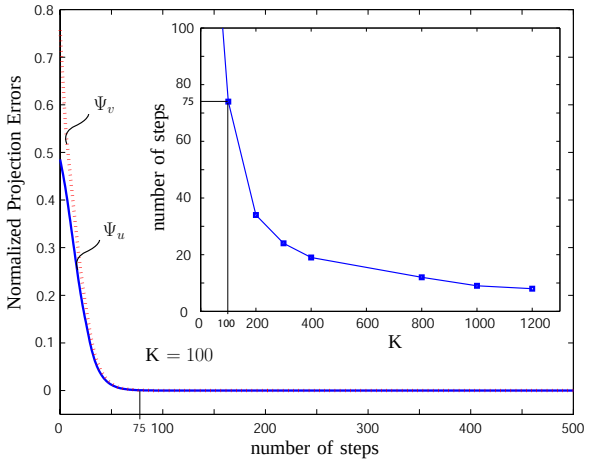

Fig. 7. Using a value of 100 for the feedback control gain K, exponential convergence of the projection errors Ψ_u and Ψ_v occurs within 75 simulation steps. As shown in the figure inset, the convergence rate can be significantly increased by increasing the gain K.

error and converge to the steady state. However, under the same conditions convergence can be achieved in 10 steps if the gain is set to $K = 1000$. In the tracking phase, i.e. in steady state after the initialization errors are compensated, each update takes only one time step.

In the discrete time implementation, the admissible values of K have an upper limit due to discretization of the algorithm. This limit depends on the integration scheme used and the step size chosen. There exist techniques whereby the convergence rate (determined by gain K) and step size may be traded off against one another while maintaining stability. For example, in [39] and [40], standard discrete time controller design techniques are utilized to obtain relationship between controller gain K and integration step size. With these techniques, the upper limit for K can be chosen such that the stability of the algorithm is guaranteed even after discretization.

There also exists a lower limit on K which is dictated by the shape and motion of the bodies and the initialization error of the witness points. This lower limit on K is required to guarantee that the convergence rate of the algorithm is faster than the disturbance due to motion of the bodies. The lower limit on K can be computed online or pre-set before simulation if the motion bounds are known before simulation time.

Note that, for multibody dynamics simulation the lower limit on K is significantly (orders of magnitude) smaller than the upper limit dictated by discretization and does not constitute a concern since choosing a gain close to the upper limit is desired for fast convergence. For example, in this particular simulation using a fixed-step second order explicit Runge-Kutta integration scheme running at 1 KHz, stable values of K range from 0.5 to 1500.

B. A Convex Planar Body of Three Convex Curves and a Point

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

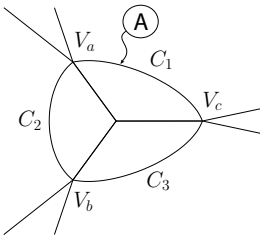

Fig. 8. A convex planar body A composed of curves C_1, C_2, C_3 joined at vertices V_a, V_b, V_c. Lines indicate boundaries between interior and exterior Voronoi regions.

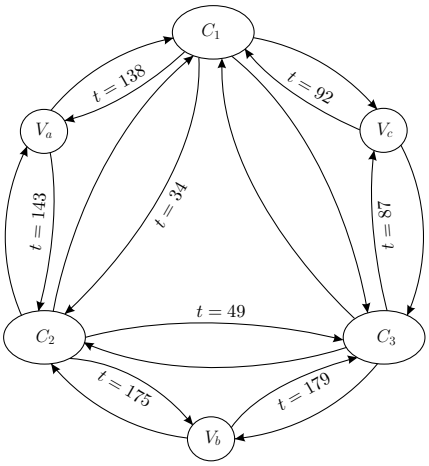

Fig. 9. The automaton used to govern the motion of the witness point on body A of Figure 8. Switching times for the simulation shown in Figure 10 are also indicated.

axes are each the locus of points equidistant to two curves of A. The automaton that describes the hybrid dynamics of a witness 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 witness point lies on a particular curve. The continuous dynamics within a particular mode govern the motion of the witness point so long as it lies on the correspondingly labelled curve. The three smaller circles each describe the (trivial) dynamics of the witness 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.

