Robust and Efficient Computation of the Closest Point on a Spline Curve

Hongling Wang, Joseph Kearney, and Kendall Atkinson

Abstract. Parametric cubic spline curves are commonly used to model the geometry of road surfaces in real-time driving simulators. Roads are represented by space curves that define a curvilinear frame of reference in which three-dimensional points are expressed in coordinates of distance along the curve, offset from the central axis, and loft from the road surface. Simulators must map from global Cartesian coordinates to local road coordinates at very high frequencies. A key component in this mapping is the computation of the closest point on the central axis of the road to a three-dimensional point expressed in Cartesian coordinates. The paper investigates a two-step method that exploits the complementary strengths of two optimization techniques: Newton's method and quadratic minimization.

§1. Introduction

Parametric cubic spline curves provide a natural basis for modeling the geometry of road surfaces in real-time driving simulators. The road model is used by programs that control the behavior of autonomous vehicles and pedestrians populating the virtual urban environment. In many simulators, roads are represented by space curves that define a central axis or spine of a ribbon-like surface [6]. A surface normal is defined at each point on the curve allowing the ribbon to twist about its spine. The ribbon establishes a curvilinear coordinate system in which points in space are expressed in coordinates of distance along the central axis, offset from the axis, and loft from the road surface. The ribbon structure provides a natural coordinate frame for computing the local geometry of navigable surfaces. This geometry is important for wayfinding of autonomous agents and also determines the spatial relationships among agents.

Curve and Surface Fitting: Saint-Malo 2002
XXX (eds.), pp. 1–3.
Copyright © 2002 by Nashboro Press, Nashville, TN.
ISBN XXX.
All rights of reproduction in any form reserved.

While some simulation computations are most effectively implemented using ribbon coordinates, other computations are most effectively implemented using Cartesian coordinates. For example, behavior modules that track roads and avoid obstacles, are most easily expressed with object locations represented in ribbon coordinates. However, the dynamics code that computes object motions from control parameters set by object behaviors is most simply written in Cartesian coordinates. Because these computations are performed at very high frequency, it is essential to have efficient and robust code to map from ribbon coordinates to Cartesian coordinates and to compute the inverse mapping from Cartesian coordinates to local ribbon coordinates.

The mapping from Cartesian to ribbon coordinates is frequently a serious computational bottleneck in driving simulators. The key component in this mapping is the computation of the closest point on the central axis of the ribbon to a three-dimensional point expressed in Cartesian coordinates. Conventional optimization techniques such as Newton's method or quadratic minimization work well most of the time. However, we've found that the standard techniques consistently fail (converge very slowly or diverge) at a small number of points on many ordinary curves. Because of the frequency with which the mappings are performed (i.e. thousands of times a second for a modestly complex simulation) even these rare problematic instances are likely to occur with regularity. This leads to unacceptable computational delays and can halt a simulation if the optimization procedure is not terminated.

To address weaknesses with standard optimization techniques, we present a two stage technique that combines quadratic minimization and Newton's method.

§2. The Problem

A parametric cubic spline curve modeling the centerline of a curved road can be expressed as [5],

$$(x(s), y(s), z(s)), \quad 0 \leq s \leq L,$$

where s denotes arc length, L is the arc length of the entire spline curve, and $x(s)$, $y(s)$, and $z(s)$ are cubic spline functions with equally spaced breakpoints $\{s_0, s_1, ..., s_n\}$ with $s_0 = 0$ and $s_n = L$. The functions x, y, and z are C^2 on $[0, L]$.

At each time step of a simulation, the dynamics module computes a new position in Cartesian coordinates for every moving object. Given an object's location in Cartesian coordinates, our problem is to find the closest point on a road centerline to the object.

