01 Jan 1990

Computational Intelligence In CAD/CAM Applications

Chaman Sabharwal
Missouri University of Science and Technology, chaman@mst.edu

Thomas G. Melson

Martin D. Fraser

Recommended Citation

C. Sabharwal et al., "Computational Intelligence In CAD/CAM Applications," *ACM Eighteenth Annual Computer Science Conference (CSC90)*, pp. 114 - 120, Association for Computing Machinery (ACM), Jan 1990.
The definitive version is available at https://doi.org/10.1145/100348.100366

COMPUTATIONAL INTELLIGENCE IN CAD/CAM APPLICATIONS

Chaman L. Sabharwal[1], Thomas G. Melson[2], Martin D. Fraser[3]

[1]University of Missouri-Rolla, 8001 Natural Bridge Road, St. Louis, MO 63121
[2]Computer Aided Technology, McDonnell Douglas Corporation, St. Louis, MO 63166
[3]Georgia State University, University Plaza, Atlanta, GA 30303

ABSTRACT

This paper presents a fundamental, direct, and powerful approach to the surface/surface intersection problem in CAD/CAM applications. The algorithm is designed and implemented in three steps: a) Preprocessing- locate the potentially intersecting sections of the surfaces and decompose the surfaces into surface elements within specified flatness tolerance; b) Intersection- decompose the possibly intersecting pairs of surface elements into continuous surface triangulations to find the approximate intersections between the pairs of surface elements; c) Postprocessing- assemble the intersection primitives into curves of intersection, refine the accuracy of computed intersection points, and compact the intersection curves. This surface/surface intersection algorithm is applicable to the widest class, C^o, of parametric surfaces, an enhancement over the existing algorithms applicable to only C^k, $k \geq 1$, surfaces. This implementation, based on computational intelligence, requires no human interaction for intersection curve pattern recognition.

INTRODUCTION

The development of an intelligent surface/surface intersection algorithm is of practical interest in the geometric modeling and manufacturing areas. The recognition and extraction of the intersection curves automatically is clearly a non-trivial task in CAD/CAM systems. The value of a software package is primarily a function of the degree to which it is used. The area of manufacturing where surface/surface intersection has been extensively applied is the aircraft design. This research evolved from a desire to present the design, in general, and an intelligent implementation, in particular.

For a pair of surfaces, three questions arise: a) Existence- does there exist an intersection between the two surfaces? b) Uniqueness- is the intersection a single curve or several curves? and c) Solution Method- is the method of solution analytic or numerical for calculating the intersection curves when they exist? These questions are addressed in the Intersection, Decomposition, and Composition sections, in that order. It is not enough to tell the computer to intersect two surfaces, the surface definition has a direct affect on the degree of automation that can be achieved. One must specify the kind of surfaces and the procedure to process them. Thus, given the type of the parametric surfaces, (e.g., Bezier, B-Spline, or free-form surfaces), there are two approaches: generative and variant. In the generative approach to the intersection problem, separate algorithms are generated for all possible surface pairs and then an algorithm appropriate for the application is used. Currently, this approach is used in Advanced Integrated Mathematical Systems (AIMS) at McDonnell Douglas Corporation. Special purpose algorithms are used for analytic and numerical solutions and a general purpose algorithm is used for sculptured surfaces. The generative approach can be computationally prohibitive in all but most trivial applications. Also, in general, a closed form solution does not exist. The general purpose algorithm, the variant approach based on the divide-and-conquer strategy, is adopted to handle all types of surfaces without explicit knowledge of the surface type whether it is analytic or synthetic.

There are many specialized algorithms for intersection between two C^k, $k \geq 1$, parametric surfaces, defined over a rectangular parametric space. These methods for the surface/surface intersection problem depend on the intrinsic characteristics of the surfaces. The methods developed at McDonnell Douglas [Melson 1978, Houghton et al 1985] and by [Barnhill et al 1987] depend on the availability of first partial derivatives. The method developed at Automation Technology Products [Filip et al 1986] requires the use of second order derivatives. The specialized and widely used methods

[Lane et al 1980, Dokken 1985] have been developed for composite surfaces, such as B-spline or Bezier surfaces, depend on the control points. The first step in the design of a solution to perform intersections between C^0 parametric surfaces was proposed by Sabharwal & Factor [1988]. An example of a C^0, but not C^1, surface is the accordion type surface [see Figure 1]. This surface is normally used as four planar surfaces in a B-rep solid modeler, it can now be treated as a single C^0 surface. For example, performing the intersection between two surfaces [see Figure 2] can be reduced to applying the intersection algorithm once to two surfaces instead of applying the algorithm four times to the surface pairs (i.e., all possible combinations of planar surfaces from each surface).