Figure 10 shows a simulation that produces the tracking behavior of a witness point on A as a point P (the other member of the closest 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 minimum distances connecting the witness points. To begin, the witness 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 of the minimum distance is either perpendicular to a curve of A or anchored on a vertex of A. The times at which transitions between modes occur are shown both in Figure 10 and on the automaton in Figure 9. For example, at $t = 34$ seconds, P

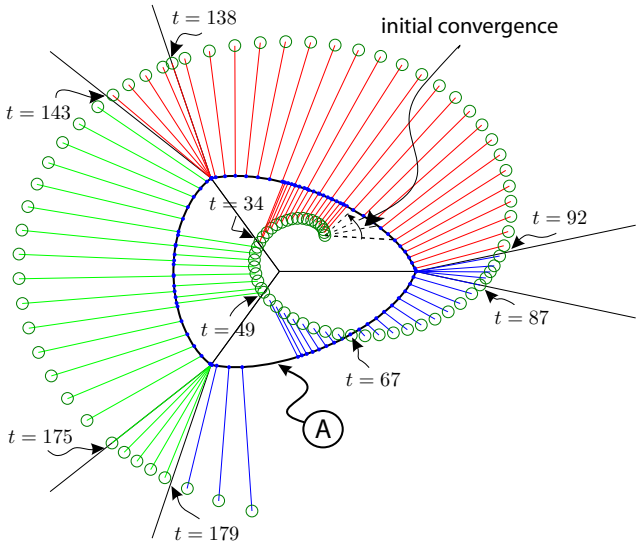

Fig. 10. The minimum distance is shown between A and a point tracing out a spiral that begins inside and ends outside A. Simulated transition times are indicated.

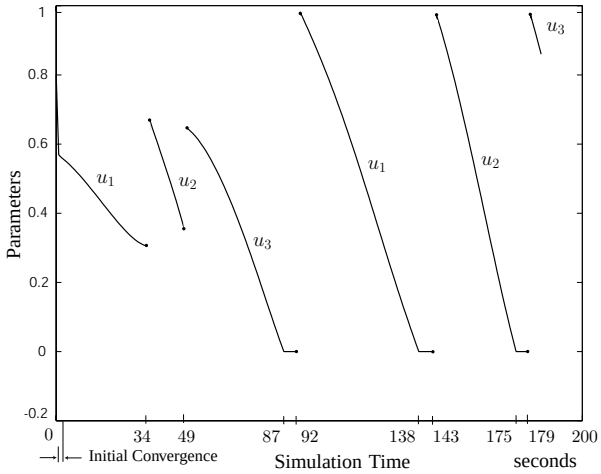

Fig. 11. The trajectory and sequence of parameters u_1, u_2, u_3 on curves C_1, C_2, C_3 for the simulation shown in Figure 10. Transition times are also indicated.

crosses the medial axis and tracking on C_1 jumps to tracking on C_2. At $t = 67$ seconds, P crosses the boundary of A, but tracking continues on C_3. At $t = 87$ seconds, the witness point anchors on V_c and at $t = 92$ 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. If all three curve parameters u_i, $(i = 1, 2, 3)$ are varied from 0 to 1, then the entire body A is traced out. The particular sequence and trajectory of values shown in Figure 11 pertains to the evolution of the witness point shown in Figure 10. Also visible in Figure 11 is the convergence of startup error at the very beginning of the simulation. As in all the simulations presented here, a very low controller gain K is utilized in this simulation to render initial convergence noticeable in the figures.

