SHAPE INTERROGATION BY MEDIAL AXIS TRANSFORM FOR AUTOMATED ANALYSIS

by

Halit Nebi Gürsoy

Engineer, Istanbul Technical University
June, 1980

S.M., N.A.M.E., Massachusetts Institute of Technology
February, 1986

SUBMITTED TO THE DEPARTMENT OF OCEAN ENGINEERING IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

November 1989

Copyright © 1989 Massachusetts Institute of Technology

Signature of Author ___________________________ Department of Ocean Engineering
November 2, 1989

Certified by ________________________________ Professor Nicholas M. Patrikalakis
Thesis Supervisor

Accepted by ________________________________ Professor A. Douglas Carmichael, Chairman
Departmental Graduate Committee, Department of Ocean Engineering

MAR 22 1990
ARCHIVES
SHAPE INTERROGATION BY MEDIAL AXIS TRANSFORM
FOR AUTOMATED ANALYSIS

by

Halit Nebi Gürsoy

Submitted to the Department of Ocean Engineering on November 2, 1989 in
partial fulfilment of the requirements for the degree of Doctor of Philosophy.

Abstract

In this thesis we develop a new interrogation method based on the medial axis transform to
extract some important global shape characteristics from geometric representations. These
shape characteristics include constrictions, maximum thickness points, associated length
scales; isolation of holes and their proximity information; and a set of topologically simple
subdomains decomposing a complex domain. The new algorithm we develop to compute
the medial axis transform of planar multiply connected shapes with curved boundaries can
automatically identify these characteristics. In order to demonstrate the effectiveness of
this interrogation method to construct automated solutions to some important design and
analysis problems, we develop a novel, efficient and automatic finite element mesh
generation scheme for complex two-dimensional domains. To demonstrate the
effectiveness of our method, the system we develop is applied to several representative
design problems. These include adaptive finite element solution of linear plane elasticity
problems, automated adaptive triangulation and faceted approximation of trimmed curved
surface patches and idealization and model creation in structural analysis processes.
Finally, other possible applications of our interrogation and modeling methods are
identified and recommendations for future research are included.
ACKNOWLEDGEMENTS

There are many people and organizations that I would like to thank for their help, support and encouragement during my graduate studies at MIT. I am grateful to each of them.

Professor Chryssostomos Chryssostomidis, who has established and directs a computer aided design laboratory with state-of-the-art equipment, has provided academic guidance since the beginning of my graduate studies at MIT. He has also introduced and encouraged me to study in the promising area of Computer Aided Design and contributed useful advice as a member of my thesis committee.

Professor Nicholas M. Patrikalakis, my thesis advisor and doctoral committee chairman, has been helpful in both the development of the research presented here and the writing of this thesis.

Members of my doctoral thesis committee, Professor David C. Gossard, Professor Tomasz Wierzbicki and Dr. Franz-Erich Wolter, have provided valuable advice, feedback and encouragement.

Mr. Michael S. Drooker and my fellow Design Laboratory mates have provided a stimulating environment. Mr. Seamus T. Tuohy and Mr. Evan C. Sherbrooke have contributed to the computer implementation.

The Turkish Scientific and Technical Research Council provided a fellowship during the first four years of my graduate studies at MIT.

Most of all, I would like to thank my family: my wife, Muzaffer, my parents, sister and brother-in-law for their support, encouragement, and patience; and my former teacher, Professor Cengiz M. Dokmeci, who has taught me principles of engineering and has been a continuous source of inspiration.

This research was supported in part by the MIT Sea Grant College Program and the Office of Naval Research, grant number N00014-87-K-0462.
# Table of Contents

Abstract 2  
ACKNOWLEDGEMENTS 3  
Table of Contents 4  
List of Figures 6  

1. Introduction 9  

2. Literature Review of Finite Element Idealization and Discretization 15  
   2.1 Finite Element Idealization 16  
   2.2 Finite Element Models 19  
   2.3 Integration of Solid Modeling with Finite Element Modeling 20  
   2.4 Review of Finite Element Mesh Generation Methods 23  
      2.4.1 Mapping Mesh Generation 25  
      2.4.2 Point Insertion Followed by Area / Volume Triangulation 28  
      2.4.3 Topology Decomposition 37  
      2.4.4 Spatial Decomposition 39  
      2.4.5 Recursive Subdivision 41  
      2.4.6 Time Complexities of Some Mesh Generation Schemes 43  
   2.5 Summary of Limitations of Existing Meshing Techniques from Automation View 44  
   2.6 Attribute Specification for Finite Element Model 46  

3. Medial Axis Transform as a Shape Interrogation Method 48  
   3.1 Introduction 48  
   3.2 Literature Review on the Medial Axis Transform 49  
   3.3 Basic Properties of the Medial Axis Transform 56  
   3.4 Applications of the Medial Axis Transform 60  

4. Computation of the Medial Axis Transform 68  
   4.1 A Computational Methodology for the Medial Axis Transform 69  
      4.1.1 Intermediate Branch Points 74  
      4.1.2 Initial Branch Points 75  
      4.1.3 Computation of Effective Offset Distance 76  
      4.1.4 Skeleton and Subdivisions 78  
      4.1.5 Approximate Medial Axis Transform For Planar Shapes With Curved Boundaries 80  
   4.2 Implementation of the Medial Axis Transform of Planar Shapes 82  
      4.2.1 Data Structures 83  
      4.2.2 An Algorithm for the Medial Axis Transform 87  
      4.2.3 Examples on the Medial Axis Transform Computation 89  
   4.3 Extension of the Medial Axis Transform to Trimmed Curved Surface Patches and Volumes 89  
      4.3.1 The Medial Axis Transform of Trimmed Curved Surface Patches 91  
      4.3.2 The Medial Axis Transform of a Closed Polyhedron 95  

5. Automatic Mesh Generation Based on the Medial Axis Transform 102
5.1 Finite Element Mesh Generation
5.2 Finite Element Triangulation of a Two-Dimensional Shape
5.3 A New Finite Element Mesh Generation Scheme Based on the Medial Axis Transform
  5.3.1 Decomposition of a Region by Means of Voronoi Regions
  5.3.2 Processing of the Subregions Obtained from the Medial Axis Transform
  5.3.3 Discretization of Boundaries and Interiors of Subdomains
  5.3.4 Triangulation of Subdomains
  5.3.5 Mesh Smoothing Process
  5.3.6 Data Structures and Algorithm for Triangulation
  5.3.7 Local Mesh Refinement
  5.3.8 Complexity Estimates of the Medial Axis Transform and Meshing Processes
5.4 Extension of the Triangulation Technique to Trimmed Curved Surface Patches
5.5 Triangular Finite Element Mesh Examples

6. Applications of the Medial Axis Transform
6.1 Adaptive Finite Element Analysis
  6.1.1 Adaptive Finite Element Analysis Schemes
  6.1.2 An Adaptive Finite Element Analysis Scheme for Plane Elasticity Problems
  6.1.3 Examples
6.2 An Idealization and Structural Model Creation Based on the Medial Axis Transform
6.3 Faceted Representation of Trimmed Curved Surface Patches

7. Conclusions and Recommendations

Appendix A. Definitions of Conic Skeleton Branches
Appendix B. Computation of Branch Points
Appendix C. Parametric Forms of Skeleton Branches
Appendix D. Approximation of a NURBS Curve in Terms of Straight Line and Circular Arc Segments
List of Figures

**Figure 1-1:** A Solid Modeler with Several Functional Layers, (Adapted from [Weller86])  
**Figure 2-1:** A Complex Planar Shape and its Decomposition, (Adapted from [Chae 88])  
**Figure 2-2:** Finite Element Mesh of the Planar Shape, (Adapted from [Chae 88])  
**Figure 2-3:** A Direct Finite Element Model  
**Figure 2-4:** Indirect Models of Several Structural Components  
**Figure 2-5:** Mapping Mesh Generation  
**Figure 2-6:** Surrounding Procedure Around a Node, (Adapted from [Van-Phai 82])  
**Figure 2-7:** Surrounding Procedure Around a Line, (Adapted from [Van-Phai 82])  
**Figure 2-8:** Point Insertion, Tessellation and Triangulation  
**Figure 2-9:** Zero Volume Elements (Adapted from [Cavendish 85])  
**Figure 2-10:** Element Creation Based on Topology Decomposition  
**Figure 2-11:** Modified Quadtree Representation of a Circle, (Adapted from [Yerry 83])  
**Figure 2-12:** Recursive Subdivisions, (Adapted from [Shephard 85a])  
**Figure 3-1:** The Medial Axis of a Planar Shape  
**Figure 3-2:** Reconstruction of a Shape Using its Skeleton and Radius Function  
**Figure 3-3:** The Voronoi Diagram and the Medial Axis of a Planar Non-Convex Shape  
**Figure 3-4:** Shape Representation by Skeleton, (Adapted from [Blum 73])  
**Figure 3-5:** Skeleton of a Symmetric Shape  
**Figure 3-6:** Decomposition of a Convex Shape by Medial Axis Transform  
**Figure 3-7:** Decomposition of a Non-Convex Shape by Medial Axis Transform  
**Figure 3-8:** The Skeleton of a Multiply Connected Shape  
**Figure 3-9:** The Medial Axis of Two Planar Curves  
**Figure 3-10:** Effect of Boundary Perturbations on the Skeleton of a Planar Shape  
**Figure 3-11:** The Medial Axis Transform Tree of a Simple Planar Shape  
**Figure 3-12:** The Skeleton of an Elongated Shape  
**Figure 4-1:** Branch Points, (Adapted from [Montanari69])  
**Figure 4-2:** Special Cases of Initial Branch Point  
**Figure 4-3:** Possible Intermediate Branch Points  
**Figure 4-4:** A Spurious Initial Branch Point  
**Figure 4-5:** Reduction in Boundary Elements  
**Figure 4-6:** Splitting of Contours in Offset Process  
**Figure 4-7:** Medial Axis Transform of a Non-Convex Shape  
**Figure 4-8:** Medial Axis Transform of a Shape with Curved Boundary  
**Figure 4-9:** Graphical Representation of the Data Structures  
**Figure 4-10:** Objects Used in Boundary Representation  
**Figure 4-11:** Medial Axis Transform of a Symmetric and Periodic Shape  
**Figure 4-12:** Decomposition of a Non-Convex Shape by Medial Axis Transform  
**Figure 4-13:** Identification of Significant Points by Medial Axis Transform
| Figure 4-14: | Idealization of a Shape by Medial Axis Transform | 4.2.3 |
| Figure 4-15: | Medial Axis Transform of a Complex Shape | 4.2.3 |
| Figure 4-16: | Mapping Between Parameter Space and Three-Dimensional Space | 4.3.1 |
| Figure 4-17: | Standard Offset of a Non-Convex Shape | 4.3.2 |
| Figure 4-18: | Offset without the Contribution of a Re-entrant Vertex | 4.3.2 |
| Figure 4-19: | Approximate Offset of a Non-Convex Shape | 4.3.2 |
| Figure 4-20: | Intermediate Branch Point on an Approximate Offset | 4.3.2 |
| Figure 4-21: | Initial Branch Point on an Approximate Offset | 4.3.2 |
| Figure 4-22: | Three-Dimensional Medial Axis Transform | 4.3.2 |
| Figure 4-23: | Non-Two-Manifold Situations | 4.3.2 |
| Figure 5-1: | Shape Decomposition by Means of The Medial Axis Transform | 5.3.1 |
| Figure 5-2: | A Simple Parametric Shape | 5.3.2 |
| Figure 5-3: | Finite Element Mesh of the Parametric Shape | 5.3.2 |
| Figure 5-4: | Finite Elements with High Aspect Ratios | 5.3.2 |
| Figure 5-5: | Modified Skeleton and Associated Discretization | 5.3.2 |
| Figure 5-6: | A Narrow Subregion and its Modification | 5.3.2 |
| Figure 5-7: | A Narrow Subregion Associated with a Re-entrant Vertex | 5.3.2 |
| Figure 5-8: | Elimination of a Short Branch | 5.3.2 |
| Figure 5-9: | Strips Created in Quadrilateral and Triangular Subdomains | 5.3.3 |
| Figure 5-10: | Node Generation Procedure Within a Subdomain | 5.3.4 |
| Figure 5-11: | A Planar Multiply Connected Shape and its Skeleton | 5.3.4 |
| Figure 5-12: | Region Subdivision and Boundary Discretization | 5.3.4 |
| Figure 5-13: | Triangulation of an Individual Subdomain | 5.3.4 |
| Figure 5-14: | Finite Element Mesh of the Planar Multiply Connected Region | 5.3.4 |
| Figure 5-15: | Relationships Between Objects Used in the Mesh Generator | 5.3.6 |
| Figure 5-16: | Triangle Bisection Operators | 5.3.7 |
| Figure 5-17: | List Manipulation During the Mesh Refinement Process | 5.3.7 |
| Figure 5-18: | Triangular Mesh of a Convex Region with 120 Elements | 5.3.8 |
| Figure 5-19: | Computation Time Results of Four Meshes of Simple Region | 5.3.8 |
| Figure 5-20: | Finite Element Mesh of a Bracket | 5.5 |
| Figure 5-21: | Finite Element Triangulation of a Non-Convex Domain | 5.5 |
| Figure 5-22: | Finite Element Triangulation of an Irregular Region | 5.5 |
| Figure 5-23: | Finite Element Mesh of a Flat Plate with Two Holes | 5.5 |
| Figure 5-24: | Finite Element Mesh of an Irregular Domain with a Hole | 5.5 |
| Figure 5-25: | Finite Element Mesh of a Multiply Connected Domain | 5.6 |
| Figure 6-1: | An Integrated CAD System, (Adapted from [Bathe86]) | 6.1.1 |
| Figure 6-2: | Stress Jumps Involved in Finite Element Solution | 6.1.2 |
| Figure 6-3: | Adaptive Analysis of a Bracket | 6.1.3 |
| Figure 6-4: | Adaptive Analysis of a L-Shaped Plate | 6.1.3 |
| Figure 6-5: | Adaptive Analysis of a Beam with a Hole | 6.1.3 |
| Figure 6-6: | An Elongated Structure | 6.2 |
| Figure 6-7: | Medial Axis Transform of the Structure | 6.2 |
| Figure 6-8: | Idealization of the Structure Using the Skeleton | 6.2 |
| Figure 6-9: | Two Mode Shapes of the Two-Dimensional Continuum Model | 6.2 |
| Figure 6-10: | Two Mode Shapes of the Beam Model | 6.2 |
| Figure 6-11: | A Planar Object and its Mechanical Model Created Using the Skeleton | 6.2 |
| Figure 6-12: | Triangulation of a Trimmed Curved Surface Patch with Four Holes | 6.3 |
| Figure 6-13: | Triangulation of an Irregular Trimmed Curved Surface Patch | 6.3 |
| Figure 6-14: | Triangulation of a Doubly Curved Shell with Two Holes | 6.3 |
| Figure 6-15: | Triangulation of a Trimmed Curved Surface Patch with a Hole | 6.3 |
| Figure A-1: | Skeleton of Two Lines          | A |
| Figure A-2: | Skeleton of a Line and a Circle | A |
| Figure A-3: | Skeleton of Two Circles         | A |
| Figure B-1: | Initial Branch Point of Two Circles | B |
| Figure B-2: | Initial Branch Point of a Line and a Circle | B |
| Figure C-1: | Parabolic Skeleton Branch       | C |
| Figure C-2: | Hyperbolic Skeleton Branch      | C |
| Figure C-3: | Elliptic Skeleton Branch        | C |
Chapter 1

Introduction

In this thesis, we investigate shape interrogation and automation of engineering analysis computations for computer aided design and manufacture by means of the medial axis transform (MAT). The medial axis of a two-dimensional closed area or a three-dimensional volume is the set of points made up of the centers of all maximal disks or spheres that fit inside the shape. The medial axis and the associated radius function of the disks or spheres define the medial axis transform [Blum 73].

Computer aided design and manufacture (CAD/CAM) based on solid modeling techniques is a relatively new discipline which is concerned with integration of computer techniques with engineering design, analysis and manufacture in a unified system. In a typical solid modeling system, it is required that tools assisting in the execution of design tasks during the design process are available. The design process is an interactive activity in which the designer carries out various design tasks including conceptual design, form creation, engineering analysis, e.g. using finite element method (FEM), and planning of fabrication process. A CAD/CAM system is also required to store and manipulate complete, unambiguous representations of the geometry of objects being designed. In such an integrated design environment, the designer would have capabilities to rapidly assess the performance of a particular design at a very early stage of the design process. A correct geometric representation extracted from the geometric database and automated engineering analysis techniques would allow the designer to carry out detailed simulations without using expensive and time consuming mechanical prototypes. Thus the designer could examine more design options and try different design configurations to come up with an optimum design. This capability is a desirable feature, especially in a novel design process.
In such a process, the designer usually has no fixed rules such as those codified by classification societies for ship construction. One of the main characteristics of this process is that it requires interactive design tools that allow quick design evaluation. During the course of the design, the designer searches a space of designs to create an acceptable design subject to several constraints. A solid modeler with advanced analysis capabilities would be an appropriate medium for such an activity. Some advantages of this design approach are

- better design configurations satisfying all design requirements could be achieved fast by expanding the number of configurations a designer can examine efficiently;
- lesser design flaws would arise in the later stages of design and in manufacture as a consequence of the detailed analysis in early stages of the design process; and,
- that better communication links between the design and manufacture departments could be established by sharing exact representation among various parts of the process.

As a consequence, better and more efficient products would be manufactured.

Currently there are two most widely used schemes for storing the geometric representations of objects in solid modelers: Boundary Representation (B-rep) and Constructive Solid Geometry (CSG) [Requicha 83]. In a Boundary Representation model, topological primitives on the surface of an object (such as faces, edges and vertices) and adjacency relationships among them are represented explicitly using various data structures [Weiler 86]. Therefore Boundary Representation models are sometimes known as evaluated, explicit and boundary based representations. In a CSG representation, an object is stored as a combination of simple primitives. The data are usually arranged in a binary tree structure in which the leaves are the shape primitives and tree nodes are the Boolean set operators constructing the object from the shape primitives. CSG models are sometimes known as unevaluated, implicit and volume based representations. Considering the shape primitives of these two approaches we can regard Boundary Representation as a low level representation compared to CSG approach.
Figure 1-1 shows a hypothetical solid modeling system whose structure comprises several functional layers. In such a system, a new layer can be built on top of the lower levels, and this adds more advanced capabilities to the system. In this system, the lowest level takes care of the representation of the topology and geometry of an object being modeled. The system also includes a set of tools to carry out various geometric operations such as intersections, offsets, sweeps, blends, filleting and chamfering. They can be regarded as low level generic operators since higher level applications such as the Boolean set operators, finite element mesh generation, numerical control (NC) tool path planning and ray tracing are built on top of them. These operators basically interrogate the geometric database to extract information for other applications.

In this thesis, we develop a new interrogation method based on the MAT to automatically extract some important global shape characteristics from the geometric database. The shape characteristics we consider in this thesis include constrictions, maximum distance points from the boundary of a region, holes, the length scales of these characteristics and a set of simple subregions decomposing a complex region. In currently available solid modeling systems, there are no methods to automatically extract such important global shape characteristics and, therefore, our method fills a significant gap in interrogation techniques.

Constrictions (or narrow parts of a region) are significant shape characteristics since they are most likely locations of stress concentration in loaded structures, and onset of separation in fluid flows. Identification of maximum thickness points in a region is useful information in various design and manufacturing processes. Identification of holes in a region and proximity information with the boundary or other holes provides useful information in design, analysis and manufacturing of structures. Decomposition of a complex shape into a set of subregions with simple topology is an active research problem in computational geometry [Toussaint 89]. The solution of this problem can be used in
Figure 1-1: A Solid Modeler with Several Functional Layers, (Adapted from [Weiler86])

many diverse fields such as computer graphics, pattern recognition, image processing, robotics, domain discretization and finite element mesh generation.

Extraction of above global shape characteristics characteristics using the MAT allows us to construct automated solutions to some important engineering analysis problems including finite element mesh generation and discretization of trimmed curved surface patches.

The main difference between our finite element mesh generation technique and other meshing schemes is that, this mesh generator aims at generation of a "coarse" mesh and at
the same time identification of significant characteristics such as constrictions, holes in a region and length scales associated with these characteristics. Using this information on shape characteristics and prescribed boundary conditions the coarse mesh is, then, locally refined to create a graded finite element mesh. Our mesh generation technique also includes local adaptive mesh refinement capability which creates compatible triangular meshes. Such a mesh creation methodology is very suitable for two different applications. First, in an adaptive finite element analysis (FEA) a mesh which provides a prescribed accuracy can be obtained by locally refining a coarse mesh depending on the level of discretization error involved in the solution [Babuska 86]. Second, the multigrid techniques which are very efficient in solving field problems involve a coarse and a fine discretization of the problem domain [Briggs 87].

Our mesh generation technique can be used to discretize trimmed curved surface patches which are very common in Boundary Representation models [Farouki 87]. These patches can be used to idealize structural components such as shells for engineering analysis. This adaptive surface approximation technique can be effectively used in some other design applications such as ray tracing and computation of integral properties of trimmed curved surface patches.

In addition to the above automated methods to solve important engineering problems, we have also explored application of the MAT in creation of idealized (indirect) structural models from geometric representations.

This thesis is organized as follows

• Chapter 2 introduces several shape characteristics which are important in automating the finite element idealization and discretization. It also provides a review of state-of-the-art methods for finite element model generation, primarily from the point of view of complete process automation.

• Chapter 3 describes the concept of the MAT and its basic properties. This chapter also reviews algorithms proposed for MAT computation and discusses potential applications of the MAT technique.

• Chapter 4 presents a new algorithm to compute the MAT of planar multiply
connected shapes bounded by closed curved boundaries. The material presented in this chapter includes our computational methodology, implementation aspects of the algorithm, some preliminary ideas potentially useful in extending the algorithm to trimmed curved surface patches and volumes, and some representative examples computed by our algorithm.

- Chapter 5, first, discusses a novel hybrid finite element mesh generation approach, and outlines our mesh generation method. This automatic meshing technique is a novel application based on the MAT and requires very little user input or manual intervention. Therefore such a meshing scheme can be very practical and useful tool in design and analysis of complex engineering systems. An efficient adaptive refinement method for triangular meshes is presented next. Extension of our triangulation and mesh refinement techniques to three-dimensional trimmed curved surfaces is also discussed. Computation time complexity of our mesh generation scheme is examined. Finally, several representative mesh generation examples demonstrating the capabilities and effectiveness of our mesh generation scheme are also included.

- Chapter 6 describes various applications drawn from engineering design and analysis to demonstrate the usefulness of our interrogation technique based on the MAT. The first application is concerned with an adaptive finite element analysis process. Our meshing scheme based on the MAT provides us an initial mesh which captures the geometric and physical characteristics of the problem domain. Our experimental adaptive finite element solver is based on a posteriori error estimation approach which uses finite element stress solutions and their smoothed distributions to estimate discretization errors involved in the numerical solution. In the second application, we demonstrate the capability of the MAT to create idealized mechanical models of engineering structures. The main concept we investigate is that the MAT of an elongated object which can be used as an indirect model of the object. Finally in the third application, we introduce a novel adaptive surface subdivision and triangulation method for trimmed curved surface patches using our automatic triangulation scheme. Our technique can be used to approximate trimmed curved surface patches in terms of a set of triangular facets.

- Chapter 7 summarizes the contributions of the present work and identifies areas for further research.

This thesis also includes the following Appendices

- Appendix A provides analytical definitions of all types of medial axis branches for boundary contours bounded by rectilinear and circular arc segments.

- Appendix B gives a solution technique for the intersection problems needed to compute intermediate branch points. A method for the computation of initial branch points is also provided.

- Appendix C provides parametric equations of conic sections appropriate for tracing branches of medial axis.

- Appendix D describes an algorithm to approximate a B-spline curve in terms of line segments and circular arcs within a prescribed accuracy.
Chapter 2

Literature Review of Finite Element Idealization and Discretization

The finite element method (FEM) is one of the most important developments in computational methods to take place during this century. Today, it is a widely used, powerful tool in many scientific and technological fields. An ongoing effort in scientific and engineering communities is to improve its capabilities and to make it more readily usable in diverse areas. In this chapter we introduce several shape characteristics which are important in automating the finite element idealization and discretization. Then, we briefly review state-of-the-art methods in finite element model generation techniques, primarily from the point of view of complete process automation.

This chapter is structured as follows. Section 2.1 identifies several shape characteristics related to finite element idealization and briefly discusses extraction of such characteristics for automatic finite element mesh generation purposes. Section 2.2 introduces a classification of finite element models used in engineering analyses. Section 2.3 reviews a methodology allowing integration of finite element modeling and solid modeling in a unified computer system. Section 2.4 provides a classification and detailed review of various finite element mesh generation techniques. Section 2.5 summarizes limitations of the existing finite element mesh generation schemes from automation point of view. Finally section 2.6 discusses approaches to specify attributes for finite element models.
2.1 Finite Element Idealization

The FEM addresses solution of boundary value or initial value problems which are
discretized by means of finite element meshes. In general, the problem is recast into a weak
form (i.e. an integral formulation of the field problem) by means of a weighted residual or
variational approach [Kardestuncer 87]. In such a discretization a mesh composed of
predefined types of finite elements is mapped onto the problem domain. This process is
genuinely based. An automated preprocessor which creates the finite element model by
interrogating geometry would be a very useful addition to currently available finite element
analysis systems. To make a preprocessor most useful, the following capabilities are
needed:

- Detection of symmetry and periodicity of problem domain helps to reduce the
  size of the problem. By making use of the symmetry of the problem, we could
  carry out the analysis in a fraction of the actual problem domain. This would
  increase solution speed and allow us to increase the number of degrees of
  freedom of the problem within given computer resources. This, in turn, would
  improve accuracy of numerical results.

- Detection of constrictions in the problem domain allows the implementation of
  physically motivated and more efficient discretization of the problem domain.
  The choice of the initial finite element mesh topology is an important factor
  from the efficiency point of view [Carey 84]. To achieve rapidly convergent
  results, we usually have to refine a finite element mesh in regions where the
  domain is narrow. Those regions, for example, are significant from the
  structural analysis point of view, because stress concentrations usually occur in
  such areas. Similar phenomena may be observed in other fields of continuum
  mechanics. For example in fluid dynamics, those regions most likely give rise
  to flow separation, and are significant from the design and analysis point of
  view.

- Extraction of special characteristics in the problem domain permits more
  effective discretization of the domain and this, in turn, would increase the
  accuracy of numerical results. For example, an opening (hole) in a plate is a
  important characteristic of the domain. In a finite element analysis depending
  on the boundary and load conditions, a finer mesh should be used around the
  opening in order to obtain accurate results.

- Decomposition of a complex shape into a set of topologically simple
  subdomains helps creation of finite element models in an efficient and
  automated manner.
Detection of the above characteristics from the geometric representation provides important information to the analysis process. If this information is used in a finite element preprocessor, better finite element models could be created. This type of information could also lead to the development of more automated finite element mesh generators. A mesh generator could be developed to adaptively select initial mesh topology and local mesh density, if length scales of constrictions and other shape characteristics of the problem domain are known. To the best of our knowledge these capabilities are lacking in currently available finite element preprocessors. In existing systems, a parameter is used to provide mesh gradation information to the mesh generator. During the mesh generation, the systems use either default values or data provided interactively by the user [Cavendish 85], [Baehmann 87].

