Extremal Distance Maintenance for Parametric Curves and Surfaces

Volkan Patoglu
Department of Mechanical Engineering
University of Michigan
Ann Arbor, MI 48109, USA
vpatoglu@umich.edu

R. Brent Gillespie
Department of Mechanical Engineering
University of Michigan
Ann Arbor, MI 48109, USA
brentg@umich.edu

Abstract

A new extremal distance tracking algorithm is presented for parametric curves and surfaces undergoing rigid body motion. The essentially geometric extremization problem is transformed into a dynamical control problem by differentiating with respect to time. Extremization is then solved with the design of a stabilizing controller. We use a feedback linearizing controller. The controller simultaneously accounts for the surface shape and motion while asymptotically achieving (and maintaining) the extremal pair. Thus collision detection takes place in a framework fully analogous to the framework used for the simulation of dynamical response.

1 Introduction

To trigger the appropriate impact response that generally accompanies a collision between objects in the physical world, there exists the matter that makes up those objects and occupies space. In simulation, however, an algorithm must be put in place to detect collisions between objects which have no matter. A collision detector must trigger the computation of interaction forces or impulses that act, in simulation, to prevent interpenetration. In virtual environments and simulations of robots interacting with their environment, collision detection is an important technology. In this paper, we present an algorithm suitable for collision detection between objects whose boundaries are represented with parametric surfaces.

The extremal distance E is defined as the minimum distance between two surfaces when they are disjoint, the local maximum penetration depth when they are interpenetrated and zero during tangential contacts [1]. We present an algorithm that maintains the two extremal points on a pair of parametric surfaces as those surfaces move. The motion of each extremal point is found as a function of the motion and shape of both surfaces. Additionally, initialization errors are tolerated and driven to zero. Although the algorithm handles only local extrema on convex shapes, with an extension involving global bounding box comparisons and surface transitions, general complex surface shapes can be handled.

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. Parametric surfaces such as non-uniform rational B-splines (NURBS) have the advantages of compact representation, higher order continuity and cost effective computation of surface derivatives and normals. Thus our algorithm is relevant to interactive simulation of models created using CAD and CAE packages, since it eliminates the need for intermediate representations such as polygonal meshes or faceted surfaces.

The new algorithm is especially useful when a human interacts with a model through a haptic interface. It may be used to efficiently track the maximum penetration distance and the appropriate common normal between the model and an image of the user's finger in the virtual environment. The penetration distance and normal vector may then be used to produce a reaction force through the haptic interface. The algorithm allows direct haptic rendering of the CAD surface rather than rendering of an intermediary surface. Another powerful property of the algorithm is its dynamic nature, which takes the motion of the surfaces into account. This dynamic structure makes it possible to easily combine extremal point tracking with physics-based simulation. Although the present work is developed for collision detection, it is based on our previous work in cobot control [2] and related work in path following for mobile robots [3].

In section 2, we review the previous work in collision detection. The proposed dynamic model is derived in section 3. Controller design and analysis takes place in section 4. In section 5, simulation results are presented and finally, section 6 concludes the paper.

2 Background

This section reviews extremal point algorithms by classifying them according to the geometric model representations they can handle. Two main categories presented are the polygonal models and the nonpolygonal models.

2.1 Polygonal Models

Previous work in collision detection has widely concentrated on computing the distance between convex polyhedral objects. Polyhedral models are extensively studied because they are easier to deal with than more general models, are almost always compatible with 3-D modelling systems, and their resolution is sufficient for many tasks. State of the art algorithms for computing the distance between convex polyhedra are the algorithm by Gilbert, Johnson and Keerthi (GJK) [4], [5] and the algorithm of Lin and Canny [6], [7].

The GJK algorithm makes use of Minkowski difference and convex optimization techniques to calculate the minimum distance. The iterative algorithm 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.

Many of the existing polyhedral collision detection algorithms also utilize scheduling and spatial partitioning techniques to speed up the solution process. Since convex optimization techniques are used in these methods, non-convex polygonal shapes must be handled by dividing them into convex parts.