C. Two Planar Bodies

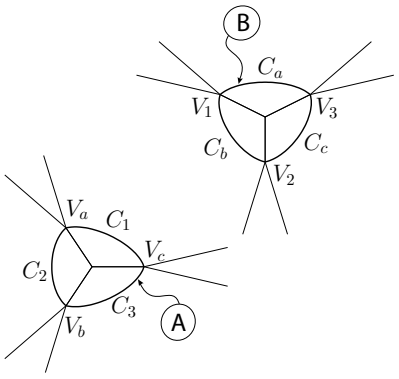

Fig. 12. Planar convex bodies A and B and their Voronoi regions.

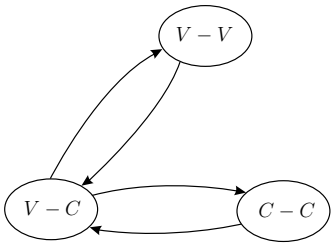

Fig. 13. The automaton (not enumerated) used to govern the motion of the witness points on bodies A and B of Figure 12.

Figure 12 shows two convex bodies each bounded by three convex curves. The exterior and interior Voronoi regions are also indicated. The hybrid dynamics of the minimum distance tracking problem is again governed by an automaton, a simplified version of which is shown in Figure 13. In this simplified version all three possible feature pairs (vertex-vertex, vertex-curve, and curve-curve) are indicated but they are not enumerated by feature name for each possible combination.

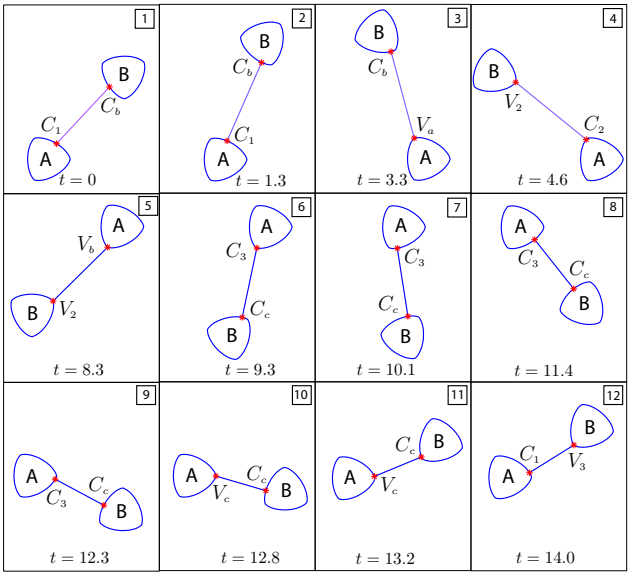

Fig. 14. The minimum distance between planar bodies A and B as A remains fixed and B spins and traces a circle around A. Twelve snapshots are shown, taken at irregular intervals. The active curves or vertices of A and B are indicated.

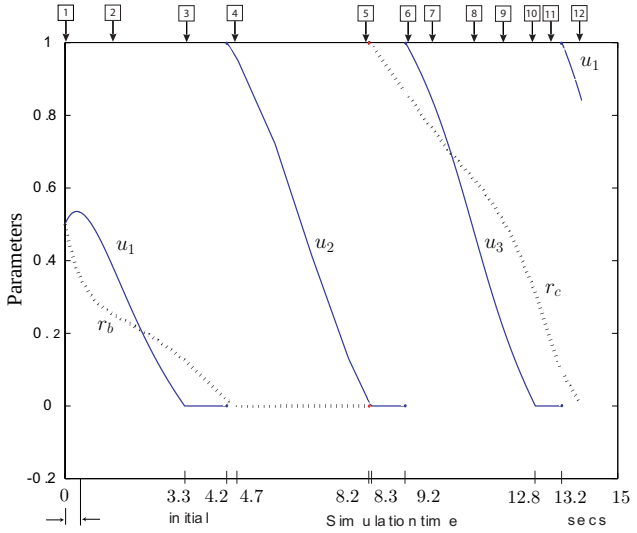

Fig. 15. The trajectory and sequence of parameters u_1, u_2, u_3 on curves C_1, C_2, C_3 and r_a, r_b, r_c on curves C_a, C_b, C_c for the simulation shown in Figure 14. Transition times are indicated.