A special case illustrating these concepts, may be seen in the finite element model of Figure 2-1 [Chae 88]. Using this automatic finite element preprocessor developed by [Chae 88], the analyst first interprets the physical domain and identifies significant points or regions. Then making use of this information, the analyst interactively subdivides the shape into a set of "simpler" subdomains and inserts so called "key nodes" along the boundary edges of the subdomains. As can be seen from Figure 2-1, nodes close to the holes and constrictions are densely placed, whereas, on the other portions of the boundary, node spacing is coarse. After the above interactive node insertion step, this mesh generator automatically creates a finite element discretization, (see Figure 2-2). Consequently, the above node insertion process determines local mesh density and size of individual finite elements in the mesh by directly employing the analyst’s interpretation of the problem domain.
Figure 2-1: A Complex Planar Shape and its Decomposition, (Adapted from [Chae 88])

Figure 2-2: Finite Element Mesh of the Planar Shape, (Adapted from [Chae 88])
In this work we study the question of automatic extraction of such information from
the geometric description of the physical domain with the objective of automating the
discretization process. If we could get quantitative geometric information, such as positions
and length scales of significant regions within the domain, before starting the mesh
generation process, we could not only carry out meshing in a more automated manner but
also create very effective models. Such a mesh, in turn, would give rise to more accurate
numerical results and this approach would further speed up the finite element analysis
(FEA) process. In this thesis, we investigate the medial axis transform (MAT) as an
interrogation tool to extract such geometric information, and implement a method to direct
the finite element mesh generation process.

In subsequent sections of this chapter we review general techniques to create finite
element models from solid models and various specific finite element mesh generation
methods.

2.2 Finite Element Models

Broadly speaking, all finite element models fall into one of two classes, either direct
models or indirect models [Shephard 85a], [Shephard 85b]. Direct models involve a clear
and direct correspondence between the geometric model and the finite element model. For
example, a solid may be analyzed with three-dimensional solid elements, (see Figure 2-3)
or a planar elasticity problem may be analyzed with two-dimensional elements. Indirect
models are those where there is not a direct or even obvious correspondence between the
geometric model and the finite element model. For example, the finite element model of a
ship hull typically consists of plate, shell, beam, and truss elements [Chen 83], [McVee 86].
These elements, (see Figure 2-4), are defined in terms of their center line properties (for
beam and truss elements), or by their middleplane properties (for plate and shell elements).
Creation of an indirect model requires idealization and interpretation of the physical domain
of interest by an analyst. As a consequence, indirect models involve a larger degree of geometric and structural idealization than direct models. Consequently, creation of indirect models is a complex task, which is more difficult to automate and, at present, requires interactive user input to a computer aided design (CAD) system. An algorithmic approach assisting in the creation of such models would make the FEM a more readily usable tool during early stages of the design process.

![Figure 2-3: A Direct Finite Element Model](image)

### 2.3 Integration of Solid Modeling with Finite Element Modeling

The amount of time and effort required to develop the actual finite element models needed as input to general purpose finite element analysis codes can be extremely large [Shephard 85b]. In recent years, this difficulty is being reduced by integration of solid modeling systems with finite element preprocessors. In most of the systems implemented, the user is allowed to directly pass basic geometric information to the finite element modeler in an interactive way. This level of integration allows the generation of finite element models without redefining the entire geometry. However, this type of integration
Figure 2-4: Indirect Models of Several Structural Components

does not take full advantage of the information available from a geometric modeler. It is only with the complete integration of solid model representations and finite element modeling techniques that automatic finite element model generation becomes possible [Shephard 85a].

Direct finite element model generation consists of two tasks. The first is the creation of finite element mesh which discretizes the object domain, and the second is the specification of analysis attribute information which specifies conditions associated with the physical problem. On the other hand, indirect finite element model generation first involves simplification and idealization of the object domain and then generation of the
mesh and specification of the analysis attribute information. The effective integration of solid modeling with finite element modeling places a number of demands on the solid model representation and the associated solid modeling functions. Since the generation of a finite element model is, to a major degree, a geometrically based process, it requires all the geometric manipulation and definition capabilities normally found in a solid modeling system.

The full integration of geometric and finite element modeling requires the ability to interrogate, and modify the geometric representation of the object being modeled. This integration can be achieved through a set of geometric communication operators. The function of these operators is to communicate information between the geometric modeling and finite element modeling systems. For purposes of finite element modeling, four types of geometric communication operators can be introduced [Shephard 85a], [Shephard 85b] to carry out the following main tasks:

1. Generation of a finite element mesh when direct finite element types are used.
4. Simplification of a geometric representation for purposes of finite element analysis.

Generally speaking, the above geometric communication operators deal with the boundary of the object. This is because the finite element modeling process is dominated by operations that deal with the boundary of the object. The majority of the finite element mesh generators work from the boundary into the interior, whereas attribute specification is primarily concerned with the specification of conditions applied on the boundary. Therefore, a boundary file is needed as an input to the mesh generator. This statement does not mean that the actual geometric representation is necessarily a Boundary Representation. This file may be either the primary representation of the object, or one derived from the primary representation by boundary evaluation [Weiler 86].
Of the four sets of operators identified above, the first three sets of operators are expected to be at a minimum necessary to create direct and indirect finite element models. In [Shephard 85a] detailed discussion and specification of the geometric communication operators that can be used in various automatic mesh generation approaches are presented. Many of the operators in these three major groups have similar or even the same functions as the operators used to create solid models with non-two-manifold topologies, [Weiler 86].

The complexity of these operators depends on the type of the automatic mesh generator being integrated with the solid modeler. As we discuss in the next section some automatic mesh generators only interrogate geometry. On the other hand, some also modify geometry in an incremental manner during the mesh generation process. Mesh generators based only on geometry interrogation approach appear to require less complex and more efficient geometric communication operators than those mesh generators based on both geometry interrogation and modification approach. Also interrogation of geometry, in general, usually takes much less computation time than geometric modifications. Therefore, mesh generators requiring only geometric interrogation are likely, in general, to be more efficient.

2.4 Review of Finite Element Mesh Generation Methods

Finite element mesh generation is concerned with the subdivision of a geometric entity, such as a curve segment, a surface patch or a volume into a set of geometrically simple shapes referred to as finite elements. This subdivision process must be controlled to ensure

- the accurate representation of all significant geometric characteristics of the problem domain by the mesh;

- the proper matching of geometric features between finite elements; and,

- that the size and distribution of elements throughout the domain being meshed satisfies specific requirements.
Over the last twenty five years various mesh generation schemes have been developed, but none has achieved general applicability for finite element meshing of complex geometries. One reason is that most existing mesh generators require a large amount of interactive user input. Another reason is that although some mesh generators, in general, create meshes with good shape characteristics, they occasionally generate meshes of poorly shaped elements or even generate an unacceptable mesh in some regions. Consequently, automatic finite element mesh generation is an active research problem in the CAD area. In the author’s opinion there are substantial opportunities for increased automation, and higher process reliability and efficiency. As indicated above, mesh generators can be grouped into two broad classes based on their interaction with geometric representation:

1. Geometric interrogation approach, in which mesh generators operate only by interrogating the original geometric representation.

2. Geometric interrogation and modification approach, in which mesh generators operate by both interrogating and incrementally modifying the geometric and topological representation of the object during the meshing process.

We can also classify existing mesh generators depending on their underlying algorithmic approaches:

- Mapping mesh generation;
- Point insertion followed by area / volume triangulation;
- Topology decomposition;
- Spatial decomposition; and,
- Recursive subdivision of a region to element level.

Mesh generation schemes based on geometry interrogation approach include mapping mesh generation and spatial decomposition. Meshing schemes using geometry interrogation and modification approach include the techniques of topology decomposition, recursive subdivision and some forms of point insertion followed by area triangulation.
2.4.1 Mapping Mesh Generation

Mesh generators based on mapping techniques were the first mesh generators to be implemented. They, in general, make use of a set of functions to map a given geometry into a simple geometry. Then this mapped region is meshed using a uniform grid and all the grid points are mapped back to the original geometry using mapping functions.

Various mesh generators have employed different mapping functions. We can classify those mesh generation schemes on the basis of the mapping approach used. Some mesh generation schemes and related mapping techniques are as follows:

- meshing based on isoparametric shape functions, [Zienkiewicz 71], [Imafuku 80], [Wellford 88];
- meshing based on multivariate blending functions, [Gordon 73], [Cook 74], [Cohen 80], [Haber 81], [Haber 82], [Cavendish 84], [Crawford 87]; and,
- Laplacian meshing, [Herrmann 76], [Lorensen 80], [Thomson 85].

The mapping techniques based on isoparametric shape functions are a subclass of the mapping techniques based on multivariate blending functions. The mesh generators using the Laplacian approach, first, generate an initial set of nodes in the domain using a simple mapping technique (such as isoparametric mapping). Then, the final mesh is computed by using the Laplacian relaxation technique. In the relaxation process, a set of difference equations governing the coordinates of the nodal points is solved using an iterative scheme such as the Gauss-Seidel method.

As an example to illustrate the mapping mesh generation approach based on isoparametric shape functions, a quadrilateral curved surface patch in the xyz reference frame can be mapped onto a parameter space square in the $\xi \eta$ plane with center at the origin, sides parallel to the $\xi \eta$ axes and equal to 2 using the following mappings [Zienkiewicz 71], (see Figure 2-5)


Figure 2-5: Mapping Mesh Generation

\[ x = \sum_{i=1}^{8} N_i(\xi, \eta) x_i \]  
\[ y = \sum_{i=1}^{8} N_i(\xi, \eta) y_i \]  
\[ z = \sum_{i=1}^{8} N_i(\xi, \eta) z_i \]

(2.1)  
(2.2)  
(2.3)

where \( x_i, y_i, \) and \( z_i \) are the Cartesian coordinates of the eight nodes defining the quadrilateral patch in the \( xyz \) reference frame and \( N_i(\xi, \eta) \) are isoparametric shape functions associated with each node.

\[ N_1(\xi, \eta) = \frac{1}{4} (1 - \xi)(1 - \eta)(1 + \xi + \eta) \]

e.g. 

\( \xi \) and \( \eta \) have values ranging from 1 to -1 on opposite sides the parameter space square. In this particular case, \( N_1 \) is associated with the node which is mapped to the point (-1, -1) in the parameter space. The isoparametric shape functions are 1 at the node associated with their index and zero at all other nodes. They also sum to 1 for all \( \xi \) and \( \eta \), thereby giving a representation invariant under rigid body motions.
In the mesh generation process, a uniform grid is generated within the parameter space square in $\xi \eta$ plane. Each grid point, $P_{kl}$, has particular $\xi_k$ and $\eta_l$ values and corresponds to a unique vertex on the quadrilateral patch in the xyz coordinate frame, (see Figure 2-5). For each grid point, the $x$, $y$ and $z$ coordinates of the vertex in the three-dimensional real space are determined by substituting the corresponding $\xi_i$ and $\eta_j$ values into equations (2.1), (2.2) and (2.3).

This mapping scheme can be extended to three-dimensional mesh generation to create finite element meshes within volumes. In this case, a region in the three-dimensional space is mapped into a parameter cube in the parameter space using a set of mapping functions.

Mapping mesh generation schemes, in general, create uniform meshes. These meshing schemes usually do not allow meshes with variable node distribution within the domain. The mesh generators proposed by [Imafuku 80] and [Wellford 88] have attempted to rectify this limitation. The mapping mesh generators create well controlled meshes within regions with simple topology, such as triangular and quadrilateral surfaces and hexahedral volumes. But they also have a severe disadvantage. They require that the region to be meshed should be decomposed into a set of such mappable regions. Decomposition of a general domain into a set of simple subregions is a very complex task in its own right. At present, this decomposition process is usually carried out by the user. When the user defines a valid set of mappable subregions, these subregions explicitly provide all the geometric information required for the meshing process. Since there is a one-to-one correspondence between a subregion and its map, a mesh generated by this approach guarantees that all elements are within the given region and there are no overlapping elements. Therefore, these mesh generators need not check validity of individual elements during the mesh generation process. As a result, they are computationally the most efficient in meshing simple regions.
We can also include here the mesh generators of [Brown 81], [Baldwin 85] based on conformal mapping. One disadvantage of this approach is that it is inherently limited to the discretization of two dimensional domains.

2.4.2 Point Insertion Followed by Area / Volume Triangulation

In this approach, the generation of the element mesh is carried out in two distinct steps. In the first step, a set of points are inserted throughout the domain, and in the second step, the points are triangulated into a finite element mesh. The point insertion process determines the desired local mesh gradations. There are several mesh generators, different in their algorithmic approaches, which make use of these two distinct operational steps.

Frederick, et al, [Frederick 70] have developed a two-dimensional mesh generator. In this approach, a mesh in a two-dimensional domain is generated surrounding all nodes by triangles. This mesh generation is based on the fact that given a set nodes inserted in the domain, a mesh can be created by completely surrounding every node using triangles, (see Figure 2-6). The meshing process starts by selecting the nearest point to a given point to be

![Diagram](image)

**Figure 2-6:** Surrounding Procedure Around a Node, (Adapted from [Van-Phai 82])

surrounded. A sequence of triangles in a specific order (e.g. counter-clockwise sequence)
around the starting point are created in such a way that they satisfy some shape conditions. During this process, a potential new triangle is checked to make sure that it does not overlap any existing triangle. This scheme requires that sufficient additional nodes outside the domain are also inserted in order to successfully surround all nodes on the boundary. At the end of the mesh generation process, the elements associated with the outside nodes are deleted.

Van-Phai [Van-Phai 82] has extended the above technique to three-dimensional meshes. In the three-dimensional case, a line bounded by two nodes is analogous to a point in two-dimensions. Namely, a line segment is surrounded by tetrahedron elements in a prescribed order, (see Figure 2-7). Given a line connecting two nodes, a base triangle is constructed by connecting the ends of the line to a nearby node which allows creation of this base triangle with good shape characteristics. Next, using this triangle and searching through the node set, a node is determined to form a tetrahedron. In this process, the new node is picked in such a way that the resulting tetrahedron satisfies some shape requirements. Then, assuming the triangular face of the tetrahedron, which passes through

![Diagram showing surrounding procedure around a line](https://example.com/diagram.png)

**Figure 2-7: Surrounding Procedure Around a Line, (Adapted from [Van-Phai 82])**
the starting line and the node determined last, to be the base triangle, another tetrahedron is
created. This process continues until the starting line is completely surrounded by a set of
tetrahedra. This meshing technique also requires outside nodes to be inserted. After the
complete mesh is constructed the elements associated with the outside nodes are deleted.
The use of outside nodes requires that around the sharp corners associated with acute
interior angles the surface should be divided into small faces in order for the scheme works
correctly. Another disadvantage of this approach is that it extensively searches through the
initial node set to determine optimum points which allow creation of tetrahedra with certain
shape properties and checks interference between a new element and other elements already
generated. In order to reduce the number of searches during the mesh generation process,
Van-Phai has suggested that the domain should be subdivided into a set subdomain before
the mesh generation. This process is interactively carried out by the analyst. Thus it
significantly reduces the effectiveness of this meshing scheme.

Lee, et al, [Lee 84] have developed a two-dimensional meshing scheme capable of
generating quadrilateral and triangular elements. This novel mesh generator is based on
CSG representation techniques. The mesh generator uses a CSG representation of the
domain and mesh density information. In the CSG representation, only rectangular
primitives are included. Therefore, the mesh generator is limited to polygonal domains. In
the node insertion step, points are uniformly inserted into primitive subdomains, which is a
relatively simple task. After the primitives are combined using boolean set operators, the
nodes within the overlapping regions are processed. In this process, nodes are either
retained or moved or deleted in order to satisfy a given mesh density requirement. In the
mesh generation process, each node is systematically visited and other neighboring nodes
are searched. Elements are generated by connecting appropriate neighboring nodes using
line segments. The scheme employs a decision making tree to generate properly shaped
elements which are either quadrilaterals or triangles. The authors have pointed out that the
use of the decision tree is a factor increasing the complexity of the mesh generation process.
Lo [Lo 85] has developed a two-dimensional mesh generation scheme with a simple but effective node insertion method. In the node insertion step, a set of horizontal parallel lines spanning the vertical dimension of the domain are drawn across the domain. The lines are uniformly spaced and the distance is equal to the average dimension of the triangular elements. Then, the intersections between the lines and the domain are determined. New nodes with uniform spacing are inserted along the intersection segments. Before the mesh generation process, all nodes are sorted into a boundary node set and an interior node set. The boundary nodes are ordered and a counter-clockwise sequence of line segments connecting them are created. In the element generation step, for a given boundary segment a node is determined from either boundary node set or interior node set. A triangle is generated by connecting the node with the end of the segment. The criteria used in selecting a node among all available interior and boundary nodes are that the node must lie to the left of the directed segment, and that the triangle has to satisfy some conditions on shape characteristics. Every time a new element is generated the sets of interior and boundary nodes are updated. This process continues until these sets become null.

In recent years, research efforts to develop mesh generation procedures based on node insertion followed by triangulation approach have employed two mathematical constructs of Computational Geometry: Dirichlet tessellation and Delaunay triangulation [Preparata 85], which are used sequentially, (see Figure 2-8). Mesh generation schemes based on this approach were developed by [Cavendish 85], [Joe 86a], [Field 86], [Sloan 87] and [Schroeder 88].

In order to introduce some necessary definitions, let us consider a two-dimensional case with a set of points \( p_1, p_2, \ldots, p_n \) in the plane \( R^2 \). We can define the sets \( V_i, 1 \leq i \leq n \), as

\[
V_i = \{ x : \| x - p_i \| < \| x - p_j \| \quad \text{for all} \quad j \neq i \}
\]  

(2.4)

where \( \| \cdot \| \) denotes Euclidean distance in \( R^2 \). \( V_i \) represents regions of the plane whose points are nearer to point \( p_i \) than to any other point in the given point set. Sometimes, \( V_i \) is also
called the influence region of point \( p_i \). \( V_i \) is an open convex polygon called Voronoi polygon whose boundaries are portions of the perpendicular bisectors of the lines connecting point \( p_i \) to point \( p_j \), when \( V_i \) and \( V_j \) are touching each other. The collection of
Voronoi polygons is called the Dirichlet Tessellation. In general, at every corner vertex of the boundary of a Voronoi polygon there are two other adjacent Voronoi polygons. Therefore, we can construct a triangle, $T_k$ connecting the points associated with such adjacent polygons. The set of these triangles, $\{T_k\}$ is called the Delaunay triangulation. This construct is a triangulation of the convex hull of the points in the plane.

The basic property of a Delaunay triangulation in two dimensions that makes it appropriate for use in mesh generation is that its triangles are as close to equilateral as possible for the given set of points. Another fundamental property of a Delaunay triangulation is that there are no points inside the circumscribing circle of a triangle in two dimensions and no points inside the circumscribing sphere of a tetrahedron in three dimensions, [Preparata 85]. This property ensures that well shaped elements, in comparison to an equilateral triangle, are generated in two dimensions. Similarly in three dimensions, the faces of Delaunay tetrahedra are also as close as possible to equilateral triangles. But this is not sufficient to ensure well shaped tetrahedral elements. As an example, given four nodes at the corners of a square in three dimensions, this type of mesh generators can create a tetrahedron with zero volume using these nodes.

There are a number of techniques for the construction of a Delaunay triangulation. The algorithm proposed by Watson [Watson 81] is a commonly used approach. This algorithm exclusively uses the basic properties of the Delaunay triangulation mentioned above. The mesh generator [Cavendish 85] which is based on Watson’s algorithm constructs the mesh by a node insertion procedure.

This approach consists of four major steps [Field 86]. In the first step, a triangle (tetrahedron) which encloses all the given points is created. In the second step, the Delaunay triangulation of the vertices of the enclosing triangle (tetrahedron) and one of the given points is formed. This task is simply carried out by connecting the point to each vertex of the enclosing triangle (tetrahedron). Also a master list of triangles (tetrahedra)
and their associated circumdisks (circumspheres) is created. In the third step, the other
given points are inserted into the triangulation one at a time. This point insertion process
involves the following tasks

1. The existing circumdisks (circumspheres) containing the point to be inserted
   are determined.

2. A list of triangles (tetrahedra) associated with the circumdisks (circumspheres) identified in the previous step is created. The union of these triangles (tetrahedra) is a polygon (polyhedron) in which the new point will be
   inserted and new elements will be created.

3. A list of boundary edges (triangles) of the insertion polygon (polyhedron) is
   created.

4. New triangles (tetrahedra) filling the insertion polygon (polyhedron) are
   created by connecting the new point to the vertices of the insertion polygon
   (polyhedron).

5. The triangles and circumdisks (tetrahedra and circumspheres) determined in
   the first two tasks are deleted from the master list.

6. All triangles (tetrahedra) and their circumdisks (circumspheres) created in the
   third task are added to the master list.

Finally in the fourth step, all triangles (tetrahedra) which share a vertex with the enclosing
triangle (tetrahedra) are deleted from the master list. At the end of this process, the union
of the remaining triangles (tetrahedra) is a Delaunay triangulation of the initial points.

It is important to note that the triangulation generated by a Delaunay process
represents the convex hull of the points used. To guarantee that the last step of the meshing
process does not remove part of the convex hull of triangulation points, the bounding
tetrahedron and the triangulation points must be determined carefully. A major difficulty of
Watson’s algorithm is encountered during the correct classification of a newly inserted
point whether it is inside, outside or on a circumsphere of the current triangulation. This
problem is amplified if the distance from a newly inserted point to a circumsphere is less
than the precision of computation. In such a situation, wrong decisions will be made during
the creation of insertion polyhedron. Consequently, an unacceptable mesh containing
overlapping tetrahedra or gaps may result. Another problem in three-dimensional mesh
generation based on this approach is that poorly shaped very thin tetrahedra (so called slivers [Cavendish 85]) might be created during the meshing process. It has been pointed out that in three-dimensional mesh generation slivers can account for as much as 10 percent of the total number of elements generated [Cavendish 85]. In three-dimensional mesh generators, such degenerate elements are treated separately. Badly shaped elements are identified by carrying out shape measure computations and removed from the mesh if possible [Cavendish 85], (see Figure 2-9). Otherwise, they are opened up by repositioning one of their nodes, (e.g. node D in Figure 2-9), to an appropriate position so that they are no longer zero volume elements.

![Diagram of slivers: removable sliver and non-removable sliver.](image)

**Figure 2-9: Zero Volume Elements (Adapted from [Cavendish 85])**

The development of algorithms for the insertion of points such that the desired mesh gradations are obtained and poorly shaped elements are avoided is an important part of mesh generation schemes based on the node insertion and area / volume triangulation approach. Several schemes have been developed for the node insertion process of mesh generators based on the Delaunay triangulation technique.

Cavendish [Cavendish 74] has proposed a semi-automatic node insertion process for
two-dimensional domains. Given a multiply connected domain, it is interactively subdivided into a set of zones. Each zone is also associated with a mesh density parameter. Using the mesh density parameters, nodes are, then, automatically generated on the boundaries of the domain and within the zones. To generate nodes within a zone, a uniform square grid is superimposed in the zone. The mesh density parameter determines the spacing of the uniform grid. For each square cell of the grid, a random node is generated. If the node is within the domain and satisfies the mesh density requirement, it is accepted. Otherwise, a finite number of attempts are made to generate an acceptable node within the cell. If those attempts fail no node is generated within the cell. The three-dimensional mesh generator developed by Cavendish, et al. [Cavendish 85] makes use of this node insertion technique. In this scheme, a volume is, first, interactively intersected by a set of parallel cut planes. On each intersection plane, nodes are generated by using the two-dimensional node insertion process summarized above.

Frey [Frey 87] has developed an automatic node insertion strategy for two-dimensional domains. This approach uses the Delaunay triangulation technique to insert new nodes. In this scheme, the user specifies a node spacing function. A set of nodes are, first, inserted along the boundaries of a two-dimensional domain. A triangular mesh is, then, created using the initial boundary nodes with the Delaunay triangulation technique. This mesh, in general, contains many triangles with unacceptable shape characteristics and does not satisfy the node spacing requirement. Those triangles are identified and a new node is determined within such a triangle to satisfy node spacing condition. A new node and a set of new elements are added to the mesh using the Delaunay triangulation technique. This iterative process continues until the prescribed node density requirement is satisfied in the domain. Since this technique involves many searches and interrogations to determine locations of nodes to be inserted, it is expected to require substantial amount of computation time for three-dimensional cases.
2.4.3 Topology Decomposition

Mesh generation schemes in this group operate by removing individual subregions from the domain one at a time until the domain is reduced to the null set. There are two distinct types of mesh generators in this category. Some of these algorithms remove individual finite elements from the domain one at a time [Wordenweber 84], [Woo 84] and [Chae 88]. Others remove larger but simple portions of the domain and then triangulate these individual pieces using a different procedure [Bykat 76], [Sadek 80] and [Joe 86b].

Topology decomposition mesh generation schemes removing individual elements from the domain typically employ a Boundary Representation model of the domain. They operate by searching for topological entities (i.e. vertices, edges and faces) that satisfy a set of connectivity and geometric requirements. Every time such an entity is removed from the domain, a new element (a tetrahedron in three dimensions and a triangle in two dimensions) is generated, (see Figure 2-10). Mesh generators based on this approach employ a number of operators each of which is used to remove a distinct type of topological entity. These operators carry out the same functional tasks as some Euler operators creating Boundary Representation models.

One entity from the set of topological entities that satisfy the given requirements is removed by an appropriate operator. At the same time, topology and geometry of the Boundary Representation model is updated. The process of looking for and removing a new element is then again applied to the domain remaining until it is reduced to a single element.

Since the amount of computation required for application of each removal operation is high, these mesh generation schemes are not computationally efficient for the creation of a fine mesh. However, these procedures can be effectively used in a two-step approach. First, the procedures remove large pieces of the object. Then, those pieces can be quickly filled with elements by using more efficient techniques, such as mapping methods.
Figure 2-10: Element Creation Based on Topology Decomposition

The second type mesh generators of this group, which deal with triangulation of two-dimensional domains, create meshes by means of two processes. The first process extracts a large subregion from the domain and the second process generates a triangular mesh within the subregion. In the meshing schemes of [Bykat 76], [Sadek 80], a set of so called "boundary layers" with a width equal to the maximum element dimension are removed from the domain one at a time and then triangulated. The mesh generator of [Joe 86b] first decomposes a two-dimensional polygon into a set of simply connected regions. Then, it reduces these regions to convex polygons which are triangulated using a quasi-uniform triangulation [Joe 86a]. Although this meshing scheme may be regarded to belong to the topology decomposition approach, the technique is efficient. The reason is that it aims at rapid triangulation of large subregions of the original domain.
2.4.4 Spatial Decomposition

The mesh generation schemes based on the spatial decomposition approach include modified forms of the quadtree in two dimensions [Yerry 83], [Bachmann 87], (see Figure 2-11), and the octree in three dimensions [Yerry 84], [Yerry 85] and [Kela 86]. The basic steps involved in the mesh generation process are summarized below for the three-dimensional case.

![Quadtree Representation](image)

Figure 2-11: Modified Quadtree Representation of a Circle, (Adapted from [Yerry 83])

The octree representation uses a recursive subdivision of the space of interest into eight octants. An object is represented as the union of a set of disjoint cubes of various sizes. These cubes are derived from recursive subdivision of parent cubes into eight octants. The entire structure is stored in a hierarchic tree with eight branches at each tree node. Each node consist of eight pointers to the tree branches and a code indicating the classification of the octant whether it is wholly inside or outside of the object or undetermined. If the code associated with a node is wholly inside or outside, the node is a leaf and the pointers are null. When a octant is classified as undetermined it is subdivided into octants which are pointed by the pointers of the octant's node. This subdivision process continues until some minimal resolution level is reached.
A mesh generator based on the spatial decomposition approach operates in two stages. The procedure generates a spatially addressable finite element mesh embedded in the lowest level of a hierarchical grid. The first stage of the meshing process involves generation of an interior mesh. On the other hand, the second stage deals with generation of a mesh between the boundary of the object and the boundary of the interior mesh.

