A COMPARISON OF THREE DIMENSIONAL PHOTOLITHOGRAPHY SIMULATORS

by

John Joseph Helmsen

Memorandum No. UCB/ERL M95/25

10 April 1995

A COMPARISON OF THREE DIMENSIONAL PHOTOLITHOGRAPHY SIMULATORS

by

John Joseph Helmsen

Memorandum No. UCB/ERL M95/25

10 April 1995

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

title p

Abstract

A Comparison of Three Dimensional Photolithography Simulators

by

John Joseph Helmsen

Doctor of Philosophy in Electrical Engineering

University of California at Berkeley

Professor Andrew R. Neureuther, Chair

Methods for negative volume artifact removal and mesh regularization are introduced and applied to triangle based surface advancement simulation in the area of photolithography dissolution in three dimensions. These methods were then compared against the popular cell and advection (level-set) techniques for solving the Hamilton-Jacobi equation. The dissolution problem is modeled by an etch rate that has been defined by the exposure and bake process. The etch rate varies according to its position in the resist. Gel-layer effects are ignored.

An order of magnitude increase in speed and a significant improvement in robustness have been achieved simultaneously for large (> 10,000) triangle ray-based representation of dissolution. This has been achieved by applying spatial decomposition methods and graph theory to loop removal. An octtree has been implemented, which is locally refined about the surface, to organize triangles by spatial coordinates. The triangles can be organized in O(NlogN) time at about 500 triangles/second. Two methods, which employ the octtree, are introduced to remove mesh loops. The first loop removal technique is a general method. The second method is specially oriented at removing loops in photolithography. Heuristic techniques are also introduced for removing thin triangles from a triangular mesh. Heuristics for removing crenulations from a triangular surface are also described.

A benchmark case that is representative of typical photolithography problems is described. A comparison has been performed on this case for three first order methods of simulating photoresist dissolution. These methods were ray-trace triangle advancement, advection contour advancement, and a cell based volume removal method. An evaluation was performed on the basis of speed, memory consumption, two types of accuracy and robustness. The speed and memory comparisons were normalized for equivalent levels of accuracy. Ray-trace was found to be 2x faster than cells and >10x faster than advection. Ray-trace was found to have equivalent memory consumption to cells and <1/10th the amount required for advection. Ray-trace and cells were found to be have greater accuracy than advection along the coordinate axes for similar grid sizes. The cell method demonstrated anisotropic behavior that was not exhibited by the other two methods. A new technique for improving the accuracy of the advection method is also introduced.

Andrew R Neureuther

Professor Andrew R. Neureuther

Committee Chairman

MINOR EDITING OF THIS VERSION
WAS CARRIED OUT BY JOHN HELMSEN
APRIL 10, 1995

Dedication

To my parents

Dr. and Mrs. Ralph and Mary Lee Helmsen

and my two brothers

Joseph and Eric

Table of Contents

List of Figures

Acknowledgments

My deepest gratitude is to my advisor, Professor Andrew R. Neureuther, for his open door policy, unwavering encouragement and invaluable guidance throughout the course of my studies. I especially wish to thank him for his willingness to tackle and solve hard problems with me where others have given up.

Likewise, Prof. Carlo Sequin has played an important role in this work by lending his time as a critical editor of my dissertation. In addition, I am also indebted to Prof. Phillip Colella for volunteering to be on my thesis committee as my outside advisor. I also would like to thank Prof. Costas Spanos for serving on my Ph.D. Qualifying Examination committee and Prof. William Oldham, who would have, but was unable to attend. Sincere gratitude is expressed to Prof. James Sethian, who, while he was not on any of my committees, has provided excellent technical advice.

Thanks is extended to Dr. Kenny Toh and Dr. Edward Scheckler for their previous work in this field. Special thanks is given to Robert Wang and John Sefler for working with me in the TCAD group. I also wish to thank the rest of Professors Neureuther's and Oldham's research group: Dr. David Newmark, Derek Lee, Bob Socha, Marco Zuniga, Michael Yeung, Bernice Lum, Min Zhou, Charles Fields, Andrew Zenk, Edita Tejnil, Rich Schenker, Anita Lee, Dr. Alfred Wong, Dr. Rich Ferguson, Dr. Nelson Tam, Dr. Alex Wong and Dr. Bill Partlo. Their encouragement, friendship and support. is appreciated.

Thank you Rita Tidwell, Rob McNicholas and Bruce Beattie for assisting me in the payroll maze and keeping the computers running.

Last, but not least, I thank my family for their love and support through the years. The sacrifices they have made and the guidance they have given is greatly appreciated.

The financial support provided by SEMATECH, the Semiconductor Research Corporation and the National Science Foundation is gratefully acknowledged.

Chapter 1 Introduction

1.1 Dissertation Motivation

The purpose of this work is to advance the state of the art in practical three dimensional simulation of fabrication processes used in the construction of integrated circuit chips. This work will focus on the dissolution step of the photolithographic process. Photolithography is one of the most widely used and studied integrated circuit fabrication processes. It is responsible for the generation of patterns on the surfaces of semiconductor wafers. These patterns define the boundaries of the semiconductor devices to be formed on the wafer. Because photolithography is fundamental to almost all semiconductor processing steps, nearly all advancements of the state of the art in photolithographic techniques directly influence the manufacture of smaller and more efficient integrated circuits. A significant number of present investigations into the dynamics of photolithography processes involve the careful consideration of three dimensional effects. Some of these effects are line foreshortening in MOSFET transistors and distortions of corners and contact cuts. These effects are enhanced by the ever decreasing ability of optical wavelengths to keep up with decreasing device sizes in modern process technology. Properly representing these structures is beyond the scope of two dimensional simulators, which are more useful for finding the cross-section of long lines. The need for accurate three dimensional simulation of photoresist dissolution is clear.

Integrated circuit manufacturing is an expensive process due to the ever increasing capital requirements of production facilities. It is far more profitable to employ semiconductor fabrication equipment to synthesize a profitable product, rather than delaying this activity by performing experimental tests and calibrating and measuring the fabrication process. While experimental work on the production process can never be completely removed, the use of computer simulation can significantly reduce fabrication line downtime. In addition, the ability of manufacturers to design new fabrication processes cheaply is significantly enhanced through the use of simulation.

1.2 Evolution of Three Dimensional Photolithography Simulation

During the 1980's many individuals in the electrical engineering community have seen the need for accurate three dimensional photolithography dissolution simulation. This has resulted in a proliferation of simulators with varying degrees of effectiveness. The simulators that were developed fell into three general classes. These classes are ray-trace, cells and level-sets. A comparison between ray-trace methods, cell methods and techniques for segment advancement without the use of rays was performed by R. Jewett [1], but the level-set technique was not included. The ray-trace method simulates the advancement of the photoresist by explicitly representing the surface with a mesh of triangles or some other geometrical object. To simulate the evolution of the surface during dissolution, the surface is advanced by moving mesh points according to the least time principle, which was first applied to photolithography by P. Hagouel [2]. Implementations of this algorithm have been performed by T. Matsuzawa [3], L. Jia. [4], K. Lee [5] and E. Barouch [6]. The most accurate first-order advancement mechanism to date is the recursive advancement algorithm by K. Toh [7]. This technique allows each point to subdivide its time step if it is advancing through an area prone to numerical error. This allows for advancement that contains a significant degree of accuracy. Toh's advancement method is the one employed in this dissertation for the purpose of comparing ray-trace against other methods. A serious disadvantage exists in this method, however, due to the formation of non-physical negative volumes called 'loops'. Maintaining the integrity of the mesh is also difficult.

E. Barouch [6] has proposed a solution employing B-splines, but this method requires an excessive increase in CPU cycles per simulation. Therefore, accurate loop removal (or loop prevention) and mesh maintenance remain as key problems in ray methods.

The second method of investigation is to divide the simulation region into many small sections called cells. Cells that are in contact with the developer are removed at a time determined by their etch rate and the status of surrounding cells. The first cell method to be applied to photolithography was developed by F. Dill [8] in two dimensions. Three dimensional cell methods have been employed in photolithography by J. Bauer [9], F. Jones [10], Y. Hirai [11], W. Henke [12], J. Pelka [13], K. Toh [7] and E. Scheckler [14]. The cell method employed in this dissertation was developed by E. Scheckler [14]. Some cell methods, such as those implemented by Henke and Scheckler, allow certain cells to continue etching past a zero amount of photoresist in a cell, creating a cell with a negative amount of photoresist. The negative photoresist amount is then set to zero and photoresist from neighboring cells is removed to keep the total amount of resist constant. This process called 'spillover', and has been shown to accelerate the execution time of cell methods significantly [13][14]. Cell methods are far easier to implement than ray methods. Cell methods also have no loop or mesh problems. They do, however, suffer from a specific lack of accuracy that cannot be corrected by refining the grid size. In cell methods, the etch rate is enhanced along preferred directions as a result of the algorithm used. Etching problems with spherical analytical solutions generate results that resemble polyhedra. This effect is called faceting. Faceting remains a key problem for all known cell methods.

In 1991, E. Barouch [15], by borrowing heavily from techniques invented by S. Osher and J. Sethian [16], introduced a method of photolithographic simulation which is based on existing numerical techniques in fluid mechanics. This method was also presented independently by M. Komatsu [18]. By representing the surface of the photoresist as a single valued contour of a monotonic function that is defined over the whole simulation region, photolithographic dissolution simulation can be performed by solving the Hamilton-Jacobi equation using simple upwind differencing schemes. This method requires no loop removal or mesh maintenance, and it has no difficulties with facet formation. Concerns have been expressed about the use of this method [17]. Because the surface is advanced by performing a computation over the entire simulation space at each time step, this method is not particularly fast. One implementation of the level-set method required 5 minutes on a supercomputer for a 100x100x100 grid [18]. This amount of computation can be excessive for the typical engineer who only has access to a workstation. Reasons also exist to suspect the accuracy of the level-set method as applied to photoresist problems, since standing waves nulls in photoresist contain high second derivatives in the etch rate. This type of behavior can cause many level-set implementations to become inaccurate [19].

A summary of existing three dimensional simulation methods is given below:

3D Photolithography Simulation Programs

Program	Date	Models and Algorithms	Availability	Comments
TRIPS-I	1985/87	Ray-Trace	Hitachi Internal	
Jia et. al.	1987	Ray-Trace		
3D-EBLS	1991	Ray-Trace	Samsung Internal	
Barouch et. al.	1989	Ray-Trace	Princeton	B-Spline
SAMPLE-3D	1990	Ray-Trace	U. C. Berkeley	Recursive Euler
LITHSIM	1980/91	Cell-Method	E. Germany	
RD3D	1980	Cell-Method	IBM Internal	
PEACE	1987	Cell-Method	Matsushita Internal	
SOLID	1990	Cell-Method	Silvaco	Integer Cells
CRATER	1991	Cell Method	U. C. Berkeley	Spillover
Barouch et. al.	1991	Level-Set	Princeton	
Komatsu	1993	Level-Set	Nikon Internal	

1.3 Research Goals and Dissertation Outline

The purpose of this work is to investigate key issues and develop methods that advance the state of the art in practical three-dimensional topography simulation, specifically for photoresist dissolution processes. Effort will be focused on algorithms and techniques that have been designed to improve the functionality of the ray-trace and level-set methods.

The first issue that was addressed in the course of the research, was to develop a more efficient and robust loop removal technique to improve the ray-trace method. The first part of the loop removal method that was considered was the intersecting triangle pair locator. Although this vital step was performed in a previous implementation, it was expected that a better method of sorting the triangles would yield positive results. A data structure, called an octtree, was tested to see if it might improve efficiency. The octtree sped up the loop removal process by 10-100 times, depending on the number of triangles. The octtree was also seen to have useful properties for many other applications. A decision was now made to concentrate on the robustness of the loop remover. The second part of the delooper, the triangle splitter, was constructed, since the loops had to be separated from the surface for removal. Once this occurred, a simple binary labeling scheme, which gave way to the winding number labeling scheme, was implemented that found the offending mesh pieces and removed them.

Once the delooper was implemented and tried on photoresist surfaces, it was found that the mesh could not be advanced further after deloop, since the new nodes that were created could not be given interpolated rays. A previous loop removal implementation solved this difficulty in two dimensions by splitting the surface into two parts and advancing each independently. To allow the surfaces to advance cleanly in three dimensions, a new method of interpolating the rays was developed. The triangle splitter was removed from the winding number delooper and replaced with a new labeling method that traversed the segments instead of the triangles. To maintain connectivity between separated parts of the mesh, for the purpose of performing additional deloops, the integer labeling scheme was invented. This deloop method, now called resist deloop, has run very well on ray-trace problems ever since. However, a new phenomena was discovered that was not apparent before due to the lack of loop removal. This problem was ray-scattering.

The resist delooper has difficulty with the type of loops that are formed by ray-scattering. These loops are small, complex, contain many thin triangles, and tend to only get worse as continued advancement takes place. Thin triangle removal was considered as a strategy to prevent loop formation, since methods already existed in two dimensions to remove thin triangles. In addition, thin triangles is another triangle mesh difficulty that is just as serious as deloop. Short distance point motion was introduced to handle curved surfaces. The problem of crenulation was also examined and its relationship to thin triangle removal was explored.

After significant effort had been put forth into developing triangle based advancement methods, simple surface advancement techniques from fluid mechanics began to appear in the photolithography community. It was decided to rigorously compare the methods, so that the relevant advantages and disadvantages for each method could be found. After the level-set algorithm had been implemented, new techniques were being developed in the fluid mechanics community for advancement of interfaces between materials using level-set techniques that contain iteration. A simple iterative scheme was tried in the original level-set code and was found to be very successful in simulating photoresist profiles with high accuracy.

The chapters are divided as follows:

Chapter 1 gives a general overview of the need for fast, accurate and robust photolithography dissolution simulation in three dimensions.

Chapter 2 is a short overview of the photolithography process.

Chapter 3 outlines the mathematical methods necessary for the ray-trace, level-set and cell photolithography simulators. Each of these methods is shown to be a plausible approach to solving the Hamilton-Jacobi equation.

Chapter 4 provides the exact details of the ray-trace, level-set and cell implementations. A description for the construction of an iterative and possibly second order method from the original first order level-set method is also provided.

Chapter 5 introduces two new techniques, which were invented by the author, for removal of negative volume areas formed during advancement of triangle meshes. One method is general, the other is specifically designed for ray-trace advancement. The general method is shown to also have the capacity to implement volume set operations.

Chapter 6 introduces some approaches for better mesh maintenance for semiconductor topographical process simulation algorithms that employ triangles based upon the observations of the author. Specific attention is paid to mesh maintenance involving thin triangles and crenulated surfaces.

Chapter 7 reports the results of the comparison between the ray-trace, cell and level-set methods. An analysis of the performance of each method is given. Specific techniques for further improvements are suggested.

Reference for Chapter 1

[1] R. Jewett, P. Hagouel, A. Neureuther and T. Van Duzer, "Line-Profile Resist Development Simulation Techniques", *Polymer Engineering and Science*, vol. 17, no. 6, June 1977.

[2] P. I. Hagouel, *X-ray Lithographic Fabrication of Blazed Diffraction Gratings*, Ph.D. Dissertation, University of California, Berkeley, 1976.

[3] T. Matsuzawa, T. Ito and H. Sunami, "Three-dimensional Photoresist Image Simulation on Flat Surfaces," *IEEE Transactions on Electron Devices*, vol. ED-32, no. 9, pp. 1781-1783, Sep. 1985.

[4] L. Jia, W. Jian-kun and W. Shao-jun, "Three-Dimensional Development of Electron Beam Exposed Resist Patterns Simulated by Using Ray Tracing Model," *Microelectronic Engineering*, vol. 6, pp. 147-151, 1987.

[5] K. Lee, Y. Kim and C. Hwang, "New Three-Dimensional Simulator for Electron Beam Lithography," *1991 International Workshop on VLSI Process and Device Modeling*, pp. 44-45, Oiso, Japan, May 26-27, 1991.

[6] E. Barouch, B. Bradie, H. Fowler and S. Babu, "Three-Dimensional Modeling of Optical Lithography for Positive Photoresists," *Interface '89: Proceedings of KTI Microelectronics Seminar*, pp. 123-136, Nov. 1989.

[7] .K. H. Toh, *Algorithms for Three-Dimensional Simulation of Photoresist Development*, Ph.D. Dissertation, University of California at Berkeley, 1990.

[8] F. Dill, A. Neureuther, J. Tuttle and E. Walker, "Modeling Projection Printing of Positive Photoresists", *IEEE Transactions on Electron Devices*, vol. ED-22, no. 7, July 1975.

[9] J. Bauer, "Modelle fuer den fotolithografischen Prozess," Feingeraetetechnik, vol.

29. pp. 127ff, 1980.

[10] F. Jones and J. Paraszczak, "RD3D (Computer Simulation of Resist Development in Three Dimensions)," *IEEE Transactions on Electron Devices*, vol. ED-28, no. 12, pp. 1544-1552, Dec. 1981.

[11] Y. Hirai, M. Sasugo, M. Endo, K. Ikeda, S. Tomida and S. Hayama, "Three Dimensional Process Simulation for Photo and Electron Beam Lithography and Estimations of Proximity Effects," *Symposium on VLSI Technology, Digest of Technical Papers*, p. 15, 1987.

[12] W. Henke, D. Mewes, M. Weiss, G, Czech and R. Schiessl-Hoyler, "Simulation of Defects in 3-Dimensional Resist Profiles in Optical Lithography," *Microelectronic Engineering*, vol. 13, pp. 497-501, 1991.

[13] J. Pelka, "SOLID: Comprehensive Three Dimensional Simulation Program for Optical Microlithography", *Information Brochure, Fraunhofer-Institut fur Mikrostrukturtechnik*, May 1990.

[14] E. Scheckler, *Algorithms for Three-Dimensional Simulation of Etching and Deposition Processes in Integrated Circuit Fabrication*, Ph.D. Thesis, University of California, Berkeley, 1991.

[15] E. Barouch, J. Cahn, U. Hollerbach and S. Orszag, "Numerical Simulation of Submicron Photolithographic Processing," *Journal of Scientific Computing*, vol. 6, no. 3, pp. 229-50, 1991.

[16] S. Osher and J. Sethian, "Fronts Propagating with Curvature-Dependent Speed: Algorithms Based on Hamilton-Jacobi Formulations," *Journal of Computational Physics*, vol. 79, pp. 12-49, 1988.

[17] J. Helmsen, M. Yeung, D. Lee and A. Neureuther, "SAMPLE-3D Benchmarks Including High-NA and Thin Film Effects", *SPIE Optical/Laser Microlithography*

VII, vol. 2197, pp. 478-88, 1994.

[18] M. Komatsu, "Three Dimensional Resist Profile Simulation", *SPIE Optical/Laser Microlithography VI*, vol. 1927, pp. 413-26, 1993.

[19] C. Hirsch, *Numerical Computation of Internal and External Flows*, Wiley, New York, 1988.

Chapter 2 Photolithography Modeling and Simulation

2.1 Introduction

This chapter is intended to introduce the standard projection printing optical photolithography process to the uninformed reader, and to describe the simulators that were employed in generating the dissolution etch rates for the surface advancement methods. It is not intended to serve as a survey of the field. For more detailed information, please reference *Introduction to Microlithography*, published by the American Chemical Society [1] and the forthcoming *Simulation of Semiconductor Lithography and Topography* by A. Neureuther [2]. Only the Berkeley lithography simulators are mentioned in this chapter.

2.2 Processes in Photolithography Modeling

The functionality of an integrated circuit is determined by both the electrical properties of the materials that have been placed on the wafer during manufacturing, and the particular geometry in which the materials are arranged. The purpose of lithographic processes is to provide a low cost method forming patterns on the surface of a wafer. This is done via a chemical that is sensitive to light bombardment. The chemical is called ‘photoresist’. Exposure to light changes the properties of the resist, so that part of it can be conveniently removed from the surface of the chip. The part that remains on the chip forms a resistive layer that allows other manufacturing processes to affect only specific sections of the surface of the integrated circuit. Two classes of photoresist exist. Positive resists, are resists that operate by becoming easy to remove when exposed to light, and negative resists, which are initially easy to remove unless exposed to light. The word light, in this case, is not restricted to only visible light. Photoresists exist for a variety of exposure systems, including ultraviolet light, electron beams and ion beams.

The three most important steps in photolithography are the exposure, post exposure baking, and removal of the photoresist. These steps are shown in Figure 1.along with the corresponding Berkeley simulators that model each step. The first step, exposure, is shown as two separate steps, since many simulators that model exposure often concentrate on computing, either the properties of incoming beam, or the effects the beam, on the chemistry of the resist. SPLAT [3] and BLEACH [5] are examples of this division of labor. The second step is an optional step in photolithography development. Post-exposure bake smooths out large variations in active compounds via diffusion. This technique is useful for suppressing standing waves and other effects of interference. The third step in photolithography simulation is dissolution simulation. In dissolution, an etchant is applied to the photoresist that causes the exposed parts (or in the case of negative photoresist, the unexposed parts) to be removed. This removal step has been simulated very successfully in two dimensions in SAMPLE. A listing of the most popular three dimensional methods of simulating photolithography dissolution is given in Chapter 3. The three Berkeley methods that have been implemented in three dimensions to solve the dissolution problem are DEVELOP (the ray-trace method), ADVECT (the level-set method) and CRATER (the cell method). These methods have been incorporated into SAMPLE-3D.