Figure 14 shows the simulation results of the minimal distance tracking problem while A remains fixed and B undergoes a motion in which it spins around its own center which in turn traces out an inward spiral centered on A. Twelve snapshots taken at irregular but indicated time intervals are shown arrayed and numbered 1 through 12 in Figure 14. The active curves or vertices, which correspond to similarly labelled modes in the automata not shown, are also indicated in Figure 14. Figure 15 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 closest points on A and B can be seen. The snapshot times are also indicated in Figure 15 as are the transition times.

D. Spatial Body and a Point

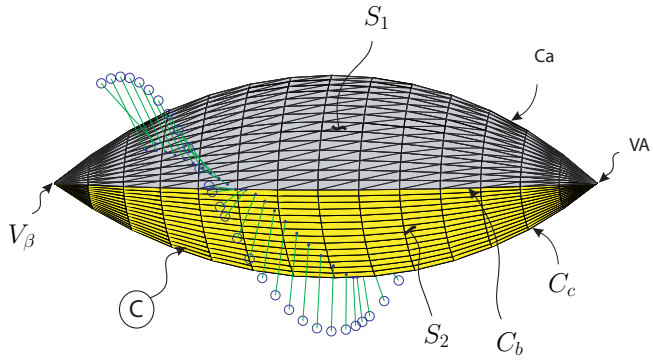

Fig. 16. The minimum distance is shown between a spatial body C and a point tracing out a helix. Body C is composed of 3 convex surface patches (only S_1 and S_2 are visible) joined at 3 curves (only C_a and C_b are visible) and 2 vertices (V_α and V_β).

Figure 16 shows the minimum distance between a spatial body C and a point that traces out a helix. Body C is formed by joining three convex parametric surface patches at their intersecting curves and vertices. The surface patches are labelled S_1, S_2, and S_3, the curves C_a, C_b, and C_c and

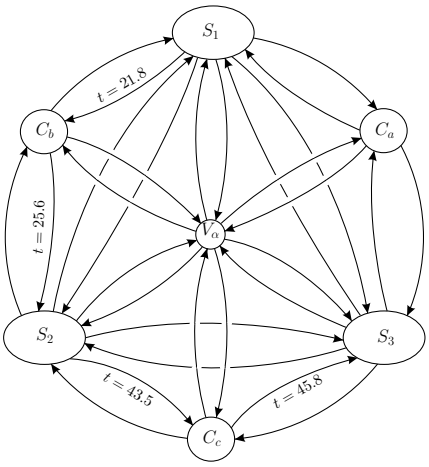

Fig. 17. The automaton used to govern the motion of the witness point on body C of Figure 16. The mode and transitions pertaining to vertex V_β are not shown. Switching times for the simulation shown in Figure 16 are also indicated.

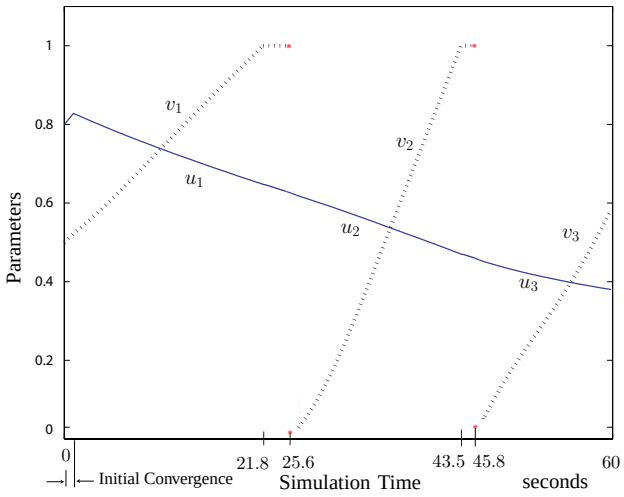

Fig. 18. Changes in curve parameters during tracking for the simulation shown in Figure 16. Transition times are indicated.