To generate the interior mesh, first the object is enclosed in a box. The box is recursively subdivided into octal cells until a prescribed level of subdivision is reached. The smallest cell size determines the element size or mesh density. Following each subdivision step, cells are classified as being inside or outside the object or undetermined. Cells classified as inside of the object at higher levels in the hierarchy are subdivided to the final level without further classification in order to reduce the subdivision difference between neighboring cells. At the end of the decomposition process, the collection of cells classified as inside of the object constitutes the internal mesh of the object. Individual cells in this mesh are directly used as hexahedral elements and cell vertices become nodes of the finite element mesh.

In the second stage, the region between the boundary of the object and the boundary of the interior mesh is filled with either hexahedral or tetrahedral elements. Three main tasks involved in this process are to systematically visit all undetermined cells on the boundary of the object, to insert nodes on the surface of the object and to associate those boundary nodes with the nodes on the boundary of the interior mesh. The undetermined cells on the boundary can be visited by tree traversal. In the node insertion process, boundary vertices (i.e. corner nodes of the faces on the boundary) contained by undetermined cells are identified to be used in the boundary mesh. Also a set of nodes are generated by determining intersections of undetermined cells with the boundary of the object. After that, the boundary mesh creation process starts. If a cell contains no corner nodes it is linked to the interior mesh to generate a hexahedral or a tetrahedral element. If
an undetermined cell contains a corner node, the element generation process is more complex. In this case, the corner node is connected to the nearest nodes on the boundary and on the boundary of the interior mesh. Thus a set of tetrahedral elements are generated. At the end of the second stage, the finite element mesh of the object is completed.

The finite element mesh generated by this approach inherits the spatial addressability and the structure of the hierarchical grid due to the fact that elements in the mesh are associated with the octants of the spatial decomposition.

The spatial decomposition methods have two main limitations in finite element mesh generation. The first limitation is that they generate meshes that are dependent on orientation and position of the initial enclosing box. The second limitation is that although, in the interior mesh elements are uniform and have good shape characteristics, shape characteristics of elements in the boundary mesh degrade.

2.4.5 Recursive Subdivision

The recursive subdivision mesh generators, [Schoofs 79], [Sluiter 82], [Bykat 83], repeatedly split a domain into subdomains until the individual parts are single elements, (see Figure 2-12). As in the topology decomposition methods, this type of mesh generators use a Boundary Representation of the domain to be meshed.

In the two-dimensional meshing scheme of [Bykat 83], planar convex regions are recursively triangulated. In the case of a non-convex region, the domain should be processed and decomposed into a set of convex regions. Given a convex region, a set of nodes are inserted along its boundary. Then the region is roughly split into two equal convex parts. Next, nodes satisfying mesh density requirements are inserted along the splitting edge of the convex parts. This splitting process is recursively applied to two halves until each half is reduced to a single triangular element.

In the three-dimensional mesh generator of [Sluiter 82], a recursive algorithm which
Figure 2-12: Recursive Subdivisions, (Adapted from [Shephard 85a])

is similar to the above scheme and presented in [Schoofs 79] is used to generate meshes composed of quadrilateral and / or triangular elements on surfaces. The generation of a mesh in a three-dimensional volume involves the following steps. First, meshes are generated on the individual faces of the domain using the surface mesh generator. Then a loop on the surface of the domain is created by using the edges of the elements on the surface. This loop is the boundary of the splitting face (in general non-planar). A mesh is generated on the splitting face again using the surface mesh generator. The splitting face is duplicated and these two faces are added to both halves of the domain. Thus the domain is divided into two subdomains. This process continues until all subdomains are reduced to individual elements. In the case of a multiply connected domain, more than one loops are required to divide the domain into subdomains.
This mesh generation method similar to the topology decomposition scheme, first, searches for a geometric features (i.e. an appropriate splitting edge in a two-dimensional region or splitting face in a three-dimensional volume) and then modifies the geometry and topology of the region being meshed by using the feature found. These two operations are the key aspects of this meshing scheme and repeatedly applied during the mesh generation process. Given a complex multiply connected volume, the computational complexity involved in these operations during the mesh generation process reduces the efficiency and robustness of this scheme. Therefore, interactive user input to prepare the region to be meshed for the mesh generation process is an essential part this mesh generator [Sluiter 82]

2.4.6 Time Complexities of Some Mesh Generation Schemes

Usually the amount of computation needed to generate a mesh of a few thousand elements for a general three-dimensional geometry may be of the same order of magnitude as a linear analysis carried out on that model. Regarding analysis of complex engineering structures such as ships and aircrafts, the amount of time needed to generate a mesh of a realistic model is of an order of magnitude of several man-months [McVee 86], [Shephard 85b]. Therefore computational efficiency of these procedures is very important.

The various algorithmic approaches exhibit different running time complexities. The approach with the greatest amount of theoretical results is the Delaunay triangulation. In the two-dimensional case, this approach exhibits an optimum O(nlogn) computational time where n is the number of nodes [Preparata 85]. In two dimensions, the number of elements is of the same order as the number of nodes. Computational results of an implemented three-dimensional algorithm [Cavendish 85] gave O(n^2) running time complexity for the mesh generation process excluding the node insertion process, where n is the number of nodes.

Topology decomposition mesh generators which generate meshes by removing
individual elements have worst case computational time complexity of order $O(n^2)$, where $n$ is the number of entities defining the boundary of the region. These mesh generators carry out extensive geometric interrogations to insure the validity of elements generated at every step. But for a coarse mesh, the number entities on the boundary is relatively small. Therefore this running time complexity is not a very significant limitation for coarse meshes.

The most efficient mesh generators implemented for the two-dimensional cases have linear time complexity. The mesh generator of [Bykat 83], which is based on recursive subdivision approach, has linear time complexity $O(n)$, where $n$ is the number of elements generated. The mesh generator developed by Joe and Simpson [Joe 86b] has linear time complexity $O(n)$, where $n$ is the number of elements. Although this mesh generator is based on topology decomposition approach we note that it aims at quick triangulation of large subregions extracted from a given domain. Mesh generators based on spatial decomposition approach [Yerry 83], and [Bachmann 87] also have linear time complexity $O(n)$, i.e. linear with the number of elements generated.

Finally mapped mesh generators have $O(n)$ time complexity. i.e. linear with the number of elements generated, and they are the most efficient methods for meshing of simple regions.

2.5 Summary of Limitations of Existing Meshing Techniques from Automation Point of View

Although the mapping mesh generation techniques are computationally efficient, they can not handle complex and multiply connected domains without user input. The meshing schemes based on the node insertion followed by area / volume triangulation approach are not computationally efficient in three-dimensional cases. They can create badly shaped
elements which require special treatment to generate an acceptable finite element mesh. In these meshing methods, the user has to interpret the problem domain, decompose it into appropriate subdomains and supply mesh gradation information. Mesh generation schemes using the topology decomposition approach are not computationally efficient for creation of fine finite element meshes. These mesh generators can create non-two-manifold topologies during meshing of multiply connected regions in three dimensions. Thus, they require more advanced data structures allowing the representation of non-two-manifold situations. Mesh generators based on the spatial decomposition technique can create badly shaped elements close to the boundary of a region. The resulting mesh layout depends on the position and orientation of the initial enclosing box used to generate the uniform grid. These mesh generators also need appropriate mesh gradation information which is usually provided by the user. The mesh generation scheme based on the recursive subdivision approach involves computationally complex operations. This mesh generation approach requires interactive user input in order for the meshing process to lead to an acceptable mesh of a complex multiply connected domain.

These observations indicate that available methods have, in general, many serious limitations in providing efficient, reliable and automated solutions of the finite element mesh generation problem for complex geometries. A possible solution to the general finite element meshing problem may be development of meshing schemes based on a hybrid approach. As suggested in the previous section, the topology decomposition approach could be used to automate the mapping mesh generation scheme. In such a hybrid scheme, topologically simple large portions of a complex domain could be extracted and a well shaped fine mesh would be generated fast within individual subregions. Thus we could develop an efficient and robust mesh generator.

In this thesis, we present such a novel hybrid mesh generation scheme which is also automatic. Our mesh generation scheme, first, uses a shape interrogation method to extract
global geometric characteristics and topologically simple subregions from a given complex domain. Those simple subregions are, then, triangulated to generate a coarse mesh. Based on our interrogation method, the meshing scheme can also automatically identify some geometric characteristics such as constrictions, holes and their length scales. Thus, a mesh capturing important geometric characteristics of a given domain can be created by our mesh generation scheme in an automated manner. A number of complex and diverse meshing examples presented in this thesis illustrate the effectiveness of our method.

2.6 Attribute Specification for Finite Element Model

The second task in the finite element model creation involves the specification of analysis attribute information. The specification of finite element attribute information is most commonly carried out using the actual finite element model. In this process, interactive three-dimensional computer graphics techniques are the most commonly used approaches in CAD systems, [Perucchio 82]. The drawback of this approach is that analysis attribute information is, then, not associated with the original geometric model, thus introducing an inconsistency into a complete, geometrically based, object definition. The attributes associated with a finite element analysis include:

- material properties
- loads
- boundary conditions
- initial conditions
- mesh gradation information
- analysis process information

Most of this attribute information is geometrically based, whereas a small portion is independent of the geometry. The specification of attribute information, which is geometry independent, is reasonably straightforward. The specification of attribute information which is geometry dependent involves a set of operators that interrogate and modify
geometric representation of a domain. For example, if an analyst wants to apply a distributed load on the surface of a model, the system, first, identifies face(s) adjacent to the location of the load, then a new face which is geometrically identical to the influence area of the applied load is created, and finally, the applied load is associated with this new face as attribute information.

A finite element attribute specification system is an integral part of an automatic finite element model generation system. Shephard [Shephard 88] has presented a basic overall structure for the specification of this form of information using a general geometry based approach within a CAD system. In this approach, the attribute information is directly tied to the geometry of an object.
Chapter 3

Medial Axis Transform as a Shape Interrogation Method

3.1 Introduction

Computer aided engineering systems promise to provide designers and engineers with a unified design, analysis and manufacture environment which allows increased automation of engineering activities. Typically, a computer model of a system is created using some geometric modeling technique such as the Boundary Representation method. Then, the model is analyzed to evaluate its performance under specified conditions. In this stage, idealization of the problem followed by discretization such as that based on finite elements and numerical solution of the resulting systems of equations are performed. These tasks are carried out in a design cycle until a satisfactory design is obtained. Finally, a manufacturing process is planned to create a physical manifestation of the design. At many steps of this complex process, designers and analysts need to interrogate the current model to extract information about various geometric characteristics. This information, in turn, is used to continue and complete the design and fabrication process. For example, solution of general arrangement and clearance problems, domain idealization for analysis, finite element discretization and numerical control tool path planning involve such interrogations, and the extraction of information about geometric characteristics.

In this thesis, we propose a method, based on the medial axis transform (MAT) which assists in the extraction of a number of geometric characteristics useful in engineering. This technique interrogates the geometric model of an object assumed to be available in a Boundary Representation form.

This chapter is organized as follows. Section 3.2 provides the basic definitions and reviews related literature and algorithms. Section 3.3 introduces and discusses some basic
properties of the MAT. Finally Section 3.4 discusses potential applications of the MAT technique.

3.2 Literature Review on the Medial Axis Transform

The MAT has been known for about twenty years, and has been used in shape representation and feature extraction in the field of digital image processing [Pavlidis 77]. On the other hand this concept has found little application in processing of continuous representations though as shown in this work it is an effective tool for such applications. This section reviews existing algorithms for the computation of medial axis transform of planar shapes.

Blum [Blum 67], [Blum 73], [Blum 78] has proposed the technique of MAT to describe biological shape. In this technique a two-dimensional shape is described by using an intrinsic coordinate system. Every point \( p \) on the plane containing the shape may be associated with a nearest point on the boundary contour \( B \). For defining the distance between an interior point and the boundary, we use the Euclidean metric. The Euclidean distance from a point \( p \) to the boundary set \( B \) is the distance from \( p \) to a nearest point \( P \) on \( B \),

\[
d(p, B) = \min \{ d(p, P) : P \in B \}
\]  

(3.1)

Such a nearest point exists because our shape boundary is a closed subset of the Euclidean space [Wolter 85a]. For a particular set of points the minimum distance is not achieved uniquely. Such points are equidistant from two or more points on the boundary contour. This set of points together with the limit points of this set constitute the medial axis or skeleton or symmetric axis of the shape (see Figure 3-1). We consider here only points in the interior domain bounded by \( B \). For example in Figure 3-1, we have the relationship for an interior point \( a \), \( d(a, B) = d(a, b) = d(a, c) \). This definition of the MAT which has been investigated also in [Wolter 85a] is equivalent to Blum’s definition of the MAT [Blum 73].
Blum defines the medial axis of a closed curve $B$ in the Euclidean plane to be the set consisting of the centers of all maximal disks which fit into the domain bounded by $B$. These two definitions are equivalent [Wolter 89]. The metric interpretation of the skeleton provides a natural basis for building a more complete description of the shape. On the skeleton $S$ of a boundary $B$, we define a function $r : S \rightarrow \mathbb{R}^+$ as follows, where $\mathbb{R}^+$ is the set of non-negative real numbers. For every $p \in S$

$$r(p) = d(p, B)$$

(3.2)

The function $r(p)$ is called \textit{radius function} or \textit{disk function} of the skeleton. According to Blum shape may be described procedurally by means of its skeleton and the associated radius function. Namely, given the skeleton and associated radius function of a figure we can exactly reconstruct it. It can be shown that given a skeleton we can uniquely recover the original shape by taking the union of all disks with radius equal to the radius function and centered on the skeleton [Duda 73] (see Figure 3-2). These and some other fundamental properties of MAT are discussed in Section 3.3.

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{medial_axis.png}
\caption{The Medial Axis of a Planar Shape}
\end{figure}

The concept of medial axis is related to the closest neighborhood problems in computational geometry [Preparata 85]. Given a set of elements $e_i$ in a region (e.g. points) we can associate every element $e_i$ with a particular subregion $r_i$ in which every point is
Figure 3-2: Reconstruction of a Shape Using its Skeleton and Radius Function

closer to that element $e_i$ than to all other elements of the set. These individual subregions $r_i$ are referred to as *Voronoi regions*. The region is the union of all Voronoi regions. The boundaries of the Voronoi regions comprise the *Voronoi diagram* of the given set. In essence, the Voronoi diagram of a set decomposes the region into a finite number of subregions. Construction of the medial axis can be referred to as the solution of the closest boundary point problem. As a matter of fact, the medial axis of a planar convex polygon decomposes it into a set of subregions each of which is the nearest neighborhood of a boundary edge. Thus, medial axis and Voronoi diagram are closely related. In the case of a convex polygon, they are identical to each other. If a polygon is non-convex, including *re-entrant corners* on its boundary contour, then the Voronoi diagram of the polygon is a superset of the associated medial axis. In this case, the difference between the medial axis and Voronoi diagram is that the medial axis does not include edges of the Voronoi diagram incident at the re-entrant corners (see Figure 3-3).

**Definition 1:** A *re-entrant corner* of a polygon is a vertex at which the internal angle is greater than $\pi$. If the angle is less than $\pi$ the corner is called convex vertex.

Montanari [Montanari 69] has developed and implemented an algorithm to exactly compute the continuous skeleton of a multiply connected planar polygonal shape. This method has been used for processing digitized images. In this algorithm, the boundary
contour is simulated as a "grass-fire wavefront" which propagates inward from the boundary. The algorithm basically computes distances of propagation and positions of so called branch points at which the topology of the wavefront changes. Branch points are also used in the present work and are defined later. Those points, in turn, are used in describing branches of the skeleton. The skeleton branches are bounded by the branch points and / or convex vertices on the boundary contour. The basic assumption of the algorithm is that the boundary contour is composed of straight line segments leading to a skeleton made up of straight line segments and arcs of parabolas. If n is the number of segments of the boundary contour, the computation of branch points in this algorithm

Figure 3-3: The Voronoi Diagram and the Medial Axis of a Planar Non-Convex Shape
exhibits a time complexity of $O(n^2)$ in the worst case. In the modification of the grass-fire wavefront, the algorithm has a time complexity which is proportional to $n$. Bookstein [Bookstein 79] has presented an approximate method to compute skeletons of simply connected polygonal shapes. This technique generates a discrete version of the skeleton. In this approach the skeleton is approximated by straight line segments which are bisectors of angles defined by non-adjacent boundary edges. This technique is more suitable to approximately compute the spine of an elongated shape.

There are several algorithms to compute Voronoi diagrams of polygonal figures. The common feature of these algorithms is that they use the "divide and conquer" paradigm, which reduces a problem of size $n$ to the solution of similar problems of integer sizes that are fractions of $n$. In this approach given a set, $E$, of $n$ elements, first $E$ is divided into two equal subsets, $E_1$ and $E_2$. This division process is recursively applied to each half of the set until it is reduced to simple cases whose solutions are known. During this subdivision process, a binary tree is used and its creation gives rise to $O(\log n)$ computation time. One of the main concepts of this approach is that solutions of simple cases are "merged" to obtain the solution of the original problem. The merging is essentially defined by a "merge curve". In this approach, Voronoi diagram of $E$ is divided into two parts by the merge curve and each part is obtained from the Voronoi diagrams of $E_1$ and $E_2$. Construction of such an algorithm with $O(n\log n)$ computation time requires that the merge curve should to be computed in linear time.

Preparata [Preparata 77] has proposed two algorithms to compute medial axes of simply connected convex and non-convex polygonal shapes. Time complexities of these algorithms for convex and non-convex simple polygons are $O(n\log n)$ and $O(n^2)$. Kirkpatrick [Kirkpatrick 79] has developed an algorithm to compute skeletons of simply connected convex and non-convex polygons. This approach is based on construction of the generalized Voronoi diagram of the polygon and has time complexity of $O(n\log n)$. Lee and
Drysdale [Lee 81] have presented an algorithm to computing the Voronoi diagram for a set of line segments and circles in the Euclidean plane with running time complexity of $O(n\log^2 n)$. Lee [Lee 82] has also proposed and implemented a second algorithm with running time complexity of $O(n\log n)$ for computing the medial axis of a simply connected polygonal shape. Fortune [Fortune 86] has presented a very different algorithm with running time complexity of $O(n\log n)$ for general Voronoi diagram problems. His method is based on a sweep-line technique in a transformed space. Srinivasan and Nackman [Srinivasan 87] have introduced an algorithm of time complexity $O(n(h+\log n))$ to compute Voronoi diagrams and skeletons of multiply connected polygons with $h$ holes. In this approach, Voronoi diagrams of individual boundaries are computed using an extension of Lee’s second algorithm. In this approach given a multiply connected polygonal figure, first the interior Voronoi diagram of the external loop and the exterior Voronoi diagrams of internal loops are computed. The Voronoi diagram of the polygon is then obtained by merging these Voronoi diagrams. Meshkat and Sakkas [Meshkat 87] have implemented the above algorithm, and proposed an approximate technique for the analysis of VLSI designs using Voronoi diagrams and associated skeletons.

The MAT, which underlies thinning algorithms, has been also widely used in processing and pattern recognition of digitized images. In general, thinning algorithms work by processing the boundary contour of a digitized image. In these algorithms, pixels on the boundary are classified. The pixels constituting the skeleton of a digitized image can be identified using a set of rules [Pavlidis 77], [Pavlidis 82]. Given a digitized image, first the skeletal pixels on the boundary contour(s) of the image are detected and the remaining pixels on the contour(s) are deleted. The procedure is applied similarly to the reduced shape and continues until the image is reduced to a form with single pixel width. The set of remaining non-deleted pixels constitutes the skeleton of a digitized image. With this technique, a digitized image is, basically, thinned layer by layer to its skeleton.
Recently, Xia [Xia 89] has developed an algorithm, which is different than thinning methods, to compute skeleton of digitized images. This approach makes use of Blum’s grass-fire propagation analogy to compute the skeleton.

For applications of the MAT, we cite the following work in shape description and feature recognition context. Various techniques have been proposed to describe shapes using the MAT for planar objects [Blum 78], [Heide 84], [Pizer 87], and also for three-dimensional objects [Boissannat 84], [Nackman 85].

A skeleton of the interior of a closed planar object is composed of a set of connected branches. In general, if the object is multiply connected such a structure is a graph, and individual branches of the skeleton are associated with different parts of a shape as their local symmetry axes [Wolter 89]. Therefore, skeleton branches separate the shape into simple segments. Thus the skeleton facilitates representation of a shape in a hierarchical manner in terms of its basic components. It can also be used as a shape analysis tool [Heide 84]. The MAT has been also proposed as a shape recognition technique [Calabi 68]. Serra [Serra 82] has presented a theoretical account of the MAT in a mathematical morphology context. As another application of the MAT, Samet [Samet 82] has adapted the medial axis transform concepts to the quadtree representation for image representation purposes. This novel data structure has been observed to enjoy favorable properties including reduction in storage requirements.

Blum [Blum 73] has suggested extension of the concept of MAT to higher dimensions. However, there is very little published work regarding the extension of MAT computation methods from two to three dimensions. O’Rourke and Badler [O’Rourke 79] have proposed an algorithm to decompose a three-dimensional object into a set of spheres of varying radii. A by-product of this algorithm is the generation of discrete points on the medial axis which are the centers of the spheres decomposing the object. Nackman [Nackman 82] has presented a formulation of the local differential geometry of the
boundary surface of an object in three dimensions based on differential properties of the medial axis and the radius function.

In this thesis one of our main objectives is to expand the range of coverage of earlier MAT algorithms from polygonal shapes to shapes involving a curved boundary. An exact solution for a multiply connected planar shape with boundaries made up of circular arcs with arbitrary radii and straight line segments, a geometry frequent in engineering, is presented in this thesis. In addition, we have also made a preliminary investigation of an approximate extension of the above algorithm to handle multiply connected planar regions bounded by non-uniform rational B-spline (NURBS) curves.

3.3 Basic Properties of the Medial Axis Transform

The medial axis transform captures the essence of shape. The skeleton and associated radius function of a shape provide an idealized, yet complete, description of the shape (see Figure 3-2). It is well known that a stick figure perfectly represents a human like object (see Figure 3-4). The skeleton and the radius function of a planar shape contain information on boundary curvature, its rate of change, flexure, and width properties of the shape [Blum 73]. The MAT preserves the symmetry and periodicity of shape. The skeleton of a symmetric shape is also symmetric with respect to the symmetry axis of the original shape (see Figure 3-5).

The skeleton of a simply connected planar shape has a tree-like structure [Wolter 89]. If the shape is convex, the skeleton and the boundary decompose the interior into a set of convex subregions. Each subregion is associated with one of the edges of the boundary (see Figure 3-6). Such a subregion represents the nearest neighborhood of the edge associated with the subregion. In this particular case the Voronoi diagram and the skeleton of the shape are identical, and Voronoi edges and skeleton branches are the same. If the shape is non-convex, the Voronoi diagram of the shape is a superset of its skeleton. In the
presence of re-entrant corners, we can associate each re-entrant corner with a particular subregion. Such a subregion is bounded by a portion of the skeleton and two line segments. These line segments are perpendicular to the boundary at the re-entrant vertex and intersect the skeleton, (see Figure 3-7). Thus the skeleton of a non-convex shape can be obtained from its Voronoi diagram by deleting the two Voronoi edges incident at each re-entrant corner.

In the case of a multiply connected planar shape the skeleton is a graph [Wolter 89]. The skeleton of a shape with a set of holes will have the same number of loops as the number of holes, each of which encloses a hole (see Figure 3-8).

Constrictions (narrow regions of the shape) can also be detected by the skeleton. A constriction is associated with a local minimum of the "thickness" of the region, and this fact is indicated by a local minimum of the radius' function of the skeleton in the vicinity of the constriction, (see Figure 3-7).
The skeleton of a free-form planar shape is composed of a set of curve segments. Let us assume that we have two planar curves $p$ and $q$ parametrically defined as

\begin{align}
  p : & x = p_x(u), \quad y = p_y(u) \\
  q : & x = q_x(v), \quad y = q_y(v)
\end{align}

We can define the medial axis of these curves using two disk functions, $D_p(u)$ and $D_q(v)$, with a variable radius $r$ in a Cartesian coordinate system, (see Figure 3-9), using the envelope theorem of differential geometry [Faux 79].

\begin{align}
  D_p &\equiv [x-p_x(u)]^2 + [y-p_y(u)]^2 - r^2 = 0 \\
  \partial D_p / \partial u &\equiv 0 \\
  D_q &\equiv [x-q_x(v)]^2 + [y-q_y(v)]^2 - r^2 = 0 \\
  \partial D_q / \partial v &\equiv 0
\end{align}
Figure 3-6: Decomposition of a Convex Shape by Medial Axis Transform

Here the envelope of $D_p(u)$ is defined by equations (3.5) and (3.6) and similarly the envelope of $D_q(v)$ is given by equations (3.7) and (3.8). The points satisfying the system of these four equations are equidistant from both curves by a variable offset distance $r$ and define the medial axis. The implicit equation of the medial axis can be obtained by eliminating the variables $u, v$ and $r$ from equations (3.5) through (3.8). The medial axis of two surfaces may be defined by a similar approach [Hoffmann 88]. As a special case, if the boundary is defined by a set of circular arcs and straight line segments then the skeleton branches are conic sections [Blum 73]. Definitions of conic skeleton branches are given in Appendix A.
3.4 Applications of the Medial Axis Transform

The MAT is an effective method to extract geometric characteristics. In this section, we identify several applications of the technique to various problems encountered in engineering design and manufacturing.

In a Boundary Representation, the data necessary to describe holes (internal loops in faces) are explicitly present in the data base of the model [Weiler 86]. But the extent of the holes in the region and the length scale of their proximity with respect to each other and the external boundary of the region is not readily available information in existing geometric models. The MAT can be effectively used to isolate holes and to extract such proximity information.

Since the skeleton of a shape is the local symmetry axis, it is at equal distance from the boundary on both sides. This basic property of the skeleton provides solutions to design and manufacturing problems involving constrictions, clearances and general arrangements.
Symmetry is a fundamental shape characteristic. Symmetry information and knowledge of presence of similar subcomponents in a shape can be employed effectively to reduce complexity and increase efficiency in many diverse engineering problems, such as in storage involving representations of patterns with symmetry [Tanimoto 81] and substructuring and use of super finite elements in a finite element analysis context [Kardestuncer 87]. For detection of symmetry various methods have been proposed [Davis 77], [Tanimoto 81] and [Wolter 85b], and the review article of [Eades 88] gives a detailed account of these techniques. The MAT may be used to extract not only symmetry information but also to discover similar subcomponents of complex shapes. Encoding skeleton information may be applicable in a multiresolution approach for shape and symmetry analysis [Pizer 87].
Figure 3-9: The Medial Axis of Two Planar Curves

The radius function associated with a skeleton branch can be used to determine constrictions at which the radius function attains a local minimum. The radius function determines the length scale (thickness) of such constrictions. Using this information we can determine proximity of various parts of the boundary of a shape with respect to each other. For example, this process allows computation of the local minimum of the distance between an internal and external contour. Such information allows direct solution of clearance problems in design and manufacturing.

If we consider the grass-fire analogue of MAT, we observe that the grass-fire ends at locations which correspond to a locally maximum distance from the boundary. Such points are called maximum thickness points (MTP). The radius function associated with such points gives the maximum local thickness of the region. This information is potentially useful in general arrangement problems in design of cluttered spaces as well as in manufacturing applications such as molding analysis.
The MAT has been extensively used in pattern recognition and image processing areas to analyze two-dimensional digital images. It has been pointed out that the MAT is extremely sensitive to disturbances of the boundary [Pavlidis 77]. Namely small perturbations on the boundary can give rise to significant changes in the skeleton. Many publications consider this to be a disadvantage of the MAT. Figure 3-10 illustrates a simple planar shape and two other figures with small perturbations on the boundary of the shape. Those boundary characteristics have distinct effects on the skeleton of the object. The sensitivity of the MAT to perturbation of the boundary shape could be exploited to compare objects to identify similarities and differences. It could also be used to classify different variations derived from a basic shape [Bookstein 79].

