AN EFFICIENT PARALLEL SOLUTION FOR EUCLIDEAN SHORTEST PATH IN THREE DIMENSIONS

(Preliminary Version)

Chanderjit Bajaj

Department of Computer Science,
Purdue University, West Lafayette, IN 47907

Abstract: We describe an efficient parallel solution for the problem of finding the shortest Euclidean path between two points in three dimensional space in the presence of polyhedral obstacles. We consider the important case where the order in which the obstacles are encountered in this shortest path is known. In particular for this case we describe an efficient parallel numerical iterative method on a concurrent-read exclusive-write synchronous shared-memory model. The iterations are essentially convergent non-linear block Gauss-Seidel. For special relative orientations of the, say n, polyhedral obstacles, we further describe a direct method that gives the exact solution in $O(\log n)$ time using n processors.

1. Introduction

The problem of finding the shortest Euclidean path between two points in 3-dimensions, bounded by a finite collection of polyhedral obstacles is a special case of the more general problem of planning optimal collision-free paths for a given robot system. In Euclidean 2-space (the Euclidean plane) the problem is easy to solve and the shortest path is polynomial time computable, Lozano-Perez, Wesley [4]. This problem for Euclidean 3-space is difficult and known shortest path sequential computations are of exponential time, Sharir, Schorr [7].

In this paper we are concerned about the efficient use of parallelism for solving this fundamental motion planning problem. We consider the important subproblem where the order in which the obstacles are encountered in this shortest path is known. Even this is seemingly intractable. The shortest path solution for this problem in general has been shown to be *not* solvable by radicals, and furthermore an exact solution is shown to be not possible even under models of computation where you additionally allow use of the non-solvable roots of the general fifth degree polynomial equation, Bajaj [1]. This rules out exact algorithms under various powerful models of computation and leaves primarily the possibility of numerical or symbolic approximation methods to the computation of the shortest path, Bajaj [2]. A $(1+\epsilon)$ polynomial time approximation scheme was recently given in Papadimitriou [6] This paper describes an efficient parallel numerical iterative method for this problem. As our model we use the concurrent-read exclusive-write synchronous shared-memory parallel machine model where on each step each processor can do a single rational addition, subtraction, multiplication, division or compute a square-root.

The shortest path between two given points, in the presence of polyhedral obstacles, is known to be *polygonal* lines (piecewise straight lines, as for the planar 2-dimensional problem), with break points that lie on the edges of the given polyhedral obstacles. Since the edges of the polyhedral obstacles are arbitrary lines in Euclidean 3-space, the problem of determining the points of contact of the shortest path with these edges can without loss of generality be versed as follows.

> *Given a sequence $L=(L_1, L_2, \cdots, L_n)$ of lines in 3-dimensional space , find the shortest path from a source point X to a destination point Y constrained to pass through interior points of each of the lines $L_1, L_2, \cdots, L_n$ in this order.*

We identify three different cases of the relative positions of the lines. All the various configurations of the n lines in 3-space consist of combinations of these basic orientations between adjacent pairs of lines, L_i, L_{i+1}, $i=1..n-1$.

(a) Pairwise lines are parallel to each other.

(b) Pairwise lines are not parallel but intersect.

(c) Pairwise lines are skew and do not intersect.

In § 2, we show that when the lines are oriented as a combination of the cases (a) and (b), then the shortest path solution can be computed exactly in time $O(\log n)$ using n processors. This corresponds to the solution being

CH2282-2/86/0000/1897$01.00 © 1986 IEEE

algebraically simple, [1]. However in § 3, for the case (c) of skew lines we describe an efficient parallel numerical iterative method. The iterations are essentially convergent non-linear block Gauss-Seidel, [5].

