Efficient Distance Computation between Natural Quadrics *

[1]Mustafa S. Fawad, [2]Mustafa Atika

[1](Proposed Presenter) Key Laboratory of Computer Science, Institute of Software,
Graduate University of Chinese Academy of Sciences, Beijing – 100039, China
(phone: +86-10-88255778; email: fawad_zaidi@yahoo.com)

[2] National University of Computer and Emerging Sciences, Karachi, Pakistan
(phone:0092-111128128; email: atika.mustafa@nu.edu.pk)

Abstract

The idea of our work is on one of the basic problems, namely the fast calculation of the distance between two objects. We have implemented an enhanced version of the shortest distance routine on the basis of mathematical equations, which allows the tracking of the distance between a pair of conics and quadrics in time that to be bounded by a constant is expected. Experimental results confirm this result, at least for moderate shortest distance calculation, and suggest that the enhanced algorithm might also come in handy. Computing the distance between two objects is a common problem, using the mathematical model of two objects, we find the point on each object such that the distance between the objects is minimized. To do this we took care of efficient computing, which is possible with simplifying the polynomials and their relative coefficients. Effectiveness of computing algorithms and executing time are also analyzed. We tested the calculation on various natural quadrics. The graphics part is implemented in ***openGL*** *and* ***QT*** *library is used for user interface. The execution time is calculated by running the distance calculation routines 10,000,000 times.*

Keywords: *Polynomial degree, equations, Lagrange formulation, real roots*

I. INTRODUCTION

During last decade an increasing interest in virtual reality (VR) technique could be observed. Virtual reality plays a highly important part in the actual reality as can be send from movies and other scientific application (i.e. Education), and VR based industries.

However, the 'classical' challenges that come when deal with the complex objects in VR world, simulating their physical behavior.

Robustness and efficiency of such a simulation heavily depend on the quality of the collision Detection precisely distance computation algorithm. They identify the earliest time of the closest distance of two objects in a given time interval. However, one has to consider that there is a trade-off between the accuracy of the approximation and the efficiency of the distance computation. In this paper we present an efficient algorithm for computing the distance between so-called 'quadratic complexes'. We reduce the distance calculation problem to the problem of solving univariate polynomial equations. We implemented on OpenGL and QT to check the practical approach to this idea and succeeded in having productive results. Reducing the high degree polynomials was indeed a challenging task, keeping in mind the approximation not affecting the quality of results. Our considerations therefore raise the question, whether or not it is worth to handle curved objects directly. It means that one has to spend more effort on the basic routines computing the distance between two patches, but can profit from higher accuracy and simplified quadratic equations to compute the closest.

* *SILVIA at International Max Planck Institute for Informatics, Saarland, Germany supported this Project, under the MSc fellowship awarded to each author of this paper*

Computing the distance between two objects is a common problem, using the mathematical model of two objects, we find the point on each object such that the distance between the objects is minimized. To do this we took care of efficient computing, which is possible with simplifying the polynomials and their relative coefficients. Details of computing algorithms and executing time are discussed in the following section.

II. OUR ALGORITHM

A. Simple Cases

We started with the simpler cases and proceeded to gradually more complex situations.

Simpler cases are dealt with ease since there polynomial equations are simpler too. We adopted the methods for simpler cases by [1]. The cases are as follows:

1. Point to Line
2. Point to Circle
3. Point to Ellipsoid
4. Point to Ellipse
5. Line to Circle
6. Line to Line

A slightly different case is of Line to Sphere, which is not as hard as is thought. It is just the extended case of line to point.

B. Complex Cases

In this section we will discuss the cases of distance of different objects with a cone. Here we have assumed that the cone is centered at origin along Z -axis. Given the general equation of this cone below:

$$\frac{x^2}{a^2}+\frac{y^2}{b^2}+\frac{z^2}{c^2}=0$$

We know from the general equation of conics:

$$X^T AX = 0$$

where,

$$A=\begin{bmatrix} A_1 & 0 & 0 \\ 0 & A_2 & 0 \\ 0 & 0 & A_3 \end{bmatrix}=\begin{bmatrix} \frac{1}{a^2} & 0 & 0 \\ 0 & \frac{1}{b^2} & 0 \\ 0 & 0 & \frac{1}{c^2} \end{bmatrix}$$

by using Lagrange formulation:

$$L=(x-p)^2+\alpha(X^T AX)$$

By minimizing the function and simplifying we get:

$$f(\alpha)=A_1 p_1^{\,2} d_2^{\,2} d_3^{\,2}+A_2 p_2^{\,2} d_1^{\,2} d_3^{\,2}+A_3 p_3^{\,2} d_1^{\,2} d_2^{\,2}=0 \text{ -}Eq1$$

where $d_i = 1+\alpha A_i$ and *i=1,2 and 3*

C. Point to cone

Eq (1) is the degree 4 polynomial in α for the case of distance between cone and point.

The nearest point on cone can be calculated by:

$$x = (\alpha A + I)p \quad \text{- - - - - - - - Eq (2)}$$

where I is the identity matrix and p is the point.

Now considering the special case of cone, i.e circular cone:
From Eq (2) we have:

$$f(\alpha) = b^2(Bp_1^{\,2}d_3^{\,2} + Bp_2^{\,2}d_3^{\,2} + A_3p_3^{\,2}b^2) = 0$$

where $A_1 = A_2 = B$ and $b = d_i$ for i=1 and 2, which is the degree 2 polynomial in α.