2.2.1 Aerial Imaging

The most popular method of exposing the photoresist in modern photolithography is the projection printing method shown in Figure 2. Projection printing is the preferred illumination process for high volume production, and is the illumination

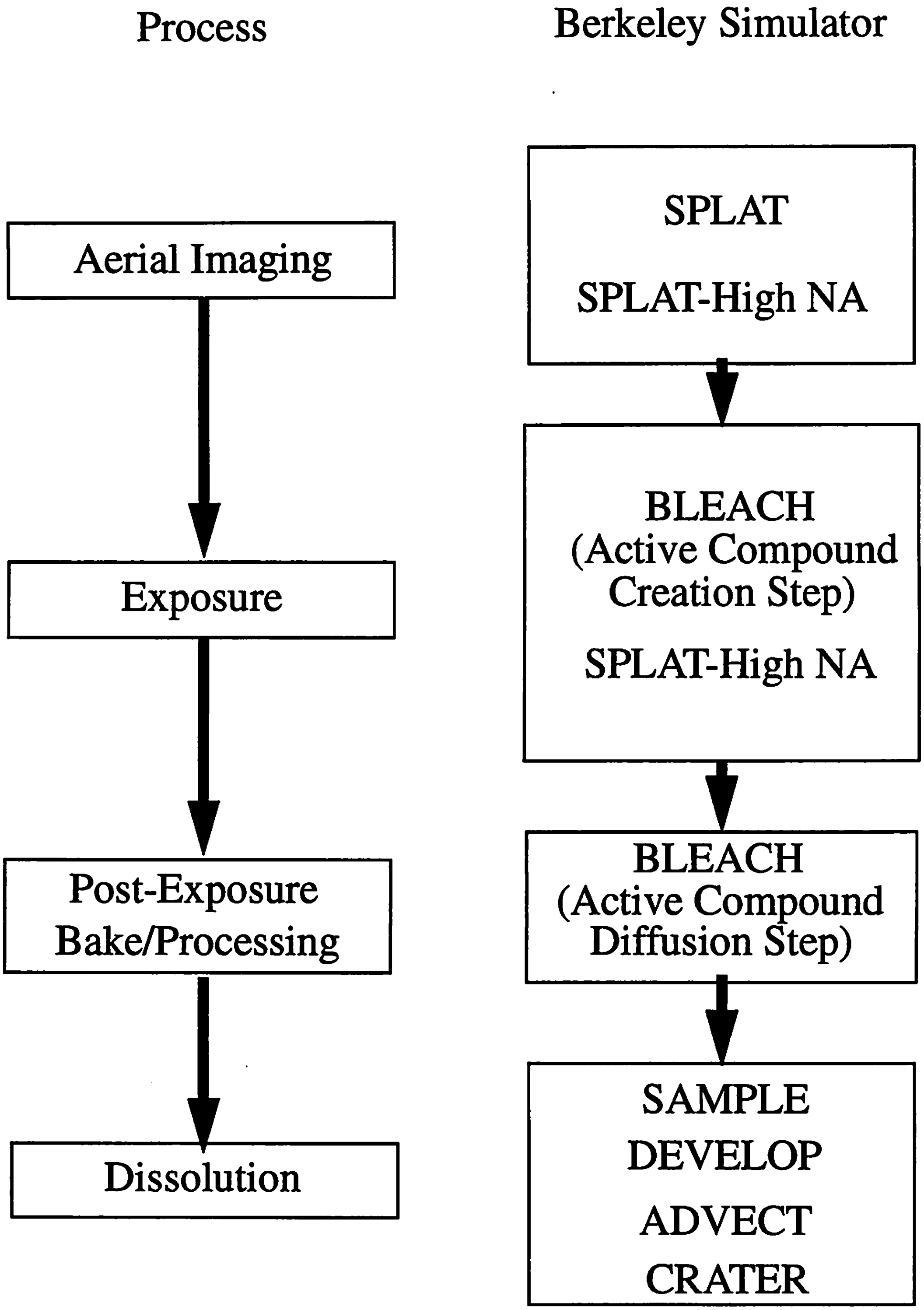


Figure 1) The Photolithographic Process Steps and their Associated Simulators

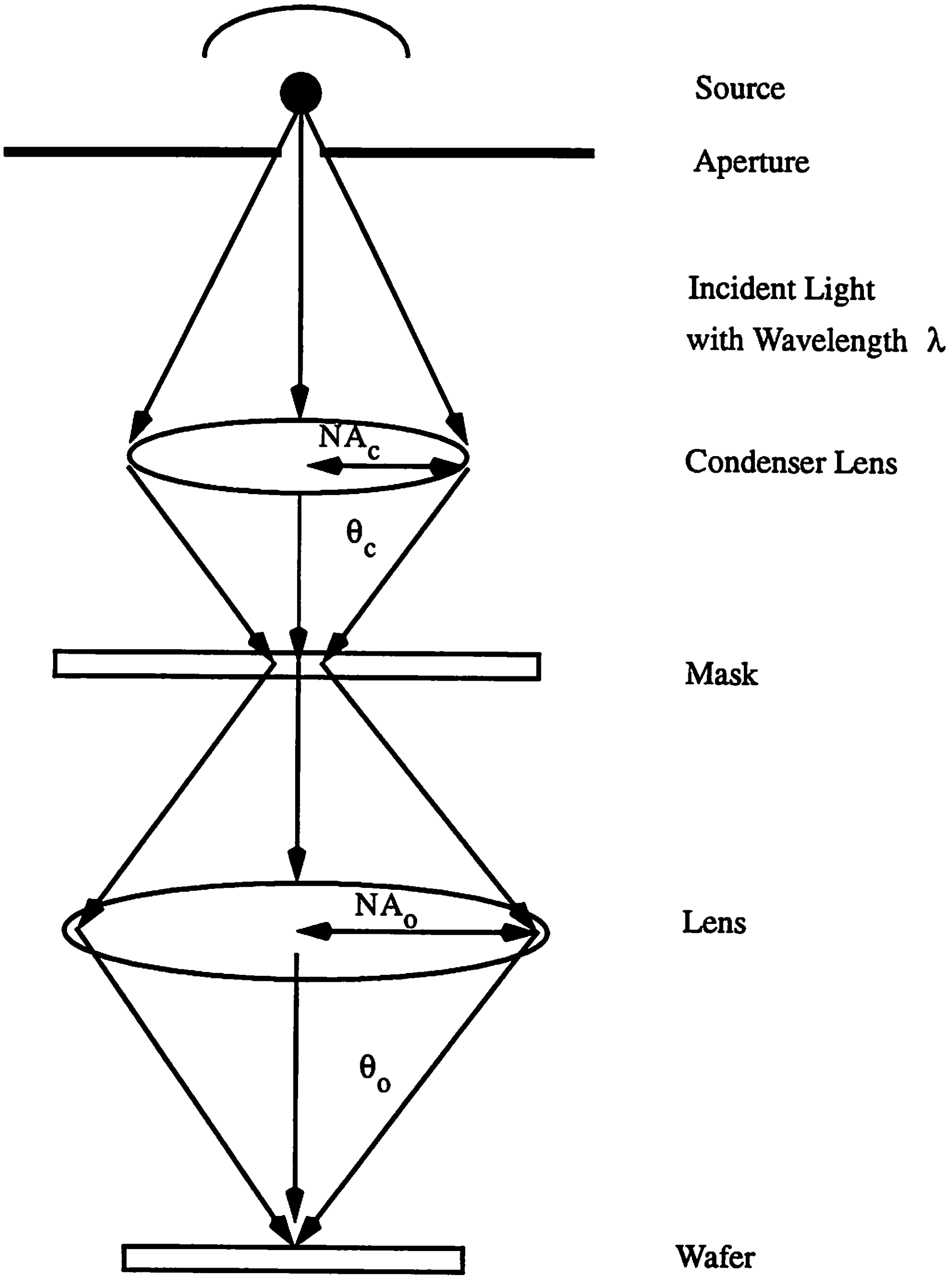


Figure 2) The Projection Printing Method of Illumination

model used for all examples generated for this dissertation. In projection printing, light arrives from a mercury arc lamp, laser or other source at the top of the picture, and illuminates the mask. The wavelength of light in this system is signified by the symbol λ. The light is focused through the condenser lens and strikes the mask as plane waves from various angles. The angles of the incoming plane waves are limited by the size of the condenser lens. Once the waves pass through the condenser lens and strike the mask, they scatter in all directions. The angles θ_c and θ_o define two other important attributes of the illumination. The 'numerical apertures' of the lenses NA_c and NA_o are given as:

$$NA_c = \sin\theta_{(} \qquad \text{[EQ: 1]}$$

$$NA_o = \sin\theta_o \qquad \text{[EQ: 2]}$$

Numerical aperture is important, since the minimum feature size or 'linewidth' (LW) that can be generated by a projection illumination system is

$$LW = \frac{k_1\lambda}{NA_o} \qquad \text{[EQ: 3]}$$

where $k_1$is a value between 0.6 and 0.8 that depends upon the resist technology. Another way that numerical aperture affects the imaging process is via partial coherence. Partial coherence models the amount of spreading in each mask pixel as it is transmitted to the wafer. The shape of the spread out pixel is similar to a gaussian at low partial coherence and to the equation $\sin x/x$ for large values of partial coherence. It is desirable to reduce the width of the gaussian function be decreased as much as possible so that the contrast of the projected image can be improved. The spread can be decreased by increasing the partial coherence. This cannot be done indefinitely, however, since 'ringing' may occur. The gaussian like spread of the pixel becomes so sharp that sidelobes form and affect the clarity of the image. The partial coherence is given by

$$\sigma = \frac{NA_c}{NA_o} \qquad \text{[EQ: 4]}$$

where σ is the partial coherence. Desirable values for the partial coherence range from 0.5 to 0.7. Finally, the numerical aperture affects the focus range of the illumination system. If the wafer is aligned slightly above or below the possible focus range, image degradation will occur. The amount that the wafer can deviate from the focal plane in either vertical direction without significant degradation is

$$DoF = \pm\frac{\lambda}{2(NA)^2} \qquad \text{[EQ: 5]}$$

where DoF represents the depth of focus. This is equivalent to a defocus allowance of one Rayleigh unit in either direction. This lack of focus is of concern, since this represents a trade off between smaller feature sizes, as shown by [EQ: 3], and the effort required to maintain focus. For present day illuminators, whose numerical apertures are approaching 0.7, this defocusing effect becomes even more severe, since the thickness of the resist approaches the depth of focus range.

Other methods, such as proximity printing, filter the incoming light through a mask that rests very near the surface of the resist. The image on the surface of the wafer is generated by the shadow that the mask casts. Proximity printing is not used significantly in modern production, because of the difficulties involved in positioning the mask above the photoresist.

The simulator that is employed in the Berkeley Topography Utilities for modeling these effects is SPLAT, which was developed by K. Toh [3], and most recently improved upon by D. Lee and M. Yeung [4] to handle numerical apertures of 0.7. SPLAT uses the parameters of the illumination system to generate an intensity plot of the projected mask image at the surface of the resist. Of course, merely computing the image at the surface of the resist is not entirely accurate, as the resist has a finite thickness, and effects related to the focal length may occur. This motivated the creation of High-NA SPLAT [4]. Some effects that occur in High-NA systems, such as changing resist refractive indices during exposure, or effects due to polarization, are not implemented in High-NA SPLAT, but comparisons between High-NA SPLAT and other simulators that solve the electric and magnetic fields in the resist more directly, have shown a great degree of similarity.

2.2.2 Photoactive Compound Creation and Diffusion

Most photoresists that are employed in integrated circuit manufacturing today are polymer resists. These photoresists are made of long chains of polymer material, that contain chemical side groups that are photochemically sensitive. In positive photoresist, the impact of a photon on the appropriate branch of the molecule causes a reaction that releases a molecule of acid. This acid then serves as a catalyst that breaks down the surrounding polymer chains and allows them to be removed in the dissolution step. In negative photoresist, the photons release materials that promote cross-linking between the resist chains. These cross-linking events enhance the ability of the photoresist to resist being dissolved during dissolution. In most resists, the amount of compound created at each location is related to the intensity of exposure at each point. Both of these reaction substances that are created during exposure are called ‘photoactive compound’ or ‘PAC’.

The exposure dependent optical properties of positive resists are generally described by the Dill [5] ABC parameters. A is the bleachable absorption, B is the nonbleachable absorption and C is the bleach rate at the exposure rate being used. These parameters allow the fraction of unexposed photoactive compound ‘M’ (0<M<1) is the resist to be calculated. The local absorption constant α at position r and time t is given by

$$\alpha(r, t) = AM(r, t) + B \qquad \text{[EQ: 6]}$$

where M(r,t) is the value of M at a specific location and time. The destruction of the remaining compound is governed by the equation:

$$\frac{d}{dt}M(r, t) = -I(r, t)\,M(r, t)\,C \qquad \text{[EQ: 7]}$$

BLEACH computes the PAC values from the image provided by SPLAT through the use of a series of one dimensional simulations that are oriented vertically in the resist. The image file is employed as the upper boundary condition on these simulations. High-NA SPLAT has to perform these calculations on its own, since the vertical approximation is no longer valid.

One important behavior of the PAC in the photoresist has occurred that must be mentioned. Because the typical refractive index of photoresist is about 1.68, and the refractive index of silicon is about 4.71, significant reflection of deposited energy may occur. This reflective energy produces interference with incoming energy. Therefore, an interference pattern or ‘standing wave’ may occur in the resist. The difference in exposure between the standing wave peaks and nulls is often about 8:1. Because of the non-linear relationship between exposure dose and etch rate in many resists, the ratio between the etch rates in the standing waves can be as much as 50:1. These standing wave etch rate variations significantly affects the performance of various dissolution simulation algorithms.

Once photoactive compound is created, it may tend to diffuse through the photoresist on its own or through the application of a post-exposure bake process step. Post exposure bake was invented for the purpose of removing standing waves by E. Walker [6]. Unfortunately, using post exposure bake also decreases the resolution of the photolithography process. It is a standard process in modern integrated circuit manufacturing. BLEACH simulates this effect by performing a convolution of the PAC function with a gaussian function whose width is defined by the user.

2.2.3 Dissolution

After illumination and post-exposure bake, the photoresist is now ready for development. This step will complete the photolithography process by removing the unwanted photoresist and leave the appropriate resist pattern on the wafer. A solution called a developer is applied to the surface of the resist. The penetration of the developer into the resist, and the ability of the resist to dissolve into the developer, is a function of the active compound concentration created during the previous two steps. The final distribution of the active compound allows an etch rate to be computed at each point in the photoresist. This etch rate is an isotropic etch rate, since the polymer chains are randomly oriented and have no crystalline structure. Along most of the surface, the region of interaction between the developer and the photoresist can be approximated as a infinitely thin plane, thus giving no influence of the shape of the surface on the etch rate. There are many models for deriving etch rates from PAC concentrations. One such model is the Kim three parameter model [7]. Given three constants R_1, which represents the fully exposed dissolution rate, R_2, which represents the fully unexposed dissolution rate and R_3, which represents the variation of the etch rate to changes in the PAC concentration M, the isotropic etch rate at every point is given by the equation

$$\frac{1}{R(M)} = \frac{1 - Me^{-R_3(1-M)}}{R_1} + \frac{Me^{-R_3(1-M)}}{R_2} \qquad \text{[EQ: 8]}$$

This equation is used by all of the simulations, other than the analytical examples, that are employed in this thesis.

Reference for Chapter 2

[1] L Thompson, C. Willson and M. Bowden, *Introduction to Microlithography, Second Edition*, American Chemical Society, Washington D.C., 1994.

[2] A. Neureuther, *Simulation of Semiconductor Lithography and Topography*, (unpublished).

[3] K. Toh and A. Neureuther, "Identifying and Monitoring Effects of Lens Abberations in Projections", *SPIE Optical Microlithography VI*, vol. 772, pp. 202-209, 1987.

[4] J. Helmsen, M. Yeung. D. Lee and A. Neureuther, "SAMPLE-3D Benchmarks Including High NA and Thin Film Effects" *SPIE Optical/Laser Microlithography* VII, vol. 2197, pp. 478-88, 1994.

[5] F. Dill, A. Neureuther, J. Tuttle and E. Walker, "Modeling Projection Printing of Positive Photoresists," *IEEE Transactions on Electron Devices*, vol. ED-22, no. 7, pp. 456-464, July 1975.

[6] E. Walker, "Reduction of Photoresist Standing-Wave Effects by Post Exposure Bake", *IEEE Transactions on Electron Devices*, vol. ED-22, no. 7, pp. 464-466. July 1975.

[7] D. Kim, W. Oldham and A. Neureuther, "Development of Positive Photoresist," IEEE Transactions on Electron Devices, vol. ED-31, no. 12, pp.1730-1735, Dec. 1984.

Chapter 3 Mathematical Basis for Dissolution Simulation

3.1 Introduction

The main aim of this chapter is to introduce the foundations of the common mathematical model of photoresist dissolution, and the basic mathematical approaches that have been developed to solve it. The mathematical model that will be employed is the least time path formulation from geometrical optics [1][2][3][4][5]. This method is identical to the Hamilton-Jacobi equation. A particular and important method of solving the Hamilton-Jacobi equation, which is the Level-Set technique that was invented by Prof. James Sethian [6][7], is also presented. From these two theoretical approaches, three classes of numerical techniques for the simulation of photoresist development have been generated with this model. The first of the three numerical techniques, called ray-trace, is based on computing the shortest time path between the surface and a point to be etched in the photoresist. This technique will be the most rigorously defined, and will be described in detail. This derivation will yield significant insight into the geometry and evolution of the photoresist surface as dissolution simulation proceeds.

A second method describes the surface as the 0.0 contour (the set of values in the field equivalent to 0.0) of a scalar field defined over the entire simulation region. This second method is known as the level-set or advection method. The motion of the surface over time is formulated in terms of the evolution of a partial differential equation (PDE) applied to this field. The time advancement of the scalar field will cause the contour associated with the surface to distort in a manner that simulates the motion of the resist-developer interface. The third approach to simulate photolithography dissolution that will be described is a popular simplification of the level-set method. This third method is known as the cell method.

Examining how these methods are derived from the basic equations of surface advancement can yield significant insight into the assumptions and trade-offs in photoresist simulators in general use. This knowledge can also provide insight into techniques that can improve the state of the art in photoresist development simulation.

3.2 Existing Photoresist Models and the Common Mathematical Form

Most existing photoresist dissolution models describe the dissolution process with an isotropic etch rate that varies as a function of position. The most often used model of photoresist dissolution is Dill's model [8], but others, such as Ferguson's [9], Tam's [10], C. Mack's [11], Kim's [12] and Hirai's [13] have the same properties. Many other resist models, including ones from electron beam, ion beam, and x-ray lithography are also described by isotropic etch rates that are defined as a function of position, such as Charlesby's models [14] and the contributions of Greeneich [15]. Further discussion on models can be found in the PARMEX User's Guide [16]. As discussed in the previous chapter, a concentration of active compound in the resist can be generated by other simulators, which perform the aerial imaging, exposure, and post-exposure bake steps. The photoresist model then uses an empirically constructed equation to relate the active compound concentration to the etch rate. This empirically constructed equation is, in the case of Dill's model, a direct mapping from the active compound concentration to an etch rate at each point in the resist, or as in the case of Ferguson's model, the number of cross-linking events at each location. The general mathematical form is an isotropic etch rate as a function of position. Exceptions to this rule do exist, especially when gel-layer effects are taken into account, and simulators have been created that represent the gel-layer more accurately. One such simulator was developed by Y. Karafyllidis and P. Hagouel [17]. These simulators, however, build upon the rate as a function of position method by adding extra functionality to existing simulation techniques. The Karafyllidis method performs multiple surface advancement simulations simultaneously to represent the gel-layer. Therefore, techniques that are developed for solving the basic photolithography problem can also contribute to algorithms that consider additional physics during dissolution.

3.3 Derivation of Ray-Trace from Basic Etch Rate Functions

The ray-trace method was originally adapted from geometrical optics and applied to photolithographic development simulation in two dimensions by P. Hagouel [4]. P. Hagouel also developed the mathematics for three dimensional ray-trace, although it was not implemented at the time. P. Hagouel first recognized the relevance of the Hamilton-Jacobi equation to photolithography [18], and used a ray following and branching approach to approximate the solution. The least time path method was also developed independently by E. Barouch [5] and A. Moniwa [19]. Ray-trace was implemented in three dimensions by A. Moniwa [19], E. Barouch [5] and K. Toh [1]. The technique described from [EQ: 3] through [EQ: 14] is the derivation given by K. Toh in Appendix A of his thesis [1]. This is in turn derived from the analysis of Carll [2]. This method determines the advancement of the interface between the developer and the photoresist by transforming it into an analogous problem. This analogous problem is determining the volume swept out by a set of light rays that travel through a medium with an inhomogeneous refractive index. Once the transformation is made, the principles of geometrical optics can be employed to form algorithms for the simulation of the development process. Additional concepts from geometrical optics [3] that were not expressed in P. Hagouel's or K. Toh's work are included here, so that a more complete description of surface advancement using this analogy can be presented.

Before etching can take place, it is necessary to define the initial conditions. It is assumed that there exists a region of space, hereafter referred to as the simulation space, where each point has the binary characteristic of being in either the etchant or the photoresist. The collections of points that represent the etchant and the bulk may be distinct and disconnected, but satisfy all the normal topological properties of collections of points that are intended to represent a physical system for simulation purposes. The collections of points satisfy natural conceptions about regions of materials. The boundary between the etchant and the resist is the surface of the resist. The surface, it will also be assumed, is differentiable in a piecewise continuous manner (i.e. It may have non-differentiable corners, but these corners are not so common as to cause parts of the surface to be fractal-like). All points that are contained in the resist region of the simulation space also have a scalar quantity associated with them. This quantity is called the 'etch rate'. This quantity is the rate of photoresist dissolution at each point. The etch rate is typically expressed as $R(x, y, z)$, but will be more commonly expressed, in this discussion, as its inverse $n(x, y, z)$ for notational convenience. This form for the etch rate was first used by F. Dill in [20].

$$n(x, y, z) = \frac{1}{R(x, y, z)} \qquad \text{[EQ: 1]}$$

The etch rate varies in a continuous and piecewise differentiable manner throughout the volume of the resist. The etch rate is either positive or equal to 0. The inverse of the etch rate is either a finite or infinite value, but always strictly positive.