Let $p_0 = (x_0, y_0, z_0)$ be the position of an object (see Figure 1). The square of the distance between position p_0 and position $(x(s), y(s), z(s))$

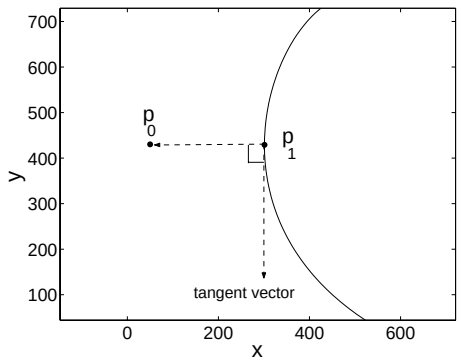

Fig. 1. Vector $\overline{p_1p_0}$ and the tangent vector of a cubic spline curve on p_1.

on a spline curve is

$$D(s) = (x(s) - x_0)^2 + (y(s) - y_0)^2 + (z(s) - z_0)^2, \tag{1}$$

where $x(s)$, $y(s)$, and $z(s)$ are cubic spline functions of the parameter s. The value s^* that minimizes $D(s)$ determines $p_1 = (x(s^*), y(s^*), z(s^*))$, the closest point to p_0 on the cubic spline curve. The vector $\overrightarrow{p_1p_0}$ is perpendicular to the tangent vector of the cubic spline curve on p_1. The distance between p_0 and p_1, which is the length of the vector $\overrightarrow{p_1p_0}$, is the smallest distance between the position p_0 and the cubic spline curve.

We approach the mapping computation as an optimization problem. To meet the stringent demands of real-time simulation, it is important that the selected optimization method converges to an accurate solution very quickly. While the average speed of this computation matters, it is of paramount importance that the maximum time does not overrun the time allotted for a simulation step by the scheduler. Thus, the demands of the application call for a method that is accurate, fast, and almost never fails. With these requirements in mind, we examine three optimization techniques: Newton's method, quadratic minimization, and a new technique that combines quadratic minimization and Newton's method.

§2. Quadratic Minimization Method

Quadratic minimization uses quadratic interpolation to minimize a one-variable function, in our case $D(s)$. Suppose that $\tilde{s}_1, \tilde{s}_2$, and $\tilde{s}_3$ are given as initial estimates of s^*, the value that optimizes $D(s)$. The quadratic

polynomial that interpolates $D(s)$ at $\tilde{s}_1, \tilde{s}_2$, and $\tilde{s}_3$ is given by,

$$\begin{aligned} P(s) = & \frac{(s-\tilde{s}_2)(s-\tilde{s}_3)}{(\tilde{s}_1-\tilde{s}_2)(\tilde{s}_1-\tilde{s}_3)} D(\tilde{s}_1) \\ & + \frac{(s-\tilde{s}_1)(s-\tilde{s}_3)}{(\tilde{s}_2-\tilde{s}_1)(\tilde{s}_2-\tilde{s}_3)} D(\tilde{s}_2) \\ & + \frac{(s-\tilde{s}_1)(s-\tilde{s}_2)}{(\tilde{s}_3-\tilde{s}_1)(\tilde{s}_3-\tilde{s}_2)} D(\tilde{s}_3). \end{aligned}$$

The minimum of $P(s)$ is used to approximate the minimum of $D(s)$. The minimum of $P(s)$ is given by

$$s^{*,k} = \frac{1}{2} \cdot \frac{y_{23} D(\tilde{s}_1) + y_{31} D(\tilde{s}_2) + y_{12} D(\tilde{s}_3)}{s_{23} D(\tilde{s}_1) + s_{31} D(\tilde{s}_2) + s_{12} D(\tilde{s}_3)}, \quad k = 1, 2, 3, \cdots, \qquad (2)$$

where $s_{ij} = \tilde{s}_i - \tilde{s}_j$ and $y_{ij} = \tilde{s}_i^2 - \tilde{s}_j^2$ for $i, j \in \{1, 2, 3\}$. We pick three values from $\tilde{s}_1, \tilde{s}_2, \tilde{s}_3$, and $s^{*,k}$ by eliminating the value which gives the largest $P(s)$ among the 4 values, and continue in a like manner until some error tolerance for $P(s)$ is achieved. It can be shown that with a sufficiently good set of initial guesses, the iteration will converge at a superlinear rate to s^* [3,4].