The Voronoi diagram decomposes a shape into a set of subregions (closest-point Voronoi regions or influence regions of the components of the boundary of a polygon). In two dimensions, each skeleton branch is associated with two adjacent subregions. The radius function associated with the skeleton branch defines the geometry of the two subregions. In a manner analogous to a CSG model, we can hierarchically represent such a decomposition and call it as the MAT tree, (see Figure 3-11). In this approach, the skeleton branches and the radius function represent shape primitives and the shape is defined as the union of the primitives. However in this representation, the shape primitives are, in general, free-form and more complex than the shape primitives of standard CSG models. As the tree is traversed in ascending order, the value of the radius function associated with each leaf of the tree (i.e. subregion of the shape) increases. In other words, the shape is represented by including shape characteristics into the MAT tree in an order starting from smaller characteristics and moving towards larger ones. Thus, this shape representation can be regarded as a multi-resolution approach.

Voronoi diagrams and regions, the skeleton and information regarding constrictions may be used to provide an elegant automated solution to general arrangement problems
Figure 3-10: Effect of Boundary Perturbations on the Skeleton of a Planar Shape

such as the design of geometry of distributive systems in cluttered spaces. For example, the above information can be used to find various paths within the region for the solution of arrangement problems of distributive systems such as piping.

The grass-fire analogue of the MAT and subregions defined by the Voronoi diagram of a planar shape may be used to plan and determine numerical control (NC) tool paths. For this purpose, Persson [Persson 78] presents a technique addressing simply connected shapes bounded by straight line segments and circular arcs. This approach is based on the
Figure 3-11: The Medial Axis Transform Tree of a Simple Planar Shape

determination of influence regions of the boundary edges of planar shapes and does not directly employ the MAT concept. However the MAT can be seen to be the fundamental underlying concept which allows us to extend his approach to multiply connected regions.

The medial axis and associated radius function may also be considered to be a shape representation method rather than a method of shape interrogation. Such a representation method would be similar to (but not exactly the same) shape description techniques based on generalized cylinders [Rosenfeld 86], [Pegna 87].
The medial axis and the associated radius function may be used to idealize shapes for structural analysis purposes. For example, an elongated slender object may be idealized as a rod using its skeleton, and the skeleton may be subsequently discretized to obtain an approximate finite element model of the object. Identification of an object as slender may be made by verifying that the ratio of the maximum of the radius function over a branch of the skeleton divided by the length of the branch is below an appropriate threshold. Therefore, the MAT can be used as technique to automatically create indirect models. The medial axis of an elongated object is shown in Figure 3-12. In this figure, the main branches of the skeleton are long in comparison to the associated radius function. The radius function directly provides us with the information to compute a good approximation of the bending, tensile and torsional stiffness properties of the structure. This approach leads to a first order solution of such complex analysis problems in an automated and efficient manner.

Finally, Voronoi regions, the skeleton and the associated radius function may be used to aid in automation of domain discretization for finite element analysis [Patrikalakis 88a], [Patrikalakis 88b], [Patrikalakis 89a]. Given a complex shape we may decompose it into subregions by means of the Voronoi diagram. As we see in this thesis, such subregions are easier to discretize in comparison to the full domain. Also the radius function can be used to automatically identify the relevant local length scales of the region and for the distribution of nodes. Since the MAT is capable of extraction of geometric characteristics, such as maximum thickness points and constrictions, and as shown in this work, it is possible to introduce a robust and automated process to select mesh gradation parameters in various parts of the region. Such a process can be viewed as the first iteration of an automatic mesh generation scheme. On the second iteration, the mesh should be locally refined depending on the boundary and loading conditions of the problem.

In this work, we develop a new finite element mesh generation scheme. This scheme
Figure 3-12: The Skeleton of an Elongated Shape

is based on the MAT technique and makes use of the concepts introduced in this section. The theoretical background and the implementation methodology of this mesh generation technique is presented in Chapter 5.
Chapter 4

Computation of the Medial Axis Transform

In this Chapter we present a methodology to compute the MAT of connected planar shapes bounded by closed curved boundaries. The boundary of a region (or shape) is defined by an exterior loop and one or more interior loops, if the region of interest is multiply connected. Each loop of the boundary is composed of an ordered set of boundary elements (curve segments and vertices). The algorithm developed for the MAT computation covers straight line, circular arc and general non-uniform rational B-spline boundary curves. Our method can be also easily extended to compute the MAT of the complement of a planar shape bounded by an arbitrary number of loops.

The algorithms and computational techniques presented in this and subsequent chapters have been implemented on a DEC Vax Station II GPX running under the Unix operating system. The figures associated with examples in this thesis have been directly captured from the CRT of a Silicon Graphics IRIS 3030 workstation driven by the other system.

This Chapter is organized as follows. The Section 4.1 presents our computational methodology for the MAT. The Section 4.2 deals with implementation aspects of the algorithm and presents the data structures used in our computer implementation and also some representative examples of the MAT computed by our algorithm. The Section 4.3 introduces some basic ideas potentially useful in extending our algorithm to trimmed curved surface patches and closed polyhedra.
4.1 A Computational Methodology for the Medial Axis Transform

The concepts developed by [Montanari 69] for planar polygonal shapes can be extended for the computation of the skeleton of the type of boundary geometry addressed in the present work. The boundary of a non-convex, multiply connected planar shape is defined by an exterior loop and a set of interior loops. The number of interior loops is equal to the number of holes in the shape, which are assumed to be disjoint.

Definition 2:

A loop is a union of a finite number of boundary elements which are ordered in such a manner that when the loop is traversed in the positive direction the interior of the shape lies to the right.

Definition 3:

An element of the boundary is either a re-entrant vertex, which is associated with material angle greater than \( \pi \), or a straight line segment, or a circular arc segment with arbitrary radius.

Line and circular arc segments are bounded by two end vertices. There are also two distinct types of circular arc segments. When we traverse a boundary loop in the positive direction, if a circular arc segment is traversed in the clockwise direction with respect to its associated circle it is convex. On the other hand, if a circular arc is traversed in the counterclockwise direction it is concave.

In the present work a free-form boundary curve (e.g. a Bezier or B-spline curve) is approximated within a prescribed tolerance using these three boundary element types. If the boundary contour of a planar shape is composed of re-entrant vertices, straight line segments and circular arcs, with arbitrary radii, then the skeleton of this shape, in general, consists of straight line segments and arcs of conics (i.e. parabola, ellipse, and hyperbola, see Figures A-1 through A-3), [Blum 73].
Definition 4:

The skeleton branch $S(e_i, e_j)$ of two boundary elements $e_i$ and $e_j$ is the locus of the points equidistant from $e_i$ and $e_j$.

Descriptions of conic skeleton branches and their parametric representations, useful for tracing purposes, are presented in Appendices A and C. The conic branches of skeleton sometimes degenerate to straight line segments or circles.

In our computational methodology we make use of the fundamental offset process with direction towards the interior of a region. This process is analogous to propagation of a grass-fire wave front towards the interior of a shape.

Definition 5:

The offset of distance $h$ of the boundary $B$ of a planar region $R$ is the envelope of the union of all closed circular disks of radius $h$, the centers of which are points of $B$.

This definition accounts for two curves on both sides of the boundary, inside and outside. We are interested only in the offset of the boundary in the interior of the shape.

We can also determine, in a relatively direct manner, the skeleton associated with the complement of a planar shape bounded by arbitrary number of disjoint loops. Given such a planar shape, we first enclose it in a box, (like for example a circle) whose dimension is several times larger than the largest length dimension of the shape. Thus we obtain a new region whose skeleton will be used to compute the the skeleton associated with the complement of the planar shape. In this approach, the boundary of the new region is made up of the enclosing box and the external loop of the shape which is represented as the interior loop of the box. We note that the boundary elements on the exterior loop of the shape should be traversed in the counter-clockwise direction so that the complement of the planar shape is to the right. After the MAT computation, we obtain the skeleton of interest
by eliminating the contributing skeleton branches of the enclosing box. To compute the exterior skeletons associated with interior loops of the planar shape, these loops are individually treated as the exterior loops of a set of simply connected disjoint regions.

Using the sign convention adopted, we observe that on an inward offset of the boundary loop, convex arcs shrink but concave arcs expand compared to the initial boundary shape. We also notice that a re-entrant vertex can be regarded as a degenerate case of a concave circular arc with zero radius, because such a vertex gives rise to a finite arc segment on the offset contour.

Given the boundary contour of a region, our objective for the computation of the MAT is to determine inward offset distances and the associated branch points at which the topology of the contour changes. These are so called effective offset distances and effective branch points.

During the course of the offset process, there are three distinct types of branch points, (see Figure 4-1) [Montanari 69].

Definition 6:

An initial branch point of a contour is a vertex at which precisely two non-adjacent elements of the offset contour are tangent to each other.

Definition 7:

An intermediate branch point of a contour is a vertex to which one or more elements of the non-vanishing offset shrink.

Definition 8:

A final branch point of a contour is a vertex which represents a vanishing offset contour.

A final branch point is, in fact, a special case of an intermediate branch point and indicates the end of the offset process. For a given boundary contour with n boundary
elements, if the number of the effective intermediate branch points is n then these branch points represent the final branch points of the contour. We observe some special cases involving final branch points. With a boundary contour composed of two circular arcs, offsets of the segments become tangent to each other at a final rather than an initial branch point, (see Figure 4-2)

For the computation of intermediate and final branch points, the boundary contour is systematically analyzed by using triplets of boundary elements.
Definition 9:

*Triplets* are a subset of the boundary of a region consisting of three adjacent boundary elements.

In this computation, the objective is to determine the offset distance at which the middle element of the triplet shrinks to a point. That point is the intermediate branch point of the triplet. This point is determined by computing the intersection of the two skeleton branches generated by the triplets. This intersection problem may be classified in one of the following types:
• straight line to straight line intersection
• straight line to conic intersection
• conic to conic intersection

An analytical solution of these intersection problems is summarized in Appendix B. For the computation of intermediate branch points, there is a special case which does not involve triplets. A convex circular arc segment on the contour collapses to its center point when the inward offset of the contour has a distance equal to the radius of the circle. Hence, for such a case, the center of the circular arc corresponds to an intermediate branch point.

The computation of initial branch points involves the computation of a tangent intersection point between two segments, at least one of them is a circular arc. An analytical solution for such intersections is summarized in Appendix B.

From the solution of intersection problems discussed above, we obtain a set of potential branch points and offset distances associated with them. The criteria explained in the next subsections are used to select admissible branch points.

Definition 10:

An admissible branch point is a branch point associated with triplets or two non-adjacent boundary elements which is within the interior of the region and equidistant from those boundary elements.

Once all admissible branch points are computed, then effective branch points and associated offset distance of the boundary contour are determined from the set of admissible branch points.

4.1.1 Intermediate Branch Points

Considering straight line to conic and conic to conic intersections, we have two and four different solutions, respectively. Some of these solutions violate a skeleton property, that an intermediate branch point must be equidistant from three or more boundary
elements. The roots not satisfying this condition are, therefore, discarded. In Figure 4-3, four hyperbolic potential skeleton branches are shown. Of the four intersection solutions \( P_1, P_2, P_3 \) and \( P_4 \), only \( P_1 \) is equidistant from the three boundary segments, and it is accepted as an admissible intermediate branch point.

![Diagram of possible intermediate branch points](image)

**Figure 4-3: Possible Intermediate Branch Points**

### 4.1.2 Initial Branch Points

The first criterion for an initial branch point to be admissible is that its normal projections onto two boundary elements, one of which is either a re-entrant vertex or a circular arc, must lie within those segments. Points which do not satisfy this requirement are discarded.
The second criterion makes sure that an initial branch point is actually created by the offset process. Generation of an initial branch point involves tangent intersection of offsets of two non-adjacent boundary elements. Creation of an initial branch point may, however, be affected by another boundary element [Montanari 69]. If the distance between an initial branch point and a boundary element other than the ones generating the branch point is less than the offset distance corresponding to the initial branch point then during the offset process, the offset contour will pass over the point. As a result, offsets of the two segments associated with the initial branch point will not become tangent to each other (recall the "grass-fire wavefront" interpretation of offsets), (see Figure 4-4). In Figure 4-4, although point P is equidistant from two non-adjacent arcs \(ca_1\) and \(ca_3\) and also a tangent point of their offsets, the distance from the point P to arc segment \(ca_2\) is less than the distance to other circular arcs. This type of point is, therefore, a spurious initial branch point and should be rejected [Patrikalakis 88]. Initial branch points without this type of interference are admissible. As a special case, if a contour has two parallel nonadjacent elements (two straight line segments or two circular arcs), there can be infinite initial branch points between the two elements. The locus of these points is an axis (a straight line or a circular arc segment) which is equidistant from the two elements. For those two boundary elements, offset stops at the offset distance corresponding to the infinite initial branch points.

4.1.3 Computation of Effective Offset Distance

Finally, to determine the effective offset distance and the effective branch point(s) of the boundary contour, the set of admissible branch points are further processed. Among all admissible branch points, the point(s) with minimum offset distance are selected as effective branch points of the boundary contour. Notice that there can be more than one such branch points with the same minimum offset distance. The effective offset distance indicates the offset value at which change(s) in the topology of the contour occur. With this information we know exactly which boundary elements drop out on the next boundary contour generated by the offset process.
Using this information a new offset contour is generated from the description of the boundary contour. There are basically two different changes in the offset topology of the boundary of multiply connected regions. We observe that, in one case, if the boundary contour is associated with $n$ effective intermediate branch points, the resulting offset contour will have $n$ boundary elements less, (see Figure 4-5). In the other case, if the boundary contour is associated with effective initial branch points, then at these vertices on the offset of the boundary contour, either two non-adjacent boundary elements of the exterior loop merge or interior loops merge with each other or with the exterior loop. As a consequence, these types of boundary interactions lead to splitting of the region into a set of subregions or reduction in the number of distinct holes of the region, (see Figure 4-6).
4.1.4 Skeleton and Subdivisions

We should note that from the offset process we obtain branch points of the skeleton and also values of the radius function, namely offset distances at those branch points. The branch points and convex vertices of the boundary contours denote the end vertices of individual skeleton branches. Then we can define the radius function associated with every skeleton branch in a continuous form as a function of offset distance or in a discrete sense as offset distance at distinct branch points.

Skeleton branches and initial boundary contour decompose a shape into a set of subregions (i.e. Voronoi regions). If the given shape is convex those resulting subregions are also convex, (see Figure 3-6). For a non-convex shape, some subregions are not convex. In Figure 3-8, for example, all subregions are non-convex. Introducing Voronoi edges and cuts at initial branch points as shown in Figures 4-7 and 5-1, we can further subdivide such non-convex subregions to obtain convex or pseudo-convex subregions.
Definition 11:

*Voronoi edges* of a given Voronoi diagram are the edges that are incident at re-entrant corners and *flat vertices* of the boundary contour of a planar shape.

Definition 12:

A *flat vertex* of a boundary contour of a planar shape is a junction point of two adjacent boundary elements at which the interior angle is equal to $\pi$.

Definition 13:

A *cut* at initial branch point is a straight line segment connecting two non-adjacent
boundary elements across the initial branch point and orthogonal to these boundary elements.

Definition 14:

A *pseudo-convex* region is an area whose closed boundary can be offset until the area becomes nil without splitting the area into separate components.

A pseudo-convex region involves no bottleneck type narrow part, and the radius functions associated with skeleton branches have no local minima other than the end points of the skeleton branches that generated the region.

As shown in Figure 4-7, subregions can be further divided into topologically *simple* subdomains using a set of straight line segments which are indicated by the dotted lines in this figure. These lines are generated by projecting the branch points on the boundary elements associated with the branch points. The resulting subdomains are either triangular or quadrilateral. These simple subdomains are used in Chapter 5 for finite element meshing.

### 4.1.5 Approximate Medial Axis Transform For Planar Shapes With Curved Boundaries

If a shape with a general curved boundary is represented in terms of non-uniform rational B-splines, our offset technique could, in theory, be generalized to this case. But such an approach would be computationally complex, and in floating point also potentially unreliable because the exact (two-sided) offsets of rational polynomial curves are very high degree algebraic curves [Farouki 89a], [Farouki 89b]. For example, the two-sided offset of a parabola is an algebraic curve of degree 6, of a rational quadratic curve 8, and of an integral cubic curve 10. Singular points of such curves and intersections of two such curves are furthermore needed. These two problems would, for example, require us to invoke complex intersection algorithms [Sederberg 88], [Prakash 88], [Patrikalakis 89b]. In this
thesis, we concentrate instead on the development of a computational technique based on boundary approximation.

For this purpose we use a modified version of the algorithm of [Patrikalakis 89c] which enables us to approximate non-uniform rational B-spline curves with a combination of circular arcs and non-uniform integral B-spline curves of second order within a prescribed accuracy. The modified version of this algorithm is described in detail in Appendix D. A brief summary is presented below.

First each rational B-spline edge is processed to discover knots with tangent
discontinuity at which the B-spline is split into separate edges. Next straight line and circular spans are discovered and extracted, by splitting, as separate edges. The remaining free-form segments of each spline are approximated using a (piecewise) linear B-spline interpolating position and first derivative at the ends and providing a prescribed position error bound at isoparametric points. When the starting edge is an integral B-spline, a tight global error bound can be also provided which guarantees the approximation error at all points and not just at a discrete set of points. The above preprocessing and approximation now allows computation of an approximate MAT of the actual shape. Obviously, the linear B-spline approximation introduces many vertices on the boundary contour, which, in turn, give rise to artificial skeleton branches. Such artificial effects may be counteracted by using a threshold technique [Montanari 69]. This threshold technique is based on computation of the minimum of the material angle between adjacent linear segments created by the approximation. Only the skeleton branches associated with vertices of material angles less than or equal to the threshold value are computed and included into the skeleton. Figure 4-8 shows an approximation of the skeleton of a shape with curved boundary made up of a degree 3 integral B-spline curve computed by our method using a threshold value of 105 degrees. We have not yet fully analyzed the conditions under which the above method provides a very good approximation of the MAT. We recommend this topic for future investigation.

4.2 Implementation of the Medial Axis Transform of Planar Shapes

An algorithm based on the above ideas for the computation of the MAT of planar multiply connected shapes with general curved boundaries has been developed and implemented in C. In our implementation, the geometric representation of the boundary contour(s) is the main input of the MAT computation. The choice of appropriate data structures is one of the most important decisions one should make before the
implementations of a geometrical algorithm. We first establish specific data types and the relationships among them using a data abstraction methodology.

4.2.1 Data Structures

In general, we can have two types of boundary contours, one external boundary contour and contours of internal loops. A typical boundary contour is composed of three different types of boundary elements. In our implementation, every internal loop is connected to the external contour along a \textit{virtual cut} using two artificial segments. These artificial segments are not used in the offset and branch point computation, and have no contribution to the skeleton. Their function is to indicate the presence of internal loops during traversal of elements on the boundary in computations of branch points. Thus, typical boundary data of a multiply connected non-convex shape are composed of straight line, circular arc and artificial segments. An abstract data type segment is defined in terms of structure constructs of the C language. These boundary elements are ordered in a
clockwise sequence so that the interior of the region lies to the right. With respect to the clockwise sequence of the boundary segments, we notice that, the center of a convex circular arc lies to the right of the contour boundary and vice versa the center of a concave circular arc to the left. As a result, when a boundary contour is offset inward, concave arcs expand and convex arcs shrink.

Doubly linked lists are used as the main data structure for the representation of boundary contours. The boundary elements of a contour are stored as the items of a doubly linked list which also has pointers to store data of branch points.

The medial axis branches and Voronoi edges decompose a shape into a set of subregions, generally called Voronoi regions. The regions are bounded by boundary elements of the initial contours and associated skeleton branches and Voronoi edges. Here we recall that a boundary element is associated with a distinct subregion. In the computation of the MAT, we determine not only descriptions of the regions but also the adjacency relationships among themselves. For this purpose, we have developed a version of the face-edge data structure proposed by [Weiler 85], [Weiler 86] for two-manifold geometric models. In this representation, the boundary of every region is composed of a boundary element and a chain of skeleton branches. Every skeleton branch has two half edges associated with the two adjacent Voronoi regions. Skeleton branches are also represented using structure constructs. Figure 4-9 illustrates the relationships among these objects.

Several dynamically allocated data structures are also employed to carry out the computation. A queue is used to store the contours to be processed. From this queue, a contour is removed and processed to compute effective branch points and the associated offset distance as discussed in Section 4.1 and Appendix B. The computed branch points of a contour are stored using a list. Then this list is processed to determine effective offset distance and branch point(s). If final branch points are found, the computation of the MAT
of the contour stops. If branch points are not final or the interior area of the contour is not nil, the contour is offset. In that case, if the contour gives rise to only intermediate branch points there will be one resulting offset contour. But if it involves initial branch points as well, the offset of the contour may be split at the initial branch points. For example, the boundary contour of a simply connected polygon with n initial branch points, will be split into n+1 offset contours. In the case of a multiply connected polygon, initial branch points
Sb : skeleton branch
VR : Voronoi region
be_i : boundary element of Voronoi region VR_i
he_{i1} : right half-edge of skeleton branch Sb_i
he_{i2} : left half-edge of skeleton branch Sb_i

Figure 4-10: Objects Used in Boundary Representation

may be generated by the interactions among internal loops and / or internal loops and the external contour. Then splitting or merging of contours occur depending on the topology of the polygon. This situation is resolved by traversing the boundary, and discovering the tangent adjacencies created at initial branch points. To represent such adjacency relationships, boundary elements have pointers to another boundary element which is the tangent element at an initial branch point.
Once a contour is offset, the newly generated offset contours are put into the queue to be processed. The branch points have pointers to the boundary segments that generate these points. The branch points also denote the end points of the skeleton branches. In Figure 4-10, the disassembled skeleton of a simple polygon illustrates the data objects used and adjacency relationships among themselves.

4.2.2 An Algorithm for the Medial Axis Transform

Our computation methodology discussed in the previous sections has led to an algorithm for MAT computation, whose pseudo-code is given below (see Algorithm 1). In our algorithm, boundary elements are treated independently to compute branch points. Then the effective offset distance and branch points are determined using the set of branch points determined in this computation. We note that a contour may be associated with more than one effective branch points though their effective offset distance must be the same. In the MAT computation, a contour may create more than one contours during the offset process depending on the topology of the contour. The MAT computation results in a set of Voronoi regions decomposing the shape. The Voronoi regions are represented as faces in a Boundary Representation model. The skeleton branches, Voronoi edges and boundary elements are the bounding edges of the Voronoi regions.

In our implementation, the MAT computation is carried out in double precision involving sixteen decimal digit arithmetic. Computation of branch points and effective offset distances requires a tolerance value for numerical comparisons. In our implementation, the maximum dimension of a shape is less than or equal to 20 and for this case the tolerance value is $10^{-5}$. 
Algorithm 1

Input: Data of boundary contour(s) of a region and
threshold angle (TA).

Output: List of skeleton branches, associated radius
function and list of Voronoi regions.

begin

Put initial contour(s) into contour queue;
while there is contour existing in contour queue do
    Get a contour from contour queue as current contour;
    if current contour is not nil then
        for each segment, other than artificial ones do
            Compute admissible intermediate branch point (BP);
            if segment is a concave circular arc then
                Compute admissible initial BP(s);
                Put BP(s) into admissible BP list;
            end
        end
        Determine effective BP(s) using admissible BP list;
        Determine offset distance using an effective BP;
        for every convex vertex on current contour do
            if material angle of convex vertex ≤ TA then
                Compute segment of skeleton branch (S);
                Put S into skeleton list;
            end
        end
    end
    if effective BP is final then
        Stop computation for current contour;
    else
        Compute new offset contour(s);
        Put new contour(s) into contour queue;
    end
end
else
    Compute skeleton branches of nil polygon;
    Put skeleton branches into skeleton list;
end

Construct list for Voronoi regions;
for every boundary element of initial contour(s) do
    Create Voronoi region using skeleton information;
    Put Voronoi region into list;
end
4.2.3 Examples on the Medial Axis Transform Computation

Some test cases of the MAT have been computed using our implementation. The results show that our implementation can reliably handle complex shapes and also that the MAT is an effective technique to automatically extract some important shape characteristics.

Figure 4-11 illustrates the skeleton of a symmetric and periodic shape. These characteristics are preserved by the MAT. Figures 4-12 and 4-13 show that non-convex shapes can be decomposed into a set of simple subdomains. As seen in these figures, the simple subdomains, which decompose planar shapes, are either triangular or quadrilateral. Also the MAT computation can identify positions of constrictions in the shape, which are bottleneck like parts of planar shapes. The cuts introduced across the constrictions allow us decompose Voronoi regions into pseudo-convex subregions. In Figures 4-11, 4-12, 4-13 and 4-15, various constrictions of complex shapes are identified by the MAT computation. In Figures 4-11 and 4-13, we notice that holes are enclosed within distinct loops of skeleton branches. This technique allows us to effectively isolate interior loops of a complex shape. Figure 4-14 illustrates the fact that the skeleton can be used to idealize an elongated shape.

4.3 Extension of the Medial Axis Transform to Trimmed Curved Surface Patches and Volumes

In this Section we present some techniques to extend the MAT computation from planar regions to trimmed curved surface patches and three-dimensional volumes. Our methodology is presented in the next two sections, respectively.
Figure 4-11: Medial Axis Transform of a Symmetric and Periodic Shape

Figure 4-12: Decomposition of a Non-Convex Shape by Medial Axis Transform
4.3.1 The Medial Axis Transform of Trimmed Curved Surface Patches

Using the Boundary Representation method, we describe a trimmed patch as a face using a set of bounding loops on a surface patch. The face of interest on the surface patch is bounded by one exterior loop and also, possibly, by a set of disjoint interior loops, all defined on the surface patch. An untrimmed parametric surface patch of the non-uniform rational B-spline (NURBS) form, $S_{k,l}(u, v)$, of maximum orders $k$ and $l$ in the parametric variables $u$ and $v$ is represented in homogeneous coordinates as

$$S_{k,l}(u, v) = \sum_{i=0}^{n} \sum_{j=0}^{m} N_{i,k}(u) N_{j,l}(v) P_{ij} \tag{4.1}$$
Figure 4-14: Idealization of a Shape by Medial Axis Transform

Figure 4-15: Medial Axis Transform of a Complex Shape
where $P_{ij} = [w_{ij}x_{ij}, w_{ij}y_{ij}, w_{ij}z_{ij}, w_{ij}]$ are the elements in an $(n+1) \times (m+1)$ array of control points and $N_{i,k}(u)$, $N_{j,l}(v)$ are B-spline basis functions defined over non-uniform knot vectors in the $u$ and $v$ directions given by $\{u_0, ..., u_{k+n}\}$ and $\{v_0, ..., v_{l+m}\}$, respectively. The B-spline basis functions can be evaluated by De Boor's recursion [DeBoor 72]. The bounding loops are defined in terms of NURBS curves defined on the parameter space of the surface patch. Such a parametric curve, $C_r(s)$ of order $r$ is represented in homogeneous parametric space as

$$C_r(s) = \sum_{k=0}^{q} N_{k,r}(s) P_k$$

(4.2)

where $P_k = [w_{k}u_{k}, w_{k}v_{k}, w_{k}]$ are the $(q+1)$ control points and, similarly, $N_{k,r}(s)$ are B-spline basis functions defined over a non-uniform knot vector $\{s_0, ..., s_{r+q}\}$. The basic properties of the B-spline basis functions used in the above equations are given in Appendix D.

The concept of medial axis transform on a curved surface patch can be generalized using the grass-fire analogy and the resulting generalized offset curves on surfaces. Such offset curves on surfaces are defined via minimal paths (geodesics) emanating from an initial curve in a direction orthogonal to the curve. Formal definition of such offsets of NURBS curves on NURBS surfaces and their approximation in terms of spline curves using an adaptive technique can be found in [Patirakalakis 89d].