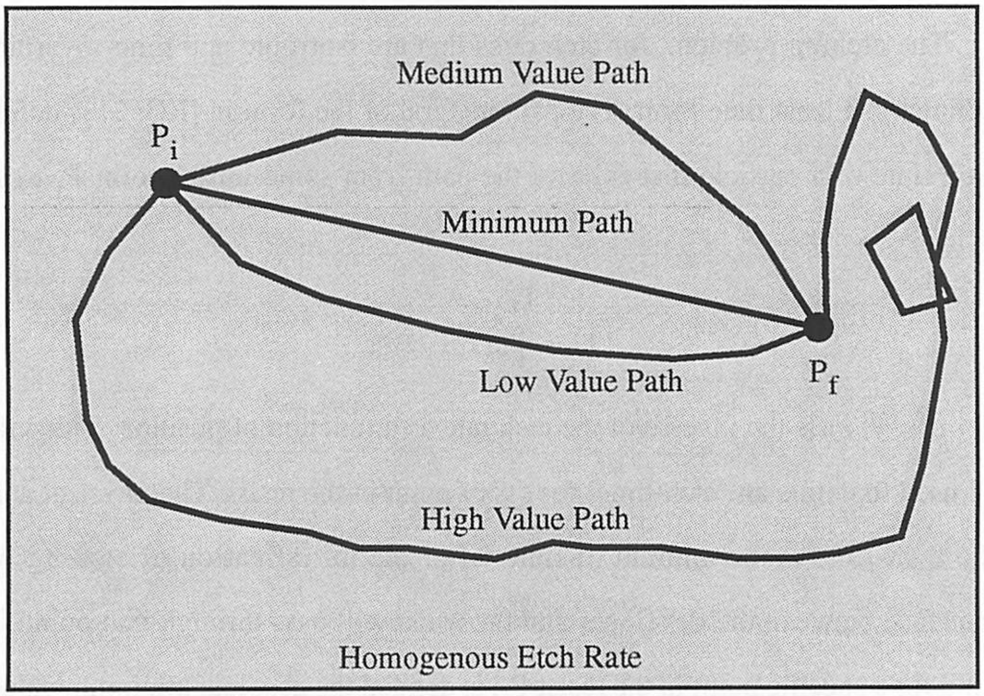
3.3.1 Derivation of the Differential Ray Function

The etching problem, for etch rates that are isotropic and time-invariant, can be defined in a least time form. The path integral of the form in [EQ: 2] is defined as the etch time of a particle that follows the path from some initial point P_i to some end point P_f.

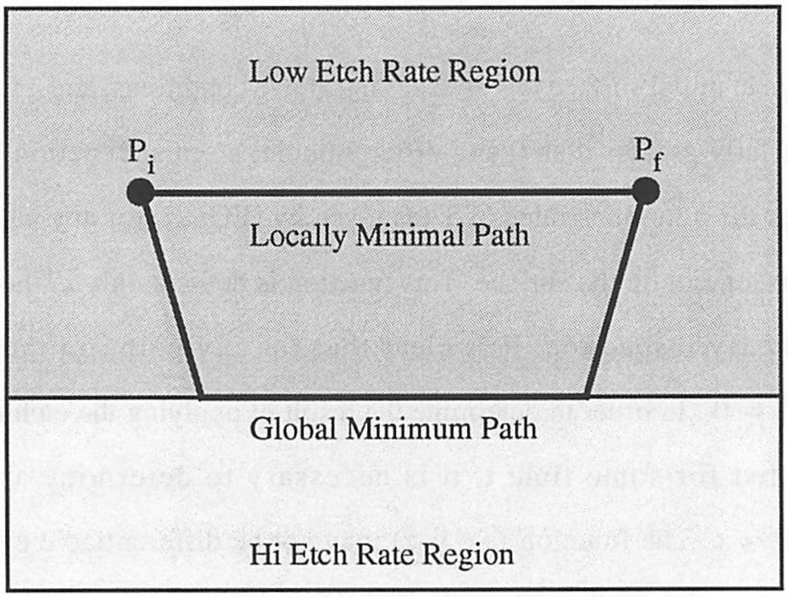
$$T = \int_{P_i}^{P_f} n(x, y, z)ds \qquad \text{[EQ: 2]}$$

n(x, y, z) is the inverse of the etch rate as a function of position. This integral can be used to define an 'etch-time' for every point in the resist. The physical analogue of the etch-time is the amount of time, after the initialization of etching, when the interface between the developer and the resist will pass through that point. The etch-time for a specific point in the resist P_i, is defined the absolute minimum value of [EQ: 2] for all paths from any initial point P_i on the initial surface. Examples of this concept are shown in Figure 1.

Given an initial surface that satisfies the above conditions, and a function n(x, y, z) that is strictly greater than 0 and differentiable, a scalar function $\zeta(x, y, z)$ can be defined as the minimum value of T (as given by [EQ: 2]) for any path for any P_i that represents a point on the surface. This function is defined only in the region occupied by resist. By inspection, it is clear that for any point on the initial surface $\zeta(x, y, z) = 0$. In order to determine the result of applying the etching process to the photoresist for some time t, it is necessary to determine all points where $\zeta(x, y, z) = t$. The function $\zeta(x, y, z)$ may not be differentiable everywhere, but it will be assumed that all points in the non-differentiable region are boundary points of the non-differentiable region. (i.e. The non-differentiable region is one dimension lower than the photoresist region.)



[EQ: 2] Evaluated in a Homogenous Etch-Rate Region Over Different Paths



The Global Minimum Path As Compared To A Locally Minimal Path

Figure 1) Explanatory Figures For [EQ:2]

More properties of the function $\zeta(x, y, z)$ must be determined to allow a simulation method to be constructed. Because few paths between P_i and P_f are locally minimal, and all minimum paths of [EQ: 2] are locally minimal, determining properties of locally minimal paths will significantly reduce the total number of paths that must be considered. If the path between some point P_i and some other point P_f is locally minimal in T, then the variation about T must be 0 to the first order for any infinitesimal change in the path.

$$\delta T = \delta\int_{P_i}^{P_f} n(x, y, z)ds = 0 \qquad \text{[EQ: 3]}$$

The differential is now brought inside the integral yielding:

$$\delta T = \int_{P_i}^{P_f}\left[\frac{\partial n}{\partial x}\delta x + \frac{\partial n}{\partial y}\delta y + \frac{\partial n}{\partial z}\delta z\right]ds = 0 \qquad \text{[EQ: 4]}$$

Now, it is well known that path integrals where ds is unitary have the form:

$$x'x'' + y'y'' + z'z'' = 0 \qquad \text{[EQ: 5]}$$

where x, y, and z are differentiated with respect to s. This can be rewritten in variational form as:

$$x'\delta x' + y'\delta y' + z'\delta z' = 0 \qquad \text{[EQ: 6]}$$

where $\delta x'$ is defined as the change in x' as the variable x is changed to $x + \delta x$. Using this relationship, [EQ: 6] can be rewritten as:

$$x'\frac{d}{ds}(\delta x) + y'\frac{d}{ds}(\delta y) + z'\frac{d}{ds}(\delta z) = 0 \qquad \text{[EQ: 7]}$$

Since [EQ: 7] is true for all curves, if it is multiplied by $n(x, y, z)$ and integrated along the path, the result is still zero.

$$\int_{P_i}^{P_f} n(x, y, z)\left[x'\frac{d}{ds}(\delta x) + y'\frac{d}{ds}(\delta y) + z'\frac{d}{ds}(\delta z)\right]ds = 0 \qquad \text{[EQ: 8]}$$

Now, since there is no variation in the endpoints, then integration by parts, with the non-integral term evaluating to 0, yields:

$$\int_{P_i}^{P_f} [\,(nx')'\delta x + (ny')'\delta y + (nz')'\delta z]\, ds = 0 \qquad \text{[EQ: 9]}$$

This zero evaluating integral is then subtracted from [EQ: 4], yielding:

$$\delta T = \int_{P_i}^{P_f} \{\frac{\partial n}{\partial x} - (nx')'\}\,\delta x + \{\frac{\partial n}{\partial y} - (ny')'\}\,\delta y + \{\frac{\partial n}{\partial x} - (nz')'\}\,\delta z ds \qquad \text{[EQ: 10]}$$

Since δT can only be 0 if each of the integration terms is also 0, the following three equations are derived:

$$\frac{\partial n}{\partial x} = \frac{d}{ds}\left[n\frac{dx}{ds}\right] \qquad \text{[EQ: 11]}$$

$$\frac{\partial n}{\partial y} = \frac{d}{ds}\left[n\frac{dy}{ds}\right] \qquad \text{[EQ: 12]}$$

$$\frac{\partial n}{\partial z} = \frac{d}{ds}\left[n\frac{dz}{ds}\right] \qquad \text{[EQ: 13]}$$

Or if written in vector notation:

$$\frac{d}{ds}\left[n\frac{dr}{ds}\right] = \nabla n \qquad \text{[EQ: 14]}$$

Where r is the position vector. This function is known as the differential ray equation, and also describes the motion of light rays in inhomogeneous refractive media. For clarity it can also be rewritten in terms of the etch rate R(x, y, z).

$$\frac{d}{ds}\left[\frac{1}{R(x, y, z)}\frac{dr}{ds}\right] = \nabla\frac{1}{R(x, y, z)} \qquad \text{[EQ: 15]}$$

This equation provides the necessary condition for the evolution of the surface to be computed via a point particle method. A group of nodes can be used to represent the initial surface and can be advanced in various random directions, according to the ray equation, to represent the etching process. If each node is halted after a time t, the volume that is swept out by these nodes will represent the section of photoresist that

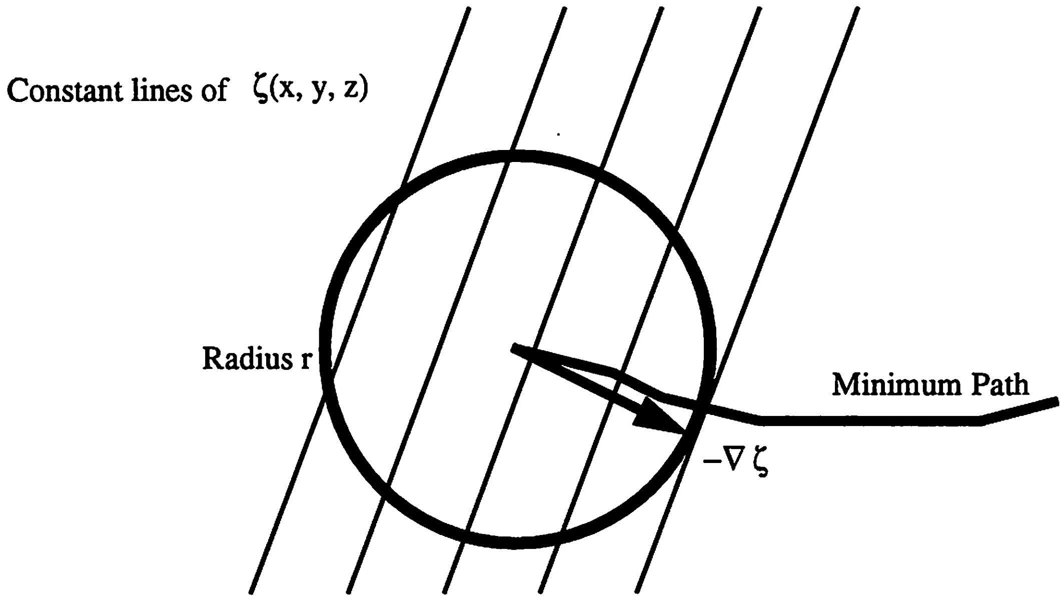


Figure 2) Differentiable Eikonal About the Endpoint of a Minimum Path

was etched during time t. Unfortunately, most of these nodes become superfluous almost immediately, since other nodes may travel into the resist more directly. Therefore, it is desirable to look for restricting conditions about the initial and final points in [EQ: 2], so that point advancement can be performed in a more efficient manner.

3.3.2 Derivation of the Eikonal

The function $\zeta(x, y, z)$, as defined previously, represents the value of the minimum path from the surface. Because $\zeta(x, y, z)$ is continuously differentiable in most parts of the simulation region, important statements about the behavior of the minimum paths in these regions can be made. For instance, minimum paths from the surface to locations in the resist always travel perpendicularly to the contours of the Eikonal function.This is a well known result in geometrical optics [3]. To prove this statement, assume that the endpoint of a minimum path, not located on the initial surface, is chosen. A sphere of a sufficiently small radius r is constructed that is centered at the endpoint (Figure 1). Given any vector $\vec{r}$, which is rooted at chosen endpoint and terminates on the surface of the sphere, the value of the least time path, i.e. the time required to traverse the path, along the vector $\vec{r}$ is equal to

$$n\|\vec{r}\| + \|\vec{r}\|^2\|\nabla n\|\cos\theta + O(\|\vec{r}\|^3) \qquad \text{[EQ: 16]}$$

where θ is the angle between $\vec{r}$ and ∇n. It can be shown from the differential ray equation that the deflection of the ray tends to zero as the radius of the sphere shrinks. Therefore, the minimum path, within an infinitesimal sphere, is straight. This information is now used to locate the point of entry into the sphere of the minimum path relative to the gradient of the eikonal function. The value of [EQ: 2] along a line segment from the center of the sphere to some point $\vec{r}$ on the surface is:

$$\zeta_f + (\vec{r} \cdot \nabla\zeta) + O(\|\vec{r}\|^2) \qquad \text{[EQ: 17]}$$

where ζ_f is the value of the eikonal at the center of the sphere. Therefore, given some point $\vec{r}$ on the surface of the sphere, the total time needed to traverse a path that intersects the sphere at $\vec{r}$ and continues to the center along a radial line segment is:

$$\zeta_f + (\vec{r} \cdot \nabla\zeta) + (n\|\vec{r}\|) + O(\|\vec{r}\|^2) \qquad \text{[EQ: 18]}$$

In the limit as $\|\vec{r}\|$ tends towards 0, the high order terms drop out. The minimum of the function, i.e. the relation of the minimum path to the eikonal, occurs when:

$$\vec{r} \cdot \nabla\zeta \qquad \text{[EQ: 19]}$$

is minimized. This only occurs when the two vectors $\vec{r}$ and $\nabla\zeta$ point in opposing directions. The minimum path, therefore, approaches the center of the sphere in the direction of the gradient of the eikonal. Furthermore, the equation is satisfied only when

$$\zeta_f + (\vec{r} \cdot \nabla \zeta) + (n\|\vec{r}\|) + O(\|\vec{r}\|^2) = \zeta_f \qquad \text{[EQ: 20]}$$

or when, if $\vec{r}$ and $\nabla \zeta$ are 180 degrees apart, constant terms are subtracted, and the limit when $\|\vec{r}\|$ tends towards 0:

$$\|\nabla \zeta\| = n \qquad \text{[EQ: 21]}$$

This is the familiar eikonal equation from geometrical optics [3]. A similar argument can also be constructed for the initial points of least time paths. This argument shows that least time paths leave initial points in the direction of the gradient of the eikonal.

From the above derivation, some conclusions can be drawn. First, since each contour of the eikonal represents the position of the surface at any particular etch time, and since the gradient of a contour is the surface normal of that contour, the rays always point into the resist in the direction of the inwardly oriented surface normal. Therefore, a suitable method for initializing rays for the ray-trace method exists if the surface normal is well-defined. The inward surface normal of the surface of the resist can also be found, at each etch time, by examining the direction of the ray after advancement. Finally, since there is a unique path for each ray on the surface, it is only necessary to sweep out a single ray from each point on the surface with a well defined surface normal. These statements are not, however, the only statements that can be made about the problem, since it is quite possible that the eikonal function, and therefore the surface, may not be differentiable at all points in the simulation region.

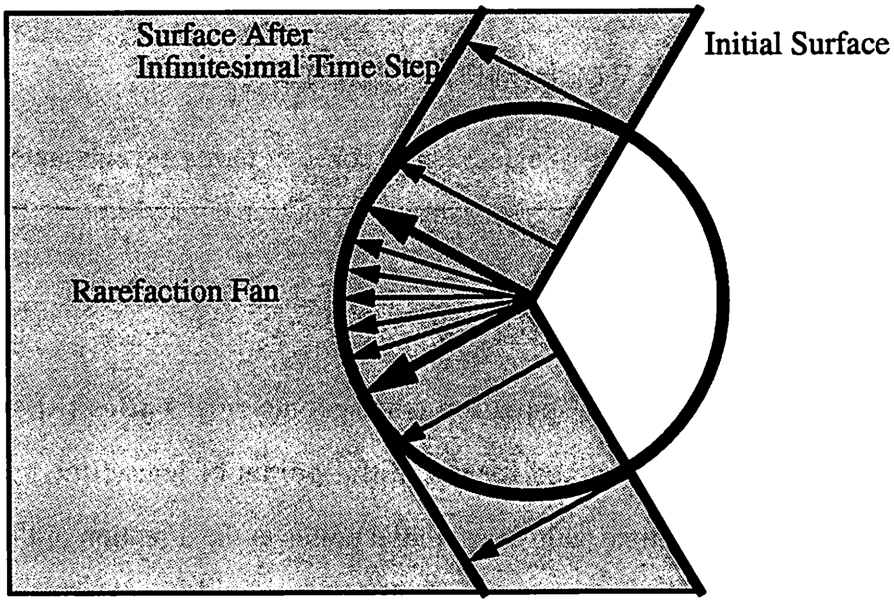


Figure 3) Evolution of an Initially Non-Differentiable Eikonal

3.3.3 The Non-Differentiable Eikonal

It was previously assumed that the eikonal can only be non-differentiable in a specific way. Every point in the non-differentiable region of the eikonal is a boundary point. Therefore, there is no sphere of finite radius that only contains points where the eikonal is non-differentiable. It is also clear, since the etchrate is non-infinite at every point in the simulation region, that the eikonal is continuous, although the derivatives of the eikonal may not be. The formulations that were previously developed are inapplicable under these conditions, since the surface normal is undefined. Therefore, certain assumptions that had been made about least time paths that validate the previous formulation, such as unique paths to final points and unique paths from initial points, are no longer appropriate.

To resolve these difficulties, methods for the initial generation of rays from a surface with a discontinuous surface normal must be determined. Two cases exist that must be considered. The first case is the solution if the internal angle of the surface is

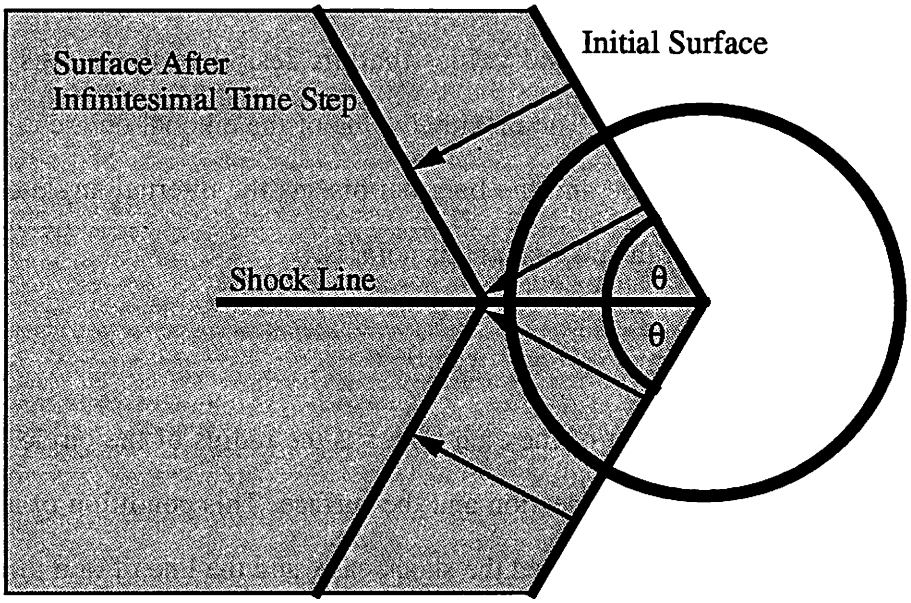


Figure 4) Evolution of an Propagating Non-Differentiable Eikonal

greater than 180 degrees. Assume there exists an infinitesimal circle about the discontinuity, so that the etch rate is constant. For all points on the arc of the circle that have an angle of less than 90 degrees in relation to the surface, it is clear that there exist paths to the resist-etchant interface with a length less than the radius of the circle. However, if the internal angle of the surface is greater than 180 degrees, there are still points on the arc that must be considered. The shortest line segments from the uncovered sections of the arc terminate on the surface discontinuity. Therefore, instead of having just one minimum path proceeding from the discontinuity, as in the continuous eikonal case, the point of discontinuity generates rays in all directions between the two limiting cases. This group of rays is called a rarefaction fan. This term was first applied to the study of surface advancement by J. Sethian [29]. The new eikonal contour that is described by this infinitesimal advancement is differentiable, so this type of discontinuity in the eikonal function does not propagate.

The second type of discontinuity that may occur is when the internal angle is less than 180 degrees. As shown in Figure 4, none of the points on the arc of the circle have the initial point of the discontinuity as their initial path point. This means that all of the points on the arc have eikonal values that are less than the radius of the circle multiplied by the etch rate. The maximal value of the eikonal on the circle occurs at the point where the circle and the line that bisects the internal angle of the surface intersect. The value of the eikonal at this point is:

$$nr\sin\theta \qquad \text{[EQ: 22]}$$

where n is the inverse of the etch rate, r is the radius of the circle, and θ is the angle between the line of bisection and the surface. This condition of a propagating non-differentiable region is called the shock case, and the line of non-differentiability is called the shock line. This term was also first used in the surface advancement context by J. Sethian. Along the shock line there are two least time paths to the initial surface with identical time values, thus providing a inverse case to the rarefaction fan. It is clear that a ray can be considered to terminate on the shock, since the gradient of the eikonal becomes undefined at this point.

Shocks can form even if they are not present in the initial conditions. In the case of a completely homogeneous etch rate and an initial surface consisting of two disconnected circles, the line of equal distance from the centers of both circles clearly has two least time paths for every point. (Figure 5) The shock first forms at the point of initial contact and spreads out radially along the equidistant plane. For this reason, methods that track least time paths clearly need an algorithm to detect these cases when they occur. It is also possible for shocks to form in a manner that resembles a reverse rarefaction fan. This may occur at locations where the surface curvature of the contours of the eikonal tend towards infinity. (Figure 6) Therefore, a method of detecting and properly dealing with this condition is also necessary for implementing a method based on a least time path formulation.