The basic considerations for designing a general purpose algorithm then become: (1) the quality of intersection in terms of robustness and accuracy; (2) the computational cost in terms of run time memory space and algorithm speed for minimal CPU time. To achieve these objectives, expert knowledge is embedded in the decision process using: (a) recursive, selective, and adaptive decomposition of surfaces; intelligent interpretation and use of surface curvature without calculating it; systematic filtering of trivially disjoint surface elements, (b) computationally efficient triangulation of surface and intersection method, (c) polytree data structure for geometric integration for curve pattern recognition, and (d) expert refinement process.

The SSINT algorithm is a complete design and implementation that handles all C^k, $k \geq 0$, parametric surfaces. These surfaces are defined on a rectangular parametric space. The earlier approaches required the evaluator to supply the position values as well as the partial derivatives. In the current approach, the parametric evaluator supplies only the surface points during decomposition and the accuracy refinement of the approximate computed points. The computations are independent of the intrinsic properties that some surfaces possess.

DECOMPOSITION OF SURFACES

An intelligent surface decomposition plays a central role in determining the success of subsequent steps in locating the surface intersections. Surface decomposition is an adaptive, selective and recursive method for filtering out trivially disjoint surface parts and determining the potentially intersecting sections of the surfaces. The divide-and-conquer strategy transforms the problem of intersecting complex surfaces into intersecting almost planar surfaces. It is worthwhile to spend time for divide (the decomposition of surfaces) to make conquer (the intersection between the surfaces) straightforward. Thus, surface decomposition centers around the possibly intersecting sections of surfaces. In order to apply decomposition intelligently to the surface/surface intersection problem [see Figure 3], several criteria are used: (a) which way to decompose, in one or both parametric directions [see Figures 4,5], (b) how to subdivide, uniformly or adaptively[see Figures 5,7], (c) how to select parts of the surfaces for decomposition[see Figures 5,8], and (d) what is the flatness threshold level of decomposition[see Figures 6,7,8,9].

Adaptive decomposition is a function of the morphology of surfaces. The primary purpose of the decomposition technique is to automatically detect where the curvature is high and then subdivide accordingly. The previous method [Sabharwal & Factor 1988] for calculating curvature was based on approximate derivatives similar to that developed by Akima [1984]. This led to a significant increase in evaluator call. Expert knowledge, based on experience and on the general principles to reason about the surface calculations, is used to simulate surface curvature eliminating these evaluator calls. An expert may use testimates, rough approximations, and general knowledge, even when such exact derivatives are readily available. In the present technique, no formal calculations for derivatives are made. The approximate calculations draw closely on the theory of analytic derivatives and are based on only the precomputed grid values utilizing relaxation method. The distinction is that the previous methods required surface derivatives and direct calculation of surface normals, whereas the new procedure does not.

Recursive decomposition partitions the surfaces into elements, at varying threshold levels of curvature, until the decomposition tolerance is reached. The threshold levels of decomposition correspond to increasing levels of flatness resolution of the surface elements.

Selective decomposition filters out all surface pairs which are irrelevant to intersection as an initial step. Selective decomposition is based on successively lowering the decomposition threshold and eliminates the costly evaluator calls pertaining to decomposition. The filtering process is done by enclosing the two surfaces in bounding volumes and determining if the bounding volumes intersect. This divide-and-conquer algorithm avoids proceeding recursively when the size of the object no longer justifies further decomposition or a surface element is no longer a candidate for intersection. In the former case, it is easier to apply the basic intersection procedure given in the Intersection section. The divide-and-conquer strategy also produces a non-binary tree of surface pairs whose leaves represent surface pair elements that are most likely to yield non-empty

intersection. However, this tree is intelligently replaced by a binary tree called polytree. Binary trees implement dictionaries as well as other complicated integration operations efficiently.