2. Exact Direct Method

Between pairs of lines in 3-space which are parallel to each other there exists a unique plane which contains both of them. The same applies to pairs of lines in 3-space which intersect. Also a point and a line in 3-space define a unique plane between them. The problem of finding the shortest path between X and Y in 3-space for cases (a) and (b), then reduces to a constrained 2-1/2 dimensional space problem as follows. Let the point X and line l_1 define the plane P_1, the lines L_i and L_{i+1} define the planes P_{i+1}, i=1..n−1, and the line l_n and the point Y define the plane P_{n+1}. The original problem is now reduced to finding the shortest path between two points X and Y in 3-space with the path constrained to the planes P_i, i=1..n+1. Then the points of contact of the shortest path with the lines L_i, the edges of the planes, are determined by first *unfolding* all the planes P_i so that they all lie on the common plane defined by say, plane P_1 containing point X. In practice we compute for each plane P_i the displacement d_i and orientation θ_i defining its position in the common plane P_1, relative to some standard fixed plane representation of P_i. The shortest path joining X and Y now becomes the shortest plane path that is the straight line, connecting X and Y^t, (the transformed point Y now on the common plane P_1 and thus coplanar with X). The points of intersection of this straight line with the unfolded lines L_i^t, when transformed back, give the points of contact with the lines L_i of our original problem. To prove correctness we note that the length of the the shortest path is kept invariant under such simple planar unfoldings and thus these unfoldings give the unique shortest path.

The unfoldings can be done in $O(log\ n)$ parallel time using n processors as follows. Initially there are a sequence of n planes P_i, i=2..n, which need to be unfolded onto the chosen common plane P_1. There is a processor mp_i assigned to each plane. In the first pass every adjacent pair of planes P_i, P_{i+1} is unfolded to become coplanar with P_i. In practice processor mp_{i+1} computes the displacement d_i and orientation θ_i defining the new position of P_{i+1}, essentially solving a quadratic equation independent of n. At the end of the pass there are now a sequence of $\lceil n/2 \rceil$ planes which need to be unfolded. The process is repeated at most $\lceil \log n \rceil$ times when all the planes become unfolded on to the common plane P_1. As each pass requires $O(1)$ time per processor the entire unfolding is complete in $O(log\ n)$ time using n processors. Each processor mp_i now computes under the shared-memory model in $O(1)$ time the point of intersection of the straight line shortest path with the unfolded line L_i^t, and then transforms it back to give the points of contact with the original lines L_i all in a grand total of $O(log\ n)$ time.

3. Iterative Numerical Method

Whenever any two adjacent lines L_i and L_{i+1} are skew to one another, there exists no common plane containing both of them. Looking at it differently, the locus (or envelope) of all possible straight line segments connecting skew lines L_i and L_{i+1} no longer defines a planar surface but a 3-dimensional volume. Hence the above exact unfolding method does not work. Nevertheless when the lines are skew there exist generalized unfoldings which however provide iterative approximations of the shortest path solution, Bajaj, Moh [3]. Here we describe a parallel numerical iterative method where each iteration is essentially a convergent non-linear block Gauss-Seidel iteration and takes only $O(1)$ time and $\lceil n/2 \rceil$ processors.

Let the interior points of each of the given (skew) lines L_i of the shortest path solution which needs to be determined, be respectively X_i, i=1..n. Each of the points X_i is a vector in 3-dimensional space. Further let l_i be the corresponding unit vectors along the lines L_i and the vectors X_0,X_{n+1} equal the initial given 3-dimensional points X and Y respectively. . We wish to minimize the function $D(X_1,...,X_n):R^n \longrightarrow R^1$ defined below, subject to the X_i constrained to their respective lines L_i.

$$D(X_1,...,X_n) = \sum_{i=0..n} d(X_i,X_{i+1}) \quad (1)$$

Here the $d(X_i,X_{i+1}) = ||X_i - X_{i+1}||$, the Euclidean distance between the two points. The above function is strictly convex, (see also [7]) and hence the unique solution is given by the following necessary and sufficient conditions. For i=1..n

$$\delta D/X_i = 0 = (X_{i-1}-X_i).l_i/||X_{i-1}-X_i|| - (X_i-X_{i+1}).l_i/||X_i-X_{i+1}|| \quad (2)$$