In this thesis, however, we do not deal with MAT based on the concept of minimal paths (geodesics) on a curved surface. Instead we perform the MAT computation in the parameter space exactly and we map the resulting skeleton branches and Voronoi diagram to three-dimensional space using the patch equation (4.1). The resulting decomposition into subregions in the parameter and three-dimensional spaces is very useful in applications.

Given the Boundary Representation of a trimmed surface patch (see Figure 4-16), we can directly extract the boundary geometry in the parameter space of the surface patch.
Thus we obtain the boundary data needed for our MAT computation algorithm. The MAT computation of this representation results in a set of Voronoi regions which are bounded by the boundary segments, skeleton branches Voronoi edges and cuts. These Voronoi regions decompose the parameter space of the trimmed patch. The union of the Voronoi regions is the face bounded by all loops on the trimmed surface patch. After using the approximate procedure of Appendix D for all bounding edges, the skeleton branches bounding the Voronoi regions are also defined in terms of conic sections (see Appendix A). The skeleton branches can be traced in parameter space using these representations. Thus, if we chose a sequence of points on a skeleton branch in u-v plane, we can map these points into the three-dimensional space using the definition of the parametric surface patch given by equation (4.1). Then connecting those points by a sequence of line segments we obtain a linear approximation of the skeleton branch in three-dimensions.

Figure 4-16: Mapping Between Parameter Space and Three-Dimensional Space

Once we decompose the parameter space of a trimmed patch in terms of a set of
Voronoi regions, we can triangulate these subregions. Thus, we obtain a set of triangles embedded into the parameter space of the trimmed patch. Then each triangle can be readily mapped into the three-dimensional space by determining the coordinates of the corner vertices using equation (4.1). This technique provides us an approximate representation of a trimmed surface patch in terms of triangular facets. By local refinement of the triangles by splitting them into smaller triangles in the parameter space, we obtain better approximations to the trimmed curved surface. We have developed an algorithm which triangulates and approximates a trimmed surface patch within a prescribed tolerance by using this methodology. This algorithm and some representative examples are presented in Section 6.3. Such a triangulation of a trimmed surface patch may be used for two purposes. First, it may be employed as a finite element discretization of untrimmed and trimmed plate and shell structures. Such a discretization can be, in turn, analyzed using triangular plate and shell finite elements, which are commonly available in existing finite element systems. Second, the approximate representation of a trimmed patch in terms of a set of triangles may also be used in approximately solving intersection problems involving trimmed curved surfaces. Due to the approximate nature of the representation, this solution method may not be very accurate. Nevertheless for low accuracies such a technique is efficient from the computation point of view and could, therefore, be useful for basic interrogation [Turner 88]. Other potential applications of the above method include integral properties computation, point-set classification and ray tracing of trimmed curved surface patches.

4.3.2 The Medial Axis Transform of a Closed Polyhedron

In theory, the MAT computation may be extended to three-dimensional problems. In this section we discuss some ideas and concepts towards this objective. But before we proceed to the three-dimensional MAT computation, it is convenient to evaluate the two-dimensional MAT computation and identify some computational difficulties. As we already discussed in Section 4.1, we make use of an offset process in the MAT
computation. In this process, if we have a non-convex shape with a number of re-entrant vertices we observe that the offset process increases the complexity of the problem even if our initial boundary is a simple polygon. Since re-entrant vertices give rise to circular arc segments on the offset contour, we deal with quadratic equations in the subsequent computations. Similarly in the three-dimensional case re-entrant vertices and re-entrant edges give rise to spherical and cylindrical surface patches, respectively. Thus re-entrant vertices and edges are critical points on the boundary, increasing the complexity of intersection problems. We also observe that quadratic equations describing the boundary of a region of interest may lead to larger round-off errors as compared to linear equations in intersection problems. Therefore, it may be difficult to implement robust and numerically stable algorithms for the three-dimensional MAT computations of complex regions. An approximate computational method may prove to be a better approach for the three-dimensional MAT.

We could devise an approximate MAT computation by redefining the offset process. A re-entrant vertex of the boundary of a planar shape gives rise to a circular arc segment when the boundary contour is offset towards the inside of the region, (see Figure 4-17). If we ignore re-entrant vertices in the offset process, then such vertices have no contribution to the offset contour, (see Figure 4-18). We observe that in this case the offset contour has the same number and type of boundary elements as the initial boundary. Thus, this modified offset process does not change the complexity of the boundary contour. In Figure 4-18 we observe that the re-entrant vertex \( V_0 \) is translated to a new position \( V_1 \) by the offset process. We also notice that this translation takes place on the line which is the bisector of the angle at the re-entrant vertex. Let \( h \) be the offset distance and \( h^* \) be the distance between vertices \( V_0 \) and \( V_1 \), then these two distance values are related by the expression

\[
h^* = h \sin(\alpha/2)
\]

(4.3)
The angle $\alpha$ is the exterior angle at the re-entrant vertex. We notice that when $\alpha$ becomes zero $h^*$ goes to infinity. Thus, for very sharp re-entrant vertices this offset process does not yield acceptable results. As a remedy to resolve this problem, we can introduce a better approximation for the offset process. In this case, a re-entrant vertex contributes to the offset contour as a line segment which is perpendicular to the bisector line and away from the vertex by the offset distance, (see Figure 4-19). As we see in the examples shown in Figures 4-17 and 4-19, this approximate offset process is capable of detecting constrictions and associated initial branch points. The advantage of this approach is that it provides an approximate MAT computation which does not increase the complexity of the Boundary Representation.

![Diagram](image)

**Figure 4-17: Standard Offset of a Non-Convex Shape**

Making use of this approximate computation approach, we could devise a new two-dimensional MAT algorithm. Given a planar shape with a curve-$\mathcal{C}$ boundary, we could first approximate the boundary using only straight line segments. Then the determination of the
Figure 4-18: Offset without the Contribution of a Re-entrant Vertex

Figure 4-19: Approximate Offset of a Non-Convex Shape
branch points involves two distinct computations as illustrated in Figures 4-20 and 4-21. The first computation deals with determination of intermediate branch points. The second computation deals with computation of initial branch points.

![Intermediate branch point](image)

Figure 4-20: Intermediate Branch Point on an Approximate Offset

It is very interesting to study if this approximate approach is appropriate for three-dimensional MAT computation. In the three-dimensional problem, our objective is to determine the skeleton and the associated Voronoi decomposition of a polyhedron, (see Figure 4-22). Similar to the two-dimensional MAT computation, we could make use of the offset process. For the Boundary Representation of the polyhedron we need to use a non-two-manifold Boundary Representation scheme, such as the radial-edge data structure [Weiler 86]. The reason is that re-entrant vertices, edges and faces give rise to non-two-manifold topologies in the offset process. In a non-two-manifold topology, either more than two faces are adjacent to each other along a common edge or the boundary of a polyhedron touches itself at a vertex or a wireframe edge is attached to a surface, (see Figure 4-23).
Figure 4-21: Initial Branch Point on an Approximate Offset

Figure 4-22: Three-Dimensional Medial Axis Transform
Figure 4-23: Non-Two-Manifold Situations

In the MAT computation, we determine offset distances for which the topology of the offset surface of the boundary changes. Similar to the two-dimensional computation, we essentially determine a number of intermediate and initial branch points, edges and faces and terminate the process by detecting final branch points, edges or faces. Here the concepts are analogous to those discussed in the two-dimensional computational methodology. Detailed study of the above ideas is recommended.
Chapter 5

Automatic Mesh Generation Based on the Medial Axis Transform

5.1 Finite Element Mesh Generation

Finite element mesh generation is concerned with subdivision of a problem domain into simpler subregions. This discretization process determines mesh node topology and geometry embedded in the original shape. As this discretization is refined, it provides a better approximation to the shape. The finite element method uses such a discretization to approximately solve a field problem.

There are several techniques developed over the last twenty years to automatically generate triangulations of a given problem domain. An extensive review of many important finite element mesh generation schemes is provided in Chapter 2. Although most of the meshing techniques are characterized as "automatic", during the starting up or at an intermediate step of the meshing process these methods usually require manual user input in order to produce an appropriate finite element mesh. A typical input is the subdivision of the original problem domain into a set of subregions so that the triangulation process in the next step does not break down. Also most of the mesh generators make use of an initial distribution of nodes along the boundary and / or within the region. These, so called, key nodes determine local mesh density and size of elements of a mesh and, therefore, significantly affect the quality of discretization.

In this chapter, a novel automatic mesh generation scheme for multiply connected planar shapes based on the medial axis transform (MAT) is presented. This mesh generator requires little user input or manual intervention, since it uses the MAT to automatically interrogate the geometric representation. Therefore, such a meshing scheme may prove to be a very practical and useful tool in design and analysis of complex engineering systems.
This chapter is structured as follows. Section 5.2 introduces a hybrid approach to the finite element mesh generation process. Next in Section 5.3, our new mesh generation scheme relying on a hybrid mesh generation methodology is presented. Section 5.4 outlines an extension of our technique for triangulation of the trimmed curved surface patches. Finally, Section 5.5 illustrates some representative examples demonstrating the capabilities of our meshing scheme.

5.2 Finite Element Triangulation of a Two-Dimensional Shape

In this Section we present basic aspects of a hybrid mesh generation process to obtain a two-dimensional triangular mesh. Of course, this process may be extended in an analogous way to three-dimensional cases using tetrahedra to discretize volumes. This approach is hybrid in the sense that it involves two distinct processes during finite element mesh generation. The first process decomposes the two-dimensional domain to be meshed into a set of topologically simple subdomains.

Definition 15:

A simple subdomain is a simply connected convex or pseudo-convex region with one boundary loop which is composed of a sequence of either three or four edges.

The second process triangulates individual simple subdomains generated by the first process. Thus a triangular mesh of the domain is obtained by taking the union of all triangulated subdomains.

First of all, the mesh generator requires the geometry of a region to be discretized as its input. This input is usually in a Boundary Representation form which can be directly obtained from a Boundary Representation model or derived from a CSG representation by means of boundary evaluation.

Then initialization of discretization begins. A complex shape with re-entrant corners
and multiple internal holes is decomposed into simpler subdomains so that an admissible mesh with triangular elements of good shape characteristics can be generated. As a heuristic rule we require all triangular elements to approximately be \textit{equilateral} triangles. The result of the decomposition process is a set of convex or pseudo-convex subdomains whose union is the original shape.

The subdomains generated in the previous decomposition process are organized into an appropriate data structure for meshing. This representation contains \textit{adjacency information} among all subdomains so that the triangular elements generated in different adjacent subdomains satisfy \textit{compatibility requirements}. In an admissible finite element mesh composed of compatible elements, two adjacent elements share all nodes on the interface edge. On the other hand, incompatible nodes can be used in a mesh provided that the degrees of freedom of those incompatible nodes are constrained with respect to the adjacent compatible nodes by using a set of equations. Although this second approach allows highly localized mesh refinement, it requires a more complex data structure to represent the resulting mesh and complicates the solution process.

After the region is decomposed into a set of subdomains each subdomain can be triangulated individually. These subdomains can be regarded as \textit{super finite elements} from the finite element discretization point of view. An approach for the triangulation of a simple subdomain can be based on discretization of the boundary of the subdomain followed by triangulation of the interior. The discretization of the boundary requires specification of mesh size and density. The discretization of the subdomain boundaries also assures compatibility between adjacent elements in different subdomains. The finite element mesh of the region is the union of all finite element meshes embedded into all subdomains. It is worth noting that other mesh generation schemes (e.g. mapping and recursive decomposition techniques) can also be used to discretize these individual simple subdomains.
Automatic triangulation of a complex shape may generate finite elements with unfavorable shape characteristics in various parts of the region. Regions with highly non-uniform shape characteristics, closely spaced holes, wavy boundary, etc can give rise to very irregular meshes. Such a finite element mesh will, in turn, give rise to less accurate numerical results. Therefore, it is common practice to apply **smoothing** to the mesh generated by triangulation in order to improve irregular shape characteristics. As an heuristic objective, we require all elements to approximately be **equilateral** triangles. Our smoothing technique to satisfy this objective is an iterative process in which the position of an interior nodes is incrementally changed by averaging its coordinates and the coordinates of all adjacent nodes. Various schemes are available for the mesh smoothing process such as Laplacian and isoparametric methods [Herrmann 76], [Lorensen 80].

Depending on the problem at hand, a finite element discretization may need local refinement in order to improve numerical results. Before the first finite element analysis, a coarse mesh should be locally refined in regions close to boundary segments associated with boundary conditions such as fixed nodes, concentrated loads and also singularities including fixed and re-entrant corners [Jaswon 77].

Also after a finite element analysis, the discretization error at individual elements can be estimated using a posteriori error indicators [Babuska 86]. Finite elements which have large error contributions should be refined by using one of several techniques in order to obtain better numerical results in subsequent analysis steps. In a practical method for the local refinement based on the h-convergence approach, triangular elements can be bisected across their longest edge. This process allows generation of compatible meshes, and at the same time does not degrade the shape characteristics of triangular elements. These are important and favorable features for a practical implementation of mesh generation schemes. The following pseudo-code summarizes the main steps involved in the hybrid finite element mesh generation process, (see Algorithm 2).
Algorithm 2

Input : Boundary Representation model of a domain to be meshed.
Output : Finite element mesh.

begin
    Decompose domain into topologically simple subdomains;
    Generate nodes along boundaries of subdomains;
    for each subdomain
        Create a triangular finite element mesh;
        for each subdomain
            if refinement is required
                Refine mesh within subdomain;
            if shape characteristics of mesh is not good enough
                Smooth mesh;
    end

5.3 A New Finite Element Mesh Generation Scheme Based on the Medial Axis Transform

Using the methodology introduced in the previous section we can develop a meshing scheme based on the MAT. The proposed finite element mesh generation scheme automatically discretizes a two-dimensional shape into a set of triangular elements. This meshing scheme accepts the MAT of a shape as its input and generates complete finite element mesh information. Major steps of this scheme are presented in detail in the following subsections.
5.3.1 Decomposition of a Region by Means of Voronoi Regions

Due to the nature of the MAT, every boundary element is associated with a unique Voronoi region of the shape. We can subdivide a Voronoi region into simple subdomains in such a way that a portion of the boundary element is associated with only one skeleton branch on the boundary of the Voronoi region. Thus we obtain a coarse discretization of the shape. One possible application of this approach is that we can easily identify subdomains associated with significant boundary conditions. Thus we determine areas for local mesh refinement in a direct and very efficient manner.

Proposition 1:

Given a Voronoi region, the region can be further subdivided so that each skeleton branch on the perimeter of the Voronoi region can be mapped onto a unique finite portion of the boundary element associated with the Voronoi region. The mapping is done by means of a projection process. In a degenerate case, if the boundary element is a re-entrant corner, all skeleton branches of the Voronoi region associated with this vertex are mapped onto this vertex.

The segmentation of the boundary is carried out by computing the projections of the end points of skeleton branches (i.e. branch points) onto the boundary element associated with those branches, (see Figure 5-1). This decomposition process results in a set of subdomains with simple topologies. These subdomains are either triangular or quadrilateral (see Figure 5-1). Triangular subdomains arise at terminal branches of the skeleton and, possibly, at re-entrant (non-convex) vertices, (see Figures 3-7 and 5-1). If a potential triangular subdomain is very narrow, (i.e. with a small acute angle), this subdomain is not generated and the adjacent quadrilateral subdomain is forced to contain this triangular subdomain. Recognition of such poorly shaped triangular subdomains and their merging with appropriate adjacent quadrilateral subdomains have been implemented in our system.
Quadrilateral subdomains arise for all other branches of the skeleton. Such a decomposition process also allows us to determine local length scale information of a shape. In this work, we use the largest value of the radius function for a given subdomain as the local length scale associated with that subdomain. This information, in turn, is used to determine the length dimension of triangular finite elements discretizing the subdomain. We also make use of the value of radius function at initial branch points to determine the local length scale of a constriction.

![Diagram showing shape decomposition](image)

**Figure 5-1: Shape Decomposition by Means of The Medial Axis Transform**

#### 5.3.2 Processing of the Subregions Obtained from the Medial Axis Transform

The MAT based process of Section 5.3.1 decomposes a complex shape into a set of topologically simple triangular or quadrilateral subdomains. Although these subdomains are topologically simple, the lengths of their edges may, sometimes, turn out to vary
significantly. For example, very narrow quadrilateral and triangular subdomains involving angles which are very different from $\pi/3$ may be occasionally generated. Presence of such badly shaped subdomains makes it difficult to create a finite element discretization composed of triangles with good shape characteristics. As a rule of thumb, we require that all triangular elements closely resemble equilateral triangles. Although we do not use quadrilateral elements in our meshing scheme, the optimum quadrilateral element shape is the square. The objective of these heuristic rules is that a well shaped finite element does not have a bias towards a particular direction in a finite element mesh layout. Such a requirement can also be expressed in terms the aspect ratios of elements. The aspect ratio, AR, of an element is defined as the ratio of the longest dimension to the shortest dimension associated with the element [Kardestuncer 87]. For structural analysis applications to obtain acceptable stress results, an upper limit on the aspect ratio may be $\text{AR} \leq 3$ [Kardestuncer 87]. Unfortunately, this criterion does not take into account the skewness of triangular and quadrilateral elements. For example, a triangular element can have an aspect ratio approximately equal to 2, and at the same time its area can be almost zero. Therefore, a better measure of aspect ratio may be defined using the skeleton and radius function of a triangular element. The aspect ratio defined as the ratio of length of longest skeleton branch to the maximum value of the radius function also accounts for skewness of elements.

It is convenient to illustrate these abnormalities involved in shape decomposition based on the MAT using a simple shape. Figure 5-2 shows a rectangle whose longer dimension $D$ is a variable and the shorter dimension is constant, $d=2a$. In Figure 5-3 corresponding to $D=2a$, i.e. a square, the solid line is the skeleton of the rectangle and dotted lines represent discretization of the region in terms of triangular elements. Here we observe that the triangular elements exhibit good shape characteristics. The aspect ratio of the triangles is $\sqrt{2}$. If we start to increase $D$, the MAT decomposition will give rise to two
very narrow subdomains. If $2a < D < 5a/2$, the discretization of those subdomains produces triangles with poor shape characteristics, (i.e. $AR > 2.24$, see Figure 5-4). In Figure 5-4, triangulation of the rectangular domain is carried out using our mesh generation method presented in the subsequent sections. For the time being, let us consider only shape characteristics of these triangular elements. We can identify such a situation which gives rise to badly shaped elements using the ratio, $R$, of the length of associated skeleton branch to the maximum value of radius function, in this particular case $0 \leq R < 1/2$. To rectify this abnormality, the short skeleton branch needs to be processed. For example as seen in Figure 5-5, we can construct an approximate skeleton by eliminating the short skeleton branch, and obtain a modified decomposition. In this example, the triangle is of aspect ratio $AR=1.6$ which is better than the previous value.

![Diagram](image)

$d = 2a$

$D \geq d$

Figure 5-2: A Simple Parametric Shape

Short boundary elements and re-entrant vertices with material angle only slightly above $\pi$ also give rise to very narrow triangular Voronoi regions, (see Figures 5-6 and 5-7).
Figure 5-3: Finite Element Mesh of the Parametric Shape

Figure 5-4: Finite Elements with High Aspect Ratios
Figure 5-5: Modified Skeleton and Associated Discretization

Such cases arise frequently for high accuracy approximation of a curved shape with linear segments. For the narrow subregion due to a short boundary elements, we can modify the skeleton branches associated with the subregion so that the triangle becomes closely equilateral, (see Figure 5-6). In this process, the apex of the triangular subregion is moved towards the boundary element until the aspect ratio of the triangle reaches an acceptable value such as 3. In the case of narrow subregion associated with a re-entrant vertex, it is possible to completely eliminate such a subregion from the decomposition. As seen in Figure 5-7, the narrow triangular subregion is shrunk to a line segment connecting the initial branch point to the re-entrant vertex.

Based on these observations, we could carry out the following operations to rectify or eliminate subdomains with unacceptable shape characteristics for finite element mesh generation purposes:

- Given the Voronoi decomposition of a shape, we first determine the ratio, R, of the lengths of the skeleton branches bounding the region with the associated
Figure 5-6: A Narrow Subregion and its Modification

Figure 5-7: A Narrow Subregion Associated with a Re-entrant Vertex
maximum radius function value, and identify Voronoi regions with small value of R (e.g. $R < 1/4$).

- If a skeleton branch with a small value of the ratio R is connected at its one end to a terminal branch, these two branches can be merged together.

- If a short branch has more than one adjacent branches at each of its two ends, the branch is eliminated using a process illustrated in Figure 5-8. This process is analogous to "kill_edge_vertex" topology operator which is used to manipulate two-manifold boundary models [Mantyla 88]. In this case, first new vertices A, B, C and D are inserted on adjacent skeleton branches at distances equal to a fraction of the local length scale value and away from the branch points. Then adjacent branches are modified by creating a new segment on each of these adjacent branches. A new straight line segment on a skeleton branch is defined by connecting the new vertex (e.g. A, B, C, and D) to the middle point, M, of the short branch. Since this modification process makes use of local length scale information, it can be made robust enough so that no intersections occur between new segments and other boundary elements of the shape. Although this process introduces approximate representation of the skeleton, such approximations are acceptable for finite element mesh generation purposes.

- If an abnormal subregion is associated with a boundary element its bounding edges are modified so that the subregion becomes more uniform, as shown in Figure 5-6. Here an aspect ratio value $AR \leq 3$ can be used as a criterion for this modification process.

**Figure 5-8: Elimination of a Short Branch**
• If a triangular abnormal subregion is associated with a re-entrant vertex, the subregion is removed from the decomposition and the skeleton branches adjacent to the subregion are modified, as shown in Figure 5-7.

This processing of a Voronoi decomposition attempts to eliminate poorly shaped subdomains of the decomposition by merging a portion or whole part of an irregular subdomain with other adjacent subdomains. After these processes, the resulting decomposition is expected to allow us to generate a mesh with better shape characteristics. We have not yet experimented extensively with the preprocessing of poorly shaped regions suggested above to identify its performance and possible limitations.

5.3.3 Discretization of Boundaries and Interiors of Subdomains

Once we process and subdivide Voronoi regions, we obtain a set of subdomains. We can discretize bounding edges of subdomains, except their boundary elements, by generating nodes along those edges. During this automatic node insertion process, a fraction of the value of radius function is used as distance between inserted nodes. Since those inserted nodes are shared by adjacent subdomains, the final mesh will satisfy compatibility requirements between adjacent elements generated in different subdomains.

Next we further subdivide individual subdomains into a set of quadrilateral strips. In degenerate cases, a quadrilateral strip becomes a single triangle which appears within triangular subdomains. In this process, a set nodal points on every boundary element associated with a subdomain is specified. Nodal points on the skeleton branch edges of a subdomain are mapped onto the boundary element of the subdomain. This process can be regarded as a discrete dual of the MAT. Namely, in the MAT computation, we determine the skeleton branch of two boundary elements which is the set of points equidistant from those elements. On the other hand in subregion discretization, we determine points on the boundary elements corresponding to points of the skeleton branch.
Proposition 2:

Every point on a skeleton branch not associated with a re-entrant vertex can be mapped onto a unique point of each boundary element associated with that skeleton branch.

The map used here is the projection of a point on the skeleton branch to an associated boundary element. In a degenerate case of a boundary element being a re-entrant corner, then this mapping would result in that corner vertex for every point on the skeleton branch associated with the re-entrant vertex. Actually, this mapping of a point from skeleton branches to boundary elements is a corollary of Proposition 1. After this mapping process, we obtain a set of points on each boundary edge and perpendicular segments connecting them to the corresponding points on the skeleton branch of the subdomain. Thus a set of quadrilateral strips are embedded into the subdomain. These quadrilateral strips have two edges orthogonal to the boundary element of the subdomain, (see Figure 5-9). Triangular elements are easily generated within such simple quadrilateral strips. If the subdomain is triangular, at most two triangular strips (they can be considered as individual elements) are also created in addition to a set of quadrilateral strips, (see Figure 5-9).

5.3.4 Triangulation of Subdomains

In the previous section, we introduced a technique to discretize the boundaries and the interior of a subdomain. The quadrilateral strips generated by the mapping have two edges which are orthogonal to boundary element of the subdomain. If there are, for example, \( n \) nodal points inserted along the skeleton branch then \((n-1)\) strips will be generated after the mapping process. New nodal points with uniform spacing can be inserted along the edges of these strips by interpolating the number of nodes along the subdomain, (see Figure 5-10). Node numbers on individual strip edges orthogonal to the boundary may be determined by using the following quasi-linear interpolation equation

\[
N_i = k + \left\lfloor i \frac{l-k}{n-1} \right\rfloor \quad 0 < i < n-1
\]  

(5.1)
where $N_i$ is the number of nodes on the $i$th edge connecting a point on the boundary element to a corresponding point on the skeleton branch at integer position $i$ in the point sequence, $k$ and $l$ are the numbers of nodes on the extreme oblique edges of the subdomain, $n$ is the number of nodes on the skeleton branch and $\lceil \cdot \rceil$ denotes ceiling. In Figure 5-10, the left and right "vertical" edges are oblique to the bottom edge because of merging of two triangular subdomains with high aspect ratio with this quadrilateral subdomain.

After inserting all nodes along the edges of the strips of a subdomain, triangulation of a subdomain can be carried out by triangulating individual strips. If the subdomain is triangular it can be transformed to a quadrilateral by removing a triangular element (which is the degenerate strip of this subdomain) from the subdomain. Triangulation of the quadrilateral strips is a relatively straightforward process which involves matching of appropriate nodes on two opposite edges of the strip. Several techniques have been also proposed for triangulation of such strips [Imafuku 80], [Schoofs 79]. These edges are perpendicular to boundary element of the subdomain. For a given strip, the four corner
vertices are interrogated to determine the one associated with the largest interior angle. Once such a vertex is identified, it is connected to an appropriate vertex on the other edge of the strip such that a triangular element is extracted from the strip at the end of this process, (see Figure 5-13). Then, this process is applied to the remaining part of the strip. Thus a set of triangles are generated by removing triangles from the strip one at a time. In our current implementation, the difference of node numbers on opposite edges of a strip is up to 3 but could easily be modified to other values.

Figures 5-11 through 5-14 illustrate the steps involved in our finite element mesh generation scheme. Figure 5-11 illustrates the skeleton of the multiply connected planar shape. Subdivision of the region and nodes inserted along the edges of the subdomains are illustrated in Figure 5-12. Figure 5-13 shows triangulation process within a subdomain. Finally, the resulting mesh which is made up of 216 elements and 134 nodes is shown in Figure 5-14.
Figure 5-11: A Planar Multiply Connected Shape and its Skeleton

Figure 5-12: Region Subdivision and Boundary Discretization
5.3.5 Mesh Smoothing Process

After all subdomains of a shape are triangulated, the mesh is smoothed to improve the shape characteristics of individual triangular elements. A simple coordinate averaging process has been found to lead to efficient smoothing. Our mesh smoothing process is similar to the smoothing methods of [Cavendish 74], [Chae 88]. In our implementation, coordinates of interior nodes are modified by averaging coordinates of the node and those of all other surrounding nodes. In this iterative process, before a new iteration starts, coordinates of all interior nodes are updated using new coordinates computed in the
previous step. This iterative process stops either after a certain number of iterations or if the maximum value of displacements of all interior nodal points at an iteration becomes less than a prescribed percentage of the maximum value of displacements of the previous iteration. For a particular point \( P \), the coordinates at iteration \( i+1 \) can be determined from the following relation.
\[ p^{i+1} = p^i + \frac{\sum_{j=1}^{m} Q_j^i}{m} \]  

where \( i \) denotes number of the iteration step and \( Q_j \) are coordinate vectors of \( m \) points adjacent to the point \( P \). Although this smoothing process is efficient it has one disadvantage. If a mesh contains extreme non-convex boundary (such as re-entrant corners) this smoothing technique may destroy the topology of mesh by moving some edges of elements in such areas to the outside of the actual region. Therefore, a robust implementation of this smoothing process requires that elements adjacent to re-entrant corners must be identified and not smoothed, or processed separately.