2.2 Nonpolygonal Models

Most of the available closest point algorithms for nonpolygonal models address the problem indirectly. One such indirect method uses adaptively refined meshes to convert the problem into a polyhedral one. Another indirect approach proposed by Adachi [8] and Stewart [9] uses intermediate tangent representations.

Although these indirect methods can be successfully implemented for some applications, there also exist cases when 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 nonpolygonal models. Gilbert et. al. extended their algorithm to general convex objects in [10]. In a related paper [11], Turnbull modifies the widely used GJK algorithm to handle convex shapes defined using NURBS. Similarly, in [12] 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 [13] and Herzen [14]. Snyder improves these methods by modelling the collision detection between time dependent surfaces as a constrained minimization problem and solves it using interval Newton methods [15].

In the field of computer graphics, Kriezis [16] and Bajaj [17] propose modelling parametric surface intersections by differential equations and using tracing/marching methods to calculate them. Although the goal of these approaches is only to calculate surface intersections, the way the problem is modelled and appropriate points are traced are closely related to our extremal pair tracking algorithm.

Thompson et. al. contribute a different kind of closest point algorithm: rather than finding the closest point at each iteration, their algorithm continually updates the closest point based on the motion of the "end-effector" point and the curve or surface shape. After initializing the algorithm with the closest point, it maintains the closest point or "tracks" the end-effector. In [18], it is called a "tracking" algorithm. Extensions to this work include [19], which handles a moving surface and [20], which makes use of higher order derivatives and tangent plane projections at singularities. Finally in [21] 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.

Note that in the final surface-to-surface tracking algorithm of [21], motion of the surfaces and extremal point tracking are performed in a decoupled manner. Successive approximations are performed by multi-dimensional Newton's method to obtain a static solution before the surfaces are allowed to move for the next update. The combined velocity formulation is an approach to handle motion of the surfaces and extremal tracking simultaneously, but it only works for the contact case and needs exact initial conditions.

Our algorithm is also a tracking algorithm. However, it is based on a dynamic formulation of the motion of the extremal points and their dependence 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 combined with the dynamic simulation. The controller output is precisely 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". Because of its dynamic formulation, the tracking algorithm can be neatly combined with the dynamic simulation process.

3 Modeling

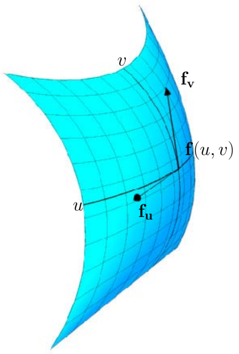

Figure 1: A parametric surface

Let there exist a parametric representation for the surface shown in Figure 1. 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.

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.

Two parametric surfaces are plotted in Figure 2 with their corresponding isoparametric curves. On these surfaces, two arbitrary points, $\mathbf{f}(u, v)$, $\mathbf{h}(r, s)$ 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 error vector $\mathbf{\Delta R}$ is normal to *both* surfaces, the requirements of the extremal distance defined in section 1 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 error 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 error vector $\mathbf{\Delta R}$ onto the tangents $\mathbf{h_s}$ and $\mathbf{h_r}$ of surface $\mathbf{h}$ as follows.

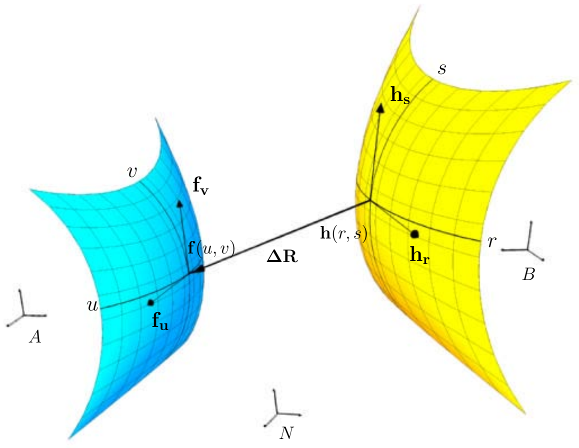

Figure 2: Two parametric surfaces with a position vector $\mathbf{\Delta R}$ between two arbitrary points

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

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

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

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

When the projection errors are all zero, the conditions for the extremal pair are met: the error 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 alternative set of equations as presented in [21]. This alternative formulation makes use of surface tangents and a normal of each surface. Although we use the set (1) - (4) in our further derivation in this paper, very similar results can be achieved using the alternative set.