D. Line to Cone

Line is represented as:

$$p(t) = c + tu$$

where c is some point on the line and u is the direction vector.
From Lagrange formulation, by minimizing and simplifying the function we get:

$$g(\alpha,t) = p_1p_1'd_2d_3A_1 + p_2p_2'd_1d_3A_2 + p_3p_3'd_1d_2A_3 = 0$$
$$f(\alpha,t) = A_1p_1^{\,2}d_2^{\,2}d_3^{\,2} + A_2p_2^{\,2}d_1^{\,2}d_3^{\,2} + A_3p_3^{\,2}d_1^{\,2}d_2^{\,2} = 0$$

Considering special case of cone i.e. circular cone, we get:

$$g(\alpha,t) = p_1p_1'bd_3B + p_2p_2'bd_3B + p_3p_3'b^2A_3 = 0$$
$$f(\alpha,t) = b^2(Bp_1^{\,2}d_3^{\,2} + Bp_2^{\,2}d_3^{\,2} + A_3p_3^{\,2}b^2) = 0$$

With this we can easily express that:
deg(f, t) = 2
deg(f, α) = 4
deg(g, t) = 1
deg(g, α) = 2

E. Circle to Cone

Circle is represented in parametric form:

$$P(t) = c + r(\cos(\phi)(u) + \sin(\phi)(v)) \quad \text{- - -- - - Eq 3}$$

where:

$$\cos(\phi) = \frac{1-t^2}{1+t^2} \text{ and } \sin(\phi) = \frac{2t}{1-t^2} \text{ and } t = [0,1]$$

$$g(\alpha,t) = b(p_1p_1'd_3B + p_2p_2'd_3B + p_3p_3'bA_3) = 0$$
$$f(\alpha,t) = b^2(Bp_1^{\,2}d_3^{\,2} + Bp_2^{\,2}d_3^{\,2} + A_3p_3^{\,2}b^2) = 0$$

$$g'(\alpha,t) = (p_1 p_1' d_3 B + p_2 p_2' d_3 B + p_3 p_3' b A_3) = 0$$
$$f'(\alpha,t) = (B p_1^{\,2} d_3^{\,2} + B p_2^{\,2} d_3^{\,2} + A_3 p_3^{\,2} b^2) = 0$$

where deg $(g', \alpha) = 1$

Solving g for α

$$\alpha = \frac{-[B(p_1 p_1' + p_2 p_3') + A_3 p_3 p_3']}{A_3 B p^T p'} \quad \text{- -- - - - Eq 4}$$

Also:

$d_3 = 1 + \alpha A_3$

$$d_3 = \frac{(B - A_3) p_3 p p_3'}{B p^T p'}$$

$$b = 1 + \alpha B$$
$$b = \frac{(A_3 - B)(p_1 p_1' + p_2 p_2'')}{A_3 p^T p''}$$

Substituting b and $d3$ in f' we get:

$$p_3^{\,2}[A_3 p_3'^{\,2}(p_1^{\,2} + p_2^{\,2}) + B(p_1 p_1'^{\,2} + p_2 p_2'^{\,2})] = 0$$

We can easily express it as:

$$p_3^{\,2} = 0$$
$$h(t) = A_3 p_3'^{\,2}(p_1^{\,2} + p_2^{\,2}) + B(p_1 p_1'^{\,2} + p_2 p_2'^{\,2}) = 0$$

where $h(t)$ is a polynomial of degree 4 in t.

Substituting t in *Eq. 2* gives point on a circle. Substituting t in *Eq. 3* gives the value of α, which can be used in *Eq. 1* to calculate the point on cone.

F. Solving the polynomials

Routines from Silvia [2] (Simulation Library for Virtual Reality and Interactive Applications) libraries are used to solve the polynomials.

The degree four or less polynomials were solved using the routines:
SolveQuadric (coeffs, roots)
SolveQuartic (coeffs, roots)

The routine returns the number of real roots of the polynomial whose coefficients are given in "coeffs" and the real roots are in "roots". For polynomials of degree higher than 4, the following routine is used:

real_roots1(coeffs, degree, roots)

where:
coeffs = coefficients of the polynomials
degree = degree of polynomial
roots = roots calculated by the routine.

The routine returns the number of real roots of the polynomial.

G. Execution Times

The execution time is calculated by running the distance calculation routines 10,000,000 times to have substantial reading and avoid micro measurements. The time depends on the following factors:

- Degree of polynomial
- Complexity of coefficient calculation

S#	Objects	Time (seconds)
1	Point – Line	25
2	Point – Circle	28
3	Point – Ellipsoid	800
4	Point - Ellipse	62
5	Point – Sphere	14
6	Point – Circular Cone	117
7	Point – Cylinder	205
8	Point – Cone	126
9	Line – Circle	415
10	Line - Sphere	37
11	Line – Circular Cone	119
12	Line – Cone	221
13	Circle - Circle	1764
14	Circle - Sphere	42
15	Circle – Circular Cone	1330
16	Circular Cone- Circular Cone	472

III. CONCLUSION

The degrees of the resulting polynomials are usually high and hence elimination methods are often not appropriate in the context of floating-point computations. However, we could show that for all cases in which the degrees of the resulting polynomials become critical, a factorization into two lower degree polynomials can be found.

ACKNOWLEDGMENT

Mustafa S. Fawad thanks Elmar Schömer, Johannes Gutenberg Universität Mainz, Germany, for his valuable guidance and support throughout our work.

REFERENCES:

1. D. H. Eberly. 3D game engine design: a practical approach to real-time computer graphics. Morgan Kaufmann, 2001.
2. Christian Lennerz, Elmar Schömer. Efficient Distance Computation for Quadratic Curves and Surfaces

Article received: 2006-02-28