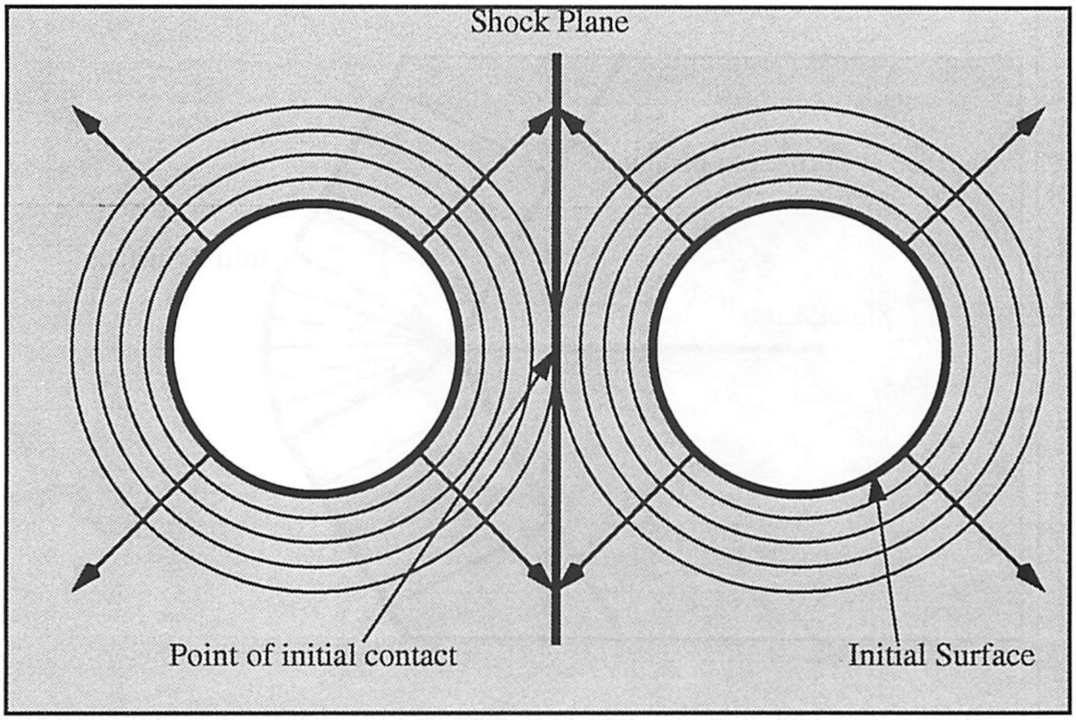


Figure 5) Formation of a Non-Differentiable Eikonal From Two Surfaces

A treatment of the full behavior of the properties of the eikonal in the non-differentiable case in three-dimensions has been given in [30]. This derivation will not be treated in extreme detail here, due to the complexity of describing behavior around saddle points. There are two obvious extensions, however, of the two dimensional case for points on nondifferentiable surfaces. First, infinitesimal spheres around non-differentiable regions in three-dimensions generally have cross sections that are directly related to the two-dimensional cases. This tends to occur when the local non-differentiable region can be approximated by a line or a plane. Second, in the case of a single point on the contour where the surface normal is undefined, rarefaction cones, shock lines, and combination shock-rarefaction fans occur. These occur, respectively, when the approximate curvature about the point is outward in relation to the resist, inward, or a saddle-point. Combinations of these cases also occur. An exact uniform

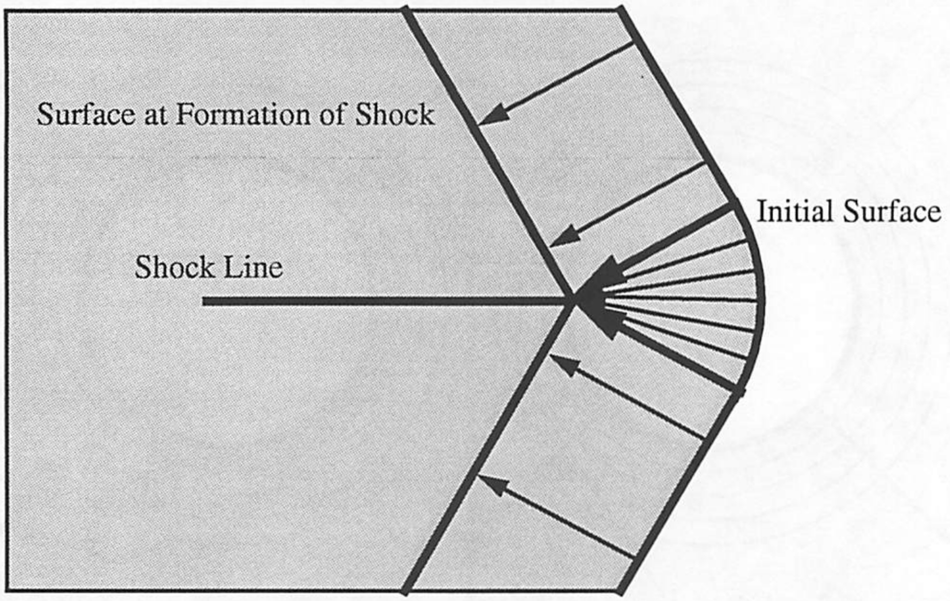


Figure 6) Formation of a Non-Differentiable Eikonal From a Single Surface

solution for purely anisotropic etching has been formulated by B. Foote [21]. It is expected that the solution for the isotropic case will contain many similarities.

3.3.4 Boundary Conditions

A boundary exists for any simulation region exists if the simulation region is non-infinite, however, the boundary need not be a simple enclosing surface of genus zero. In photoresist simulation, the boundary may represent topography created by materials that are not affected by the etching process, such as silicon or oxide. In the case of the standard simulation boundary, it is desirable to have the least time path terminate at the boundary. If the results of the simulation are dependent on the continuation of the least time path past the defined boundary, it is recommended that the simulation boundary be extended further to encompass the region of interest. Least time paths should not be created at the boundary, unless they are part of the initial conditions, since this

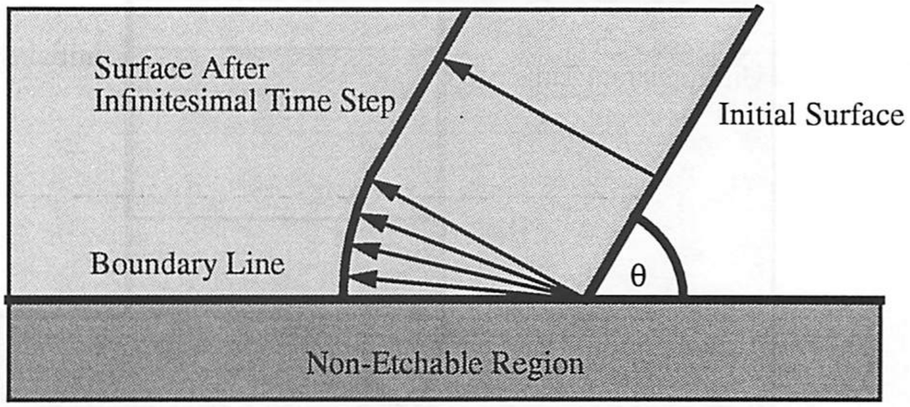


Figure 7) Acute Angle Prohibition at a Boundary

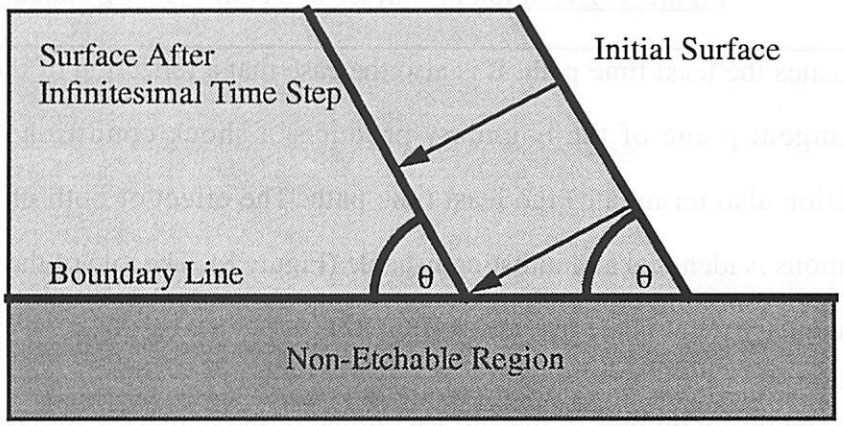


Figure 8) Evolution of Surface at a Planar Boundary

formation of paths represents activity outside of the simulation region. It is also clear that the surface will not preserve intersections of the surface into the boundary with acute angles from the side of the etchant, since intersection of the surface with the boundary will become perpendicular for infinitesimal time steps.(Figure 7)

For the purposes of this discussion, it will be assumed that the surface normal of the resist near the boundary is continuously defined. The limiting case of an infinitesimal sphere on the boundary is considered. The material on the opposite side of the boundary is considered to have an etch rate of 0. This etch rate value effectively

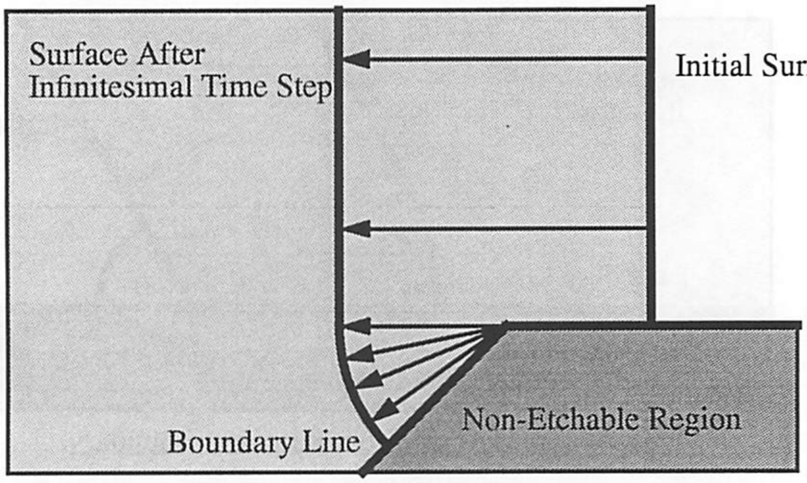


Figure 9) Evolution of Surface at a Non-Planar Boundary

terminates the least time path. It is also the case that a reflection of the eikonal across the tangent plane of the boundary produces a shock condition. This boundary condition also terminates the least time path. The effect of both of these boundary conditions is identical and indistinguishable (Figure 8). The rate of the advancement of the boundary point is the inverse of [EQ: 22].

When discontinuites of the surface normal of the boundary are taken into account, it is clear that curvature of the boundary away from the simulation region may cause difficulties for simulators that explicitly trace out the least time path, since a path may 'split' into multiple paths (Figure 9). Ray generation may occur if the surface curves continuously as well. These effects are not of concern, in general, for most situations in photolithography simulation. In the case of reflective notching, however, this effect may be of significant concern. Explicit tracking of points on the boundary surface is a possible method of simulating this effect accurately.

3.4 The Level-Set Method

Instead of sweeping out the advancing resist-etchant interface via point advancement, it is possible to solve for the eikonal directly in the form of a partial differential equation. This is the level-set formulation invented by Prof. Sethian [6][7]. Advancement of the level-set is performed using advection techniques from fluid mechanics. The partial differential equation is solved over the entire simulation space as a real valued field, with the surface represented as a contour of that field. In this manner, the need for explicit surface representation during simulation, as in the previous least path formulation, is removed.

3.4.1 How the Level-Set Method Works

Consider a monotonically increasing function u(x) that passes through 0 at the origin, (Figure 10) (Figure 11) and consider the partial differential equation:

$$\frac{\partial u}{\partial t} + F(x)\frac{\partial u}{\partial x} = 0 \qquad \text{[EQ: 23]}$$

where F(x) is the 'etch rate'. It is clear that for any linear u(x), the rate of motion of the u(x)=0 point is to the right at the rate F(x). For:

$$u(x) = ax + b \qquad \text{[EQ: 24]}$$

The intercept of the x-axis is at

$$x = \frac{-b}{a} \qquad \text{[EQ: 25]}$$

Given F(x) = r where r is a constant, [EQ: 23] evaluates to:

$$\frac{\partial u}{\partial t} = -r\frac{\partial u}{\partial x} \qquad \text{[EQ: 26]}$$

or:

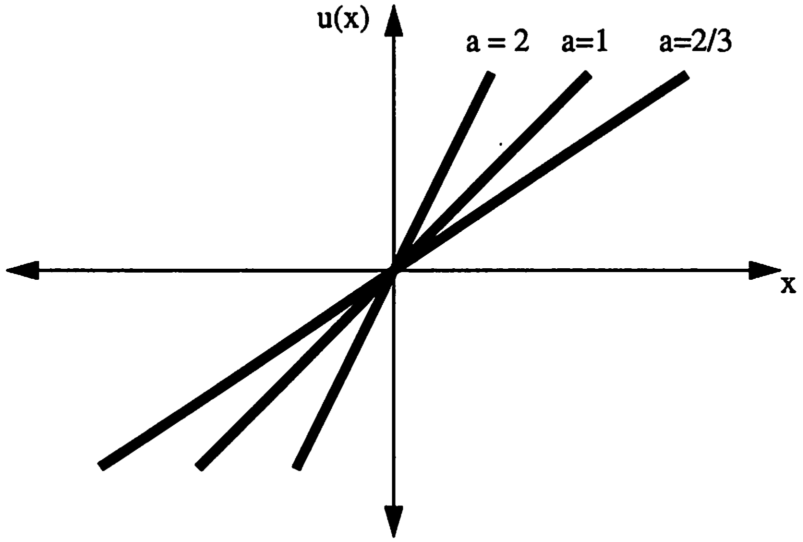


Figure 10) Possible Initializations for Surface Point at Origin

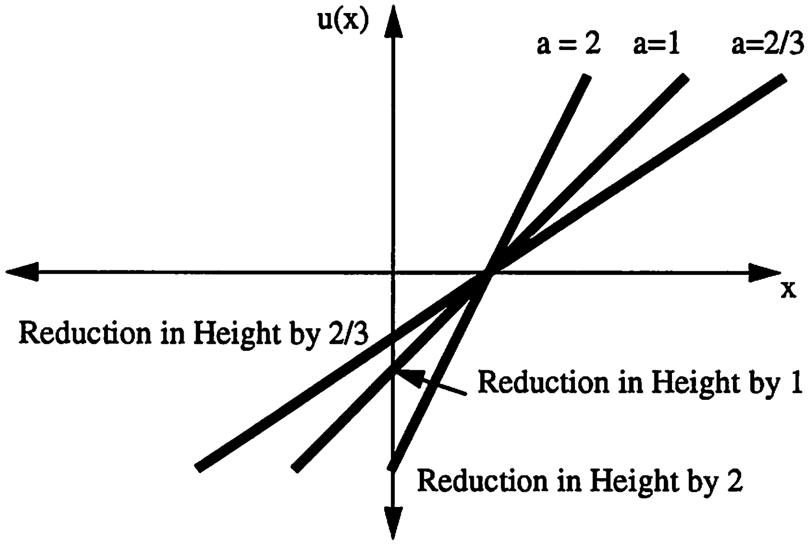


Figure 11) Advancement Via $\frac{\partial u}{\partial t}+\frac{\partial u}{\partial x} = 0$ for One Time Unit

$$\frac{\partial u}{\partial t} = -ra \qquad \text{[EQ: 27]}$$

giving:

$$u(x, t) = a(x - rt) + b \qquad \text{[EQ: 28]}$$

This equation demonstrates that the 0 point advances at a speed r for any choice of a.

In two and three dimensions, the advancement occurs in the direction of the surface normal at the local etch rate. Therefore, in any small region about the surface, the advancement can be defined as:

$$\frac{\partial u}{\partial t} + F(x, y, z)\frac{Du}{D\vec{n}} = 0 \qquad \text{[EQ: 29]}$$

where $\vec{n}$ is the surface normal, but since the surface normal is the gradient of the equation, the equation can be rewritten as:

$$\frac{\partial u}{\partial t} + F(x, y, z)\,\|\nabla u\| = 0 \qquad \text{[EQ: 30]}$$

3.4.2 Initialization of The Level-Set Technique

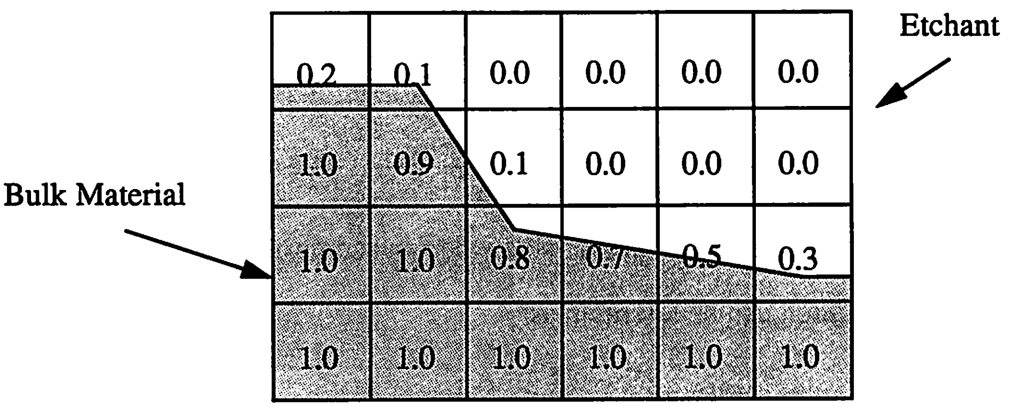
In order perform photolithographic simulation with the level-set model in two and three dimensions, it is necessary to not only define the initial surface, but to initialize the function u(x,y,z) over the entire simulation region. The two conditions on u are 1) That u(x,y,z) = 0 at the location of the surface, and that 2) The gradient of the function u(x,y,z) be in the direction of the inward surface normal of the photoresist Fortunately a simple method of initialization satisfies both criteria:

$$u(x, y, z) = \pm\text{distance} \qquad \text{[EQ: 31]}$$

where distance is the Euclidean distance from the surface. The function is positive in the direction of desired etching, and negative in the region of the etchant.

3.4.3 Boundary Conditions

The boundary conditions for level-set solvers are, in general, similar to the conditions defined for ray-trace approaches. The boundary condition across a planar boundary is reflective both in terms of the etch rate and the function u(x,y,z). For some applications of level-set solvers, other boundary conditions are necessary. These conditions are normally representative of large etched regions that are not explicitly



0.2	0.1	0.0	0.0	0.0	0.0
1.0	0.9	0.1	0.0	0.0	0.0
1.0	1.0	0.8	0.7	0.5	0.3
1.0	1.0	1.0	1.0	1.0	1.0

Figure 12) A Typical Surface Represented by Cells

represented in the simulation region. These boundary conditions are application specific and ought to be considered in light of the specific implementation under consideration. The specific boundary conditions employed for the photoresist problem are described in Chapter 4.

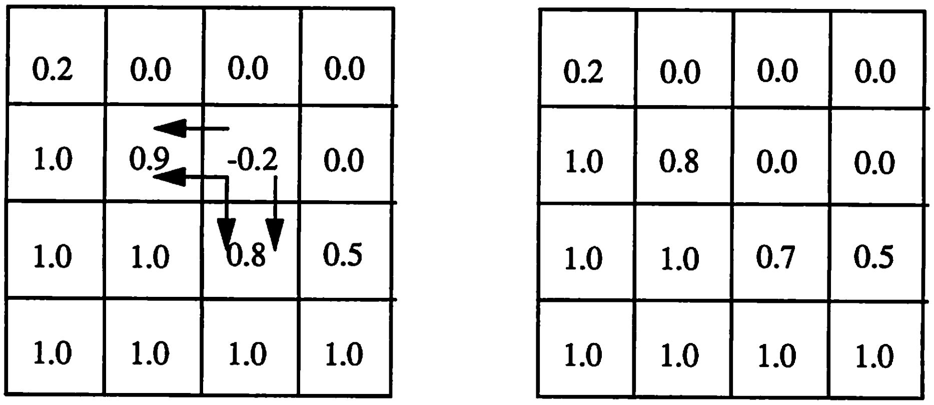
3.5 The Cell Method

Cell methods, in general, are methods that represent the material to be etched as a collection of 'material amounts' on a Euclidean grid [22][23][24][25][26][27][28]. Cells that are completely filled with photoresist are given a value of 1.0, and cells that are completely empty receive a value of 0.0. In some cell methods, the cells with intermediate states receive time stamps for the expected time when the 0.0 will be reached. In other methods, a standard time stepping approach is used. Values between 0.0 and 1.0 are used in each cell to represent the amount of material left at the end of each time step (Figure 12). Advancement methods for cells are generally ad-hoc, but the best ones act similarly to numerical techniques commonly used for level-sets. They are not exactly equivalent to the level-set method though, since the solution of the system only takes place in a tight band near the surface.

3.5.1 Advancement with Cells

All cell methods perform advancement computations with two required steps, and often employ a third correcting step. The first step for cell advancement is the analysis of the status of each cell before advancement. This step is related to the computation of the surface normal in the level-set method. Each particular cell, based on the amount of etching inside the cell, and the amount of etching that has been undergone in the cells nearby, has an interpolated surface shape generated inside. This shape is then used in the second step, also known as the removal step, to compute the amount of material removed from the cell during that time step. A significant number of cell-based etching schemes have improved their performance through the use of 'overetching' (Figure 13). The overetching technique allows simulation method to not consider each cell individually when it reaches its 0.0 value. In the removal step, some methods continue etching past the value of 0.0, so that a negative value for the amount of material in the cell is generated. In this case, the cell is renormalized to 0.0 by removing material from neighboring cells to make up the difference. Originally designed to allow cell methods to take larger time steps, overetching done properly also tends to maintain a better approximation of the surface normal between time steps. With the inclusion of overetching, cell methods can perform a better job of approximating a banded level-set method.