INTERSECTION

Next, the possibly intersecting surface elements, within the specified flatness tolerance, are to be considered for actual intersection. Traditionally, the techniques [Melson 1978, Barnhill 1987] for intersection depend on a start point. The ability to visually detect the start points is well suited to humans. Thus, human interaction is required to determine the start point or points when there are several curves of intersection. The object is to bypass the intermediate step of locating the start point. Then the Newton-Raphson technique is used to trace the path until a complete curve is obtained. The success of this technique depends on the availability of exact derivatives. The estimated start point is decisive to the determination of the entire curve of intersection, since the accuracy of the curve depends on the accuracy of the starting point.

An intelligent method, independent of surface derivatives, for intersecting two surfaces is the triangular decomposition technique. Individual surface elements are decomposed into triangles to yield continuous triangulation of surfaces [see Figure 13]. The parametric representation of triangles is used to intersect triangle pairs. The implementation of parametric representation shows that it is on the average 50% faster than the heuristic approach to intersect triangle pairs [see Figures 10,11]. The intersection between pairs of triangles, if it exists, is represented by two end points of the intersection line segment, called the curve "primitive".

Four problems arise when the triangle pair intersection technique is used. First, surface "cracks" appear in the triangular approximation of the surfaces[see Figure 12]. The surface "cracks" are eliminated by judicious triangulation [see Figure 13] of surface elements by including additional decomposition points on the surface boundary where the adjoining surface elements have vertices. Secondly, multiple copies of identical intersections may be calculated if the triangles intersect along the edges [see Figure 14]. The previous method creates a new data structure to store potentially duplicate primitives. Due to space and time considerations, this algorithm applies an intelligent beam search through the intersection primitives to eliminate duplicate intersections. Thirdly, when two surface elements are planar, redundant collinear points may be calculated [see Figure 15]. They can be removed by eliminating the intermediate points for each set of three collinear points [see Figure 16]. Finally, the computed points are approximations, not actual points on the curves of intersection. This is the accuracy problem of computed intersection points [see Figures 17,18,19]. The algorithm for improving the accuracy of approximate intersection curves is designed in the Refinement section.

The intersection primitives are stored only in terms of the end points of intersection primitives in a polytree of surface pairs. The polytree data structure facilitates the process of integrating intersection primitives into curves, because the search procedure takes you quickly to the intersection primitives without looking at the data at the internal nodes.

COMPOSITION OF INTERSECTION ELEMENTS

Another issue with intersection with the triangulation method is that the totality of intersection primitives is not organized in the form of curves. The composition step synthesizes the computed intersections primitives. The composition process collects the clusters of intersection curve primitives, coalesces them into a set of continuous intersection curves and compacts the intersection curves by eliminating the redundant collinear points Figures 15,16]. The intersection primitives are stored in a data structure that promotes a method for limiting the search space that needs to be investigated. The geometric composition is an intelligent composition method that prevents fruitless searching of the entire polytree of intersection primitives. The end points of the primitives in the intersection curves are connected on the basis of absolute "nearness" with machine epsilon. This algorithm takes advantage of the spatial coherence of the intersection primitives so that the points are connected in a hierarchical manner. In this branched reasoning tree, composition becomes slightly more recursive but it is worth it.

The composition process, in principle, is the same throughout the historical development of the algorithm. It is similar to feature detection techniques used in geometric modelers. However, the polytree data structure used here promotes the method of limmiting the search space. The current implementation differs significantly from the previous implementations. Previously, primitives were first linked manually at the leaf level and recursively at the parent levels of the polytree. The algorithm has been revised and improved. The new algorithm is also recursive but it connects the intersection primitives automatically at all levels of the polytree of surface pairs. The points on intersection curves are approximations of the actual points on intersection curves. Thus, the computed points do not necessarily lie on either surface. This requires improving

the accuracy of these points to representative intersection points closer to the two intersecting surfaces.