the vertices V_α and V_β. The visible components are indicated in Figure 16 and the corresponding modes are shown in the automaton of Figure 17, except for the mode for V_β. The mode V_β and its transitions look just like those for V_α but were omitted to make the figure readable. The sequence and trajectory of the surface parameters u_i and v_i, taken pairwise with $(i = 1, 2, 3)$ to locate the witness point on C are shown in Figure 18. Periods of time in which the witness point lies on the bounding curves can be recognized as constant values of 0 or 1.

E. Two Spatial Bodies

Figure 19 shows 5 snapshots taken at irregular intervals for the minimum distance tracking between two spatial bodies A and B. In this sample simulation, Body A is fixed and Body B moves around Body A in a specified time dependent motion. As in the previous example, both bodies are formed by joining three convex parametric surface patches at their intersecting curves and vertices. The curves and vertices are not marked but

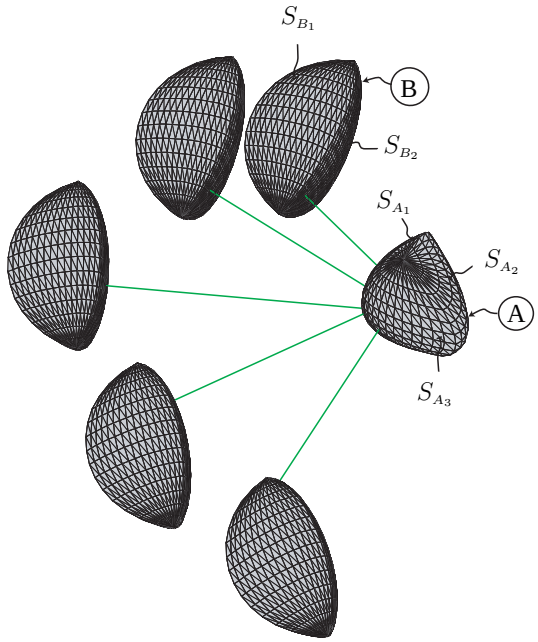

Fig. 19. Witness points move on bodies A and B transitioning between convex surface patches when necessary as B traces a pre-specified trajectory around A.

the visible surface patches are indicated in Figure 19. Surface patches of Body A, with surface parameters u and v are labelled S_{A_1}, S_{A_2}, and S_{A_3} whereas surface patches of Body B, with surface parameters r and s are labelled S_{B_1}, S_{B_2}, and S_{B_3}.

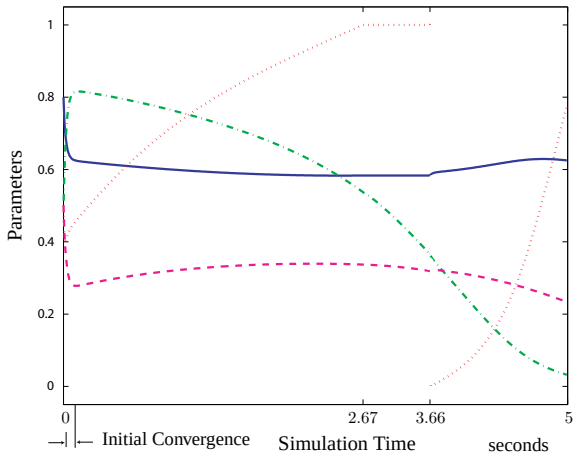

Fig. 20. Changes in surface parameters during initial convergence and tracking for the simulation shown in Figure 19. Transition times are indicated.

The sequence and trajectory of the surface parameters u_i, v_i, r_i and s_i of both bodies taken pairwise to locate the witness points on A and B are shown in Figure 20. One can also recognize that at $t = 2.67$ the witness point on surface patch S_{A_1} reaches its boundary curve and evolves on that boundary until $t = 3.66$ when the witness point moves onto the next surface patch S_{A_2}.