Given a set of equations, 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 is the approach undertaken in [21].

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 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 differential equations for the projection errors.

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 2). This choice results in simpler matrix expressions.

Consider the case where each surface 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 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 ${}^{\mathbf{N}}\mathbf{V}^{\mathbf{B_o}}$.

$$\Psi^u = (\mathbf{f} - \mathbf{h}) \cdot \mathbf{f_u} \tag{5}$$
$$\Psi^v = (\mathbf{f} - \mathbf{h}) \cdot \mathbf{f_v} \tag{6}$$
$$\Psi^r = (\mathbf{f} - \mathbf{h}) \cdot \mathbf{h_r} \tag{7}$$
$$\Psi^s = (\mathbf{f} - \mathbf{h}) \cdot \mathbf{h_s} \tag{8}$$

Taking time derivatives of equations (5) to (8), and rearranging, one can present an expression for the projection error derivatives as

$$\dot{\Psi} = M\,U + b \tag{9}$$
$$\dot{x} = U \tag{10}$$
$$y = \Psi \tag{11}$$

where

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

and M and b are shown at the top of the next 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 and these estimates can be calculated as a by-product of the control effort that regulates the projection errors to zero. Details of this procedure is shown in the next section.

4 Control

Equation (9) defines 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 can be exactly calculated.

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 [22].

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

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

and apply this control input to (9). Then the nonlinear model 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 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. However, it is possible to synthesize different outer loop controllers to satisfy various design objectives.

Exponential stability of the overall controller 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 its order and input-output linearization leads to input-state linearization [22].

Figure 3 shows the block diagram of the completed controller design. Here again, the inner loop renders the system as a linear model utilizing the input vector μ whereas

$$M = \begin{bmatrix} \mathbf{f_u}\cdot\mathbf{f_u}+\mathbf{\Delta R}\cdot\mathbf{f_{uu}} & \mathbf{f_v}\cdot\mathbf{f_u}+\mathbf{\Delta R}\cdot\mathbf{f_{uv}} & -\mathbf{h_r}\cdot\mathbf{f_u} & -\mathbf{h_s}\cdot\mathbf{f_u} \\ \mathbf{f_u}\cdot\mathbf{f_v}+\mathbf{\Delta R}\cdot\mathbf{f_{uv}} & \mathbf{f_v}\cdot\mathbf{f_v}+\mathbf{\Delta R}\cdot\mathbf{f_{vv}} & -\mathbf{h_r}\cdot\mathbf{f_v} & -\mathbf{h_s}\cdot\mathbf{f_v} \\ \mathbf{f_u}\cdot\mathbf{h_r} & \mathbf{f_v}\cdot\mathbf{h_r} & -\mathbf{h_r}\cdot\mathbf{h_r}+\mathbf{\Delta R}\cdot\mathbf{h_{rr}} & -\mathbf{h_s}\cdot\mathbf{h_r}+\mathbf{\Delta R}\cdot\mathbf{h_{rs}} \\ \mathbf{f_u}\cdot\mathbf{h_s} & \mathbf{f_v}\cdot\mathbf{h_s} & -\mathbf{h_r}\cdot\mathbf{h_s}+\mathbf{\Delta R}\cdot\mathbf{h_{rs}} & -\mathbf{h_s}\cdot\mathbf{h_s}+\mathbf{\Delta R}\cdot\mathbf{h_{ss}} \end{bmatrix}$$