Quadratic minimization needs three initial estimates of s^*. In our application, we usually have a good guess of which segment, $[s_i, s_{i+1}]$, contains s^* based on the simulation state at the previous time step. An object typically enters a road at one end or the other (i.e. on the first or last segment.) As the object moves along a road, we track its position and velocity. Knowing s^* at the previous step and the object's velocity, we can predict the value of s^* at the current step.

Because the spline segments are all of equal length, we can calculate the index, i, of the segment containing the initial estimate,

$$i = \lfloor \frac{s^{*,0}}{l} \rfloor, \qquad (3)$$

where l is the arc length of each segment of the spline curve. We use as our three initial estimates of s^* the values s_i, $\frac{s_i + s_{i+1}}{2}$, and s_{i+1}.

When s^* lies near a segment boundary, error in the initial estimate may cause us to choose the wrong segment. This is detected when the iteration converges to a value outside the segment boundaries $[s_i, s_{i+1}]$. In this case, we attempt to solve s^* on adjacent segments.

Sometimes we are unable to predict s^* from previous states (for example, when an object moves from offroad terrain to a road.) When we are unable to compute a good initial estimate, we attempt to solve for s^* on each successive segment of the curve.

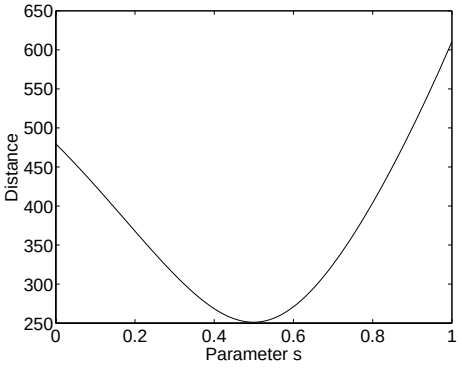

Fig. 2. Distance curve between p_0 and points on the spline segment in Figure 1.

The road curves we seek to model are typically smooth and have low curvature relative to their width. The width of a road surface must be less than the radius of curvature of the road axis spline to prevent self intersections. As a consequence, there is a single nearest point on the spline for all points on the surface of a road. Thus, the mapping from Cartesian coordinates to ribbon coordinates is unique.

We expect quadratic minimization method to work well for our problem because the minimum distance between a point p_0 on the surface of the road and the spine of the road is normally well-approximated by a parabola. For example, Figure 2 graphs the minimum distance from a point p_0 to a spline segment.

We tested the quadratic minimization method on a variety of cubic spline curves representative of road curves used in driving simulation. For each curve, we randomly generated a cloud of points near the curve and computed, for each of these points, the closest point on the curve. Experimental results showed that quadratic minimization converged to an accurate solution in fewer than 8 iterations for about one third of the test points. In the remaining cases, a solution was usually found although it sometimes took hundreds of iterations to converge. In a small percentage of cases the method diverged and no solution was found. Closer examination of cases in which the method diverged or converged very slowly revealed that the early iterations made progress toward a solution, but as the optimal value was approached it jumped about in a small interval surrounding the optimum.

§3. Newton's Method

The value s^* that minimizes $D(s)$ in formula (1) satisfies $D^{'}(s^*) = 0$. We can use Newton's method to find a root of this equation. This leads to

the iteration formula

$$s^{*,m+1} = s^{*,m} - \frac{D^{'}(s^{*,m})}{D^{''}(s^{*,m})}, \quad m = 0, 1, 2, \cdots \tag{4}$$

Similar to the quadratic method, the initial estimate $s^{*,0}$ is based on the value of s^* computed on the last time step of the simulation. Likewise, adjacent segments are considered when the method returns a value out of the initial segment's range. This method is quadratically convergent [4].