The major drawback to the cell methods that have been implemented to perform photolithography, is that although they can be made to run quickly, the accuracy of the method is suspect. Cell methods tend to be highly accurate when applied to etching problems that extend in 1 dimension in the direction of the basis vectors of the grid coordinates. Accuracy of this type is generally the main criterion in the original design of these methods. Because these methods do not perform careful approximations of the surface normals, however, they tend to introduce etching errors that have a



0.2	0.0	0.0	0.0
1.0	0.9	-0.2	0.0
1.0	1.0	0.8	0.5
1.0	1.0	1.0	1.0

Before Overetching

0.2	0.0	0.0	0.0
1.0	0.8	0.0	0.0
1.0	1.0	0.7	0.5
1.0	1.0	1.0	1.0

After Overetching

Figure 13) An Example of Renormalization of Cells After Overetching

significant degree of grid dependence. These grid dependent errors are contained in the design of the advancement method, so these effects do not decrease with increased grid resolution. Overetching tends to reduce these effects, but they are extremely difficult to remove completely without a fully developed theory of surface advancement.

One technique that has been employed to reduce anisotropy is cell removal that is Huygen sphere based cell removal [3][31]. Spheres are extruded from the center of each etched cell that borders the surface. This method has been implemented in the SOLID simulator by J. Pelka [25] and K. Toh [1]. The size of the extruded sphere is equivalent to the etch rate multiplied by the time step. Each cell whose center lies inside an extruded sphere is removed. The implementation performed by K. Toh consumed a significant amount of memory. SOLID avoids this by using an adaptive grid. To reduce anisotropic effects significantly, however, the dimensions of the cells must be small in relation to the size of the spheres. This greatly increases the computation time. SOLID has been reported to require 2 to 20 minutes on a VAX station 3540, which is a parallel multiprocessor machine [26]. The amount of time required for a single processor workstation is far in excess of this. Better methods of computing the surface normal from cells have been implemented by E. Puckett [28], but were not available at the time of this research.

3.5.2 Cell Boundary Conditions

The boundary conditions that cells use are, in general, simple. For purposes of initialization, the initial surface is considered to be faces of the cells that are exposed to the etchant, or a layer of cells with the value 0.0 that cover the initial surface. The initial surface, in the non-planar case, may initialize the cells by computing the fraction of the volume of the cell that is in the unetched region defined by the surface. The simulation of boundary conditions is generally performed by giving certain cells an etch rate value of 0. Non-planar boundary conditions can be performed by approximating the boundary with unetchable cells, or cells that may be etched to only a certain value.

Reference for Chapter 3

[1] K.K.H Toh, "Algorithms for Three-Dimensional Simulation of Photoresist Development", *Ph.D. Dissertation*, University of California at Berkeley, 1990

[2] L.B. Carll, *A Treatise on the Calculus of Variations*, pp. 335-344, John Wiley & Sons, New York, 1881.

[3] M. Born, E.Wolf, *Principles of Optics, Sixth Edition,* Pergammon Press, London 1980.

[4] P. Hagouel, "X-Ray Lithographic Fabrication of Blazed Diffraction Gratings", *Ph.D. Dissertation*, University of California, Berkeley, 1976.

[5] E. Barouch, B. Bradie, S. Babu, "Resist Development Described by Least Action Principle-Line Profile Prediction", Journal of Vacuum Science & Technology B, vol. 6, no. 6, pp. 2234-7, Nov. 1988.

[6] J. Sethian, "An Analysis of Flame Propagation", *Ph.D. Dissertation*, University of California at Berkeley, 1982.

[7] J. Sethian, "Numerical Algorithms for Propagating Interfaces: Hamilton-Jacobi Equations and Conservation Laws", *Journal of Differential Geometry*, 1990, pp. 131-161.

[8] F.H. Dill, W.P. Hornberger, P.S. Hauge, and J. M. Shaw, "Characterization of Positive Photoresist," *IEEE Transactions on Electron Devices,* vol. ED-22, pp. 456-464, July 1975.

[9] R. Ferguson, J. M. Hutchinson, C.A. Spence, and A.R. Neureuther, "Modeling and Simulation of a Deep-UV Acid Hardening Resist," J*ournal of Vacuum Science and Technology B*, vol 8, pp. 1423-1427

[10] N. N. Tam, "Resist mechanisms and models in electron-beam lithography," *Ph.D.*

Dissertation, University of California at Berkeley, 1991.

[11] C. Mack, "Development of Positive Photoresists," *Journal of Electro-Chemical Society*, vol. 134, no. 1, pp. 148-152, 1987.

[12] D. Kim, W. Oldham, A. Neureuther, "Development of Positive Photoresist," *IEEE Transactions on Electronic Devices*, vol. 31, pp. 1730-5, Dec. 1984.

[13] Y. Hirai, M. Sasago, M. Endo, K. Tsuji and Y. Mano, "Process Modeling for Photoresist Development and Design of DLR/sd (Double-Later Resist by a Single Development) Process," *IEEE Transactions on Computer-Aided Design*, vol. CAD-6, no. 3, pp. 403-409, 1987.

[14] A. Charlesby, *Atomic Radiation and Polymers*, Pergammon Press, London, 1960.

[15] J. Greeneich, "Developer Characteristics of Poly(methyl methacrylate) electron resist:", *Journal of the Electrochemical Society*, vol. 122, 970-976, 1975.

[16] *Parmex 1.0 User Guide*, Electronics Research Laboratory, University of California, Berkeley 1989.

[17] Y. Karafyllidis, P. Hagouel, "Simulation of Multiple Etch Fronts," *Microelectronics Journal*, vol. 22, pp. 97-104, 1991

[18] P. Hagouel and A. Neureuther, "Modeling of X-ray Resists for High Resolution Lithography", *American Chemical Society 170th Meeting*, vol. 35, no. 2, pp. 298-305, Aug. 1975.

[19] A. Moniwa, T. Matsuzawa, T. Ito and H. Sunami, "A Three-Dimensional Photoresist Imaging Process Simulator for Strong Standing Wave Effect Environment", *IEEE Transactions on CAD*, vol CAD-6, no. 3, May. 1987.

[20] F. Dill, A. Neureuther, J. Tuttle and E. Walker, "Modeling Projection Printing of Positive Photoresists", IEEE Transactions on Electron Devices, vol. ED-22, no. 7, July 1975.

[21] B. Foote, *M.S. Thesis*, University of California, Berkeley, Sept. 1990

[22] E.W. Scheckler, "Algorithms for Three-Dimensional Simulation of Etching and Deposition Processes in Integrated Circuit Fabrication", *Ph.D. Dissertation*, University of California, Berkeley, Nov. 1991.

[23] W. Henke, D. Mewes, M. Weiss, G. Czech, and R. Schiessl-Hoyler, "Simulation of Defects in 3-Dimensional Resist Profiles in Optical Lithography", *Microelectronic Engineering*, vol. 13, pp. 497-501, 1991.

[24] W. Henke, D. Mewes, M. Weiss, G. Czech, and R-Schiessl-Hoyler, "A Study of Rectile Defects Imaged into Three-Dimensional Developed Profiles of Positive Photoresist Using the SOLID Lithography Simulator", *Microelectronic Engineering*, vol. 14, pp. 283-297, 1991.

[25] J. Pelka, "SOLID: Comprehensive Three Dimensional Simulation Program for Optical Microlithography", *Information Brochure, Fraunhofer-Institut fur Mikrostrukturtechnik*, May 1990.

[26] J. Pelka, Simulation of Ion-Enhanced Dry-Etch Processes, *Proceedings of the SPIE*, vol. 1392, pp. 55-66, 1991.

[27] Y. Hirai, S. Tomida, K. Ikeda, M. Sasago, M. Endo, S. Hayama, and N. Nomura, "Three-Dimensional Resist Process Simulator PEACE (Photo and Electron Beam Lithography Analyzing Computer Engineering System)", *IEEE Trans. on CAD*, vol. 10, pp 802-807.

[28] E. Puckett, "A Volume-of-Fluid Interface Tracking Algorithm with Applications to Computing Shock Wave Refraction", *Proceedings of the 4th International Symposium on Computational Fluid Dynamics*, pp. 933-938, 1991.

[29] J. Sethian, "Curvature and the Evolution of Fronts", *Communications in Mathematical Physics*, v. 101, pp. 487-499, 1985.

[30] S. Osher and J. Sethian, “Fronts Propagating with Curvature-Dependent Speed: Algorithms Based on Hamilton-Jacobi Formulations,” *Journal of Computational Physics*, vol. 79, pp. 12-49, 1988.

[31] A. Chorin, “Flame Advection and Propogation Algorithms,” *Journal of Computational Physics*, vol. 35, pp. 1-11, 1980.

Chapter 4 Implementation of Methods

4.1 Introduction

Three methods exist in SAMPLE-3D that simulate the resist dissolution process. These methods are ray-trace, level-set and the cell method. Each of these methods solves the same mathematical equation in a different manner, as discussed in Chapter 3. This chapter will discuss the specifics of the implementation of each method in SAMPLE-3D. This chapter will also give the details of the rate function input file that was used for each method.

4.2 The Rate Function

In SAMPLE-3D the etch rate that is required by the dissolution simulators is computed in a separate program. This program is BLEACH. BLEACH takes aerial image input from SPLAT and computes the resulting chemical densities in the resist. BLEACH may also perform an optional post-exposure bake step, and evaluate the diffusion, creation and consumption of active compounds in the resist. The etch rate is derived from the concentrations of active compounds at the end of the post-exposure bake step.

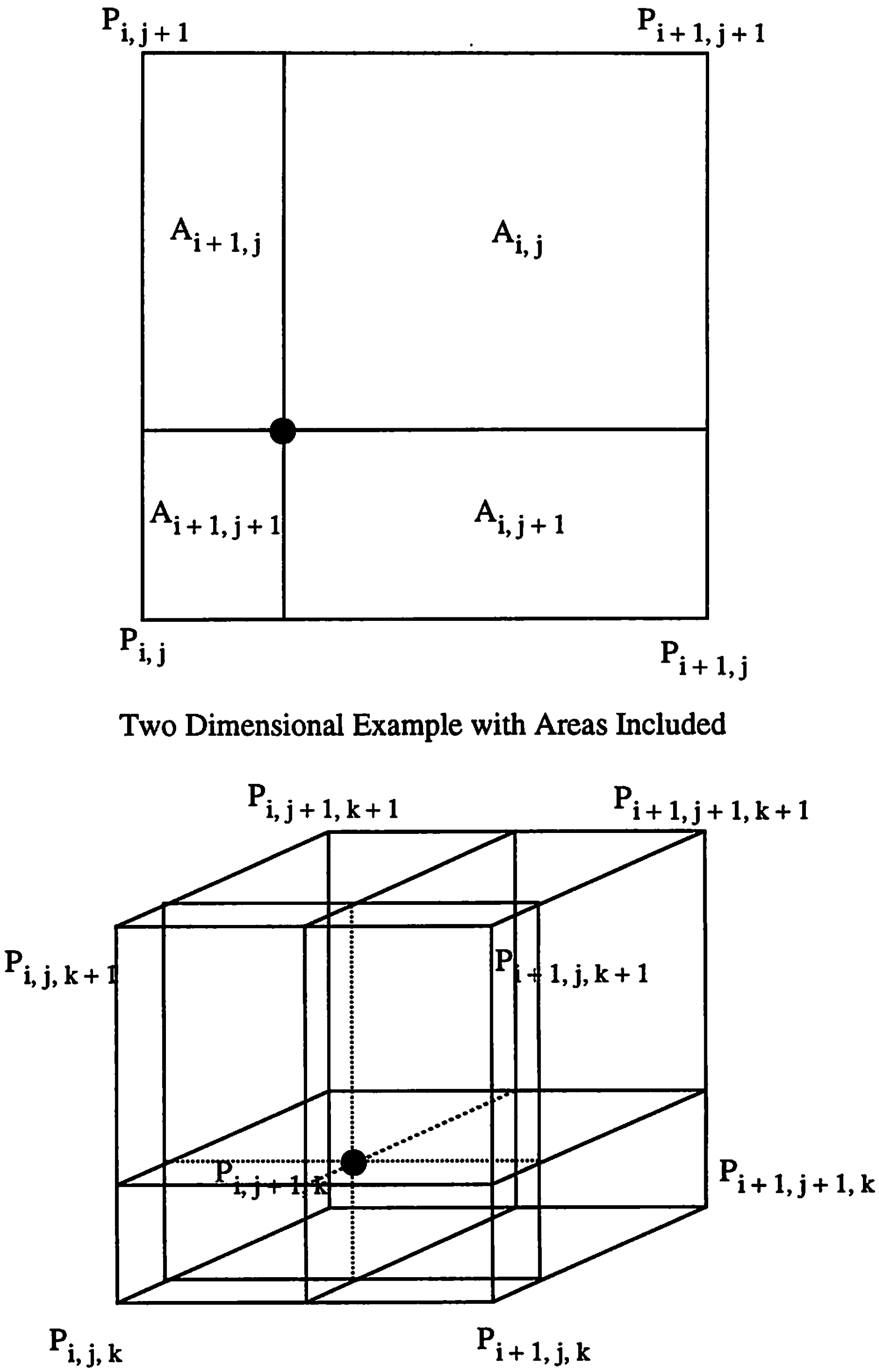
The rate function that is computed by bleach is given to the dissolution simulators in the form of a binary format file. Because BLEACH uses a three dimensional Euclidean grid to compute active compound concentration, the data is written as an array. A small section of header information is written at the beginning of the file, so that the size and dimensions of the array are also included. Ray-trace (DEVELOP), level-set (ADVECT) and the cell-method (CRATER) all use the same code to read the rate information from the file. The etch rate at a point in the dissolution simulation region that is correspondent with an input file grid point receives the etch rate value expressed in the file. If the point is not coincident with a grid point, the rate at that location is interpolated from the surrounding grid points. The method of etch rate interpolation is shown in Figure 1. A two dimensional representation of the interpolation method is shown in the top half of Figure 1. The equation that describes how the etch rate is interpolated from Figure 1 is [EQ: 1]. The etch rate at the large point is:

$$A_{i,j}P_{i,j} + A_{i+1,j}P_{i+1,j} + A_{i,j+1}P_{i,j+1} + A_{i+1,j+1}P_{i+1,j+1} \qquad \text{[EQ: 1]}$$

where:

$$A_{i,j} + A_{i+1,j} + A_{i,j+1} + A_{i+1,j+1} = 1 \qquad . \qquad \text{[EQ: 2]}$$

The values $A_{i,j}$, $A_{i+1,j}$, $A_{i,j+1}$ and $A_{i+1,j+1}$ are the areas of the rectangles formed by the boundaries of the particular cell where the interpolation is being performed, and the coordinate lines that pass through the interpolation point. These areas are normalized to form the basis for a weighted average so that the etch rate can be computed with input from each of the four grid points [EQ: 2]. Each grid point etch rate value $P_{i,j}$, $P_{i+1,j}$, $P_{i,j+1}$ and $P_{i+1,j+1}$ is multiplied by the normalized area of its corresponding rectangle and summed to determine the interpolated etch rate, as shown in [EQ: 1]. The quadrant associated with each grid point is always diagonally opposite from that point. On each edge of the grid, the interpolation is equivalent to a simple linear interpolation. In the bottom part of Figure 1 is a subdivision of a three dimensional cell that represents the three dimensional linear interpolation that is used in SAMPLE-3D for the etch rate. This interpolation is similar to the two dimensional case, except that eight terms are used and the multiplicative factors for the etch rates at the grid points are the normalized volumes that are fully opposite from each grid point. This interpolation method is identical to the two dimesnial interpolation method for



Two Dimensional Example with Areas Included

Three Dimensional Example

Figure 1) Interpolation of the Rate Function the faces of the cube. This method also generates a linear interpolation of the etch rate on each of the 12 edges.

The ray-trace method requires the gradient of the etch rate to be derived from the etch rate data. This is performed by computing the gradient from the etch rate of the neighboring points using the analytical derivative of the three dimensional version of [EQ: 1]. While the computed gradient is the exact gradient of the etch rate function as interpolated, the gradient is not necessarily continuous across the boundary of a particular grid cell. This is a possible source of error in the present ray-trace method. However, it is not clear that other interpolation methods, while they may generate a continuous gradient, are preferable, since they may be computationally expensive or may yield a set of gradients that do not conform to the derivative of the interpolated etch rates.

4.3 Implementation of the Ray-Trace Method

4.3.1 Triangles, Segments and Nodes

The SAMPLE-3D ray-trace advancement method employs a triangular mesh with a winged edge data structure. The typical configuration of the initial surface is shown in Figure 2. In photolithography simulation, it is assumed that the initial surface of the resist is a plane at a height specified by the process. The basic components of the mesh are the triangles, segments and nodes. There is an additional mesh object that does not appear as a visible object in the mesh. This object is called a node-segment object.

Each of the four objects has a unique data structure, although there are some similarities. Each object contains an object ID, which is an integer that identifies the object as unique among other objects of its type. (Objects may have the same ID identifier if they are of different types, for instance triangle 1074 and segment 1074, but no two

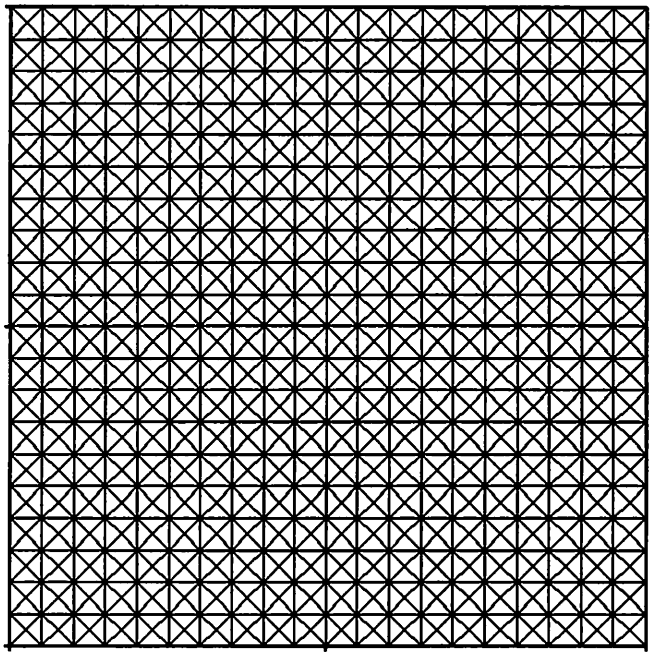
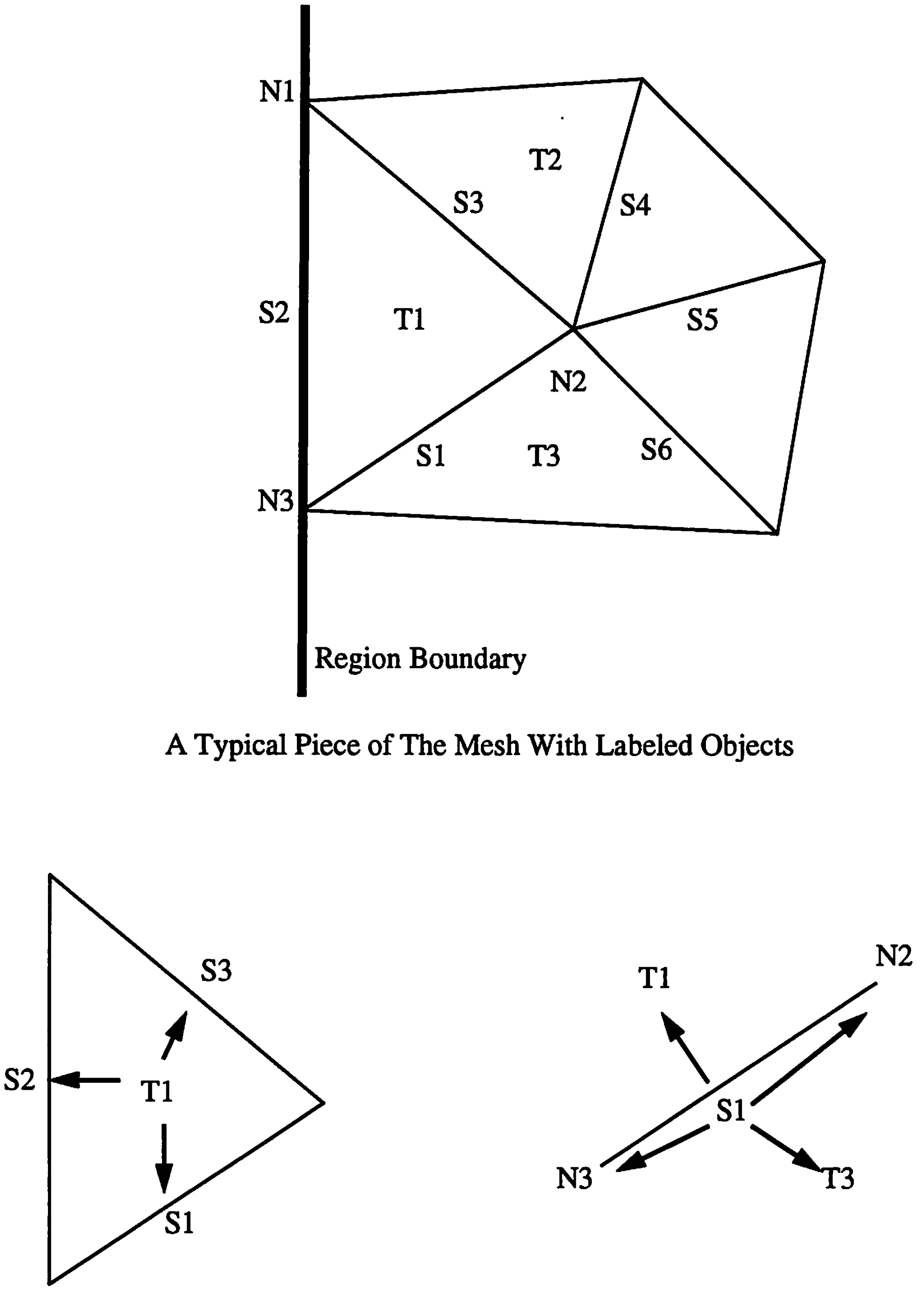


Figure 2) Initial Mesh Configuration (Top View)

objects of the same type can have the same ID.) Each object type contains two pointers that are used to form doubly linked lists. Each object type has its own doubly linked list that contains all objects of that type in the mesh. There are 8 global variables associated with the lists, which are the 4 pointers to the heads of the lists and the 4 pointers to the tails of the lists. Finally, each mesh object also contains connectivity pointers that describe the topology of the mesh. An example that illustrates how these pointers express the mesh connectivity is shown in Figure 3 and in Figure 4.