$$b = \begin{bmatrix} ({}^{\mathbf{N}}\mathbf{v}^{\mathbf{A}} - {}^{\mathbf{N}}\mathbf{v}^{\mathbf{B}} + {}^{\mathbf{A}}\omega^{\mathbf{N}} \times (\mathbf{p}^{\mathbf{OA_o}} - \mathbf{p}^{\mathbf{OB_o}}) - {}^{\mathbf{A}}\boldsymbol{\omega}^{\mathbf{B}} \times \mathbf{h}) \cdot \mathbf{f_u} \\ ({}^{\mathbf{N}}\mathbf{v}^{\mathbf{A}} - {}^{\mathbf{N}}\mathbf{v}^{\mathbf{B}} + {}^{\mathbf{A}}\omega^{\mathbf{N}} \times (\mathbf{p}^{\mathbf{OA_o}} - \mathbf{p}^{\mathbf{OB_o}}) - {}^{\mathbf{A}}\omega^{\mathbf{B}} \times \mathbf{h}) \cdot \mathbf{f_v} \\ ({}^{\mathbf{N}}\mathbf{v}^{\mathbf{A}} - {}^{\mathbf{N}}\mathbf{v}^{\mathbf{B}} + {}^{\mathbf{B}}\omega^{\mathbf{N}} \times (\mathbf{p}^{\mathbf{OA_o}} - \mathbf{p}^{\mathbf{OB_o}}) - {}^{\mathbf{B}}\omega^{\mathbf{A}} \times \mathbf{f}) \cdot \mathbf{h_r} \\ ({}^{\mathbf{N}}\mathbf{v}^{\mathbf{A}} - {}^{\mathbf{N}}\mathbf{v}^{\mathbf{B}} + {}^{\mathbf{B}}\omega^{\mathbf{N}} \times (\mathbf{p}^{\mathbf{OA_o}} - \mathbf{p}^{\mathbf{OB_o}}) - {}^{\mathbf{B}}\omega^{\mathbf{A}} \times \mathbf{f}) \cdot \mathbf{h_s} \end{bmatrix}$$

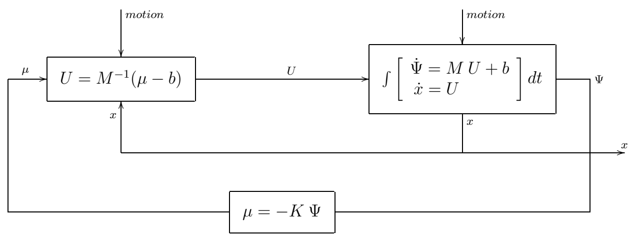
motion
motion
μ
$U = M^{-1}(\mu - b)$
U
$\int \begin{bmatrix} \dot\Psi = M\,U + b \\ \dot x = U \end{bmatrix} dt$
Ψ
x
x
x
$\mu = -K\,\Psi$

Figure 3: Control block diagram showing an inner feedback linearization loop and an outer linear control loop

the outer loop achieves desired dynamics of the linear system via full state feedback with gain matrix K. In practice, the values of the gain matrix K are chosen to produce a rate of convergence that outruns any potential disturbance due to motion and shape.

Furthermore, the desired outputs, i.e. the parametric values of the extremal pair, are continually maintained with the knowledge of 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.

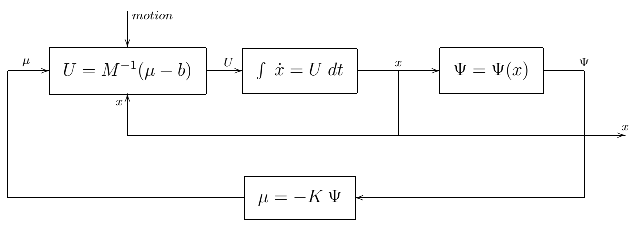

Figure 4: Control block diagram showing an alternative implementation

In fact, even a simpler implementation is possible. Figure 4 demonstrates this equivalent case. Since the projection errors Ψ can be directly calculated through equations (5) to (8), the derived dynamic model can be replaced by these set of nonlinear equations. Note that, although the derived dynamic model is replaced, the controller design stays unchanged.