The above optimality conditions can be interpreted geometrically. For each i, the vectors $(X_{i-1}-X_i)$ and (X_i-X_{i+1}) subtend equal angles at the lines L_i for the shortest path solution. As X_i is constrained to the line L_i with unit vector l_i we let $X_i=\alpha_i l_i+\beta_i$, where α_i is a scalar and β_i is some vector point on the line L_i. On substituting this in (2) and simplifying (see Appendix A), we obtain for each i=1..n,

$$a_i\alpha_i^2 + b_i\alpha_i + c_i = 0 \quad (3)$$

where the a_i, b_i and c_i are at most fourth degree polynomial functions of X_{i-1} and X_{i+1} as shown below.

$$a_i = ((X_{i-1}-\beta_i).l_i)^2 + (\beta_i-X_{i+1}).(\beta_i-X_{i+1})$$
$$- ((\beta_i-X_{i+1}).l_i)^2 - (\beta_i-X_{i-1}).(\beta_i-X_{i-1})$$

$$b_i = 2((X_{i-1}-\beta_i).l_i)[((X_{i-1}-\beta_i).l_i)((\beta_i-X_{i+1}).l_i)$$
$$- (\beta_i-X_{i+1}).(\beta_i-X_{i+1})]$$
$$- 2((X_{i+1}-\beta_i).l_i)[(X_{i+1}-\beta_i).l_i)((\beta_i-X_{i-1}).l_i)$$
$$- (\beta_i-X_{i-1}).(\beta_i-X_{i-1})]$$

$$c_i = ((X_{i-1}-\beta_i).l_i)^2(\beta_i-X_{i+1}).(\beta_i-X_{i+1})$$
$$- ((X_{i+1}-\beta_i).l_i)^2(\beta_i-X_{i-1}).(\beta_i-X_{i-1})$$

We now obtain our parallel iterative computation of the X_i, i=1..n, as follows. (In the following let m take on all the odd values from 1..n and n take on all the even values from 1..n.) Given initial values of each of the X_i^0, in the general k th iteration,

(1) Compute for all the m in parallel, the values α_m^{k+1} and set $X_m^{k+1} = \alpha_m^{k+1} l_m + \beta_m$.

(2) Then compute for all the n in parallel, the values α_n^{k+1} and set $X_n^{k+1} = \alpha_n^{k+1} l_n + \beta_n$.

Each of the α_i is governed by a quadratic equation (3) above and can be computed exactly by the quadratic formula. The two solutions of α_i obtained is a direct consequence of the squaring of equation (a) in Appendix A during simplification of (2) to remove the square roots of the Euclidean distance norm. One of these solutions corresponds to the right hand side of equation (a) being taken positive and the other to the right hand side being taken negative. We of course at each stage, conforming to our original minimality condition (2), select the α_i solution corresponding to the positiye right hand side of the equation (a). The correctness of the above parallel algorithm follows directly from the fact that each of the α_i is a function of only X_{i-1} and X_{i+1}.

From Ortega, Rheinboldt [5, p219], we obtain the following facts. If F: $R^n \longrightarrow R^n$ has components $f_1, \ldots, f_n$ then the basic step of the non-linear Gauss-Seidel iteration is to solve the i th equation

$$f_i(X_1^{k+1},\ldots,X_{i-1}^{k+1},X_i,X_{i+1}^k,\ldots,X_n^k) = 0$$

for X_i and then to set $X_i^{k+1} = X_i$. Thus in order to obtain all the X^{k+1} from the X^k we solve successively the n one-dimensional non-linear equations above for i=1..n. Since in our case each of the f_i is a function of only X_{i-1}, X_i and X_{i+1} and since all the f_i are identical, we can solve $\lfloor n/2 \rfloor$ of them in parallel at a time. Also as is the Gauss-Seidel principle of using new information as soon as it is available, in our case step (2) of the algorithm uses the most recent iterate values of the X_i. To draw the analogy tighter between the above parallel algorithm and the Gauss-Seidel method, consider further the non-linear block Gauss-Seidel procedure, Ortega, Rheinboldt [5, p225]. In this a complete iteration requires the solution of s non-linear system of dimension q_i, i=1..s. The components f_i of F are grouped into mappings F_i: $R^n \longrightarrow R^{q_i}$, and the X partitioned into $(X^1, \ldots, X^s)$ with $X^i \in R^{q_i}$. Then

$$F_i((X^1)^{k+1},\ldots,(X^i)^{k+1},(X^{i+1})^k,\ldots,(X^s)^k) = 0$$

The above algorithm is thus non-linear block Gauss-Seidel for $q_i = \lfloor n/2 \rfloor$ and $m = 2$, the two blocks being the systems for the odd and even values of i=1..n of (3) above.