Each triangle in SAMPLE-3D contains pointers to its neighboring segments. The triangle T1 in Figure 3 contains three pointers. These point to the segments S1, S2 and S3. Each segment contains pointers to both the two neighboring triangles and the two nodes that form the endpoints of the segment. If the segment is a boundary segment, as in Figure 4, then the second triangle pointer of the segment is set to NULL to signify

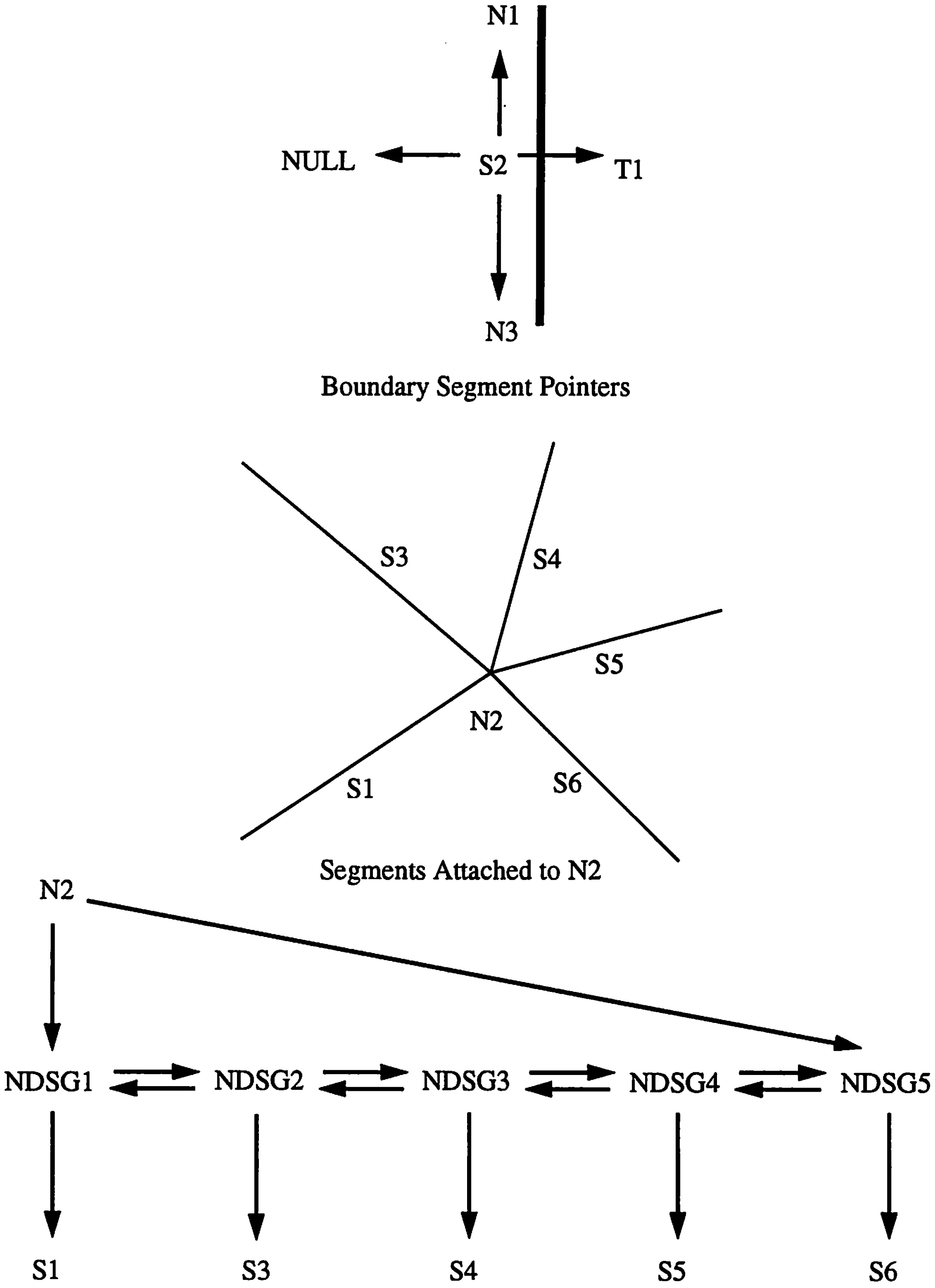


A Typical Piece of The Mesh With Labeled Objects

Triangle Pointers

Segment Pointers

Figure 3) SAMPLE-3D Mesh Objects



Boundary Segment Pointers

Segments Attached to N2

The Node-Segment List

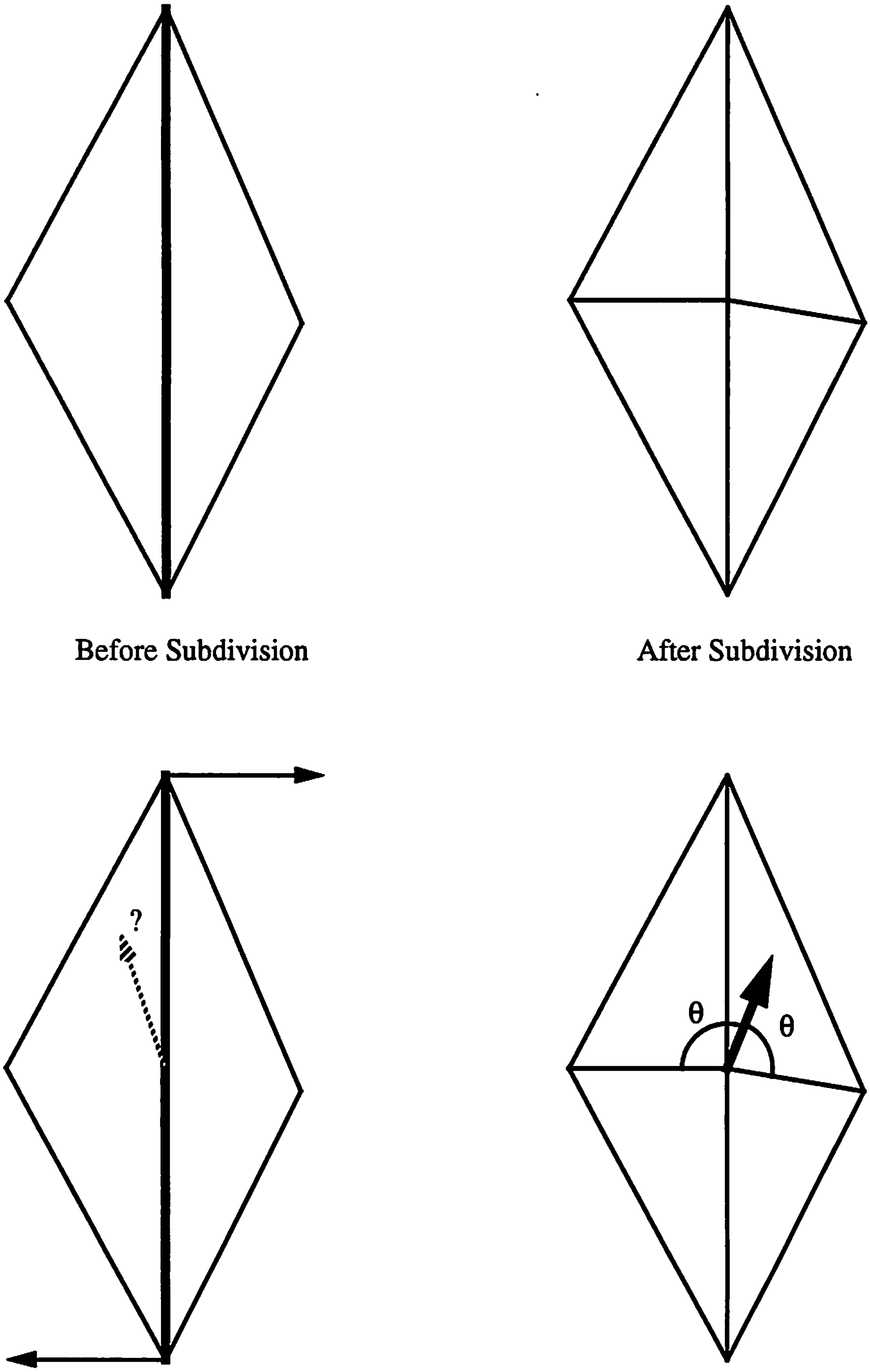
Figure 4) SAMPLE-3D Mesh Objects (Continued) that no triangle exists across the boundary. Nodes, because an arbitrary number of segments may be connected to them, need an additional object, called a node-segment object, to assist in storing connectivity information. Each node contains two fields that point to the head and tail of a doubly linked list. The list is formed of node-segment objects. Other than the four pointers that form the node specific list and the list of node-segement objects, only one pointer is contained in this object. This pointer points to one of the segments that is attached to the node as seen in Figure 4. The entire list of node-segment objects associated with a specific node has exactly one node-segment object for each segment that is attached to the node. The list of segments is unordered. Specific routines are employed to determine which nodes are attached to a specific triangle. It is generally not necessary to determine which triangles are attached to a specific node for performing ray-trace functions.

In addition to connectivity pointers, each node contains two other important fields. The first is the coordinate information that represents the location of the node in the simulation region. This information is a standard coordinate representation that contains three floating point numbers that represent the x, y and z coordinates. The coordinates of the triangles and segments are determined by referring to the coordinates on each node. This makes the mesh simple to advance, since only the coordinates on the nodes must change to advance the mesh to its new position. Each node also contains a unit vector that represents the inverse surface normal at that position. This vector represents the direction of advancement during the next time step, and is called the 'direction vector'. The direction vectors are initialized as x = 0.0, y = 0.0 and z = -1.0 to correspond with the initial surface shown in Figure 2. The mesh is advanced by updating the coordinate and direction vector for each node for each time step.

4.3.2 Basic Mesh Maintenance

K. Toh implemented two basic mesh maintenance operations in SAMPLE-3D [1]. These methods were segment merging and segment division. These operations were performed for the purpose of removing small segments and subdividing long segments. The operations were performed to regularize the mesh, so that rays did not become too close or too distant. To remove small segments, one node on the segment is removed. This is performed by removing the node and reconnecting the segments that were attached to the node that is being removed to the node of the segment that remains. The two triangles that were on either side of the segment are removed. The two segments of the removed triangles that were not the small segment that is being removed, are combined into one segment. A more complete discussion of the process, which includes three-dimensional effects, is given in Chapter 6.

To subdivide large segments, a new node is placed at the midpoint of the large segment, thereby forming two smaller segments. Each of the triangles on either side of the segment were also subdivided into two smaller triangles, as seen in Figure 4. Because a new node is formed during subdivision, a ray vector must be given to it. This is performed by ray interpolation. In the original version of SAMPLE-3D, the ray vectors were set by linearly averaging the vectors of the nodes at opposite ends of the original large segment. This method was found to be inappropriate for resist deloop, since interpolating points from large segments that are attached to banana nodes generates meaningless results. This difficulty was overcome by using the local geometry instead of the neighboring rays for interpolation. The unit surface normals of the two origonal neighboring triangles are averaged and normalized to form the ray. If the long segment is on the boundary, only one original neighboring triangle to the long segment exists. In this case, the ray is placed in the direction of the surface normal of the triangle. This method of generating rays is based on the rarefaction fan discussion in Chapter 3.



Before Subdivision

After Subdivision

Old Interpolation

New Interpolation

Figure 5) Segment Subdivision

4.3.3 Ray Trace Advancement Equations

The ray-trace method as applied to photolithography in three dimensions was originally developed by Kenny Toh [1]. The mathematics that describe the advancement of the surface are shown in Chapter 3. The ray advancement equation that was developed in Chapter 3 was:

$$\frac{d}{ds}\left[\frac{1}{R(x,y,z)}\frac{dr}{ds}\right] = \nabla\frac{1}{R(x,y,z)} \qquad \text{[EQ: 3]}$$

where $R(x, y, z)$ is the etch rate, s is the arc length and r is the position vector. The notation s will be used in place of $\frac{dr}{ds}$ for the purpose of clarity. s is the unit tangent vector for the path of the ray and is called the 'direction' vector. [EQ: 3] is simplified by applying the chain rule to the right side:

$$\frac{d}{ds}\left[\frac{1}{R}s\right] = -\frac{1}{R^2}\nabla(R)\ . \qquad \text{[EQ: 4]}$$

This equation is now rewritten in its discretized form. If Δs is the distance etched in some time step ΔT in an average etch rate R_{ave}, i.e. $\Delta s = R_{ave}\Delta T$, then

$$\frac{1}{\Delta s}\Delta\left[\frac{1}{R}s\right] = -\frac{1}{R_{ave}^2}\nabla(R)\ . \qquad \text{[EQ: 5]}$$

By rearranging the terms, the equation may be simplified.

$$\Delta\left[\frac{1}{R}s\right] = -\frac{1}{R_{ave}^2}\nabla(R)R_{ave}\Delta T = -\frac{1}{R_{ave}}\nabla(R)\Delta T \qquad \text{[EQ: 6]}$$

If the equation [EQ: 6] is applied between to two points P_1 and P_2, then it becomes,

$$\frac{s_2}{R_2} - \frac{s_1}{R_1} = -\frac{1}{R_{ave}}\nabla(R)\Delta T, \qquad \text{[EQ: 7]}$$

$$\frac{s_2}{R_2} = -\frac{1}{R_{ave}}\nabla(R)\Delta T + \frac{s_1}{R_1}, \qquad \text{[EQ: 8]}$$

$$s_2 = -\frac{R_2}{R_{ave}}\nabla(R)\Delta T + s_1\frac{R_2}{R_1}, \qquad \text{[EQ: 9]}$$

$$s_2 - s_1 = -\frac{R_2}{R_{ave}}\nabla(R)\Delta T + s_1\left[\frac{R_2}{R_1} - 1\right]. \qquad \text{[EQ: 10]}$$

The average etch rate R_{ave} may be written as

$$\left[\frac{1}{R}\right]_{ave} = \frac{1}{2}\left[\frac{1}{R_1} + \frac{1}{R_2}\right]. \qquad \text{[EQ: 11]}$$

Therefore, the discrete etch rate equation becomes

$$s_2 - s_1 = -0.5\frac{(R_1 + R_2)}{R_1}\nabla(R)\Delta T + s_1\left[\frac{R_2}{R_1} - 1\right] \qquad \text{[EQ: 12]}$$

The vector s_2 is then renormalized to a unit vector after the application of this equation. This equation relates the difference between the unit vectors s_1 and s_2 to the gradient of the etch rate. The advancement method can now be summarized. First, the point is advanced from location $\mathbf{r}_1$ in the direction of its direction vector:

$$\mathbf{r}_2 = \mathbf{r}_1 + s_1 R_1 \Delta T \qquad \text{[EQ: 13]}$$

After advancement, the deviation of the direction vector is calculated according to [EQ: 12]. If the length of the deviation, i.e. $\|s_2 - s_1\|$, is less than 0.1, then $\mathbf{r}_2$ becomes the coordinate of the point after advancement. If, however, $\|s_2 - s_1\|$ is greater than 0.1, then the advancement is recomputed. The time step is halved and two successive advancement steps are employed to advance the point. This halving of the time step is performed recursively until all advancement steps have direction vector deviations of less than 0.1. This recursive time step capability allows the rays to advance accurately through regions of rapidly changing etch rate. This recursive time step ability is not equivalent to a second order method. Since an approximation of the subdivision of the time step as suggested by [EQ: 12] can be precomputed before any surface advancement, this implementation of ray advancement is a first order method that employs an adaptive gridding scheme.

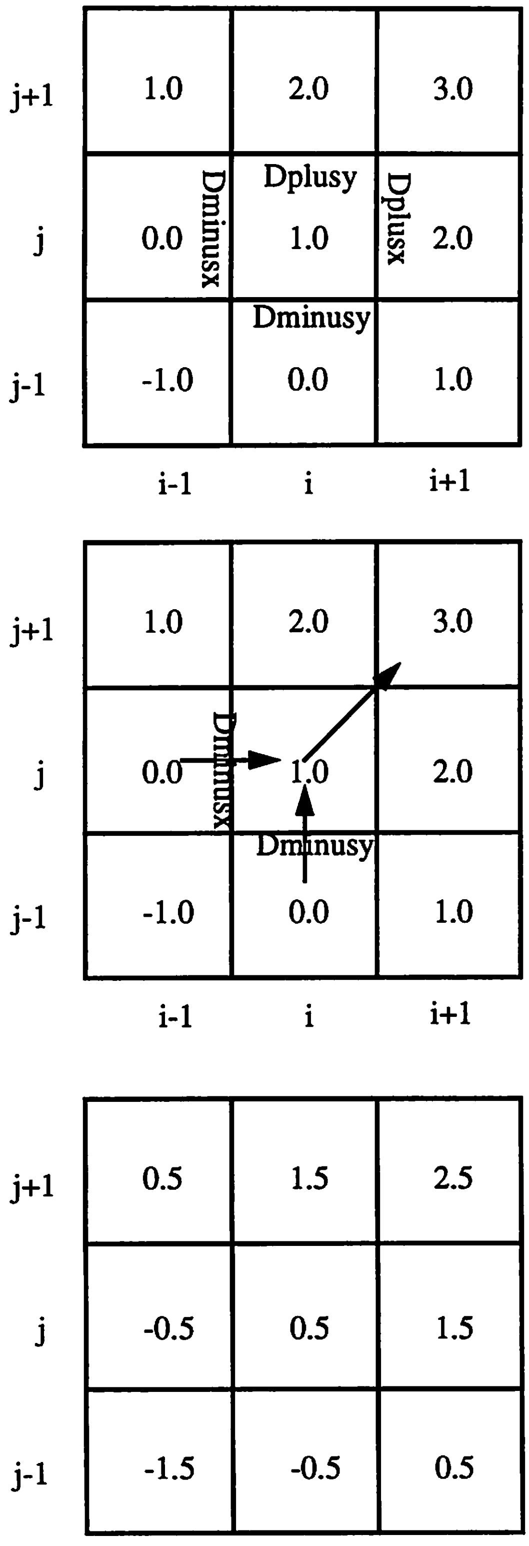
The whole surface is advanced by applying the above procedure to every node in the mesh simultaneously. Proper mesh maintenance, a topic that will be elaborated upon in Chapter 6, requires that no point move more than a certain specified distance. In this case, the specified distance is 15% of the ideal segment length. This is enforced by first finding the etch rate at every node in the mesh, and then computing the time step for advancement by dividing the specified distance by the maximum rate. Each point will, therefore, move a distance equal to or less than the specified distance. Besides allowing proper mesh maintenance, adaptive time steps are also useful in photolithography simulation, since the maximum etch rate at the surface can vary widely during simulation. Therefore, an adaptive time step can allow simulation to proceed much more quickly than advancement that is limited by the fastest etch rate ever encountered during simulation.

4.4 Level-Sets

4.4.1 Computing the Evolution of the Values on the Grid

This entire method, except for the section on iteration, is taken from the work of J. Sethian and J. Strain in [2] where it was employed for simulating crystal growth. The level-set method simulates the advancement of the surface through the use of a function that represents the surface as the 0-contour of the function, i.e. the surface is located wherever the function takes on the value of 0. This simulation takes place on a regular Euclidean grid. As previously noted in Chapter 3, this function must be initialized to values proportional to the distance from the surface. Because the initial surface is a flat plane at the top of the simulation region, the cells are assigned values equivalent to the distance of the center of each cell from the top of the simulation region. The etch rates are assigned for each Euclidean cell by computing the value of the etch rate at the center of the each cell.