We implemented Newton's method to optimize the distance expression (1). We tested Newton's method with the same curves and sample points that we used to test quadratic minimization. Experiments showed Newton's method converges in most, but not all, cases. Generally, Newton's method found a solution more quickly than quadratic minimization – usually in 3 to 4 iterations. However, for some of the sample points Newton's method required dozens or even hundreds of iterations to converge. The problem cases seem to be caused by poor initial estimates. When, after a slow start, the method approached the optimal value it converged very quickly to the final solution. In a very small number of cases, Newton's method diverged jumping to values far away from the optimal value.

§4. Combining Newton's Method and Quadratic Minimization

Neither Newton's method nor quadratic minimization perform satisfactorily for real-time simulation. The average rate of convergence of quadratic minimization is too slow for our application. Both methods are plagued by the occurrence of cases in which convergence is unacceptably slow and both methods diverge in a small number of cases.

The good news is that Newton's method works well when given a sufficiently good initial estimate; sometimes an accurate solution is found in a single iteration. This is because Newton's method takes the first-order term and second-order term of Taylor's expansion while truncating the higher order terms. Therefore, as we approach the optimal value with formula (4), the error caused by truncating higher order terms is quite small. On the other hand, the error can be quite large when the initial estimate is far away from the optimal value.

Comparing the convergence properties of the two methods, we observe that their strengths complement one another. Quadratic minimization is good at refining coarse estimates. Newton's method is good at converging to the optimal value quickly with a good initial guess. This leads us to consider combining the two methods to leverage their complementary strengths in overcoming their weaknesses. The composite algorithm begins with quadratic minimization method to find a rough estimate that serves as an initial guess for Newton's method.

Based on our experiments, we find that quadratic minimization generally finds an acceptable initial value for Newton's method after four iterations. By using four iterations, we allow the possibility of updating all of the initial values. Each iteration in the quadratic minimization method produces a new estimate of the optimal value and throws away the worst of the current estimates. After 3 iterations we have produced 3 new guesses. If some or all of the 3 initial values are poor estimates of the optimal value, we have an opportunity to replace them all with new, better estimates and base the 4th estimate on these new values.

§5. Results

We demonstrate the performance of quadratic minimization, Newton's method, and our new composite method on the parametric spline curve shown in Figure 3. The curve is composed of 8 parametric cubic segments. We randomly generated 30,000 points in a band around the curve and computed, for each point, the nearest point on the spline curve. Methods were initialized with values on the segment, i, from which the s^* to be estimated was selected. Figure 4 presents the convergence rates for the three methods. The results are summarized in Table 1.

The termination criteria for all three algorithms was set to $|s^{*,k+1} - s^{*,k}| \leq (s_{i+1} - s_i) \cdot 10^{-8}$ where $[s_i, s_{i+1}]$ is the range of the parameter value for the spline segment where the final solution lies.

The most striking aspect of the test results is that the new method found a solution in less than 8 iterations in all 30,000 cases. In contrast, both quadratic minimization and Newton's method get mired in a significant number of cases. The new method outperforms quadratic minimization in every respect; it finds solutions faster on average and its worst case performance is capped at a reasonable value. For many points, the new method is an iteration or two slower than Newton's method. However, the tradeoff is that a solution is always found in modest number of iterations. Because of the need to bound the length of a simulation time step, it is highly desirable to minimize the maximum time of component computations. Thus, the elimination of failures and the reduction in the time to compute the hardest cases outweighs the small increase in time for the easy cases. Overall, the new method provides an attractive alternative for real-time applications.

§6. Conclusion

The closest point computation is a core component of real-time ground vehicle simulation. It forms an essential step in the process of mapping from Cartesian coordinates to road coordinates needed to place synthetic agents on the road network. To satisfy the requirements of real-time simulation, the closest point computation must be efficient and extremely robust.