VI. Discussion and Conclusions

We have contributed a narrow phase collision detection algorithm with certain attractive properties that follow from its

formulation as a dynamic control problem. These properties include a means of determining the highest gain K that maintains stability under a given discretization and a large and easily characterized basin of attraction of the stabilized closest points.

Since the continuous dynamics of the projection errors (Eqs. (13) and (14)) are linear, stability analysis is simple: the characteristic equation produced by discretizing the linear dynamics according to a given numerical method may simply be inspected for its pole locations. So long as the poles remain inside the unit circle and given step size h, the gain K may be increased to achieve the highest stable algorithm speed. For more detail on the use of discrete time controller design techniques for the analysis of convergence rates under various integrators, see [41] [42] [43]. Once stability of the continuous dynamics Eqs. (13) and (14) is established, convergence of the witness points to the closest points depends on an analysis of Eqs. (12) and (14). A Lyapunov function produced by squaring the surface parameter errors can be used to show convergence given positive definite matrix M and sufficiently large gain matrix K. The positive definiteness of M depends on the surface shape and current surface parameter error, and can be used to characterize the basin of attraction of the closest points.

Our chief contribution, then, is an algorithm whose limits of performance can be exposed using straightforward analysis. Once these are in hand, the algorithm can be driven to its limits in speed. Note that the determination of stability-preserving gains K for algorithms based on Newton iteration is a much more complicated affair, since these are discrete and nonlinear methods. Also, the basin of attraction of the closest points within a Newton iteration approach is a complicated and not necessarily connected set.

When it comes to comparing the ultimate computational efficiency of our algorithm with that of the previously available methods, our argument relies on noting the close relationship between our feedback control method and the Newton iteration and gradient methods. We have reported a few results on the speed of our algorithm in machine-independent fashion (Figure 7). Our algorithm subsumes all the computational efficiency features of the Newton and gradient based algorithms. For example, without compensation for the motion of the bodies and under discretization using Euler's method, our method (in particular, as represented in Figure 3) reduces to that published in [28]. Thus the computational efficiency of [28] is inherited by our algorithm and can only be improved upon with a broader choice of numerical method and explicit limits of performance.

The features discussed so far pertain to the control-based algorithm that handles the closest points on two convex parametric surface patches. We presented a sister algorithm that handles switching among convex surface patches, bounding curves, or vertices that make up each of two convex bodies. This closest feature algorithm significantly extends the attractive properties outlined above, for they effectively increase the basin of attraction of the closest points beyond the closest features themselves.

We showed how the discrete closest feature and continuous stabilizing closest point algorithms may be combined in a hybrid dynamical system, wherein the membership of the witness point on the opposite body of a particular Voronoi region controls the switching among features on each body. Transitions between surface patches are handled in a manner that decreases inter-feature distance, according to the properties of the V-Clip [9] or the Lin-Canny [8] algorithm.

In further support of applying a control theoretic approach to collision detector technology, note that similar feedback stabilization techniques have yielded significant advantage when applied to certain problems in robotics and multibody dynamics. 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 [44]. Baumgarte [45] introduced the use of feedback to stabilize the solution of a set of differential equations to the manifold in which the solution is constrained to lie by the constraint equations. Recently, Chiou et.al. [40] and Kurdila et.al. [46] demonstrated the use of input-output feedback linearization techniques to stabilize constraint violations in multibody dynamic analysis. In these papers, it is proved that Baumgarte stabilization can be recovered from the input-output feedback linearized system by a particular choice of linear feedback controller (full state feedback). It is also shown that alternative feedback choices result in more efficient, stable, and robust methods.