Once the function and the etch rates are defined, advancement can begin. At each time step, a basic operation is performed on each cell. This operation first computes the Euclidean norm of the gradient of the function at that point. The Euclidean norm of the gradient is employed in conjunction with the etch rate to update the value of the cell for the next time step. The method for computing the gradient is shown in Figure 6 in two-dimensional form [2]. Figure 6 represents a linearly increasing function whose gradient points to the upper right. To compute the gradient, it is necessary to employ the values of the neighboring cells in the computation. Because the monotonicity of the level-set function must be preserved, only neighboring cell values are considered in computing the gradient that are smaller than the value in the cell whose gradient is being calculated. Four difference operators are employed. The operators are employed across each cell edge to measure the partial derivative in that direction. These difference operators are marked as Dminusx, Dplusx, Dminusy and Dplusy in Figure 6



A) Before Advancement

B) Gradient Calculation

C) Advancement by time step of $\frac{\sqrt{2}}{4}$

Figure 6) Level-Set Advancement Method

and stand consecutively for the relations $D_x^{minus}\phi_{i,j}$, $D_x^{plus}\phi_{i,j}$, $D_y^{minus}\phi_{i,j}$, and $D_y^{plus}\phi_{i,j}$. The operators are defined by the equations:

$$D_x^{minus}\phi_{i,j} = \frac{\phi_{i,j}^n - \phi_{i-1,j}^n}{\Delta x} \qquad \text{[EQ: 14]}$$

$$D_x^{plus}\phi_{i,j} = \frac{\phi_{i+1,j}^n - \phi_{i,j}^n}{\Delta x} \qquad \text{[EQ: 15]}$$

$$D_y^{minus}\phi_{i,j} = \frac{\phi_{i,j}^n - \phi_{i,j-1}^n}{\Delta y} \qquad \text{[EQ: 16]}$$

$$D_y^{plus}\phi_{i,j} = \frac{\phi_{i,j+1}^n - \phi_{i,j}^n}{\Delta y} \qquad \text{[EQ: 17]}$$

where Δx and Δy are the dimensions of the cells in the x and y directions and $\phi_{i,j}^n$ is the value of the cell in the column i in row j at time step n. In the three-dimensional case, six operators exist, one for each face of the three-dimensional cell. These operators are $D_x^{minus}\phi_{i,j,k}$, $D_x^{plus}\phi_{i,j,k}$, $D_y^{minus}\phi_{i,j,k}$, $D_y^{plus}\phi_{i,j,k}$, $D_z^{minus}\phi_{i,j,k}$ and $D_z^{plus}\phi_{i,j,k}$. The norm of the gradient can now be computed:

$$xdir = ((\max(D_x^{minus}\phi_{i,j}, 0))^2 + (\min(D_x^{plus}\phi_{i,j}, 0))^2) \qquad \text{[EQ: 18]}$$

$$ydir = ((\max(D_y^{minus}\phi_{i,j}, 0))^2 + (\min(D_y^{plus}\phi_{i,j}, 0))^2) \qquad \text{[EQ: 19]}$$

$$|\nabla\phi_{i,j}| = \sqrt{(xdir + ydir)} \qquad \text{[EQ: 20]}$$

An additional zdir term, which is created in the same manner as [EQ: 18] and [EQ: 19], is required for three-dimensions. In three dimensions, xdir, ydir and zdir are all gathered under the square root sign in [EQ: 20]. Now that the gradient has been computed for each point, advancement can take place. It is necessary that the time step satisfy the relations in [EQ: 21], [EQ: 22] and [EQ: 23], where maxrate is the fastest etch rate in the resist. It has been suggested that a value for Δt that half of the one suggested by the relations is desirable [3]. The value used in the examples in Chapter 7 is $0.5 * \Delta z/\text{maxrate}$.

$$\Delta t < \frac{\Delta x}{\text{maxrate}} \qquad \text{[EQ: 21]}$$

$$\Delta t < \frac{\Delta y}{\text{maxrate}} \qquad \text{[EQ: 22]}$$

$$\Delta t < \frac{\Delta z}{\text{maxrate}} \qquad \text{[EQ: 23]}$$

The Hamilton-Jacobi equation can now be solved over the mesh. At each time step, for all i, j and k, all the cells are updated according to the following formula:

$$\phi^{n+1}_{i,j,k} = \phi^{n}_{i,j,k} - R_{i,j,k}\Delta t\left|\nabla \phi^{n}_{i,j,k}\right| \qquad \text{[EQ: 24]}$$

where n is the time step and $R_{i,j,k}$ is the etch rate at cube i, j, k. This equation is repeated until the desired simulation time is reached.

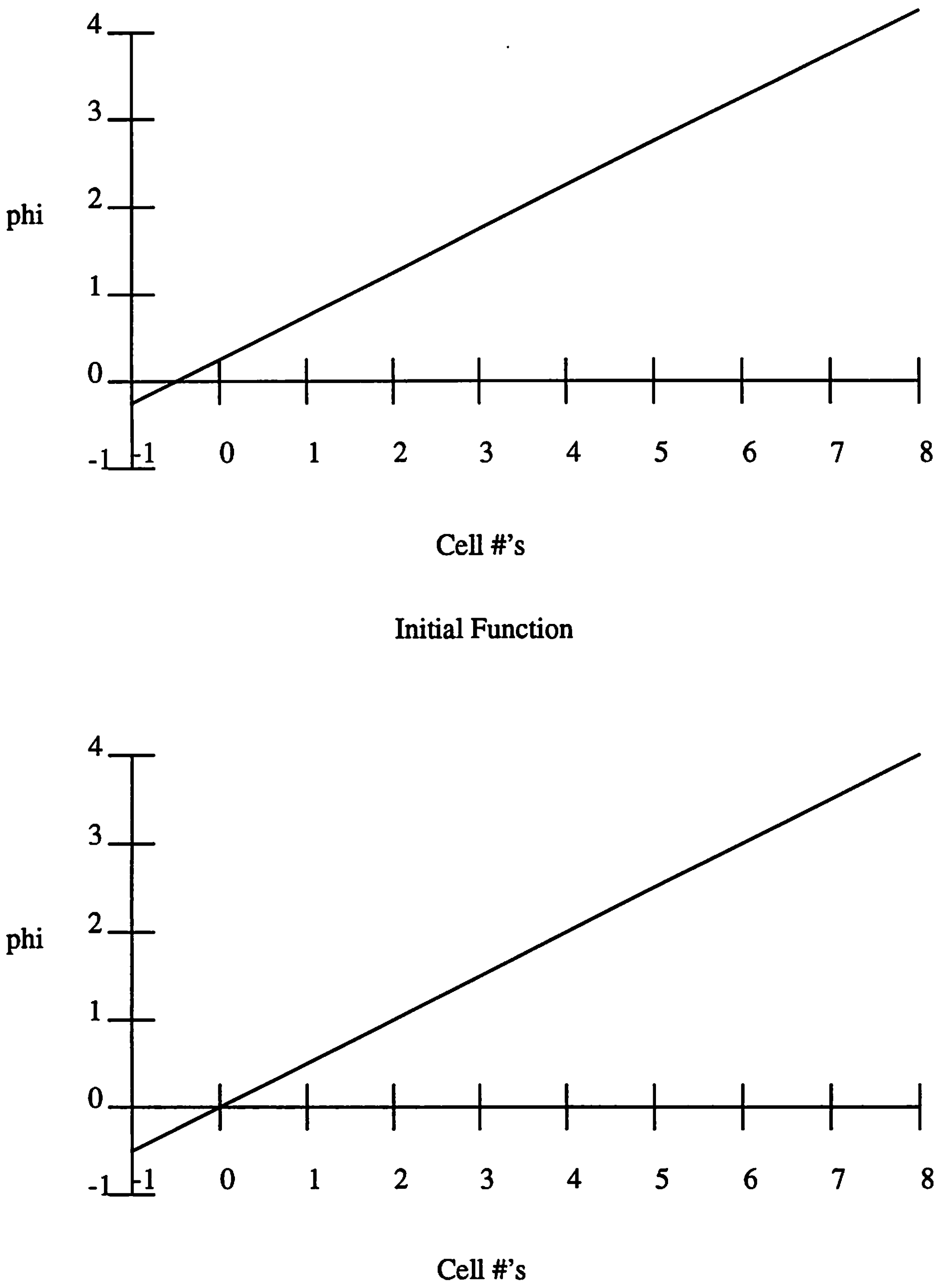
4.4.2 Boundary Conditions

The simulation region contains 6 boundaries, one for each side of the simulation region. The boundaries that are perpendicular to the initial surface are reflective boundaries, since the rate function that is generated by SPLAT and BLEACH is reflective. As per the discussion in Chapter 3, the zero etch rate region at the bottom of the simulation region that represents the silicon substrate is also an reflective boundary condition. If the side of a cell is part of the boundary, then the side of that cell is assumed to have a cell with the same value as the original on the other side of the boundary.

The boundary condition that represents the original location of the initial surface (i.e. the top of the simulation region) is required to introduce new contours into the system, so that the 0 contour can easily evolve into the rest of the resist. This is performed as illustrated in Figure 7. This figure is analogous to a 1x1x8 section of cells that extends from the surface to 8 cells deep into the resist. This particular example assumes an etch rate of 2 cells/sec exists and a time step 0.25 sec. The cell markers represent the centers of the cells. The surface is initially set so that the 0 contour is at the left hand side of cell 0. This requires cell -1, the boundary cell to be set to -0.25. For each time step, the boundary cell is decreased in value by the value of the time step. The bottom half of Figure 7.shows the advanced surface with a properly formed contour. In ADVECT, the border cells have the values of -t-0.5*Δz. This ensures that the border cells always have values that are less than the values in the resist. A physical analogy can be made by assuming that the new contours represent a the continuing penetration of more etchant into the resist.

4.4.3 Iterative Approach

A new iterative approach to the level-set technique has been implemented. This method is based on concepts from J. Sethian in [4] that were applied to grid generation. This is the first time that this concept has been applied to photolithography simulation. While it appears very promising, the results are tentative. This method operates in the same manner as the original level-set scheme with a single difference. In the original method, the contour that was evolved was the 0 contour and the method terminated at time t, where t is the simulation time. In this method, evolution continues in the same manner as the original method until time t + n. The -n contour is returned as the result. This method, therefore, uses the first n time units of the simulation to set the initial conditions. It is assumed that the accuracy improvement occurs, because the



Initial Function

Function After One Time Step

Figure 7) Level-Set Upper Boundary Condition best initial condition for evolving the surface contour probably is the eikonal function. The first n time steps are used, therefore, to generate an approximation of the eikonal.

4.5 Cell Method

Finally, the cell method that was developed by Ed Scheckler [5] was employed. This method divides the region, like the level-set method, into a Euclidean grid. In this case, however, instead of evolving a scalar function across the grid, the grid represents the photoresist volume directly. Each cell represents a small piece of volume in the simulation region. Each cell contains a number between 0.0 and 1.0 called the 'volume fraction' that represents the fractional volume of the photoresist contained in the cell. A value of 0.0 represents no photoresist, and a value of 1.0 represents a cell completely filled with photoresist. The cell method advances the surface by lowering the volume fraction in cells that are in contact with the surface. The technique for lowering the volume fraction varies from method to method. In this particular method, a rate of volume removal will be computed by examining the number of exposed faces of a cell and its etch rate.

The method for approximating the volume removal rate is shown in Figure 8. If a neighboring cell is devoid of material, the cell face shared with that neighbor is exposed. All exposed faces are moved into the cell, with a distance given by the product of the etch rate at the cell and the time step. If only one face is exposed, then the amount of volume removed is the product of the face area and the distance traveled. If two edge-sharing faces are exposed, a correction factor of $1/\sqrt{2}$ for cubic cells is included in the rate to slow the advance of the planes. If three vertex-sharing planes are exposed, the correction factor is $1/\sqrt{3}$ for cubic cells. The volume removed is estimated as the amount swept out by all of the moving planes:

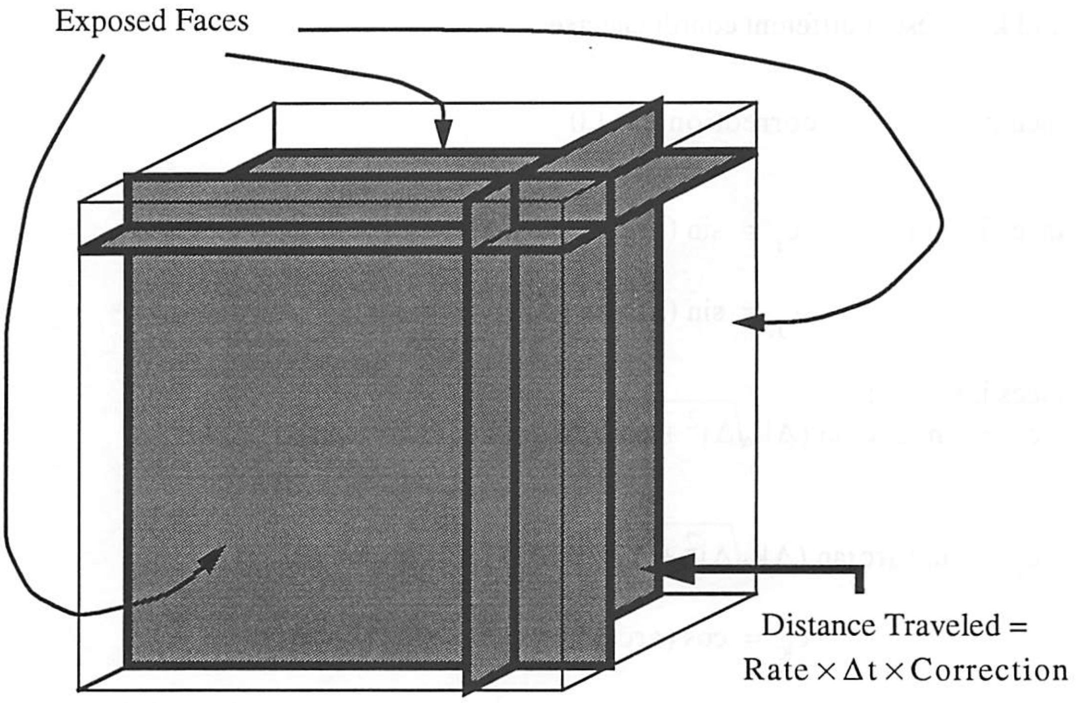


Figure 8) Cell Advancement Algorithm

$$
\begin{aligned}
\mathrm{Volume} = {} & d_{xl}d_{yl}d_{zl} + d_{xl}d_{yl}d_{zh} + d_{xl}d_{yh}d_{zl} + d_{xl}d_{yh}d_{zh} + d_{xh}d_{yl}d_{zl} \qquad \text{[EQ: 25]} \\
& + d_{xh}d_{yl}d_{zh} + d_{xh}d_{yh}d_{zl} + d_{xh}d_{yh}d_{zh} \\
& + (\Delta x - d_{xh} - d_{xl})\,(d_{yl}d_{zl} + d_{yl}d_{zh} + d_{yh}d_{zl} + d_{yh}d_{zh}) \\
& + (\Delta y - d_{yh} - d_{yl})\,(d_{xl}d_{zl} + d_{xl}d_{zh} + d_{xh}d_{zl} + d_{xh}d_{zh}) \\
& + (\Delta z - d_{zh} - d_{zl})\,(d_{xl}d_{yl} + d_{xl}d_{yh} + d_{xh}d_{yl} + d_{xh}d_{yh}) \\
& + (\Delta x - d_{xh} - d_{xl})\,(\Delta z - d_{zh} - d_{zl})\,(d_{yl} + d_{yh}) \\
& + (\Delta y - d_{yh} - d_{yl})\,(\Delta z - d_{zh} - d_{zl})\,(d_{xl} + d_{xh}) \\
& + (\Delta x - d_{xh} - d_{xl})\,(\Delta y - d_{yh} - d_{yl})\,(d_{zl} + d_{xh})
\end{aligned}
$$

where d_{il}, d_{ih} are the etch distances (rate x Δt) of the two planes perpendicular to the i-axis, and Δx, Δy and Δz are the cell dimensions. The correction factors for one, two or three adjacent faces are derived as if the etch front were advancing along an inclined plane placed on top of the cells, and are given as follows, where each of i, j and k represent different coordinate axes:

face i: $\text{correction}_i = 1.0$

faces i and j: $c_i = \sin(\arctan(\Delta i/\Delta j))$

$$c_j = \sin(\arctan(\Delta i/\Delta j))$$

faces i, j and k:

$$c_i = \sin(\arctan(\Delta k\sqrt{\Delta i^2 + \Delta j^2}/\Delta i\Delta j))\cos(\arctan(\Delta i/\Delta j))$$

$$c_j = \sin(\arctan(\Delta k\sqrt{\Delta i^2 + \Delta j^2}/\Delta i\Delta j))\sin(\arctan(\Delta i/\Delta j))$$

$$c_k = \cos(\arctan(\Delta k\sqrt{\Delta i^2 + \Delta j^2}/\Delta i\Delta j))$$

The volume removal calculation is always performed as if the cell were completely full until it is removed entirely.

The overetch technique, that was discussed in Chapter 3, is illustrated in Figure 9. Because the cell method can operate more quickly if the time step is not limited by the first cell that reaches a value of 0.0 during etching, some values are allowed to decrease beyond 0.0 and become negative. To reset this volume fraction to 0.0, the extra necessary photoresist is equitably removed from all the cells that share a face with the original cell. For typical time steps, this excess volume is typically only a few percent of the total cell volume.

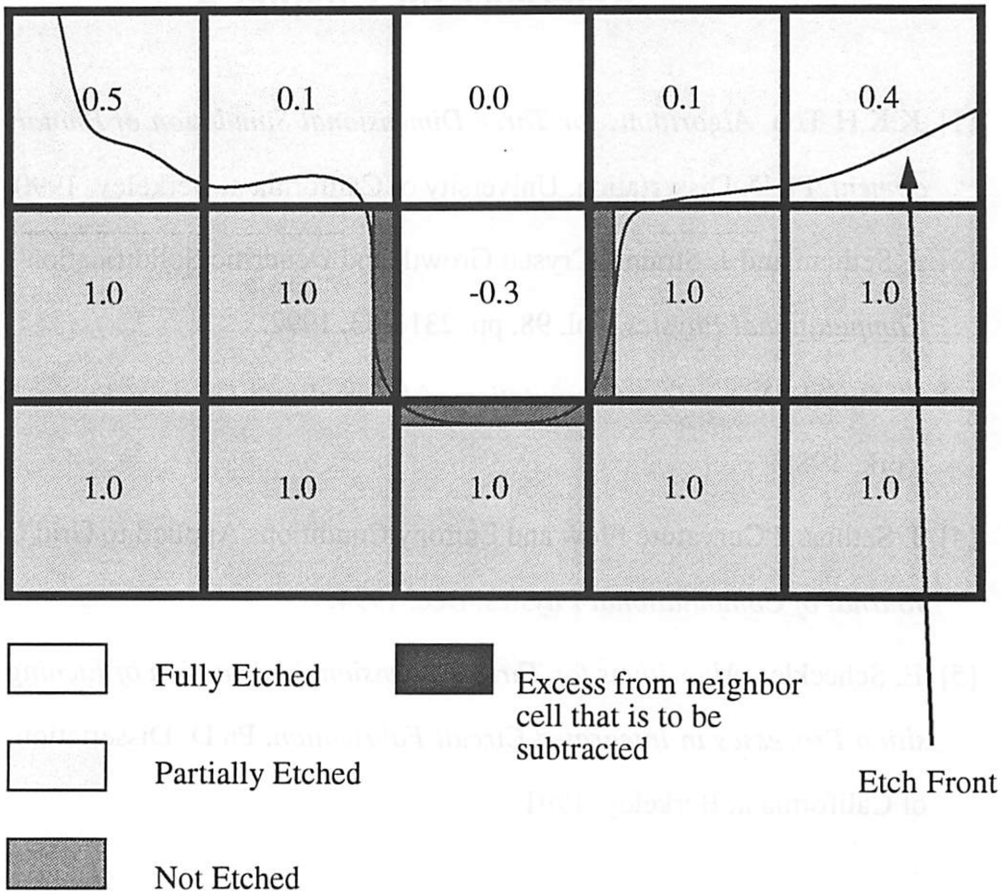


Figure 9) Cell Spillover Mechanism

Reference for Chapter 4

[1] K.K.H Toh, *Algorithms for Three-Dimensional Simulation of Photoresist Development*, Ph.D. Dissertation, University of California at Berkeley, 1990.

[2] J. Sethian and J. Strain, "Crystal Growth and Dendritic Solidification", *Journal of Computational Physics*, vol. 98, pp. 231-253, 1992.

[3] C. Hirsch, *Numerical Computation of Internal and External Flows*, Wiley, New York, 1988.

[4] J. Sethian, "Curvature Flow and Entropy Conditions Applied to Grid Generation", *Journal of Computational Physics*, Dec. 1994.

[5] E. Scheckler, *Algorithms for Three-Dimensional Simulation of Etching and Deposition Processes in Integrated Circuit Fabrication*, Ph.D. Dissertation, University of California at Berkeley, 1991.

Chapter 5 Deloop

5.1 Need for Deloop in Surface Advancement Algorithms

While level-set and cell techniques handle topological changes automatically, surface advancement techniques have different requirements. The surface advancement method causes perturbational changes in the surface representation at each time step, while preserving local connectivity. Although these changes preserve connectivity, they do not necessarily represent a valid surface. During the perturbation, the surface may form self-intersecting regions. Two types of regions may be formed. (Figure 1) The first type occurs when the region intersects itself locally. The surface,

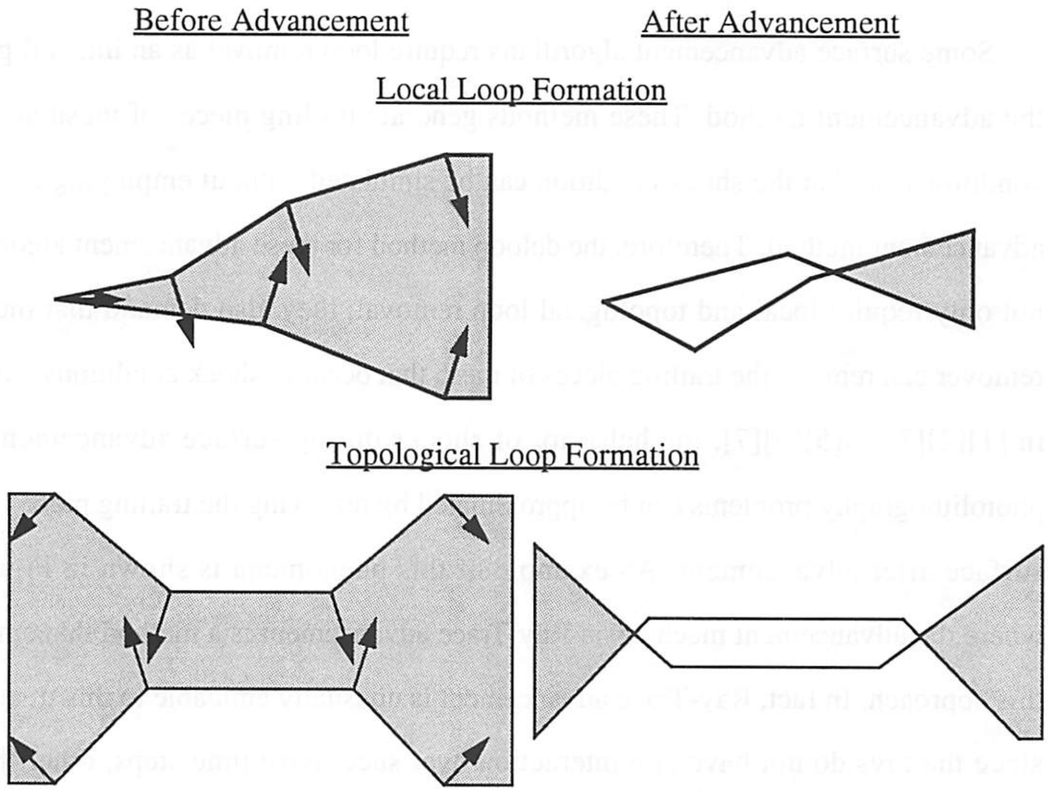


Figure 1) Loop Formation During Etching

under certain perturbation conditions, may pass through itself during an advancement step. This is called 'local loop formation' due to the loop-like cross section that is observed after this advancement. Certain advancement methods prevent local loop formation, so this type of loop does not occur in all methods. Topological alterations, which can occur with any advancement method, such as hole formation and material detachment, cannot be described as local loops, but are similar to them in two important ways. First, topological alterations are similar to loops, because the mesh intersects itself after advancement. Second, to get a self-consistent non-intersecting surface after advancement that represents the actual material removed, pieces of the surface mesh must be removed. A method is given in this chapter that corrects both of these advancement difficulties. Because the method was originally invented to remove local loops, it is called 'Deloop' [15].