The above smoothing process puts some requirements on geometric representation and data structures used in modeling. We have implemented an efficient version of this process, which necessitates explicit representation of adjacency information of all nodal points surrounding any nodal point. In Boundary Representation terminology [Weiler 85], this is denoted as the vertex-vertex adjacency relationship \( v(V) \). \( V \) represents a set of vertices which are adjacent to a particular vertex \( v \). Examples of mesh smoothing are illustrated in Figures 5-18 and 5-21 to 5-24.

5.3.6 Data Structures and Algorithm for Triangulation

Our mesh generation scheme allows mesh smoothing and adaptive local mesh refinement. Efficient implementation of operations requires that adjacency information among triangular elements must be available explicitly in the data structures. In our implementation, we have adopted a Boundary Representation scheme to represent the resulting finite element mesh. In this representation, an individual triangle is treated as a topological element "face". Our implementation is based on a version of face-edge data structure [Weiler 85]. Figure 5-15 illustrates the structures used in our implementation.
Adjacency relationships between adjacent triangles are explicitly represented. This information allows us to implement very efficient mesh smoothing and local refinement processes. We use a dynamically allocated doubly linked list structure to store triangular elements. The following pseudo-code illustrates the main steps involved in our coarse finite element mesh generation scheme, (see Algorithm 3).

![Diagram of triangular element and half-edge structures]

**Figure 5-15**: Relationships Between Objects Used in the Mesh Generator
Algorithm 3

Input : Data of boundary contour(s) of a region.
Output : List of triangular finite elements.

begin

Construct skeleton, list of Voronoi regions using Algorithm 1;

Create list for simple subdomains;

for each three sided Voronoi region
    Add region into subdomain list;

for each remaining Voronoi region {
    Subdivide region into three / four sided subdomains;
    Add subdomains into subdomain list;
}
for each edge between adjacent subdomains
    Generate uniformly distributed nodes;

Create list for triangular elements;

for each subdomain in list {
    if subdomain is four sided {
        Generate set of quadrilateral strips;
        Triangulate strips in quadrilateral subdomain;
    }
    if subdomain is three sided {
        Generate set of strips;
        Convert triangular subdomain into quadrilateral by extracting one degenerate triangular strip as a triangular element;
        Triangulate strips in remaining quadrilateral subdomain;
    }
    Add triangles into list;
}

Smooth mesh;

end
5.3.7 Local Mesh Refinement

One of the objectives of our implementation is to create a compatible mesh when a local mesh refinement process is carried out. In a compatible mesh, degrees of freedom associated with finite element nodes are common to all adjacent elements. For local adaptive mesh refinement, we use a bisection technique. In this approach, a triangle is split into two halves across its longest edge. Our triangle refinement approach is similar to the mesh refinement method proposed by [Rivara 84]. However, our mesh representation methodology and refinement process which are based on Boundary Representation techniques are different from Rivara's scheme.

When a triangle is split into two halves across its longest edge and if a second triangular element adjacent to the split edge exists, the second element must be also processed since the middle node inserted on the split edge is not a common vertex. Therefore, this refinement process has, to some extent, a propagative nature. Numerous examples of our mesh generator chosen for their complexity and diversity led to the conclusion that this mesh refinement approach, in general, exhibits a local refinement character and does not effect the whole domain being meshed.

Figure 5-16 illustrates two possible cases and operators involved in this mesh refinement scheme.

**Case A**: When splitting a triangular element into two triangles, the incompatible node is on the longest edge of the triangle. This case is either the starting or terminal point of the refinement process. When the refinement process starts, it, first, splits a triangle along its longest edge by introducing an incompatible node. Thus, on the next step the triangular element which is adjacent to the edge with incompatible node is determined and refined. If this edge is the longest of the next triangle to be processed the refinement terminates after the next triangle is split into two triangles. If the edge containing the incompatible node is on the boundary of the mesh the refinement also terminates.
Case B: When splitting a triangle into three triangles, the incompatible node is on one of the shorter edges of the triangle. This case is, in general, an intermediate refinement step. It can be a terminal step only if the longest edge of the triangle is on the boundary of the mesh. Given a triangle with a incompatible node on one of its shorter edges, first the triangle is split into two triangles by introducing a new incompatible node on its longest edge. Then identifying the new triangle which contains the first incompatible node, this new triangle is split into two triangles by connecting the two incompatible nodes. Thus three new triangles are generated in total. In the next step, the triangle adjacent to the edge containing the new incompatible node is determined and processed similarly.

Since our representation is based on a Boundary Representation scheme, we can make an analogy between the refinement operators and standard Euler operators modifying a manifold topology in a boundary model. The process which splits a triangle is similar to a "make_vertex_edge_face" type Euler operator which involves creation of a new vertex, edge and face [Mantyla 88]. Thus an advantage of our mesh refinement procedure is that it can easily be implemented in a boundary based solid modeling system if appropriate low-level Euler operators are provided.

During the mesh refinement process, the doubly linked list structure is manipulated in such a way that when a triangle is refined it is removed from the list and the newly generated triangles are inserted at the end of the list. Figure 5-17 illustrates our list management approach in the mesh refinement process. This approach is computationally efficient and involves linear computation time with respect to the number of newly created elements.

Basic aspects of our list manipulation process are discussed here. In our implementation, the list has pointers to its head and tail elements and also a pointer is used to indicate current position on the list. The current element pointer (CEP) indicates the element to be refined. We need one additional pointer which points to the previous element
Figure 5-16: Triangle Bisection Operators

Case A:

(1)  (2)  (3)

Case B:

(1)  (2)  (3)  (4)
Figure 5-17: List Manipulation During the Mesh Refinement Process

Case A:

Case B:

topologically adjacent element of \( T_M \)
on the list and is called previous element pointer (PEP), (see Figure 5-17). Suppose we want to refine the element with offset m from the head of the list. Thus CEP points to triangle $T_m$ and PEP points to triangle $T_{m-1}$. During refinement, there are two different cases involving the following processes (see also Figures 5-16 and 5-17):

- **Case A**: $T_m$ is removed from the list and two newly created triangles are added to the end of the list. Using the adjacency information of the edge containing the incompatible node, CEP is set. If it is a null pointer this indicates the terminal case, namely that the edge with the incompatible node is on the boundary. In such a case, the refinement propagation terminates, and CEP is reset to point to a triangle which is the element next to triangle $T_{m-1}$ (i.e. $T_{m+1}$). Otherwise, CPE points to the triangle which was topologically adjacent to the $T_m$ along its longest edge. This triangle can be at any position in the list.

- **Case B**: This case is more complex than the above. At the beginning CEP points to $T_m$ and PEP points to $T_{m-1}$. First $T_m$ is split and three new triangles are added into the end of the list. The adjacency relationship of the longest edge of $T_m$ is employed to identify the next triangle. CEP is set to point to this element. If CEP is a null pointer, this indicates that the longest edge is on the boundary and, therefore, the refinement process terminates for triangle $T_m$. In that case, CEP now points to a triangle which is the element next to triangle $T_{m-1}$ (i.e. $T_{m+1}$). Otherwise, the new triangle pointed by CEP is identified. This triangle can be at any position in the list. The triangle pointed by CEP is split in a similar manner.

In this list manipulation strategy, we always keep track of the element to be refined next. Thus we do not have to repeatedly traverse the list structure in order to determine the next element to be processed. Appropriate use of the pointers allows us to implement the refinement scheme in an efficient manner.

### 5.3.8 Complexity Estimates of the Medial Axis Transform and Meshing Processes

A running time complexity analysis of our algorithms gives the following results. In the MAT algorithm, we observe that computation of admissible initial branch points of a given contour with n boundary elements has $O(n^2)$ time complexity in the worst case. Admissible intermediate branch points of a given contour with n boundary elements are computed in $O(n)$ time. The offset process and these branch point computations continue until the offset contour becomes a trivial shape (i.e. either a nil polygon or a triangular region) which, in general, takes certain number of steps proportional to n.
The coarse mesh generation process is an $O(m)$ process with respect to the total number of triangular elements, $m$, being generated. Triangulation of individual subdomains has a linear time complexity with respect to the number of elements generated. The reason is that our meshing scheme is geometrically based and no search operations are carried out during the meshing process as, for example, in topologically based schemes. Thus this approach gives rise to a linear running time complexity. As discussed in the previous section, the mesh refinement process is also an $O(q)$ process with respect to the number $q$ of new triangular elements being created.

Considering the overall performance of our implementation, the overhead involved in our MAT algorithm is subdominant with respect to the mesh creation and refinement time complexity. In a typical mesh generation problem $n \ll m, q$, and, therefore, the bulk of the computational effort is spent for actual mesh generation and refinement, which exhibit a linear time complexity.

To test our claim, we have used a simply connected region and generated four meshes with increasing element densities. Figure 5-18 illustrates the initial and smoothed meshes generated by our algorithm for one particular mesh density. This particular mesh is made up of 120 elements and 79 nodes. The four finite element meshes generated in this numerical experiment contain 72, 96, 120 and 144 elements and are associated with running time 2.6, 3.4, 4.4 and 5.4 seconds respectively. These running time results are shown in Figure 5-19. These results have been obtained using a DEC Vax Station II GPX running under the Unix operating system. The elapsed running time for each case has been determined using the utility "time" of the Unix operating system. The linear character of the running time curve with respect to number of elements supports our claim regarding the time complexity of our mesh generator.
Figure 5-18: Triangular Mesh of a Convex Region with 120 Elements

(a) mesh before smoothing

(b) mesh after smoothing
Figure 5-19: Computation Time Results of Four Meshes of Simple Region

5.4 Extension of the Triangulation Technique to Trimmed Curved Surface Patches

In Chapter 4, we have presented a technique to approximately extend the MAT to trimmed curved surface patches. In this section we discuss some implementation issues.

The MAT can be used to decompose a complex shape into a set of simple subregions. The triangulation process of such simple regions is a relatively straightforward task. In previous sections of this Chapter, we present such a triangulation technique. Several applications of the triangulation process include finite element mesh generation and approximation of trimmed curved surface patches.
Given a Boundary Representation of a trimmed curved parametric patch, we can readily compute the MAT of the shape on the parameter space. The MAT computation results in the skeleton, associated radius function and a set of subregions decomposing the parameter space of the trimmed patch. Using our triangulation technique we can next generate a mesh on the parameter space of the trimmed surface patch. If required by our application, smoothing can be invoked to improve the shape characteristics of the triangles. Once individual triangles are generated on the parameter space, then they are readily mapped into the three-dimensional space using the surface description of the trimmed patch. This triangulation is the initial discretization of the trimmed patch. In this process, the trimmed surface patch is approximated using a set of triangular planar facets. Since surface curvature information of the patch is not used by the MAT, we may need to refine the faceted approximation so that it satisfies certain accuracy requirements. For this purpose, we define two error norms. An error norm is used to account for the Euclidean distance between the centroid of a triangular facet and a corresponding point on the surface obtained from the surface equation using the parameter values of the centroid. Another error norm can be introduced to identify the error involved in unit normal of the surface. In this case, the unit normal vector of a triangular face is determined and it is again compared with the unit normal vector of the surface at a point which corresponds to the centroid of the triangle. If a triangle has distance error and normal vector angular discrepancies greater than prescribed error bounds, then the triangle is further subdivided to obtain a better approximation for the curved surface patch. The local refinement technique described in Section 5.3 is used for this purpose.
5.5 Triangular Finite Element Mesh Examples

We have tested our coarse mesh generation scheme by applying the triangulation system to many complex and diverse examples. The following representative cases illustrate the usefulness and efficiency of our system. Figure 5-18 illustrates a triangular coarse mesh of a convex shape. Figures 5-20, 5-21 and 5-22 illustrate coarse triangulation of non-convex shapes. In Figures 5-23, 5-24 and 5-25, coarse triangulations of multiply connected region with one or more holes are shown. Specification and computational times of these examples are given in Table 1. The computation times represent total time of mesh generate process including the MAT computation. These small computation times indicate the high potential of our method for real time design and analysis applications. The computation times in Table 1 do not include smoothing. Smoothing computation time is, however, a small fraction of the mesh generation time. These results have been obtained using a DEC Vax Station II GPX running under the Unix operating system.

<table>
<thead>
<tr>
<th>Mesh</th>
<th>Subdomains</th>
<th>Elements</th>
<th>Nodes</th>
<th>Time (sec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 5-20</td>
<td>14</td>
<td>88</td>
<td>59</td>
<td>3</td>
</tr>
<tr>
<td>Figure 5-21</td>
<td>34</td>
<td>192</td>
<td>123</td>
<td>9</td>
</tr>
<tr>
<td>Figure 5-22</td>
<td>36</td>
<td>208</td>
<td>136</td>
<td>11</td>
</tr>
<tr>
<td>Figure 5-23</td>
<td>36</td>
<td>320</td>
<td>195</td>
<td>21</td>
</tr>
<tr>
<td>Figure 5-24</td>
<td>40</td>
<td>324</td>
<td>200</td>
<td>22</td>
</tr>
<tr>
<td>Figure 5-25</td>
<td>56</td>
<td>416</td>
<td>270</td>
<td>35</td>
</tr>
</tbody>
</table>
Figure 5-20: Finite Element Mesh of a Bracket
Figure 5-21: Finite Element Triangulation of a Non-Convex Domain

(a) mesh before smoothing

(b) mesh after smoothing
Figure 5-22: Finite Element Triangulation of an Irregular Region

(a) mesh before smoothing

(b) mesh after smoothing
Figure 5-23: Finite Element Mesh of a Flat Plate with Two Holes

(a) mesh before smoothing

(b) mesh after smoothing
Figure 5-24: Finite Element Mesh of an Irregular Domain with a Hole

(a) mesh before smoothing

(b) mesh after smoothing
Figure 5-25: Finite Element Mesh of a Multiply Connected Domain
Chapter 6

Applications of the Medial Axis Transform

In this chapter we present various applications drawn from engineering design and analysis areas to illustrate the usefulness of our interrogation technique based on the medial axis transform (MAT). The first application addresses adaptive finite element analysis of linear elasticity problems. Our meshing scheme based on the MAT provides an initial mesh appropriate for an adaptive refinement solution method. The mesh refinement capabilities of our meshing system are also used during an h-version adaptive analysis process. The second application illustrates the capability of MAT to create idealized models of engineering structures useful in preliminary analysis and simulation. Finally in the third application, we use our triangulation scheme to discretize and approximate trimmed curved surface patches in terms of a set of planar facets within a prescribed precision.

6.1 Adaptive Finite Element Analysis

In recent years, an important research aim of the finite element analysis (FEA) community has been to develop more automatic analysis procedures by integrating FEA with computer aided design (CAD) systems [Wordenweber 84], [Shephard 84], [Shephard 85a], [Shephard 85b], [Kela 86], [Taig 86], [Fenves 86], [Chae 88] and [Zienkiewicz 89]. In this section, we review the main aspects of the various adaptive analysis schemes and present the technique which we developed for our experimental adaptive finite element solver.
6.1.1 Adaptive Finite Element Analysis Schemes

The emphasis in the development of adaptive FEA procedures has largely been on linear problems because adaptive procedures for nonlinear analysis can only be tackled in depth once such procedures are available for linear problems. Figure 6-1 gives a schematic overview of an efficient structural CAD system integrated with a finite element code. Such a integrated adaptive CAD system is not currently available [Bathe 86], but would be most valuable in engineering design. The major link between the CAD system and the finite element system is the mesh generation and model creation process. Here the finite element system could be one general purpose FEA code. In an adaptive FEA scheme, the objective is to automatically produce results satisfying a prescribed degree of accuracy.

The basic steps involved in a typical adaptive process may be summarized as follows.

1. Start with definition of the problem and an initial finite element mesh.

2. Perform the required FEA.

3. Using an error estimator, predict the discretization errors to determine those portions of the analysis model that are not yielding the required degree of accuracy.

4. Improve the portions of the mesh that are not satisfactory, return to the second step and continue this process until the desired level of accuracy is achieved.

This is an iterative process in which distinct steps of analysis, error estimation and mesh improvement are carried out in sequence. For mesh improvement, there are essentially four options [Babuska 86]:

- **h-version**: Finite element size is reduced in subsequent analysis step.

- **p-version**: Degree of interpolation functions is increased in subsequent analysis step.

- **r-version**: Nodal points in the domain are repositioned.

- **Remeshing**: Previous mesh is discarded and analysis is started again with a new mesh layout.

The single most important feature of an adaptive finite element code is the prediction of errors present in the current solution. Errors involved in the numerical solution of a finite element formulation can be separated into three main groups [Utku 83]:
Figure 6-1: An Integrated CAD System. (Adapted from [Bathe86])

Physical Problem → Change of Physical Problem →

Mechanical Model Governed by Differential Equations
Assumptions on
- Geometry
- Kinematics
- Material law
- Loading
- Boundary conditions
  Etc.

→ Improve Mechanical Model →

Finite Element Solution
Choice of
- Finite elements
- Mesh density
- Solution parameters
  Representation of
- Loading
- Boundary conditions
  Etc.

→ Refine Mesh, Solution Parameters, Etc. →

Assessment of Accuracy of Finite Element Solution of Mechanical Model

→ Interpretation of Results → Design Improvements
  Structural Optimization

→ Refine Analysis →
-144-

- Discretization errors: These are caused by representing a continuum using a finite number of degrees of freedom in the discretized system.

- Round-off errors: These are caused by the limitation of digital computers in representing real numbers.

- Solution errors: There are various causes for these errors. For example, solution errors in the constitutive modeling are due to linearization and integration of the constitutive relations. Solution errors in the calculation of the dynamic response arise in the numerical integration of the equations of motion with respect to time. And, finally, solution errors arise from iterative solutions because convergence is measured on increments in the solution variables that are small but not zero.

A priori finite element error analysis can only indicate the convergence rate of the numerical solution [Strang 73]. Therefore, quantitative error estimation is based on a posteriori error analysis which makes use of actual finite element results to estimate the discretization error. Depending on the degree and location of discretization errors, the finite element mesh is refined locally to obtain more accurate results in the next analysis cycle. In constructing an error estimator in the displacement based finite element approach, several criteria have been developed during the last decade [Babuska 86]:

- Predicted strain energy density variations;
- Stress discontinuities along element boundaries; and,
- Magnitude of violation of internal element equilibrium.

To some extent, all these criteria have been shown to work well in practice. In existing adaptive FEA schemes, the h-version and p-version refinement techniques are the most widely used. It has been shown that the p-version adaptive analysis has better convergence characteristics in comparison to the h-version [Babuska 86]. One major disadvantage of the h-version refinement approach is that in the vicinity of singularities in the problem domain, (such as re-entrant vertices, constrained corners and concentrated loads), the rate of convergence is slow. Therefore, problems with severe singularities require very fine finite element meshes and this, in turn, reduces the efficiency of the numerical solution. On the other hand the p-version refinement approach involves a
smaller number of finite elements and high order interpolation functions. In an experimental two-dimensional adaptive system, these two techniques are used in tandem so that their favorable properties are better exploited [Zienkiewicz 89]. In that solution scheme, the h-version refinement process is initially employed to obtain a better finite element mesh layout and to identify singularities in the problem domain. Then, the refinement process switches to the p-version using the mesh obtained from the previous process as its starting mesh. Numerical results given in [Zienkiewicz 89] indicate that this approach is computationally efficient.

6.1.2 An Adaptive Finite Element Analysis Scheme for Plane Elasticity Problems

In this section, we discuss the technique we developed to determine discretization errors in a finite element solution of linear elasticity problems. We also use this methodology to solve plane stress elasticity problems.

Given a Cartesian coordinate system, (x, y, z), we deal with the solution of elasticity problems whose governing differential equations describing equilibrium of a volume element are of the following form [Timoshenko 70]

\[
\sigma_{ij,j} + p_i = 0 \quad i,j = x, y, z
\]  

(6.1)

where \(\sigma_{ij}\) (\(\sigma_{ij} = \sigma_{ji}\)) are components of the symmetric stress tensor, and \(p_i\) are components of applied body force per unit volume, \(j\) denotes partial derivative with respect to \(j\), and a repeated index denotes summation over \(x, y\) and \(z\).

Equations (6.1) are solved in a domain \(\Omega\), subject to prescribed conditions on its boundary \(\Gamma\).

\[
u_i = \bar{u}_i \text{ on } \Gamma_u \quad \sigma_{ij} n_j = t_i = \bar{t}_i \text{ on } \Gamma_t \quad \Gamma = \Gamma_u \cup \Gamma_t \quad \Gamma_u \cap \Gamma_t = 0
\]  

(6.2)

where \(u_i\) are components of displacement, \(t_i\) components of boundary traction (i.e. force per unit area), \(n_j\) components of the unit normal vector of the boundary, and \(\Gamma_u\) and \(\Gamma_t\) portions of the boundary with displacement and stress boundary conditions.
In this formulation, components of strain are defined as follows
\[ \varepsilon_{ij} = \frac{1}{2} (u_{i,j} + u_{j,i}) \quad \varepsilon_{ij} = \varepsilon_{ji} \]  
(6.3)
and stress is related to strain by the following constitutive equation
\[ \sigma = D \varepsilon \]  
(6.4)
where \( D \) is the material matrix of the domain, \( \sigma = [\sigma_{ij}] \), \( \varepsilon = [\varepsilon_{ij}] \) are the stress and strain tensors.

In the finite element discretization, we approximate the solution function \( u(x, y, z) \) using a trial function, \( u^*(x, y, z) \)
\[ u(x, y, z) = u^*(x, y, z) = N(x, y, z) \ U \]  
(6.5)
where \( N(x, y, z) \) are shape functions and \( U \) is a vector of unknown nodal displacements.

We can obtain the global system of equations of the finite element discretization using the above trial function by a standard weighted residual process, such as the Galerkin method, or by a variational approach in the following form [Zienkiewicz 83]
\[ K \ U = F \]  
(6.6)
where \( K \) is the stiffness matrix and \( F \) is the force vector
\[ K = \int_{\Omega} B^T D B \ d\Omega \]  
(6.7)
\[ F = \int_{\Omega} N^T \bar{p} \ d\Omega + \int_{\Gamma_i} N^T \Gamma \ d\Gamma \]  
(6.8)
and superscript \( T \) denotes matrix transpose, and matrix \( B \) contains derivatives of shape functions such that
\[ \varepsilon^* = B \ U \]  
(6.9)
Using the finite element solution, we can determine the stress as follows.
\[ \sigma^* = D \varepsilon^* \]  
(6.10)
The approximate solutions \( u^*, \sigma^* \) and \( \varepsilon^* \) differ from the exact values \( u, \sigma \) and \( \varepsilon \), and the differences are the discretization errors involved in the finite element solution. We note that the numerical finite element solution also includes round-off errors. Following [Utku 83], we define absolute solution errors for displacements and stresses by
\[ e = u - \bar{u}^* \] \hspace{1cm} (6.11)

\[ e_\sigma = \sigma - \sigma^* \] \hspace{1cm} (6.12)

We note that in the above equations \( u \) and \( \sigma \) are actual values, which are not available. Therefore, it is necessary to predict these error values by other means.

These forms of the error are pointwise definitions. A more useful form of the error may be defined in a manner similar to the energy norm of the solution [Zienkiewicz 83]:

\[ \| u \| = [ \int_\Omega (Bu)^T D (Bu) \ d\Omega ]^{1/2} \] \hspace{1cm} (6.13)

\[ \| u^* \| = [ \int_\Omega (Bu^*)^T D (Bu^*) \ d\Omega ]^{1/2} \] \hspace{1cm} (6.14)

\[ \| e \| = [ \int_\Omega (Be)^T D (Be) \ d\Omega ]^{1/2} = [ \int_\Omega e_\sigma^T D^{-1} e_\sigma \ d\Omega ]^{1/2} \] \hspace{1cm} (6.15)

Using a finite element discretization, the error norm in the whole domain can be calculated by adding up the contributions of error norms of all elements used in the discretization, i.e.

\[ \| e \| = [ \sum_{i=1}^m \| e_i \|^2 ]^{1/2} \] \hspace{1cm} (6.16)

where \( i \) spans all elements in the finite element discretization and \( m \) is the number of elements in the mesh.

Using the energy and error norms, we also define the relative percentage error, which is a measure of the relative error involved in the solution [Utku 83]

\[ \eta = \frac{\| e \|}{\| u \|} \times 100 \] \hspace{1cm} (6.17)

In an "optimal" finite element mesh, the relative percentage error associated with each element is almost the same.

In the displacement based finite element formulation of elasticity problems, \( C^0 \) (displacement) continuity is assumed in the trial functions of the finite element approximation. Therefore, numerical results, in general, give rise to discontinuous approximations of \( \sigma \), (see Figure 6-2). This character of the solution is well known since
the early days of FEA. Therefore, stress smoothing techniques have been commonly used in practice to provide improved quantitative results from the discontinuous stress functions [Hinton 79], [Langtangen 89]. The smoothed stress values turn out to be much better approximations of the actual values of stress than the finite element results before smoothing, [Hinton 79]. Based on this observation, we can assume that such a smoothed stress distribution can be used to predict the solution error. If we substitute the smoothed stress value, $S$, for the exact stress value in equation (6.12), we obtain a prediction for the error in stress.

$$e_\sigma = S - \sigma^*$$  \hspace{1cm} (6.18)
Rank and Zienkiewicz [Rank 87] have shown that such an error prediction method is equivalent to the other a posteriori error estimators used for error predictions in adaptive FEA [Babuska 86].

Now we can determine the energy and error norms using equations (6.14) and (6.15) and predict the percentage error involved in the solution [Babuska 86].

\[ \eta = \left[ \frac{\|e\|^2}{(\|u^*\|^2 + \|e\|^2)} \right]^{1/2} \times 100 \]  

(6.19)

Once we have a finite element solution, the smoothed stress distribution can be readily obtained. Using the above expressions, we can come up with a prediction of the error involved in the solution. In an adaptive FEA based on h-convergence, and after determining the error norm, we carry out mesh refinement to reduce the size of elements associated with an unacceptable error value before we start the next solution step. Suppose we try to achieve a relative percentage error value, \( \eta_p \), in the finite element solution. Then we require that the adaptive FEA process ends when the following condition is satisfied the resulting mesh:

\[ \eta \leq \eta_p \]  

(6.20)

We also assume that the solution error is uniformly distributed throughout the mesh in a converged adaptive FEA. This assumption of uniform distribution of the solution error is not valid for finite elements near singularities of the problem domain. Based on this assumption, we obtain a condition which specifies the acceptable error norm of individual finite elements [Zienkiewicz 89]

\[ \|e_i\| \leq \bar{e} = \eta_p \left[ (\|u^*\|^2 + \|e\|^2) / m \right]^{1/2} \]  

(6.21)

where right hand side of the inequality represents the mean relative error norm associated with the current mesh. We can readily check whether an element needs to be refined using a ratio of the error norm to this mean value:

\[ \rho_i = \|e_i\| / \bar{e} \]  

(6.22)

If the ratio is greater than unity for a given element, the error associated with that element is
higher than the prescribed tolerance, $\eta_p$. Therefore the element should be refined to a certain subdivision level whose degree is determined by the ratio $\rho$. If a problem domain contains singularities, finite elements near such singularities will be associated with large values of the ratio $\rho$ (i.e. $\rho \gg 1$). Thus, the singularities of the problem domain may be identified.