REFINEMENT

The refinement process analyzes the computed points to determine if they can be brought closer to the surfaces. It improves the accuracy of computed intersection points that constitute curves of intersection. For complex surfaces, these approximate points work well for quick refinement to new points so that the new points lie on the surfaces or are as close to the surfaces as possible. For simple surfaces, these estimates represent the exact points. A new criteria to find the points on the surfaces or close to the surfaces within a specified tolerance is given . In the previous methods, first surface points P_1 and P_2 that are closer to P are determined. The refinement of the approximate intersection points to new points closer to the surfaces is obtained by calculating the intersection point among three planes: tangent plane at P_1, tangent plane at P_2, and the plane through P_1, P_2, and P. The tangent planes to the surfaces which require analytic calculation of normals to the surfaces. In the current implementation, the need for tangent planes has been eliminated. Expert knowledge is extracted from the locations of surface points P_1 , P_2, the computed point P and used to calculate the new point P_{new} closer to the surfaces. This eliminates the calculation of tangent planes which depend on surface derivatives and their intersection. For each computed point calculate two points, one point on each surface, which are at a minimum distance from the given point. Refine the given point as follows. In the plane of the given point P and the two surface points P_1 and P_2, find P_{new} such that P_{new} P_1 and P_{new} P_2 are orthogonal to P_1P and P_2P respectively [see Figure 17]. By replacing P with P_{new}, the loop for refinement terminates once the new point is within specified tolerance of the two surfaces. Consider special cases when P_1P or P_2P becomes degenerate [see Figure 18].

Final step is refining the midpoints: if P_1 and P_2 are two points, P_{mid} is the middle point, the this point is refined, if it is not within specified tolerance of the two surfaces, to P_{new} [see Figure 19]. This process is applied recursively to all primitives so the curves are truly representative of intersection curves.

CONCLUSION

This paper presented a method for cross intersection between both smooth and non-smooth parametric surfaces in three steps: Preprocessing, Intersection, and Postprocessing. These steps include Decomposition, Intersection, Composition, and Refinement. The present work is a complete design and implementation which is a major enhancement over the previous surface/surface intersection implementations at McDonnell Douglas [Melson 1978, Houghton et al 1985, Sabharwal & Melson 1989]. This method is applicable to all parametric surfaces depending only on the surface definition and not the surface smoothness. This technique eliminates complete digitization or the use of a Newton-Raphson type method. This method is completely independent of the surface derivatives on which the Newton-Raphson technique depends. This algorithm was tested on a variety of surfaces about which the results were known. The test results were quite satisfactory. This presentation automates the surface/surface intersection algorithm as used in CAD/CAM applications. No human interaction is required for intersection curve pattern recognition.

ACKNOWLEDGMENT

Janet E. Oakes provided useful discussions during the preparation of this paper. Edward A. Motl provided useful interfaces to Evans and Sutherland graphics devices.

REFERENCES

Akima, Hiroshi, *On estimating partial derivatives for bivariate interpolation of scattered data*, **Rocky Mountain Journal of Mathematics**, 14, 1984, pp. 41-52.

Barnhill, R. E., Farin, G., Jordan, M., & Piper, B. C., *Surface/surface intersection*, **Computer Aided Geometric Design**, 4, 1987, pp. 3-16.

Dokken, T., *Finding intersections of B-spline represented geometries using recursive subdivision techniques*, **Computer Aided Geometric Design**, 2, 1985, pp. 189-195.

Filip, D., Magedson, R., & Markot, R., *Surface algorithms using bounds on derivatives*, **Computer Aided Geometric Design**, 3, 1986, pp. 295-311.

Houghton, E. G., Emnett, R. F., Factor, J. D., & Sabharwal, C. L., *Implementation of a divide-and-conquer method for intersection of parametric surfaces*, **Computer Aided Geometric Design**, 2, 1985, pp. 173-183.

Lane, J. M. and Riesenfeld, R. F., A *theoretical development for the computer generation and display of piecewise polynomial surfaces*, **IEEE Trans. Pattern Analysis and Machine Intelligence** 2(1), 1980.

Melson, Thomas G., *Surface/surface intersection*, **MCAIR Report** 78-011, 1978, McDonnell Douglas Corporation, St. Louis, MO.

Sabharwal, C. L. & Factor, J. D., *Cross intersection between any two C^0 surfaces*, **Proc. Austr. International Comp Graph Conference,** AUSGRAPH'88, 1988, pp. 37-42.

Sabharwal, C. L. & Melson, T. G., *Intersection between non-smooth surfaces*, **MDC Report** B1472, 1989, McDonnell Douglas Corporation, St. Louis, MO.

Sabharwal, C. L. & Melson, T. G., *Implementation of cross intersection between triangular surfaces*, **Proc. of the ACM Computer Science Conference**, 1988, pp. 92-101.