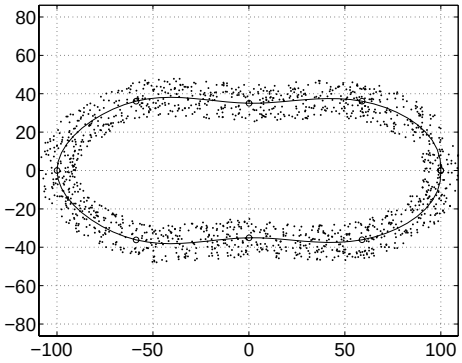

Fig. 3. An cubic spline curve example composed of 8 parametric cubic curve segments and some of the randomly chosen points.

algorithm	quadratic minimization	Newton's method	the new algorithm
rate of fast convergence (<=8 iterations)	34.17%	89.53%	100%
rate of slow convergence (>8 iterations)	65.79%	10.24%	0
divergence	0.04%	0.22%	0

Tab. 1. Performance of different methods for the cubic spline curve example in Figure 3.

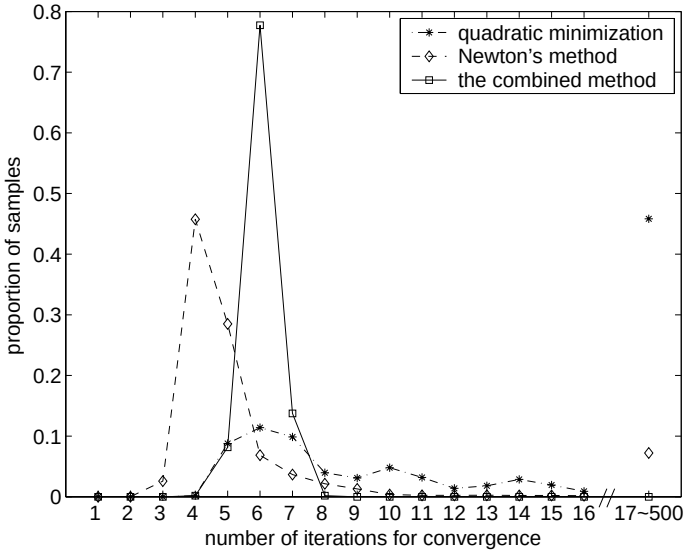

Fig. 4. A Histogram displaying, for each of the three methods, the distribution of convergence rates for 30,000 test points using the cubic spline curve shown in Figure 3.

The method presented in this paper is well tailored to the needs of real-time simulation. By combining quadratic minimization and Newton's method, we've found a technique that very reliably converges to an accurate solution in a small number of iterations. The method has undergone rigorous testing in our real-time ground vehicle simulator, Hank. In 10 months of daily runs (some for periods of many hours) we have had no failures. This practical experience over an extended period of time gives us great confidence in the robustness and usefulness of the approach.

Acknowledgments. This work was supported in part through National Science Foundation grants INT-9724746, EAI-0130864, and IIS-0002535. Jim Cremer and Pete Willemsen made significant contributions to the development of the Hank simulator.

References

1. Atkinson, K., Modelling a road using spline interpolation, Reports on Computational Mathematics # 145, Department of Mathematics, The University of Iowa, (2002).
2. Atkinson, K., *An Introduction to Numerical Analysis,* John Wiley & Sons, Hoboken, NJ, 1989.
3. Luenberger, D., *Linear and Nonlinear Programming,* Addison-Wesley, Reading, MA, 1984.
4. Wang, H., An analytical solution for free-form roads in driving simulation, Technical Report 01-04, Department of Computer Science, The University of Iowa, (2001).
5. Wang, H., Kearney, J., and Atkinson, K., Arc-length parameterized spline curve for real-time simulation, 5th international conference on Curves and Surfaces, (2002).
6. Willemsen, P., Kearney, J., and Wang, H., Ribbon networks for modeling navigable paths of autonomous agents in virtual urban environments, to appear in IEEE Virtual Reality Conference, 2003.
7. Willemsen, P., Behavior and Scenario Modeling For Real-Time Virtual Environments, dissertation, The University of Iowa, 2000.

Hongling Wang, Joseph Kearney, and Kendall Atkinson
Department of Computer Science
The University of Iowa
Iowa City, IA 52242
howang|kearney|atkinson@cs.uiowa.edu