6.1.3 Examples

The finite element mesh generator described in Chapter 5 has been coupled with a finite element solver for plane stress problems. In such problems and if $x$-$y$ is the plane of interest, then the only non-zero stress components are $\sigma_{xx}$, $\sigma_{yy}$ and $\sigma_{xy}$. In our implementation, the finite element solver uses six-node triangular isoparametric elements. Input data of the FEA are preprocessed and nodes are renumbered so that the bandwidth of the global stiffness matrix becomes minimum. From efficiency point of view, this is a very important consideration even for two-dimensional linear problems. Our automatic node renumbering scheme is based on the method presented in [Collins 73]. Element stiffness matrices are numerically evaluated using a three-point Gauss quadrature rule. The resulting system of linear equations is solved using the Gauss elimination method. The error indicator used in this experimental system is based on the techniques presented in the previous section. In our implementation, the smoothed stress distribution is obtained using a simple process. First of all, stress components computed at discrete points in the element (i.e. at three Gauss quadrature points) are extrapolated to the nodes of the element. Then for a given node, smoothed stress values are calculated by taking the average of weighted stress contributions of all adjacent elements. Simple averaging of stress at nodal points works correctly only if a mesh is composed of elements with the same size. We assume that the problem domain has uniform thickness and use the area of an element as its weight factor to account for the size of adjacent elements in this smoothing process. This smoothing approach is similar to the "lumped mass least-squares" approximation in one-dimension presented in [Langtangen 89].
The following three representative model problems have been adaptively solved by our experimental system. In each of these cases, the problem domain has a Young's modulus, $E = 2 \times 10^5$ MPa, a Poisson's ratio, $\nu = 0.3$ and a thickness, $t = 1$ cm. The adaptive scheme has provided a solution with relative percentage error (RPE) less than 10% in all three cases.

In the first example, a bracket is analyzed, (see Figure 6-3). The error indicator has identified stress concentration in the narrow part, and local refinement is performed in that region. In the second example, (see Figure 6-4), an L-shaped plate is analyzed. In this problem, the re-entrant vertex is associated with a singularity. In the adaptive analysis, this singularity is effectively detected by the error indicator and mesh refinement is carried out around the re-entrant vertex. In the third example, a beam with a circular hole at its center is analyzed, (see Figure 6-5). A clamped boundary condition is specified at one end and a uniformly distributed shear force is applied at the other end. The clamped end boundary condition gives rise to two singularities and a boundary layer which are identified by the error indicator. The adaptive solution also identifies some critical elements around the circular hole in the problem domain. All these test cases demonstrate the efficiency and usefulness of our automatic mesh generator and adaptive finite element solution scheme.

6.2 An Idealization and Structural Model Creation Based on the Medial Axis Transform

In previous Chapters, various properties of the MAT were introduced. Inspecting skeletons of planar shapes, it is easy to observe that the skeleton and associated radius function of a shape may be useful in providing a engineering idealization of shape. For example, a slender (i.e. elongated) planar structure can be idealized as a beam using the skeleton as the axis of the beam. The skeleton and associated radius function can be used to define the parameters of the beam model for approximate or preliminary structural analysis.
Figure 6-3: Adaptive Analysis of a Bracket

$E = 2 \times 10^7 \text{ N/cm}^2$
$t = 1 \text{ cm}$
$\nu = 0.3$

$3 \text{ cm}$
$p = 50 \text{ N/cm}^2$

88 elements
59 nodes
RPE = 16%

525 elements
1122 nodes
RPE = 9%
Figure 6-4: Adaptive Analysis of a L-Shaped Plate

$E = 2 \times 10^7 \text{ N/cm}^2$

$\sigma = 1 \text{ cm}$

$\nu = 0.3$

- 88 elements
- 59 nodes
- RPE = 13%

- 147 elements
- 332 nodes
- RPE = 9%
Figure 6-5: Adaptive Analysis of a Beam with a Hole

\[ E = 2 \times 10^7 \text{ N/cm}^2 \]
\[ t = 1 \text{ cm} \]
\[ \nu = 0.3 \]

204 elements, 460 nodes, RPE = 12%

364 elements, 810 nodes, RPE = 7%
As an example, we investigate the response of the planar structure shown in Figure 6-6. Here the structure is composed of three elongated subcomponents and its MAT is shown in Figure 6-7. We note that, except close to the ends of the shape, the skeleton can be regarded as a wireframe representation of the structure. By trimming away the terminal branches of the skeleton, we construct a simple finite element model of the structure using its skeleton branches which are discretized in terms of beam elements, (see Figure 6-8).

\[ p = 40,000 \text{ N/m}^2 \]

\[ E = 2 \times 10^{11} \text{ N/m}^2 \]

\[ t = 1 \text{ m} \]

\[ \nu = 0.3 \]

\[ \rho = 7800 \text{ kg/m}^3 \]

![Diagram of an elongated structure](image)

Figure 6-6: An Elongated Structure

To assess the accuracy of such an approximation, we determine the response of this structure for the model problem defined in Figure 6-6 as well for the associated eigenproblem. The structure is first represented as a wireframe model using Hermitian beam elements. It is also analyzed in terms of a two-dimensional continuum model using plane stress elements. Both models have been analyzed under the static loading of Figure 6-6 and, also, using a linear free vibration analysis determining the first two natural frequencies and corresponding mode shapes. The numerical computations have been
Figure 6-7: Medial Axis Transform of the Structure

Figure 6-8: Idealization of the Structure Using the Skeleton
carried out by using a well known finite element code [ADINA 84]. Experimenting with several different finite element discretizations of these models, we have obtained convergent results. We have compared extremum values of the normal stress at several locations and the first two natural frequencies of these two models. Comparison of the results is summarized in Table 2. Mode shapes of the fundamental frequencies of the continuum model and the beam model are shown in Figures 6-9 and 6-10, respectively. Very similar results are obtained from the beam and two-dimensional continuum model approaches.

| TABLE 2 |
| (CM : continuum model,  BM : beam model) |

<table>
<thead>
<tr>
<th></th>
<th>CM</th>
<th>BM</th>
<th>100</th>
<th>CM-BM</th>
<th>/CM</th>
</tr>
</thead>
<tbody>
<tr>
<td>displacement point</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>0.106</td>
<td>0.118</td>
<td></td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>stress points</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>0.490</td>
<td>0.513</td>
<td></td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>-0.374</td>
<td>-0.416</td>
<td></td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>0.373</td>
<td>0.415</td>
<td></td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>-0.489</td>
<td>-0.511</td>
<td></td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>frequencies</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mode 1</td>
<td>6.9</td>
<td>7.1</td>
<td></td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Mode 2</td>
<td>32.4</td>
<td>34.6</td>
<td></td>
<td>7</td>
<td></td>
</tr>
</tbody>
</table>
Figure 6-9: Two Mode Shapes of the Two-Dimensional Continuum Model

Figure 6-10: Two Mode Shapes of the Beam Model
Based on this analysis, we may draw several conclusions. If a structure is made up of elongated subcomponents, namely, if one dimension of a subcomponent is much longer than all other dimensions, the skeleton seems to provide a very useful and appropriate approximation of the actual structure for a quick preliminary analysis. On the other hand, if the structure components are not elongated, the skeleton is not expected to lead to an appropriate beam model, and the structure should be treated as a two-dimensional continuum. In such cases, the MAT can also be used to identify this situation. Namely, we may consider a structure not to be slender, when the length of skeleton branches is comparable to the value of the associated maximum radius function over the corresponding branches. Conversely, this helps us to distinguish whether a structure may be regarded as a slender object. We can, however, always employ the subdivision obtained from the MAT computation to create a finite element discretization, if we treat the structure as a continuum model.

The above two discretization approaches may be also employed to create very effective mechanical models of structures with non-uniform shape properties (e.g. structures composed of slender and solid components). As an example, the two-dimensional body shown in Figure 6-11 has two distinct subcomponents. One part is elongated and slender, essentially one-dimensional, and the other part is two-dimensional. The elongated part is identified by the MAT because it gives rise to a relatively long skeleton branch associated with a small radius value. On the other hand, the two-dimensional part gives rise to skeleton branches whose length dimensions are comparable to the values of the associated radius function. An idealized mechanical model of this object is shown in Figure 6-11. The square region is decomposed into a coarse finite element mesh by the skeleton branches.

Considering the approximate model creation based on the MAT approach we should be cautious in deciding how skeleton branches should be included in the approximate
model. If we examine the MAT of a rectangular region, (see Figure 3-11), we observe that the convex corners generate skeleton branches which are relatively shorter than the main portion of the skeleton. Along these skeleton branches the radius is a linear function and its value changes from zero at the boundary vertices to the half of the thickness of the rectangle at the branch points. If we include those skeleton branches as beam elements, the resulting model would not be an accurate representation of the physical structure. Using the CSG type concept of Figure 3-11, the model would, for example, have more material near the branch points than the actual structure. The reason is that around the branch points disks associated with different branches overlap and give rise to representation of more material than the actual region. An approach to rectify this misrepresentation of the region
might be to modify the radius functions of minor skeleton branches close to branch points so that the actual material of a problem domain is represented as accurate as possible in the approximate model. The radius function can be "thinned" within those areas. We also note that the transverse dimension of such a beam is comparable to its length and, therefore, a beam model approach may be inappropriate from mechanical point of view for this case.

6.3 Faceted Representation of Trimmed Curved Surface Patches

Our planar mesh generation scheme has been extended to trimmed curved surface patches using the concepts introduced in Chapters 4 and 5. The surface patch is parametrically defined by equation (4.1). In this implementation, given a trimmed surface patch we, first, determine boundary data required for the MAT computation. The MAT computation results in a set of Voronoi regions which decompose the parameter space of the trimmed patch. Then, our coarse finite element mesh generation scheme creates a triangulation in the parameter plane of the trimmed patch. This computation results in a list of triangles. Those triangles are treated individually and their vertices are mapped into three-dimensional space by using the surface definition. Thus, a faceted approximation of the trimmed curved surface patch in the form of a coarse faceted triangulation is obtained.

To evaluate the level of accuracy attained in this approximation, we use a norm based on Euclidean distance. Given a triangular approximation, we compute the distance from the centroid of an individual triangle to the corresponding point on the trimmed surface patch. If this distance value is greater than a prescribed tolerance, a refinement flag associated with the triangle is set. In our implementation, value of the refinement flag is determined by the ratio of the distance value to the prescribed tolerance. The floor value of the refinement flag indicates the level of subdivision to be carried out for the triangle. By traversing the doubly linked list structure, every triangle is similarly processed to determine the value of the refinement flag.
Once refinement flags of individual triangles are set, the list structure is traversed again to carry out the actual refinement process. If a triangle has a refinement flag greater than or equal to unity, it is split. The resulting child triangular elements assume refinement parameter values one less than the parent triangle's refinement flag value. The refinement process continues until the refinement flag values of all triangles are reduced to zero.

The following pseudo-code summarizes the steps of our technique to triangulate trimmed curved surface patches with complex curved boundaries, (see Algorithm 4). Our scheme to triangulate trimmed curved surface patches has been implemented and tested with a number of complex and diverse examples. Figures 6-12 to 6-15 illustrate triangulation of some such trimmed patches with complex boundaries. In these cases, the surfaces are defined as integral bicubic B-spline patches.

Algorithm 4

Input : Boundary Representation of trimmed curved surface patch and distance TOLERANCE.
Output : List of triangular facets.

begin

Compute Voronoi decomposition in parameter space using Algorithm1;

Construct list of triangles as a coarse triangulation in parameter space using Algorithm3;

for every triangle { 
    Compute distance error norm (DISTANCE_ERR);

    if DISTANCE_ERR > TOLERANCE
        [comment: set refinement flag of triangle (FLAG)]
        FLAG ← [DISTANCE_ERR / TOLERANCE];

    }

while exists a triangle with FLAG ≥ 1 in list

    Refine triangle;

end
Figure 6-12: Triangulation of a Trimmed Curved Surface Patch with Four Holes

Figure 6-13: Triangulation of an Irregular Trimmed Curved Surface Patch
Figure 6-14: Triangulation of a Doubly Curved Shell with Two Holes

Figure 6-15: Triangulation of a Trimmed Curved Surface Patch with a Hole
Chapter 7

Conclusions and Recommendations

In this chapter, we summarize the major results of this thesis. We also identify other potential applications and related future research topics on shape interrogation and representation based on the medial axis transform (MAT).

In this thesis, we show that the MAT can be used as an effective interrogation method in geometric modeling systems to extract important global shape characteristics from geometric representations. The automatic extraction of information on constrictions, maximum thickness points and their length scales; holes and their proximity information; and the decomposition of complex shapes into simpler subregions by means of the MAT technique is expected to be a very useful capability in future CAD/CAM systems. Such shape characteristics can be used effectively in solutions of many engineering problems.

In this thesis, we develop a novel reliable algorithm which allows us to compute the MAT of multiply connected planar domains with curved boundaries composed of circular arcs of arbitrary radii and straight line segments. Our algorithm effectively extracts the above shape characteristics. In order to prove the effectiveness and usefulness of our MAT computation technique, we also propose higher level interrogation and analysis methods based on our algorithm.

In this thesis, we develop a novel, efficient and automatic finite element mesh generation method for complex two-dimensional shapes based on the MAT. The major characteristic of this approach is that the MAT is used as an interrogation tool to quantitatively determine significant shape characteristics and also as a technique to create a coarse finite element mesh. In this approach we employ an important capability of the MAT which can decompose a complex region into a set of convex and pseudo-convex
subregions. Such subregions, in turn, can be decomposed into topologically simple subdomains of triangular or quadrilateral form. Such a decomposition can be directly used in a p-version finite element analysis which requires a coarse discretization of the problem domain in terms of finite elements involving high degree polynomial basis functions. Also a coarse discretization of these simple subdomains can be employed as a good initial mesh in a h-version adaptive finite element solution.

Creation of a coarse finite element mesh followed by local adaptive mesh refinement is a very effective technique in solving complex field problems arising in engineering design and analysis. Existing finite element solvers can be transformed into very effective analysis tools, if they are integrated with automatic mesh generators and post-processors carrying out a posteriori error analysis. Such an adaptive analysis process starts with a coarse grid which is locally refined and leads to a solution within prescribed error bounds. Also during the last decade, research on multigrid methods has proven that this methodology significantly enhances computation efficiency in analysis of complex engineering problems. This solution technique employs coarse and locally refined discretizations of problem domains. Our automatic mesh generation technique would be a very effective tool to create such discretizations needed by multigrid solution systems.

In our mesh generation process, the MAT is used to identify local length scales of a region. This information, in turn, is used to automatically select the finite element mesh size, which is usually supplied interactively by the analyst in existing finite element preprocessing systems. Another possible direct application of our shape decomposition method in finite element meshing is to automate mapping mesh generation schemes. These mesh generators, which require that regions are decomposed into mappable subregions, currently rely on manual user input for the shape decomposition process. By contrast, in our hybrid meshing scheme, a complex shape is, first, automatically subdivided into simple subdomains by means of the MAT. Next, these topologically simple subdomains are
individually meshed employing an efficient mesh generation technique. Application of our mesh generation scheme to many complex and diverse examples shows the usefulness and effectiveness of this new methodology in finite element mesh generation.

To further demonstrate the capabilities of our technique, we develop and present three different application algorithms: adaptive finite element solution of plane elasticity problems, automatic adaptive triangulation and faceted approximation of trimmed curved surface patches and idealization and model creation in structural analysis problems.

Our experimental adaptive finite element analysis system for plane elasticity problems uses an error indicator based on smoothed finite element stress solutions. The discretization error involved in the solution is estimated using the discrepancies between finite element stress solutions and their smoothed distributions. Although this approach is approximate, it is more efficient than other a posteriori error indicators which determine discretization errors by integrating stress jumps across adjacent finite element faces. This feature is very important if we want to use adaptive analysis techniques for the three-dimensional problems. For such analysis tasks, even a single cycle finite element analysis requires substantial amount of computation time. Therefore, any additional inefficient computation would render the adaptive solution approach impractical in actual design. To conduct adaptive finite element solutions we develop a method to bisect triangular elements along their longest edge, which propagates refinement to create a compatible finite element mesh.

The application of the MAT to assist creation of idealized structural models is a novel concept. Currently no design system offers appropriate algorithms to automatically carry out such tasks. In general, the designer or analyst is solely responsible for the execution of these high-level abstractions and the labor intensive model creation tasks. The MAT may prove to allow the implementation of systems in which complex decision making processes can be reduced to well defined algorithmic tasks, lending themselves to automation.
Numerical results of a model problem investigating this capability of the MAT for an elongated structure suggest that an idealization method based on the MAT could be very efficient, especially in novel design processes which require rapid first-order evaluation of many design alternatives.

Triangulation of trimmed curved surface patches is another significant extension of our triangulation method based on the MAT. In this novel approach, a trimmed curved surface patch is adaptively approximated in terms of a set planar triangular facets. Our scheme decomposes the parameter space of the trimmed patch in terms of simple triangular and quadrilateral subdomains. Then, individual subdomains are triangulated. The resulting mesh is mapped into the three-dimensional space using the surface equation to obtain an approximation to the trimmed curved surface patch. Next the surface of the trimmed patch is interrogated to determine the accuracy of the approximation. Using user-specified tolerances, and comparing distance and unit normal vector discrepancies between points on the surface and corresponding points on the triangular facets, the accuracy of the approximation can be determined. Depending on local error values, the surface approximation can be adaptively refined. In this adaptive refinement process, a method based on bisection of triangular facets which propagates the refinement to create a compatible triangular mesh is used. Our adaptive surface decomposition and approximation technique can be used to automatically create finite element meshes of arbitrary shell structures with complex geometries in an efficient manner. Other applications include point-set classification, ray tracing and computation of integral properties of trimmed curved surface patches.
Based on the results of this thesis, we recommend the following topics for future research.

Extension of MAT computation to trimmed curved surface patches and three-dimensional volumes bounded by planar and curved faces is a worthwhile objective. Such a computation is, however, expected to be a very complex algorithmic process. Such extension requires use of non-two-manifold Boundary Representations [Weiler 86], [Rossignac 89]. Possible approaches to the solution of the problem of computing the MAT of curved entities include boundary approximation with simpler (i.e. linear) geometries and differential equation methods. The relative merits of these approaches are unknown. Derivation of the differential properties of the MAT is recommended as it could find application in surface interrogation. Development of parallel algorithms for MAT computation to allow real-time design and analysis is also recommended.

Extension of our finite element meshing method based on the MAT to three-dimensional volumes is a worthwhile objective. Identification of subregions with unacceptable shape characteristics and their processing needs careful study. Development of parallel algorithms for two and three dimensional mesh generation schemes based on the MAT to allow real-time analysis is also recommended.

Shape idealization (abstraction) using the MAT is a novel and worthwhile topic of further research with many applications in design, analysis and manufacturing. The homotopy properties of the MAT and their relation with shape abstraction are suggested for further investigation. Development of robust methods based on the MAT to recognize shape features such as symmetry, periodicity, similar subcomponents of a shape, depressions, protrusions and their length scales is also recommended. As an example of the usefulness of such features, super finite elements could be automatically extracted from complex shapes.

Collision avoidance and motion planning in robotics, and determination of optimum
layouts of distributive systems within congested environments in network analysis are other potential applications of the MAT which we recommend for future study.

The MAT offers an alternative method of shape representation beyond existing Constructive Solid Geometry and Boundary Representation schemes. The potential usefulness of such a representation is apparent from all applications identified in this thesis. Data structures and algorithmic issues arising from such an alternative representation scheme deserve further investigation. In addition, efficient reconstruction of the Boundary Representation of a shape from a MAT Representation is recommended for further study.
Appendix A
Definitions of Conic Skeleton Branches

In this Appendix analytical definitions of all types of skeleton branches for boundary
contours bounded by rectilinear and circular arc segments are derived.

Straight Line:

The equations of the locus of points P(x, y) which are equidistant from two straight
lines, shown in Figure A-1, with equations:

\[ ax + by + c = 0, \quad dx + ey + f = 0 \]  \hspace{1cm} (A.1)

can be expressed as:

\[ \frac{ax + by + c}{\sqrt{a^2 + b^2}} = \pm \frac{dx + ey + f}{\sqrt{d^2 + e^2}} \]  \hspace{1cm} (A.2)

These are the equations of the bisectors of the acute and obtuse angles between the two
lines.

Parabola:

We next derive the equations of the locus of points which are equidistant from a
straight line and a circle, shown in Figure A-2, with equations:

\[ ax + by + c = 0, \quad (x - A)^2 + (y - B)^2 = R^2 \]  \hspace{1cm} (A.3)

The offsets of the line and circle with the same offset distance \( h \geq 0 \) can be written in
general form as follows:

\[ ax + by + c = \delta \sqrt{a^2 + b^2} h, \quad (x - A)^2 + (y - B)^2 = (R + \varepsilon h)^2 \]  \hspace{1cm} (A.4)

where \( h \) is the offset distance and \( \delta \) and \( \varepsilon \) are sign functions \( (\delta, \varepsilon = \pm 1) \) indicating the
direction of offsetting. Eliminating \( h \) from equations (A.4) and using \( \eta = \frac{\varepsilon}{\delta} \), we obtain:

\[ \eta (ax + by + c) + R \sqrt{a^2 + b^2} = \sqrt{a^2 + b^2} \sqrt{(x - A)^2 + (y - B)^2} \]  \hspace{1cm} (A.5)

This equation can be reduced to the following quadratic form:
\[ \alpha x^2 + \beta xy + \gamma y^2 + ex + fy + g = 0 \]  
(A.6)

where

\[ \alpha = b^2, \quad \beta = -2ab, \quad \gamma = a^2 \]  
(A.7)

\[ e = -2ac - 2A(a^2 + b^2) - 2aR\sqrt{a^2 + b^2} \eta \]  
(A.8)

\[ f = -2bc - 2B(a^2 + b^2) - 2bR\sqrt{a^2 + b^2} \eta \]  
(A.9)

\[ g = (A^2 + B^2 - R^2)(a^2 + b^2) - 2cR\sqrt{a^2 + b^2} \eta - c^2 \]  
(A.10)

Since \( \beta^2 = 4\alpha \gamma \), this quadratic equation represents a parabola. Depending on the sign function there are two parabolas as shown in Figure A-2.

### Ellipse and Hyperbola:

We next derive the equations for the locus of points equidistant from two circles, shown in Figure A-3, with equations

\[ (x-A)^2 + (y-B)^2 = R^2, \quad (x-C)^2 + (y-D)^2 = r^2 \]  
(A.11)

The offsets of these two circles with the same offset distance \( h \geq 0 \) are of the following form:

\[ (x-A)^2 + (y-B)^2 = (R + \varepsilon h)^2, \quad (x-C)^2 + (y-D)^2 = (r + \delta h)^2 \]  
(A.12)

where \( \varepsilon \) and \( \delta \) are sign functions \( (\varepsilon, \delta = \pm 1) \) indicating the direction of offsetting.

Eliminating \( h \) from the equations (A.12), we obtain:

\[ \eta \sqrt{(x-C)^2 + (y-D)^2} = (\eta r - R) + \sqrt{(x-A)^2 + (y-B)^2} \]  
(A.13)

where \( \eta = \frac{e}{g} \). This equation can be reduced to the following general quadratic form:

\[ \alpha x^2 + \beta xy + \gamma y^2 + ex + fy + g = 0 \]  
(A.14)

with coefficients

\[ \alpha = (A-C)^2 - (\eta r - R)^2, \quad \beta = 2(A-C)(B-D) \]  
(A.15)

\[ \gamma = (B-D)^2 - (\eta r - R)^2, \quad e = 2E(A-C) + 2A(\eta r - R)^2 \]  
(A.16)

\[ f = 2E(B-D) + 2B(\eta r - R)^2, \quad g = E^2 - (A^2 + B^2)(\eta r - R)^2 \]  
(A.17)

where

\[ 2E = C^2 + D^2 - A^2 - B^2 - (\eta r - R)^2 \]  
(A.18)

The value of the sign function \( \eta \) determines the type of conic. If we examine the characteristic value of the conic:
\[ \Delta = \beta^2 - 4\alpha\gamma \] \hspace{1cm} (A.19)

we obtain
\[ \Delta = 4C_2(C_1 - C_2) \] \hspace{1cm} (A.20)

where
\[ C_1 = (A-C)^2 + (B-D)^2 \geq 0, \quad C_2 = (\eta r - R)^2 \geq 0 \] \hspace{1cm} (A.21)

If \( \eta = -1 \) then \( C_2 > C_1 \) and, therefore, \( \Delta < 0 \). Hence, the conic represents an ellipse. On the other hand if \( \eta = 1 \), then \( C_1 > C_2 \) and, therefore, \( \Delta > 0 \). Hence, the conic is a hyperbola.

In the special case of two concentric circles (namely \( A = C \) and \( B = D \)), then the conic becomes a circle with the following coefficients
\[ \alpha = 1, \quad \beta = 0, \quad \gamma = 1, \quad e = -2A, \quad f = -2B, \quad g = A^2 + B^2 - \overline{R}^2 \] \hspace{1cm} (A.22)

where
\[ \overline{R} = \frac{R + r}{2} \] \hspace{1cm} (A.23)

![Diagram](attachment:diagram.png)

**Figure A-1:** Skeleton of Two Lines
Figure A-2: Skeleton of a Line and a Circle

Figure A-3: Skeleton of Two Circles
Appendix B

Computation of Branch Points

In this Appendix, solution techniques for the intersection problems needed to compute branch points are given.

Straight Line to Straight Line Intersection:

This intersection problem is governed by the following system of linear equations:

\[ ax + by + c = 0, \quad dx + ey + f = 0 \]  \hspace{1cm} (B.1)

If \( \Delta = ae - db \neq 0 \), then the intersection point is given by:

\[ x = \frac{bf - ce}{\Delta}, \quad y = \frac{cd - af}{\Delta} \]  \hspace{1cm} (B.2)

If \( \Delta = 0 \), then the two lines do not intersect.

Straight Line to Conic Intersection:

This intersection problem is governed by the following two equations:

\[ ax + by + c = 0, \quad \alpha x^2 + \beta xy + \gamma y^2 + ex + fy + g = 0 \]  \hspace{1cm} (B.3)

Solution of this intersection problem requires consideration of special cases. If \( a, b \neq 0 \), then \( y \) is eliminated from the equations (B.3). The resulting equation, which is at most quadratic in \( x \), is solved for \( x \) values. After that corresponding \( y \) values are computed by substituting values of \( x \) into the first of the equations (B.3).

If \( b = 0 \), then the intersection has in \( x = -c/a \). The values of \( y \) are computed from the second equation of (B.3) using the value of \( x \). If \( a = 0 \), then the intersection has in \( y = -c/b \), and the values of \( x \) are computed using the second equation of (B.3).

Conic to Conic Intersection:
This intersection problem is governed by the following two, in general quadratic, equations:
\[ \alpha x^2 + \beta xy + \gamma y^2 + \epsilon x + \delta y + g = 0, \quad \rho x^2 + \sigma xy + \tau y^2 + \nu x + i y + j = 0 \] (B.4)

From these two equations \( y \) is eliminated by forming the following two equations:
\[ Ay^2 + By + C = 0, \quad Dy^2 + Ey + F = 0 \] (B.5)

where,
\[ A = \gamma, \quad B = \beta x + f, \quad C = \alpha x^2 + \epsilon x + g \] (B.6)
\[ D = \tau, \quad E = \sigma x + i, \quad F = \rho x^2 + \nu x + j \] (B.7)

Eliminating \( y \) from the equations (B.5) following the method given by Sederberg et al [Sederberg 84], we obtain
\[
\begin{bmatrix}
C & B & A & 0 \\
0 & C & B & A \\
F & E & D & 0 \\
0 & F & E & D
\end{bmatrix}
\begin{bmatrix}
1 \\
0 \\
1 \\
0
\end{bmatrix}
= \begin{bmatrix}
0 \\
0 \\
0 \\
0
\end{bmatrix}
\]

The determinant of the coefficient matrix gives a polynomial of at most degree four in \( x \). This polynomial can be easily formed using symbolic computation [MACSYMA 85] and its roots can be evaluated using a root solver [NAG 88]. After determining \( x \) values, corresponding \( y \) values may, for example, be computed using equations (B.4).

**Computation of Initial Branch Points:**

For the computation of initial branch points there are two cases.