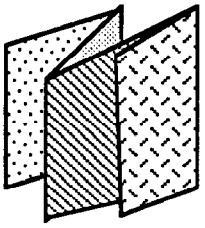

Figure 1. C^0 Surface

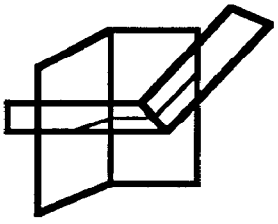

Figure 2. Intersection Between C^0 Surfaces

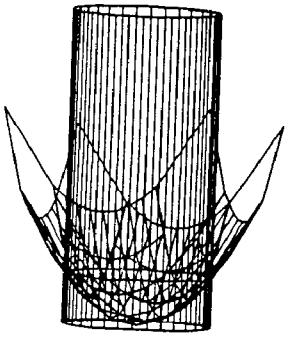

FIGURE 3. Intersection Of A Cylinder And A Paraboliod Of Revolution With Decomposition Threshold Of 10 Degrees.

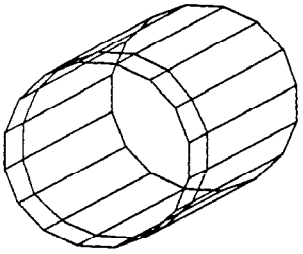

FIGURE 4. The Surface Of Cylinder Subdivided In Only One Parametric Direction With Decomposition Threshold Of 20 Degrees.

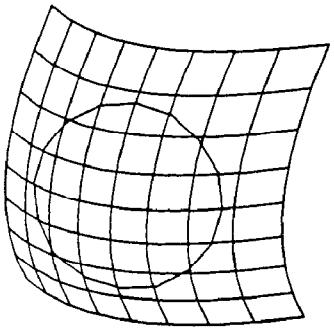

FIGURE 5. The Surface Of Paraboloid Of Revolution Uniformly Subdivided In Both Parametric Directions With Decomposition Threshold Of 20 Degrees.

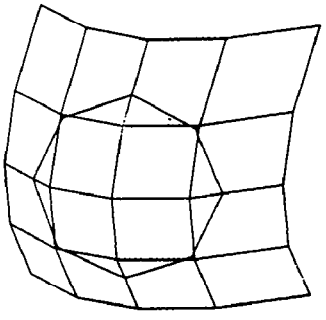

FIGURE 6. The Surface Of Paraboloid Of Revolution Adaptively Subdivided In Both Parametric Directions With Decomposition Threshold Of 40 Degrees.

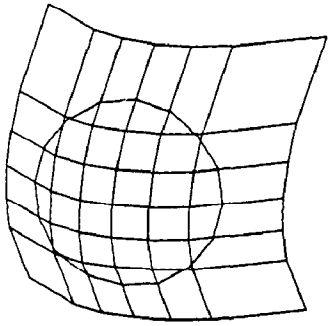

FIGURE 7. The Surface Of Paraboloid Of Revolution Adaptively Subdivided In Both Parametric Directions With Decomposition Threshold Of 20 Degrees.

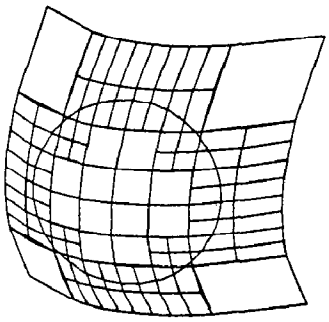

FIGURE 8. The Surface Of Paraboloid Of Revolution Selectively Subdivided In Both Parametric Directions With Decomposition Threshold Of 10 Degrees.

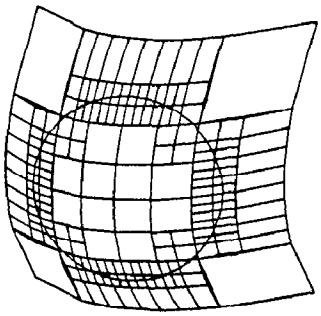

FIGURE 9. The Surface Of Paraboloid Of Revolution Selectively Subdivided With Decomposition Threshold Of 5 Degrees To Yield Realistic Intersection Curve.