$$a_1\alpha_1^2 + b_1\alpha_1 + c_1 = 0$$
$$a_3\alpha_3^2 + b_3\alpha_3 + c_3 = 0$$
$$a_5\alpha_5^2 + b_5\alpha_5 + c_5 = 0$$

$$a_2\alpha_2^2 + b_2\alpha_2 + c_2 = 0$$
$$a_4\alpha_4^2 + b_4\alpha_4 + c_4 = 0$$
$$a_6\alpha_6^2 + b_6\alpha_6 + c_6 = 0$$

The above iterative parallel procedure was implemented and efficiently converged to the optimal solution for a variety of input skew lines and various values of precision and error bounds. The results are summarized here for both the l_1 and the l_∞ error norms. The initial values for these iterations were computed in a single parallel step using equation (3) and in each case letting $X_{i-1} = X_0$ and $X_{i+1} = X_{n+1}$, the two given input points.

Acknowledgements: I wish to thank Walter Gautschi for enlightening discussions in numerical analysis.

4. References

[1] Bajaj, C., *The Algebraic Complexity of Shortest Paths in Polyhedral Spaces,* Proc. 23rd Allerton Conference on Comm., Control and Computing, p510-517, 1985.

[2] Bajaj, C., *Limitations to Algorithm Solvability: Galois Methods and Models of Computation,* Purdue University Computer Science Tech. Report, TR-567, 1986.

[3] Bajaj, C., and Moh, T., *Generalized Unfoldings for Shortest Paths in Euclidean 3-Space,* Purdue University Computer Science Tech. Rept., TR-526, 1985.

[4] Lozano-Perez, T., and Wesley, M.A., *An algorithm for planning collision free paths among polyhedral obstacles,* CACM 22, p560-570, 1979.

[5] Ortega, J., and Rheinboldt, W., *Iterative Solution of Nonlinear Equations in Several Variables,* Academic Press, New York, 1970.

[6] Papadimitriou, C., *An Algorithm for Shortest Path Motion in Three Dimensions,* Information Processing Letters, 20, p259-263, 1985.

[7] Sharir, M., and Schorr, A., *On shortest paths in polyhedral spaces,* Proceedings 16th STOC, p144-153, 1984.

Table 1: Summary of results under the l_1 norm

Error	*N=2*	*N=3*	*N=4*	*N=5*	*N=6*	*N=7*	*N=8*	*N=9*	*N=10*
1.0e-01	1	4	3	3	3	4	4	4	4
1.0e-02	4	5	5	5	5	5	5	5	5
1.0e-03	5	7	6	7	7	7	7	7	7
1.0e-04	5	8	8	9	9	9	9	9	9
1.0e-05	6	10	9	11	10	10	10	10	10
1.0e-06	7	11	11	13	12	12	12	12	12
1.0e-07	8	13	12	15	14	14	14	14	14
1.0e-08	9	14	14	17	16	16	16	16	16
1.0e-09	10	16	15	20	17	17	17	17	17
1.0e-10	11	18	17	22	19	19	19	19	19
1.0e-11	12	19	18	24	21	21	21	21	21
1.0e-12	13	21	20	26	22	23	23	23	23
1.0e-13	14	22	21	28	24	24	24	24	24
1.0e-14	15	24	23	30	26	26	26	26	26
1.0e-15	16	25	24	32	28	28	28	28	28