**Case 1** involves tangency of two concave circular arcs, (see Figure B-1). The offset distance corresponding to the associated initial branch point is computed by the following equation
\[ h = \frac{\sqrt{(x_1-x_2)^2 + (y_1-y_2)^2} - r_1 - r_2}{2} \] (B.8)
Distances between the tangent point and centers of the circles are determined by the following relationships:
\[ l = l_1 + l_2, \quad l_1 = r_1 + h, \quad l_2 = r_2 + h \]  \hspace{1cm} (B.9)

In non-dimensional form we have the following expressions for \( l_1 \) and \( l_2 \):
\[ \overline{l}_1 = \frac{r_1 + h}{r_1 + r_2 + 2h}, \quad \overline{l}_2 = \frac{r_2 + h}{r_1 + r_2 + 2h} \]  \hspace{1cm} (B.10)

Using these non-dimensional quantities and the coordinates of the centers of the two circles, we determine the coordinates of the initial branch point from:
\[ x_T = x_1\overline{l}_2 + x_2\overline{l}_1, \quad y_T = y_1\overline{l}_2 + y_2\overline{l}_1 \]  \hspace{1cm} (B.11)

**Case 2** involves tangency between a concave circular arc and a straight line (see Figure B-2). First, intersection point \( S \) between the line segment and its normal passing through the center of circle is determined. Then, the offset distance corresponding to the initial branch point is determined by the following equation:
\[ h = \frac{\sqrt{(x_s - x_c)^2 + (y_s - y_c)^2}}{r} - r \]  \hspace{1cm} (B.12)

Non-dimensional distances between the tangent point and straight line and the center of circle are determined by the following equations:
\[ \overline{l}_1 = \frac{r + h}{r + 2h}, \quad \overline{l}_2 = \frac{h}{r + 2h} \]  \hspace{1cm} (B.13)

The coordinates of the initial branch point are computed as follows:
\[ x_T = x_c\overline{l}_2 + x_1\overline{l}_1, \quad y_T = y_c\overline{l}_2 + y_1\overline{l}_1 \]  \hspace{1cm} (B.14)
Figure B-1: Initial Branch Point of Two Circles

Figure B-2: Initial Branch Point of a Line and a Circle
Appendix C

Parametric Forms of Skeleton Branches

In this Appendix we derive appropriate parametric equations of parabola, hyperbola and ellipse for tracing branches of skeleton.

Parabola:

As shown in Figure C-1, a parabola is defined by its Bezier control points, \( P_1, P_2 \) and \( P_3 \). Here \( P_1 \) and \( P_3 \) are starting and ending points of the skeleton branch respectively. Point \( P_2 \) is at the intersection of tangent lines passing through the end points \( P_1 \) and \( P_3 \). Using the three points and a parameter \( t \), we can write the parametric form of parabola

\[
x(t) = C_1 t^2 + C_2 t + C_3, \quad y(t) = C_4 t^2 + C_5 t + C_6
\]  

(C.1)

where \( 0 \leq t \leq 1 \) and

\[
C_1 = x_3 - 2x_2 + x_1, \quad C_2 = 2(x_2 - x_1), \quad C_3 = x_1
\]  

(C.2)

\[
C_4 = y_3 - 2y_2 + y_1, \quad C_5 = 2(y_2 - y_1), \quad C_6 = y_1
\]  

(C.3)

Hyperbola:

As shown in Figure C-2, a hyperbola can be written in a global coordinate frame using its parametric definition in the local coordinate frame and applying coordinate transformation. In the local coordinate frame we have the following parametric equations:

\[
x = a \cosh \theta, \quad y = b \sinh \theta
\]  

(C.4)

where \( 0 \leq \theta \leq 2\pi \). Using a parameter \( t = e^\theta \) we can also write the above equations as follows

\[
x = \frac{a \left( t^2 + 1 \right)}{2t}, \quad y = \frac{b \left( t^2 - 1 \right)}{2t}
\]  

(C.5)

In the global coordinate frame, equations (C.5) transform into the following forms:

\[
x = x_o + \frac{a \cos \alpha - b \sin \alpha}{2} t + \frac{a \cos \alpha + b \sin \alpha}{2} \frac{1}{t}
\]  

(C.6)
\[ y = y_o + \frac{a \sin \alpha + b \cos \alpha}{2} t + \frac{a \sin \alpha - b \cos \alpha}{2} \frac{1}{t} \]  \hspace{1cm} (C.7)

Now we will determine the values of parameter \( t \) to trace the hyperbola from point \( P_1 \) to point \( P_2 \). Using equations (C.5)

\[ t = \frac{a b}{b X - a Y} \]  \hspace{1cm} (C.8)

If the parameter values associated with points \( P_1 \) and \( P_2 \) are \( t_1 \) and \( t_2 \), respectively, then the hyperbolic branch of a skeleton is traced in the interval \( t \in [t_1, t_2] \).

**Ellipse:**

Similar to hyperbola, an ellipse is defined in a local coordinate frame by the following equations (see Figure C-3):

\[ X = a \cos \theta, \quad Y = b \sin \theta \]  \hspace{1cm} (C.9)

In the global coordinate frame, equations (C.9) transform into the following forms:

\[ x = x_o + a \cos \alpha \cos \theta - b \sin \alpha \sin \theta, \quad y = y_o + a \sin \alpha \cos \theta + b \cos \alpha \sin \theta \]  \hspace{1cm} (C.10)

For a given coordinate pair \((X, Y)\) we can solve equations (C.9) to determine \( \theta \) from

\[ \theta = \arg \left[ \frac{X}{a} + i \frac{Y}{b} \right] \]  \hspace{1cm} (C.11)

where \( i \) is the imaginary unit and \( \arg \) is the argument of a complex number in \((-\pi, \pi]\). If the parameter values associated with points \( P_1 \) and \( P_2 \) are \( \theta_1 \) and \( \theta_2 \), respectively, then the elliptic branch of a skeleton is traced in the interval \( \theta \in [\theta_1, \theta_2] \) from point \( P_1 \) to point \( P_2 \).
Figure C-1: Parabolic Skeleton Branch

Figure C-2: Hyperbolic Skeleton Branch
Figure C-3: Elliptic Skeleton Branch
Appendix D
Approximation of a NURBS Curve in Terms of Straight Line and Circular Arc Segments

In this appendix, we present a technique to approximate a non-uniform rational B-spline (NURBS) curve in terms of a set of straight line and circular arc segments. Such an approximation is used as the input of our method to compute the medial axis transform (MAT) of planar shapes. Our method is based on a more general approximation technique developed by [Patrikalakis 89c], which allows conversion of high degree NURBS curves into non-uniform integral B-spline curves of lower degree with a prescribed accuracy.

In our approximation scheme, first each NURBS curve is processed to determine knots with tangent discontinuity at which the curve is split. To identify straight line segments, the B-spline curve is checked one span at a time. For each span, the curve is evaluated at a set of uniformly distributed points in the parameter domain to determine the position on the curve. If the distance of these points from the line passing through the end points of the span is less than a tolerance, the span is identified as a straight line segment. A tight upper bound of the above distance is the maximum of the distance of the control points contributing to the span definition from the straight line through the end points of the span. To identify circular arcs, each span of the NURBS curve is processed in a similar manner to determine curvature values at a set of uniformly distributed points in the parameter space. If curvature values are of constant sign and within a small tolerance range of their average value, this span is, then, considered to be a circular arc.

The remaining free-form segments are approximated using a piecewise linear B-spline which interpolates position and first derivative at the ends and providing a prescribed position error bound at isoparametric points. Before we present this technique, it is convenient to introduce some definitions and properties of B-spline curves.
An integral B-spline curve of order \( m \) (degree \( m-1 \)), \( R_m(t) \), is defined as follows

\[
R_m(t) = \sum_{i=0}^{n-1} P_i N_{i,m}(t) \quad (D.1)
\]

where

- the \( P_i \) are control points in a Cartesian system;
- \( t \) is a real parameter, \( t \in [t_0, t_k] \);
- \( N_{i,m}(t) \) are scalar non-negative piecewise polynomials in the variable \( t \), of order \( m \) (degree \( m-1 \)), called B-splines which form a basis of spline functions.

The basis functions are completely defined by the order \( m \) and a knot vector \( T \). For open (non-periodic) curves, the knot vector is given by

\[
T = [t_0 = t_1 = \ldots = t_{m-1} < t_m \leq t_{m+1} \leq \ldots \leq t_{n-1} < t_n = \ldots = t_{n+m-1}] \quad (D.2)
\]

The B-splines have the following properties

\[
N_{i,1}(t) = \begin{cases} 1 & \text{if } t_i \leq t < t_{i+1} \\ 0 & \text{otherwise} \end{cases} \quad (D.3)
\]

\[
N_{i,m}(t) = \frac{t-t_i}{t_{i+m-1}-t_i} N_{i,m-1}(t) + \frac{t_{i+m}-t}{t_{i+m}-t_{i+1}} N_{i+1,m-1}(t) \quad \text{if } m > 1 \quad (D.4)
\]

\[
\sum_{i=0}^{n-1} N_{i,m}(t) = 1, \quad t_0 \leq t \leq t_{n+m-1} \quad (D.5)
\]

To evaluate equation (D.4) the convention \( 0/0 = 0 \) is used whenever such a ratio appears.

The first two parametric derivatives of a B-spline curve \( R_m(t) \) can be written as follows [DeBoor 78]

\[
\frac{dR_m(t)}{dt} = (m-1) \sum_{i=1}^{n-1} \frac{P_i - P_{i-1}}{t_{i+m-1} - t_i} N_{i,m-1}(t) \quad (D.6)
\]

\[
\frac{d^2R_m(t)}{dt^2} = (m-1)(m-2) \sum_{i=2}^{n-1} \left[ \frac{P_i - P_{i-1}}{t_{i+m-1} - t_i} - \frac{P_{i-1} - P_{i-2}}{t_{i+m-2} - t_{i-1}} \right] \frac{1}{t_{i+m-2} - t_i} N_{i,m-2}(t) \quad (D.7)
\]

Rational B-spline curves [Tiller 83] provide a generalization of integral B-spline curves. They permit representation of a wider class of free-form curves and exact representation of classical algebraic curves such as conics. For example, circular arc segments can be represented exactly with rational B-spline curves. A rational B-spline curve of order \( M \) over the control polygon \( P \) with \( n \) vertices and knot vector \( T \) is defined as
\[ R_M(t) = \frac{\sum_{i=0}^{n-1} h_i P_i N_{i,M}(t)}{\sum_{i=0}^{n-1} h_i N_{i,M}(t)} \quad (D.8) \]

In the above equation, \( h_i \) are positive real numbers (weights). The integral B-spline curve is a special case of the rational, obtained by setting \( h_i=1 \) and observing (D.5). It can be verified from (D.8) that a rational B-spline curve is tangent to the first and the last segment of the control polygon. Rational B-spline curves enjoy all properties of integral B-spline curves.

Now we introduce the approximation technique. Let \( R_N(t) \) be a rational B-spline curve, \( N \) its order and \( R \) the control polygon with \( n \) vertices \( R_i, 0 \leq i \leq n-1 \) and \( T \) is the open knot vector \( \{t_0, t_1, \ldots, t_{n+N-1}\} \). The parameter \( t \) may be normalized so that it takes values in the interval \([0, 1]\). Our objective is to approximate this rational B-spline curve using a second order integral B-spline curve \( S_2(t) \) with a control polygon \( P \) having \( m \) vertices \( P_i, 0 \leq i \leq m-1 \) and an open knot vector \( S_2(s_0, s_1, \ldots, s_m) \). Here we determine \( m, P_i \) and \( s_j \) such that interpolation of position and first parametric derivative at both ends and the following condition are satisfied

\[ |R_N(t) - S_2(t)| < \epsilon \quad t \in [0, 1] \quad (D.9) \]

where \( \epsilon \) is a prescribed tolerance. The continuity conditions imposed at the ends of the curve are

\[ S_2(0) = R_N(0) \quad (D.10) \]
\[ S_2'(0) = R_N'(0) \quad (D.11) \]
\[ S_2(1) = R_N(1) \quad (D.12) \]
\[ S_2'(1) = R_N'(1) \quad (D.13) \]

where the prime denotes differentiation with respect to \( t \).

By evaluating equations (D.1) and (D.6) at \( t=0, 1 \) we obtain

\[ P_0 = R_0 \quad (D.14) \]
\[ P_{m-1} = R_{n-1} \quad (D.15) \]
\[ P_1 = R_0 + R_N'(0)[s_2 - s_1] \quad (D.16) \]
\[ P_{m-2} = R_{n-1} - R_N(1)[s_m - s_{m-1}] \]  

\[ (D.17) \]

The approximating control polygon is determined by an iterative process. During the first iteration, a knot vector is chosen with no internal knots. The remaining control points are determined by interpolating position at \( m-4 \) distinct points, \( \xi_2, \ldots, \xi_{m-3} \), called nodes and given by \( \xi_i = s_{i+1} \). Interpolating \( R_N(t) \) at \( \xi_i \) we obtain

\[ \sum_{j=0}^{m-1} P_j N_{j,2}(\xi_i) = R_N(\xi_i) \]  

\[ (D.18) \]

which is transformed to

\[ \sum_{j=2}^{m-3} P_j N_{j,2}(\xi_i) = R_N(\xi_i) - \sum_{j=0}^{1} P_j N_{j,2}(\xi_i) - \sum_{j=m-2}^{m-1} P_j N_{j,2}(\xi_i) \]  

\[ (D.19) \]

From this equation we can form the following linear system by setting \( i = 2, \ldots, m-3 \)

\[ [N] \{P\} = \{R\} \]  

\[ (D.20) \]

The coefficient matrix \([N]\) associated with this system is non-singular and banded [DeBoor 78]. By solving the system of equations we determine the unknown control points \( P_2, \ldots, P_{m-3} \).

After solving \((D.20)\) both B-spline curves are sampled and condition \((D.9)\) is evaluated at isoparametric points. If this condition is not satisfied at a particular point \( t_p \), a new knot at \((s_j + s_{j+1})/2\) is added to the knot vector, where \( s_j \leq t_p < s_{j+1} \). If there are more than one such points within a span only one knot is added at the middle of that span. If there are no new knots added to the knot vector this indicates that the iterative process has converged. Otherwise the next iteration begins, in which the updated knot vector is used as the starting knot vector. If \( R_N(t) \) is an integral B-spline curve a tight global upper bound for the position error, valid for all isoparametric points, can be established using the convex hull property [Patrikalakis 89c].
References

[ADINA 84] ADINA Engineering. 

Accuracy Estimates and Adaptive Refinements in Finite Element 
Computations. 

M. A. 
Robust, Geometrically Based, Automatic Two-Dimensional Mesh 
Generation. 
International Journal for Numerical Methods in Engineering 

[Baldwin 85] Baldwin K. H., Schreyer H. L. 
Automatic Generation of Quadrilateral Elements by a Conformal 
Mapping. 

[Bathe 86] Bathe K. J. 
Some Advances in Finite Element Procedures for Nonlinear Structural 
and Thermal Problems. 
Computational Mechanics and Trends, Proceedings of Symposium on 
Future Directions of Computational Mechanics, ASME, Winter 
Annual Meeting, A. K. Noor (Editor), ASME, New York :183-216, 
1986.

[Blum 67] Blum H. 
A Transformation for Extracting New Descriptors of Shape. 
Models for the Perception of Speech and Visual Form, Weinant Wathen- 

[Blum 73] Blum H. 
Biological Shape and Visual Science (Part I). 

[Blum 78] Blum H., Nagel R. N. 
Shape Description Using Weighted Symmetric Axis Features. 

[Boissannat 84] Boissannat J. D. 
Geometric Structures for Three-Dimensional Shape Representation. 

[Bookstein 79] Bookstein F. L. 
The Line Skeleton. 
[Briggs 87] Briggs W. L.
_A Multigrid Tutorial._

[Brown 81] Brown P. R.
A Non-Interactive Method for the Automatic Generation of Finite Element Meshes Using the Schwarz-Christoffel Transformation.

[Bykat 76] Bykat A.

[Bykat 83] Bykat A.
Design of a Recursive, Shape Controlling Mesh Generator.

[Calabi 68] Calabi L., Hartnett W. E.
Shape Recognition, Prairie Fires, Convex Deficiencies and Skeletons.

[Carey 84] Carey G. F., Oden T.
_Finite Elements, Computational Aspects Volume III._

[Cavendish 74] Cavendish J. C.
Automatic Triangulation of Arbitrary Planar Domains for the Finite Element Method.

[Cavendish 84] Cavendish J. C., Hall C. A.
A New Class of Transitional Blended Finite Elements for Analysis of Solid Structures.

[Cavendish 85] Cavendish J. C., Field D. A., Frey W. H.
An Approach to Automatic Three-Dimensional Finite Element Mesh Generation.
[Chae 88] Chae S. W.

[Chen 83] Chen, Y. K., Kutt L. M., Piaszczyk C. M., Bieniek M. P.
Ultimate Strength of Ship Structures.
Society of Naval Architects and Marine Engineers Transactions 91:149-168, 1983.

[Cohen 80] Cohen H. D.
A Method for the Automatic Generation of Triangular Elements on a Surface.

[Collins 73] Collins R. J.
Bandwidth Reduction by Automatic Renumbering.

[Cook 74] Cook W. A.
Body Oriented (Natural) Coordinates for Generating Three Dimensional Meshes.

[Crawford 87] Crawford R. H., Waggenspack W. N., Anderson D. C.
Composite Mappings for Planar Mesh Generation.

[Davis 77] Davis L. S.
Understanding Shape: II. Symmetry.

[DeBoor 72] De Boor C.
On Calculating with B-Splines.

[DeBoor 78] De Boor C.

[Duda 73] Duda R. O., Hart P. E.
Pattern Classification and Scene Analysis.
[Eades 88] Eades P.
Symmetry Finding Algorithms.

[Farouki 87] Farouki R. T.

[Farouki 89a] Farouki R. T., Neff C. A.
Algebraic Properties of Plane Offset Curves.

[Farouki 89b] Farouki R. T., Neff C. A.
Analytic Properties of Plane Offset Curves.

[Faux 79] Faux I. D., Pratt M. J.
*Computational Geometry for Design and Manufacture.*

[Fenves 86] Fenves S. J.
A Framework For Cooperative Development of a Finite Element Modeling Assistant.

[Field 86] Field D. A.
Implementing Watson’s Algorithm in Three Dimensions.

[ Fortune 86] Fortune S.
A Sweepline Algorithm for Voronoi Diagrams.

[Frederick 70] Frederick C. O., Wang Y. C., Edge F. W.
Two-Dimensional Automatic Mesh Generation For Structural Analysis.

[Frey 87] Frey W. H.
[Gordon 73] Gordon W. J.
Construction of Curvilinear Coordinate Systems and Applications to Mesh Generation.

[Haber 81] Haber R. B., Shephard M. S., Abel J. F., Gallagher R. H., Greenberg D. P.
A Generalized Two-Dimensional, Graphical Finite Element Preprocessor Utilizing Discrete Transfinite Mappings.

[Haber 82] Haber R. B., Abel J. F.
Discrete Transfinite Mappings for the Description and Meshing of Three-Dimensional Surfaces Using Interactive Computer Graphics.

[Heide 84] Heide S. S.
A Hierarchical Representation of Shape from Smoothed Local Symmetries.

[Herrmann 76] Herrmann L. R.
Laplacian-Isoparametric Grid Generation Scheme.

[Hinton 79] Hinton E., Owen D. R. J.
*An Introduction to Finite Element Computations.*

[Hoffmann 88] Hoffmann C.
A Dimensionality Paradigm for Surface Interrogations.
*Computer Science, Purdue University* TR 88-837, 1988.

[Imafuku 80] Imafuku I., Kodera Y., Sayawaki M.
A Generalized Automatic Mesh generation Scheme for Finite Element Method.


[Joe 86a] Joe B.
Delaunay Triangular Meshes in Convex Polygons.
[Joe 86b] Joe B., Simpson, R. B.
Triangular Meshes for Regions of Complicated Shape. 
*International Journal for Numerical Methods in Engineering*

[Kardestuncer 87] Kardestuncer H., Norrie D. H.
*Finite Element Handbook.*

Towards Automatic Finite Element Analysis. 

[Kirkpatrick 79] Kirkpatrick D. G.
Efficient Computation of Continuous Skeletons. 

[Langtangen 89] Lanftangen H. P.

[Lee 81] Lee D. T., Drysdale R. L.
Generalization of Voronoi Diagrams in the Plane. 

[Lee 82] Lee D. T.
Medial Axis Transformation of a Planar Shape. 

[Lee 84] Lee Y. T., De Pennington A., Shaw N. K.
Automatic Finite-Element Mesh Generation from Geometric Model a Point-Based Approach. 

[Lo 85] Lo S. H.
A New Mesh Generation Scheme for Arbitrary Planar Domains. 

[Lorensen 80] Lorensen W.
Grid Generation Tools for the Finite Element Analyst. 

[MACSYMA 85] Symbolics Inc.
*VAX UNIX MACSYMA Reference Manual.* 
<table>
<thead>
<tr>
<th>Reference</th>
<th>Author(s)</th>
<th>Title</th>
<th>Publication Details</th>
</tr>
</thead>
</table>
[Patrikalakis 89a] Patrikalakis N. M., Gürsoy H. N.
Shape Feature Recognition by Medial Axis Transform.

[Patrikalakis 89b] Patrikalakis, N. M., Prakash, P. V.
Intersections for Trimmed Surface Patches.
8th International Symposium on Offshore Mechanics and Arctic
Engineering, The Hague, The Netherlands. VI, Computer

[Patrikalakis 89c] Patrikalakis N. M.
Approximate Conversion of Rational Splines.

[Patrikalakis 89d] Patrikalakis N. M., Bardis L.
Offsets of Curves on Rational B-Spline Surfaces.

[Pavlidis 77] Pavlidis T.
Structural Pattern Recognition.

[Pavlidis 82] Pavlidis T.

[Pegna 87] Pegna J.
Variable Sweep Geometric Modeling.

[Persson 78] Persson H.
NC Machining of Arbitrarily Shaped Pockets.

[Perucchio 82] Perucchio R., Ingraffea A. R., Abel J. F.
Interactive Computer Graphic Preprocessing for Three-Dimensional
Finite Element Analysis.
International Journal for Numerical Methods in Engineering

[Pizer 87] Pizer S. M., Oliver W. R., Bloomberg S. H.
Hierarchical Shape Description Via the Multiresolution Symmetric Axis
Transform.
IEEE Transactions on Pattern Analysis and Machine Intelligence

[Prakash 88] Prakash P. V., Patrikalakis N. M.
Surface-to-Surface Intersections for Geometric Modeling.
Preparata F. P.
The Medial Axis of a Simple Polygon.
*Lecture Notes in Computer Science* Mathematical Foundations of
Computer Science, Goos G., Hartmanis J. (Editors), Springer-Verlag,

Preparata F. P., Shamos M. I.
*Computational Geometry: An Introduction.*

Rank E., Zienkiewicz O. C.
A Simple Error Estimator in the Finite Element Method.

Requicha A. A. G., Voelcker H. B.
Solid Modeling: Current Status and Research Directions.

Rivara M. C.
Algorithm for Refining Triangular Grids Suitable for Adaptive and
Multigrid Techniques.
*International Journal for Numerical Methods in Engineering*

Rosenfeld A.
Axial Representations of Shape.

Rossignac J.R., O'Connor, M.A.
SGC: A Dimension-Independent Model For Pointsets With Internal
Structures and Incomplete Boundaries.

Sadek E.
A Scheme for the Automatic Generation of Triangular Finite Elements.
*International Journal for Numerical Methods in Engineering*

Samet H.
Quadtree and Medial Axis Transform.
*Proceedings, 6th International Conference on Pattern Recognition* IEEE

A General Purpose Two-Dimensional Mesh Generator.

Schroeder W. J., Shephard M. S.
Geometry-Based Fully Automatic Mesh Generation and the Delaunay
Triangulation.
*International Journal for Numerical Methods in Engineering*
[Sederberg 84] Sederberg T. W., Anderson D. C., Goldman R. N. 
Implicit Representation of Parametric Curves and Surfaces. 

[Sederberg 88] Sederberg T. W. 
An Algorithm for Algebraic Curve Intersection. 

[Serra 82] Serra J. 
*Image Analysis and Mathematical Morphology.* 

[Shephard 84] Shephard, M. S., Yerry, M. A. 
Finite Element Mesh Generation For Use With Solid Modeling and Adaptive Analysis. 

[Shephard 85a] Shephard, M. S. 

[Shephard 85b] Shephard, M. S. 

[Shephard 88] Shephard M. S. 
The Specification of Physical Attribute Information for Engineering Analysis. 

[Sloan 87] Sloan S. W. 
A Fast Algorithm for Constructing Delaunay Triangulation in Plane. 

[Sluiter 82] Sluiter, M. L. C., Hansen, D. C. 
A General Purpose Automatic Mesh Generator for Shell and Solid Finite Elements. 

[Srinivasan 87] Srinivasan V., Nackman L. R. 
Voronoi Diagram for Multiply Connect Polygonal Domains I: Algorithm. 

[Strang 73] Strang G., Fix G. 
*An Analysis of the Finite Element Method.* 
[Taig 86] Taig T. C.
Expert Aids to Reliable Use of Finite Element Analysis.

[Tanimoto 81] Tanimoto S. L.
A Method for Detecting Structure in Polygons.

*Numerical Grid Generation Foundations and Applications*.

[Tiller 83] Tiller W.
Rational B-Splines for Curve and Surface Representation.

[Timoshenko 70] Timoshenko S. P., Goodier J. N.
*Theory of Elasticity*.

[Toussaint 89] Toussaint G. T.
Computational Geometry: Recent Developments.

Accurate Solid Modeling Using Polyhedral Approximations.

[Utku 83] Utku S., Melosh R. J.
Errors in Finite Element Analysis.

[Van-Phai 82] Van-Phai, N.
Automatic Mesh Generation with Tetrahedron Elements.

[Watson 81] Watson D. F.
Computing the n-Dimensional Delaunay Tessellation with Applications to Voronoi Polytopes.

[Weiler 85] Weiler K. J.
Edge-Based Data Structures for Solid Modeling in Curved-Surface Environments.
[Weiler 86] Weiler K. J.
*Topological Structures for Geometric Modeling.*

[Wellford 88] Wellford L. C., Gorman M. R.
*A Finite Element Transitional Mesh Generation Procedure Using Sweeping Functions.*

*Cut Loci in Bordered and Unbordered Riemannian Manifolds.*


1989
Private Communication, October.

[Woo 84] Woo T. C., Thomasma T.
An Algorithm for Generating Solid Elements in Objects with Holes.

[Wordenweber 84] Wordenweber, B.
Finite Element Analysis for the Naive User.

[Xia 89] Xia Y.
Skeletonization Via the Realization of the Fire Front’s Propagation and Extinction in Digital Binary Shapes.

[Yerry 83] Yerry M. A., Shephard M. S.
A Modified Quadtree Approach to Finite Element Mesh Generation.

[Yerry 84] Yerry M. A., Shephard M. S.
Automatic Three-Dimensional Mesh Generation by the Modified Octree Technique.
[Yerry 85] Yerry M. A., Shephard M. S.
Trends in Engineering Software and Hardware, Automatic Mesh
Generation for Three Dimensional Solids.

[Zienkiewicz 71] Zienkiewicz O. C., Phillips D. V.
An Automatic Mesh Generation Scheme for Plane and Curved Surfaces
by Isoparametric Coordinates.
*International Journal for Numerical Methods in Engineering* 7:461-477,
1971.

[Zienkiewicz 83] Zienkiewicz O. C., Morgan K.
*Finite Elements and Approximation.*

[Zienkiewicz 89] Zienkiewicz O. C., Zhu J. Z. and Gong N. G.
Effective and Practical h-p Version Adaptive Analysis Procedures for the
Finite Element Method.
*International Journal for Numerical Methods in Engineering*