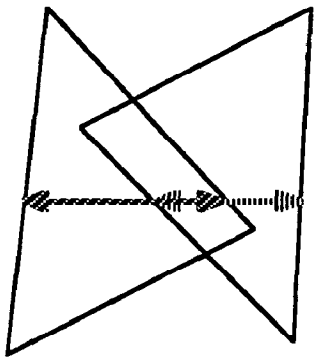

Figure 10
Intersection Via Heuristics

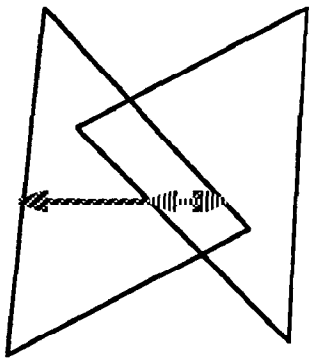

Figure 11
Intersection Via
Parametric Method

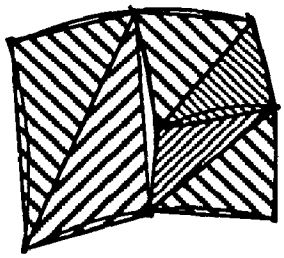

Figure 12. surface "crack"

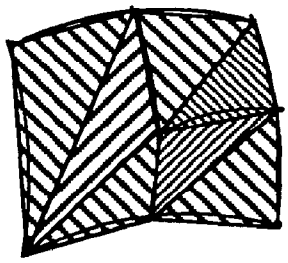

Figure 13.
surface continuity

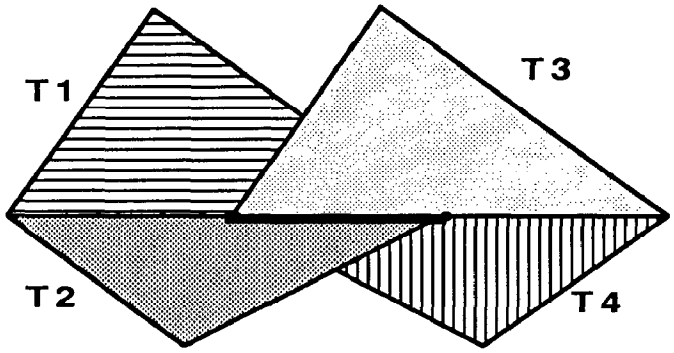

Figure14. Multiple intersections. (T1, T2 triangulate one surface element, and T3, T4 triangulate second surface element. Intersection of two surface elements yields four identical intersections of T1,T3; T1,T4; T2,T3; T2,T4.)

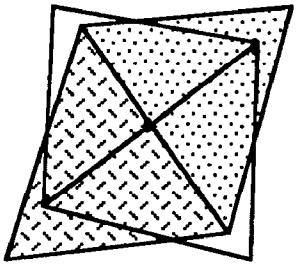

Figure 15

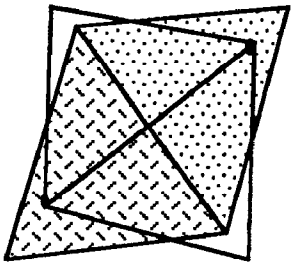

Figure 16

Redundant (Figure 15), No Redundant (Figure16) Collinear Intersection Points

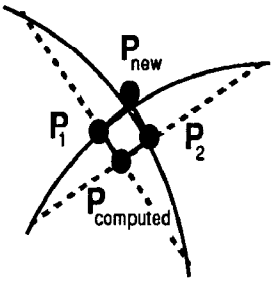

Figure 17

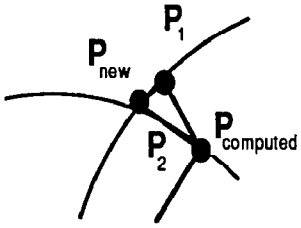

Figure 18

Computed Intersection Point Not On Any Surface (Figure 17), On One Surface (Figure 18)

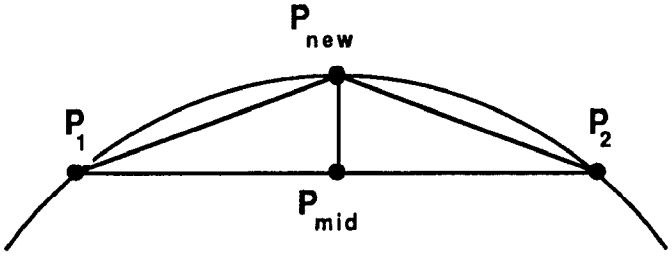

Figure 19 . Refining The Middle Point.