It is also possible to combine the extremal pair tracking algorithm with dynamics as discussed in section 1. One such case is shown in Figure 5. 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. Decisions about external control forces are made by the motion controller. The motion controller can be designed to achieve different tasks, for example to capture influences of human acting through a haptic interface. For this implementation, all integration (update and maintenance) operations are combined in a single operator.

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 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.

5 Simulation Results

We developed a computer simulation to verify the validity of the dynamic formulation and the effectiveness of the control algorithm discussed in the previous sections. Our simulations are implemented in Matlab/Simulink and sample results are presented below.

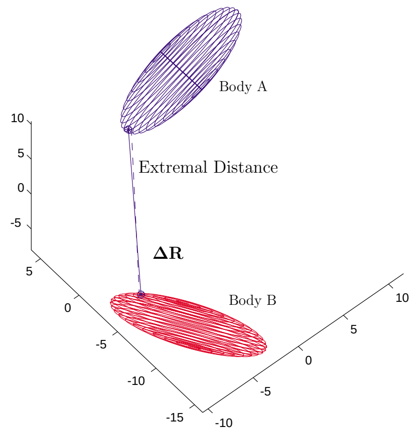

Figure 6: Simulation construction: two ellipsoids in rigid motion

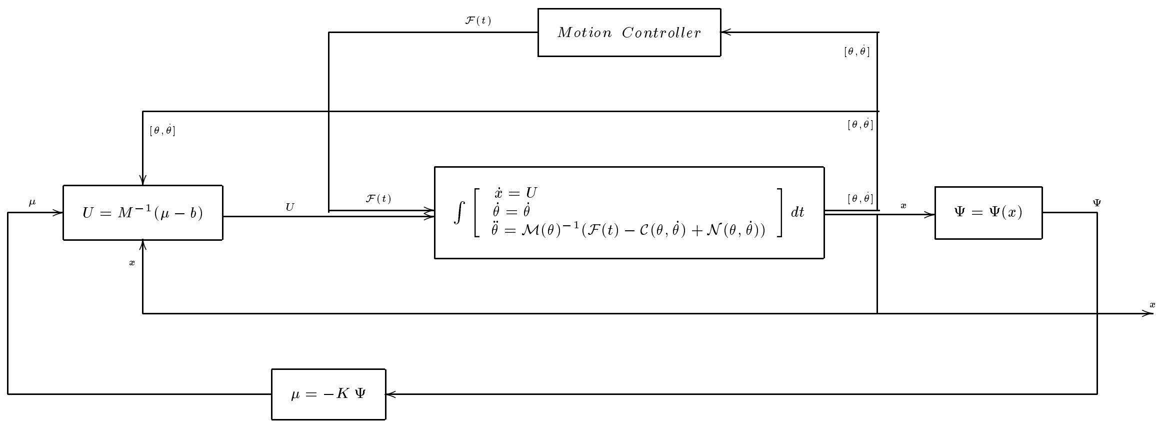

Figure 5: Combining extremal pair tracking with dynamics

As a sample simulation case, two ellipsoids drawn in an isometric view are shown in Figure 6. To indicate the extremal distance (found by prior application of the algorithm) a dashed line is shown. The extremal pair is located at the intersection of the dashed line with the ellipsoids.

In our implementation, we initialize the algorithm with two points different from the extremal pair. These points are marked by stars ($\star$) in Figure 6. The initial error vector (denoted by the solid line) $\mathbf{\Delta R}$ is also drawn.

Next, both of the ellipsoids are allowed to move as rigid bodies with specified time dependent motion (velocities and angular velocities) and the extremal tracking algorithm is started. Figure 7 demonstrates exponential convergence of the normalized projection errors, where a projection error is normalized by dividing by the norm of the current error vector $\mathbf{\Delta R}$ and the norm of the appropriate tangent vector. From Figure 7, we can conclude that even though the initialization is not exact, the algorithm exponentially converges to the extremal pair and maintains them as the surfaces move. The convergence rate is adjustable by tuning the linear feedback gain K.