Table 2: Summary of results under the l_∞ norm

Error	*N=2*	*N=3*	*N=4*	*N=5*	*N=6*	*N=7*	*N=8*	*N=9*	*N=10*
1.0e-01	1	4	3	3	3	3	3	3	3
1.0e-02	4	5	4	4	5	5	5	5	5
1.0e-03	5	6	6	7	6	7	6	6	6
1.0e-04	5	8	7	9	7	8	8	8	8
1.0e-05	6	10	9	11	10	10	10	10	10
1.0e-06	7	11	10	13	12	12	12	12	12
1.0e-07	8	13	12	15	13	13	13	13	13
1.0e-08	9	14	13	17	15	15	15	15	15
1.0e-09	10	16	15	19	17	17	17	17	17
1.0e-10	11	17	16	21	18	19	19	19	19
1.0e-11	12	19	18	23	20	20	20	20	20
1.0e-12	13	21	19	25	22	22	22	22	22
1.0e-13	14	22	21	28	24	24	24	24	24
1.0e-14	15	24	22	30	25	26	25	26	26
1.0e-15	16	25	24	32	27	27	26	27	27

Appendix A

We now rationalize and simplify the optimality equation (2) of the text.

$$(X_{i-1}-X_i).l_i/||X_{i-1}-X_i|| \;=\; (X_i-X_{i+1}).l_i/||X_i-X_{i+1}|| \quad (a)$$

As X_i is constrained to the line L_i with unit vector l_i we let $X_i=\alpha_i l_i+\beta_i$, where α_i is a scalar and β_i is some vector point on the line L_i. On substituting this in (a) and simplifying we obtain

$$(X_{i-1}-\alpha_i l_i-\beta_i).l_i/||X_{i-1}-\alpha_i l_i-\beta_i||$$
$$= (\alpha_i l_i+\beta_i-X_{i+1}).l_i/||\alpha_i l_i+\beta_i-X_{i+1}||$$

$$((X_{i-1}-\beta_i).l_i-\alpha_i)/\sqrt{(X_{i-1}-\beta_i).(X_{i-1}-\beta_i)-2\alpha_i(X_{i-1}-\beta_i).l_i+\alpha_i^2)}$$
$$=((\beta_i-X_{i+1}).l_i+\alpha_i)/\sqrt{(X_{i+1}-\beta_i).(X_{i+1}-\beta_i)-2\alpha_i(X_{i+1}-\beta_i).l_i+\alpha_i^2)}$$

Squaring both sides of the above equation and rationalizing

$$((X_{i-1}-\beta_i).l_i-\alpha_i)^2[(X_{i+1}-\beta_i).(X_{i+1}-\beta_i)-2\alpha_i(X_{i+1}-\beta_i).l_i+\alpha_i^2)]$$
$$=((\beta_i-X_{i+1}).l_i+\alpha_i)^2[(X_{i-1}-\beta_i).(X_{i-1}-\beta_i)-2\alpha_i(X_{i-1}-\beta_i).l_i+\alpha_i^2)]$$

Simplifying the above equation we obtain a polynomial equation in terms of the scalar α_i,

$$a_4\alpha_i^4+a_3\alpha_i^3+a_2\alpha_i^2+a_1\alpha_i+a_0 \;=\; 0$$

where the coefficients are given by

$$a_4=0 \qquad\qquad a_3=0$$

$$a_2=((X_{i-1}-\beta_i).l_i)^2+(\beta_i-X_{i+1}).(\beta_i-X_{i+1})$$
$$-((\beta_i-X_{i+1}).l_i)^2-(\beta_i-X_{i-1}).(\beta_i-X_{i-1})$$

$$a_1=2((X_{i-1}-\beta_i).l_i)[((X_{i-1}-\beta_i).l_i)((\beta_i-X_{i+1}).l_i)$$
$$-(\beta_i-X_{i+1}).(\beta_i-X_{i+1})]$$
$$-2((X_{i+1}-\beta_i).l_i)[(X_{i+1}-\beta_i).l_i)((\beta_i-X_{i-1}).l_i)$$
$$-(\beta_i-X_{i-1}).(\beta_i-X_{i-1})]$$

$$a_0=((X_{i-1}-\beta_i).l_i)^2(\beta_i-X_{i+1}).(\beta_i-X_{i+1})$$
$$-((X_{i+1}-\beta_i).l_i)^2(\beta_i-X_{i-1}).(\beta_i-X_{i-1})$$