In closing, the work reported in this paper and our continuing work is focused on producing narrow phase algorithms with properties that can be exploited by a suitable broad phase algorithm to produce robust collision detectors for the more complicated problems of non-convex bodies and extremal distance between interpenetrating bodies. In the non-convex cases, multiple extrema exist and during switching, the closest points do not necessarily lie within neighboring features. In particular, it is the basin of attraction of closest points which we strive to extend despite body motion and sharp curvature so that the broad phase need not initialize close to the global extrema. A simpler yet robust broad phase and minimum number of narrow phase algorithms running in parallel means higher computational efficiency.

ACKNOWLEDGMENTS

The authors gratefully acknowledge the support of the National Science Foundation under award number IIS:PECASE #0093290, and some very fruitful discussions with Elmer Gilbert and Jessy Grizzle and the very helpful comments of anonymous reviewers.

REFERENCES

[1] P. Hubbard, "Collision detection for interactive graphics applications," *IEEE Transactions on Visualization and Computer Graphics*, vol. 1, no. 3, pp. 218–230, 1995.

[2] 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 Trananactions on Visualization and Computer Graphics*, vol. 4, no. 1, pp. 21–36, 1998.

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

[4] A. Gregory, M. C. Lin, S. Gottschalk, and R. Taylor, "Fast and accurate collision detection for haptic interaction using a three degree-of-freedom force-feedback device," *Computational Geometry: Theory and Applications*, vol. 15, no. 1-3, pp. 69–89, 2000.

[5] M. Lin and S. Gottschalk, "Collision detection between geometric models: A survey," in *Proc. IMA Conference on Mathematics of Surfaces*, pp. 37–56, 1998.

[6] E. G. Gilbert, D. W. Johnson, and S. S. Keerthi, "Fast procedure for computing the distance between convex objects in three-dimensional space," *IEEE Journal of Robotics and Automation*, vol. 4, no. 2, pp. 193–203, 1988.

[7] C. J. Ong and E. G. Gilbert, "Fast versions of the Gilbert-Johnson-Keerthi distance algorithm: Additional results and comparisons," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 4, pp. 531–539, 2001.

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

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

[10] S. Ehmann and M. C. Lin, "Accelerated proximity queries between convex polyhedra using multi-level voronoi marching," *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2101–2106, 2000.

[11] S. Ehmann and M. C. Lin, "Accurate and fast proximity queries between polyhedra using convex surface decomposition," *Computer Graphics Forum (Proc. of Eurographics'2001)*, vol. 20, no. 3, pp. 500–510, 2001.

[12] J. M. Snyder, "Interactive tool for placing curved surfaces without interpenetration," in *Proceedings of the ACM SIGGRAPH*, pp. 209–218, 1995.

[13] Y. Adachi, T. Kumano, and K. Ogino, "Intermediate representation for stiff virtual objects," in *Proceedings Virtual Reality Annual International Symposium*, pp. 203–210, 1995.

[14] P. Stewart, Y. Chen, and P. Buttolo, "CAD data representations for haptic virtual prototyping," in *Proceedings of ASME Design Engineering Technical Conference*, 1997.

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

[16] C. Turnbull and S. Cameron, "Computing distances between NURBS-defined convex objects," in *Proceedings IEEE International Conference on Robotics and Automation*, vol. 4, pp. 3685–3690, 1998.

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

[18] M. C. Lin and D. Manocha, "Fast interference detection between geometric models," *Visual Computer*, vol. 11, no. 10, pp. 542–561, 1995.

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

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

[21] 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 *Proceedings of ACM SIGGRAPH*, pp. 321–334, 1993.

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

[23] R. E. Barnhill and S. N. Kersey, "A marching method for parametric surface/surface intersection," *Computer Aided Geometric Design*, vol. 7, no. 1, pp. 257–280, 1990.

[24] C. L. Bajaj, C. M. Hoffmann, R. E. Lynch, and J. E. H. Hopcroft, "Tracing surface intersections," *Computer Aided Geometric Design*, vol. 5, no. 4, pp. 285–307, 1988.