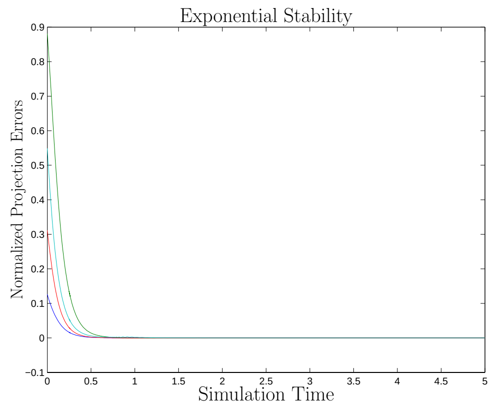

Figure 7: Normalized projection errors versus simulation time

6 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 very 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 0093290, and some very fruitful discussions with Elmer Gilbert.

References

[1] D. Baraff, "Curved surfaces and coherence for nonpenetrating rigid body simulation," in *SIGGRAPH Computer Graphics Proceedings*, vol. 24, pp. 19–28, 1990.

[2] R. Gillespie, J. Colgate, and M. Peshkin, "A general framework for cobot control," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 4, pp. 391–401, 2001.

[3] N. Sarkar, X. Yun, and V. Kumar, "Control of mechanical systems with rolling contraints: Application to dynamic control of mobile robots," *Int. Journal of Robotics Research*, vol. 13, no. 1, pp. 55–69, 1994.

[4] E. Gilbert, D. Johnson, and S. Keerthi, "A fast procedure for computing distance between convex objects in three-dimensional space," *IEEE Journal of Robotics and Automation*, pp. 193–203, 1988.

[5] C. Ong and E. 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. 531–39, 2001.

[6] M. C. Lin and J. F. Canny, "Efficient algorithms for incremental distance computation," in *IEEE Conf. on Robotics and Automation*, pp. 1008–14, 1991.

[7] M. C. Lin and J. F. Canny, "A fast algorithm for incremental distance calculation," in *IEEE Int. Conf. on Robotics and Automation*, pp. 1008–14, Sacramento, California, 1991.

[8] Y. Adachi, T. Kumano, and K. Ogino, "Intermediate represenation for stiff virtual objects," in *Proc. Virtual Reality Annual Int. Symp.*, pp. 203–10, 1995.

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

[10] E. Gilbert and C. 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.

[11] C. Turnbull and S. Cameron, "Computing distances between NURBS-defined convex objects," in *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 3686–90, 1998.

[12] M. Lin and D. Manocha, "Fast interference detection between geometric models," *The Visual Computer*, vol. 11, pp. 542–61, 1995.

[13] T. Duff, "Interval arithmetic and recursive subdivision for implicit functions and constructive solid geometry," *ACM Computer Graphics*, vol. 26, no. 2, pp. 131–39, 1992.

[14] B. V. Herzen, A. H. Barr, and H. R. Zatz, "Geometric collisions for time-dependent parametric surfaces," in *Proc. of ACM SIGGRAPH*, pp. 171–80, 1990.

[15] 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. 321–34, 1993.

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

[17] C. Bajaj, C. Hoffmann, R. Lynch, and J. Hopcroft, "Tracing surface intersections," *Computer Aided Geometric Design*, vol. 5, pp. 285–307, 1988.

[18] T. V. Thompson, D. E. Johnson, and E. Cohen, "Direct haptic rendering of sculptured models," in *Proc. Symposium on Interactive 3D Graphics*, pp. 167–76, 1997.

[19] T. V. Thompson, D. D. Nelson, E. Cohen, and J. Hollerbach, "Maneuverable NURBS models within a haptic virtual environment," in *6th Annual Symp. Haptic Interfaces for Virtual Environment and Teleoperator Systems*, vol. 61, pp. 37–44, 1997.

[20] D. E. Johnson and E. Cohen, "An improved method for haptic tracing of a sculptured surface," in *Symp. on Haptic Interfaces, ASME International Mechanical Engineering Congress and Exposition*, pp. 243–48, Anaheim, CA, 1998.

[21] 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*, pp. 101–8, 1999.

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