[25] T. V. Thompson II, D. E. Johnson, and E. Cohen, "Direct haptic rendering of sculptured models," in *Proceedings Symposium on Interactive 3D Graphics*, pp. 167–176, 1997.

[26] T. V. Thompson II, D. D. Nelson, E. Cohen, and J. Hollerbach, "Maneuverable NURBS models within a haptic virtual environment," in *Proceedings of ASME International Mechanical Engineering Congress and Exposition*, vol. 61, pp. 37–44, 1997.

[27] D. E. Johnson and E. Cohen, "An improved method for haptic tracing of a sculptured surface," in *Proceedings of ASME International Mechanical Engineering Congress and Exposition*, vol. 64, pp. 243–248, 1998.

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

[29] V. Patoglu and R. Gillespie, "Extremal distance maintenance for parametric curves and surfaces," in *Proc. 2002 IEEE International Conference on Robotics and Automation*, pp. 2817–2823, 2002.

[30] R. Ramamurthy and R. T. Farouki, "Voronoi diagram and medial axis algorithm for planar domains with curved boundaries -I. theoretical foundations," *Journal of Computational and Applied Mathematics*, vol. 102, pp. 119–141, 1999.

[31] H. Alt and O. Schwarzkopf, "The Voronoi diagram of curved objects," in *Proceedings of the Eleventh Annual Symposium on Computational Geometry*, pp. 89–97, 1995.

[32] P. Bhattacharya and A. Rosenfeld, "Convexity properties of space curves," *Pattern Recogn. Lett.*, vol. 24, no. 15, pp. 2509–2517, 2003.

[33] G. Elber, "Accessibility in 5-axis milling environment," *Computer Aided Design*, vol. 26, no. 11, pp. 796–802, 1994.

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

[35] J. Gallier, *Geometric Methods and Applications: For Computer Science and Engineering*. Springer, 2001.

[36] R. Adams and B. Hannaford, "Control law design for haptic interfaces to virtual reality," *IEEE Transactions on Control Systems Technology*, vol. 10, pp. 3–13, 2002.

[37] M. Branicky, V. Borkar, and S. Mitter, "A unified framework for hybrid control: model and optimal control theory," *IEEE Transactions on Automatic Control*, vol. 43, no. 1, pp. 31–45, 1998.

[38] R. B. Gillespie, M. Hoffman, and J. Freudenberg, "Interface for hands-on instruction in system dynamics and embedded control," in *Proceedings of IEEE Virtual Reality Conference*, vol. 22-23, pp. 410–415, 2003.

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

[40] J. C. Chiou and S. D. Wu, "Constraint violation stabilization using input-output feedback linearization in multibody dynamic analysis," *Journal of Guidance, Control and Dynamics*, vol. 21, no. 2, pp. 222–228, 1998.

[41] E. V. Solodovnik, G. J. Cokkinides, and A. P. S. Meliopoulos, "On stability of implicit numerical methods in nonlinear dynamical systems simulation," in *Proceedings of the Thirtieth Southeastern Symposium on System Theory*, vol. 3, pp. 27–31, 1998.

[42] S. T. Lin and J. N. Huang, "Parameter selection for Baumgarte's constraint stabilization method using predictor-corrector approach," *AIAA Journal of Guidance, Control and Dynamics*, vol. 23, no. 3, pp. 566–570, 2000.

[43] S. T. Lin and M. C. Hong, "Stabilization method for the numerical integration of controlled multibody mechanical systems: A hybrid integration approach," *JSME International Journal Series*, vol. 44, no. 1, pp. 79–88, 2001.

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

[45] J. Baumgarte, "Stabilization of constraints and integrals of motion in dynamical systems," *Computer Methods in Applied Mechanics and Engineering I*, pp. 1–16, 1972.

[46] S. Kurdila, N. Fitz-Coy, D. McDaniel, and G. Webb, "Lie algebraic control for the stabilization of nonlinear multibody system dynamics simulation," *Nonlinear Dynamics*, vol. 20, pp. 55–84, 1999.