Some surface advancement algorithms require loop removal as an integral part of the advancement method. These methods generate trailing pieces of mesh at shock conditions, so that the shock condition can be simulated without employing a special advancement method. Therefore, the deloop method for these advancement algorithms not only require local and topological loop removal, they also demand that the loop remover can remove the trailing pieces of mesh that occur at shock conditions. As seen in [1][2][3][4][5][6][7]. the behavior of shocks during surface advancement for photolithography problems can be approximated by removing the trailing pieces of the surface after advancement. An example of this phenomena is shown in Figure 2, where the advancement mechanism Ray-Trace advancement is a method that employs this approach. In fact, Ray-Trace advancement is unusually amicable to this treatment, since the rays do not have any interaction over successive time steps, other than at shocks. Therefore, loop removal can track shock conditions even when it is performed only at every 10th or 20th time step.

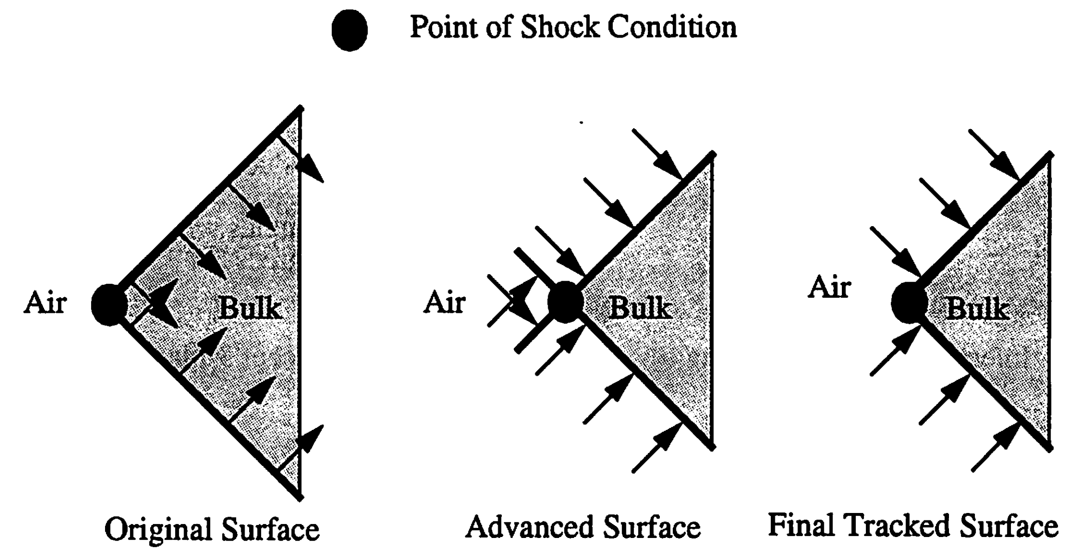


Figure 2) Advancement of Shock Front as Defined by Removal of Rays

Because a deloop algorithm is an essential part of any surface advancement method,[1][3] the loop removal method must satisfy three criteria that are desirable in any program. The method must be fast, since it will be performed often during advancement, possibly at every time step. The method must be robust algorithmically in order to handle complicated structures, since few assumptions can be made about the shapes of surfaces that will be formed in general field applications. Third, it must be numerically robust, so that vertex coincidences or zero area facets will not cause the program to malfunction. The methods presented will satisfy these requirements.

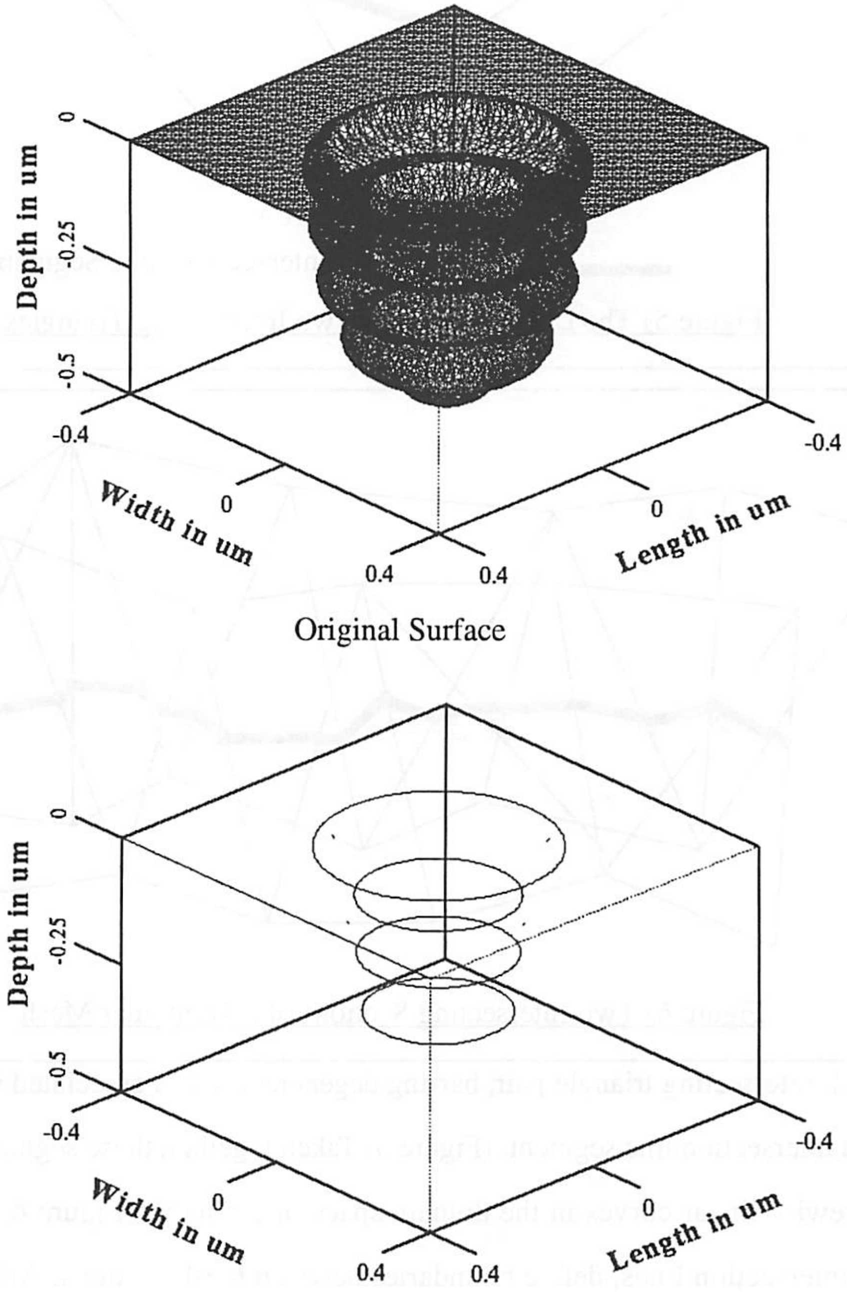
In this chapter, two versions of deloop will be presented. The first will be a generic version that is oriented towards loop removal and topological changes in surface advancement in general. The second is a specialized version that is specifically made for advancement in photoresist. This chapter will also discuss extensions of loop removal to a full set operation implementation that resembles a solid modeler.

Code Section	Level of Implementation
Intersection Location	Handles all cases due to node pushing Has problem with triangles with angles < 10^{-3} degrees
Intersection Line Tracking	Handles most cases, except 3-plane intersections
Triangle Subdivision	Handles most cases, except 3-plane intersections
Removal	Basic binary loop removal implemented Other functions like detached surface detection are implemented

Figure 3) Deloop Routines That Have Been Implemented

5.2 General Characteristics of any Deloop Method

In any loop removal method, there exist three separate steps that must be performed. The first step is to determine whether any loops exist, and if loops do exist, where they are located. This function can be performed by determining if there are any facets that intersect with other facets (Figure 4). This test is nearly identical to the test for loops. (A finding of no intersection may occasionally mean that the entire mesh is a loop.) To determine the locations of the loops, all pairs of intersecting facets must be determined. It is extremely important that this intersection search be performed as fast and as efficiently as possible. It is theoretically possible to do this in $O(N \log N)$ time, where N is the number of triangles in the mesh.



Original Surface

The Lines of Self-Intersection of the Above Surface

Figure 4) Example Surface and its Intersection Lines

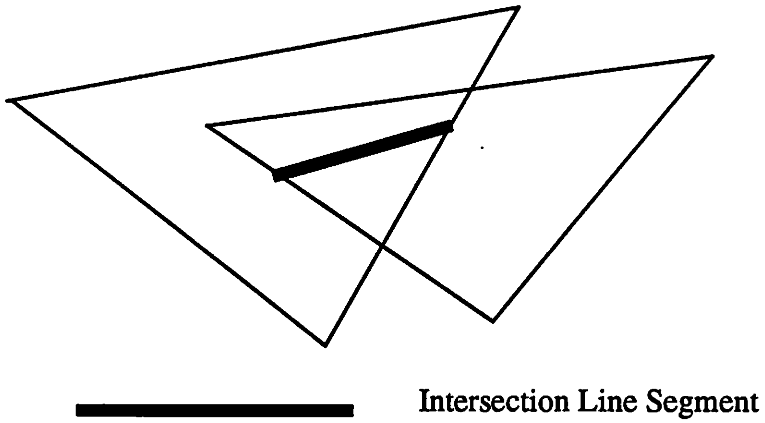


Figure 5) The Line Segment of Two Intersecting Triangles

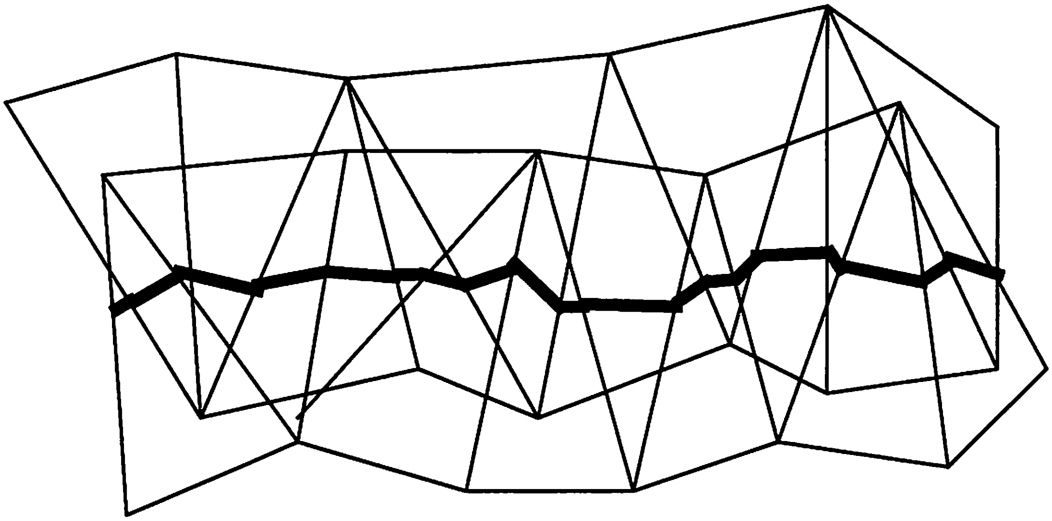


Figure 6) Two Intersecting Sections of a Triangular Mesh

Each intersecting triangle pair, barring degeneracies, has associated with it a well-defined intersection line segment. (Figure 5) Taken together, these segments form a set of piecewise linear curves in the domain space of the mesh (Figure 6). These lines, called intersection lines, define boundaries between mesh sections. All points in any particular section have the same topological identity, i.e. all the triangles in a particular section will be either loop or surface. These sections must be explicitly or implicitly represented in the mesh, before identification of loops and surface can take place.

Because of this requirement, the second part of any deloop algorithm is the partitioning of the mesh along the intersection lines into separate parts.

Third, the loop and surface pieces must be correctly identified, and the loop sections removed. It is important, that the sections of the mesh be labeled correctly, even in extremely degenerate cases. It is also desirable that pieces of the material being etched that have detached themselves from the bulk be properly identified.

5.3 The Triangle Intersection Test and the Octtree

A naive facet intersection algorithm would test all pairs and thus take $\frac{N^2}{2}$ operations, where N is the number of facets that the algorithm takes as input, i.e. the number of triangles in the mesh. The previous implementation of triangle intersection in SAMPLE-3D employed this method. This approach is clearly inefficient, since most pairs of triangles in a 'typical' mesh involve triangles that are widely separated in space relative to the dimensions of the triangles. Typical meshes are considered to be the type normally encountered in photolithography and topography simulation. The number of intersecting facet pairs in these meshes is considerably smaller than the number of facets. If the algorithm is modified so that the intersection test for each triangle is performed only against spatially nearby facets, then far fewer operations are required to find all intersecting pairs of facets. With proper ordering of the facets and clever exploitation of spatial coherence, the number of operations required can be reduced to $O(N\log N)$. This $O(N\log N)$ limit is expected to be the theoretical lower limit on the number of operations, because there is a necessary O(N) step of reading the input, and because an intermediate data structure requiring O(logN) time for insertion and deletion of a single object is mandatory. The data structure that has been chosen to represent spatial coherence is the octtree.

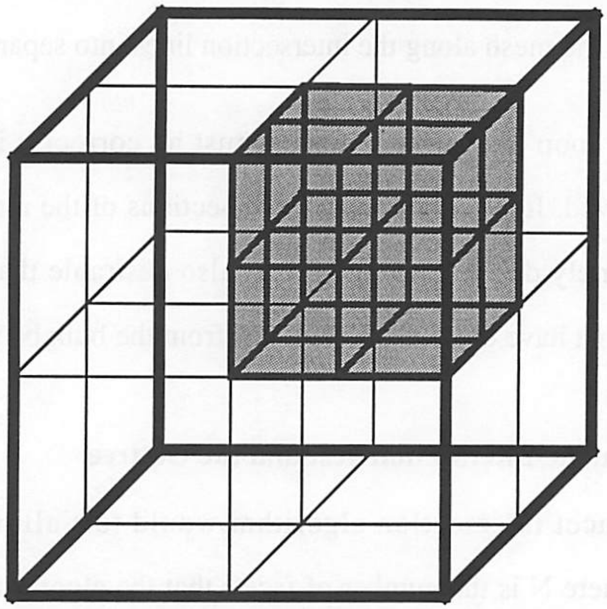


Figure 7) Octtree Spatial Division with Division of One Subnode Highlighted

5.3.1 What is an Octtree?

To perform a spatial sort on the triangles, an octtree is employed [9]. The octtree may be considered as a three-dimensional equivalent of a binary tree (Figure 7). The root node of the octtree represents the entire simulation space. This node has eight children, or subnodes. The subnodes of the root node represent subspaces, also called octants. Typically the octants are formed by slicing the complete domain by three mutually intersecting perpendicular planes. Although their dimensions are halved, these subspaces are otherwise geometrically similar to the full simulation space. The subspaces are arranged in a 2x2x2 formation. Each subnode in turn has 8 octants that represent further divisions of the simulation region. This division continues recursively to a preset depth, or until further subdivision is deemed unnecessary. Subnodes that have no children are called leaves.

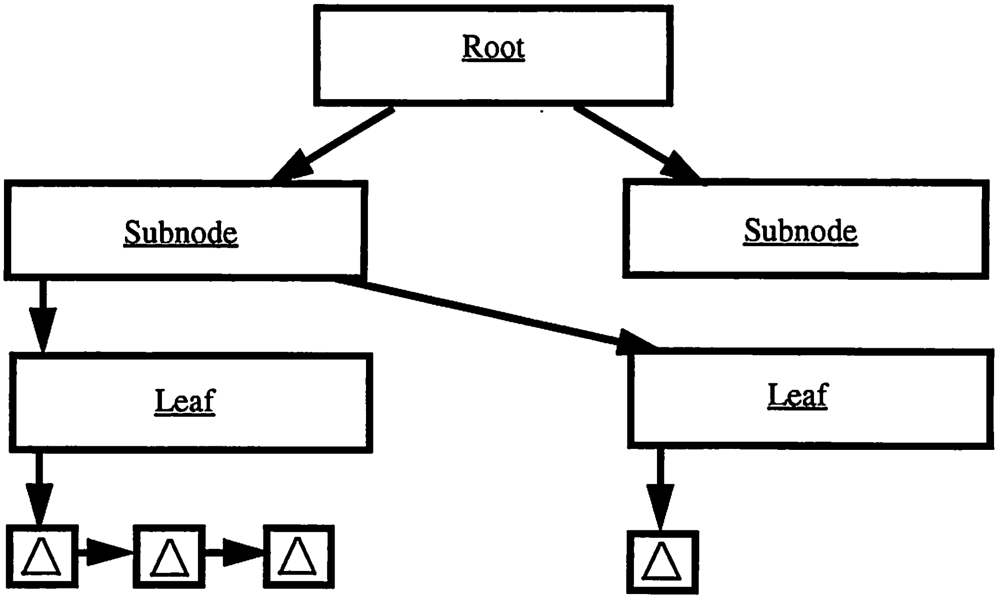


Figure 8) Storage of Triangle Locations in the Octtree Data Structure

5.3.2 Sorting Triangles with an Octtree

Determination of pairs of intersecting triangles is made efficient by having all triangles located in this octtree.(Figure 8) Starting with an initially empty octtree, each triangle is inserted into it in turn. Before the triangle is inserted, it is converted to a polygonal representation. This representation is an ordered list of points that define a polygon on a plane in three-dimensional space. The plane is the plane of the triangle. The polygon represents the intersection of the set of points contained in the triangle with the region of space associated with the node of the octtree. The triangle is first inserted into the root node, and, using the polygon, the subnodes that the triangle intersects with are found. Subnode intersection is performed by dividing the polygon into smaller polygons through plane divisions (Figure 9). This generates residue polygons that represent the intersection of the triangle with the regions of space associated with the subnodes. The division into eight polygons is performed in three steps. The original polygon is divided into two parts by one of the three bisecting

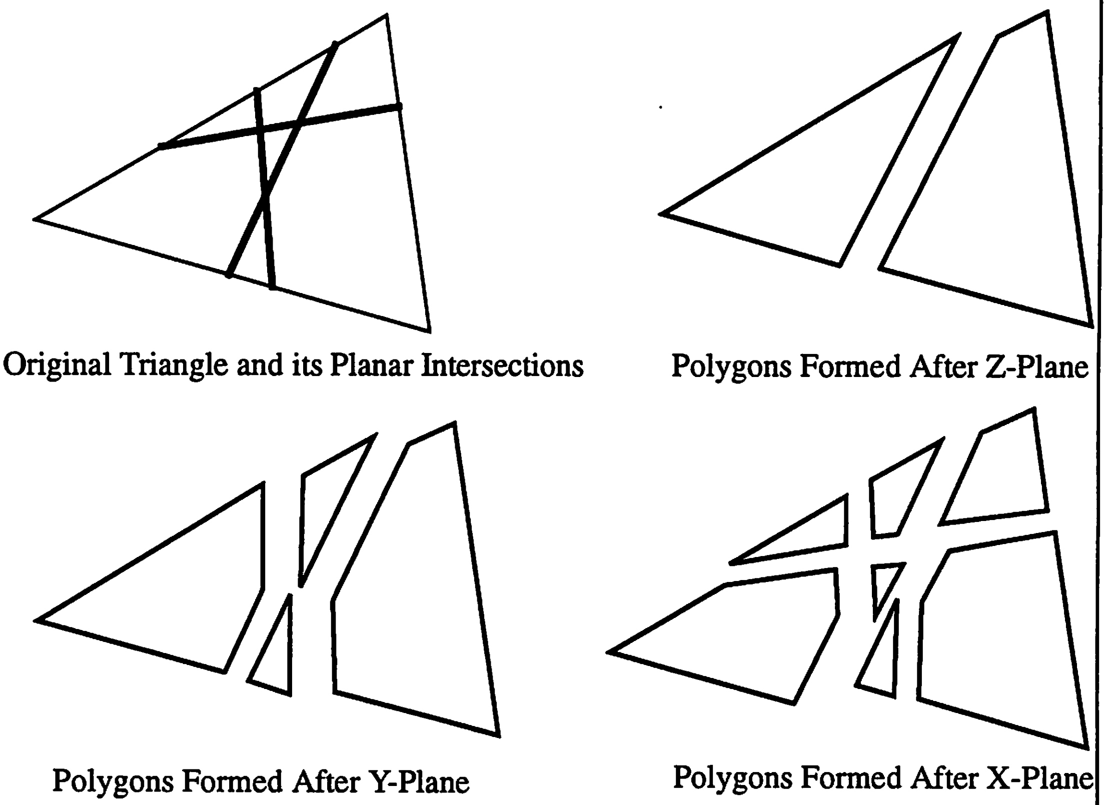

Original Triangle and its Planar Intersections
Polygons Formed After Z-Plane
Polygons Formed After Y-Plane
Polygons Formed After X-Plane

Figure 9) Subdivision of Triangle into Polygons During Triangle Insertion

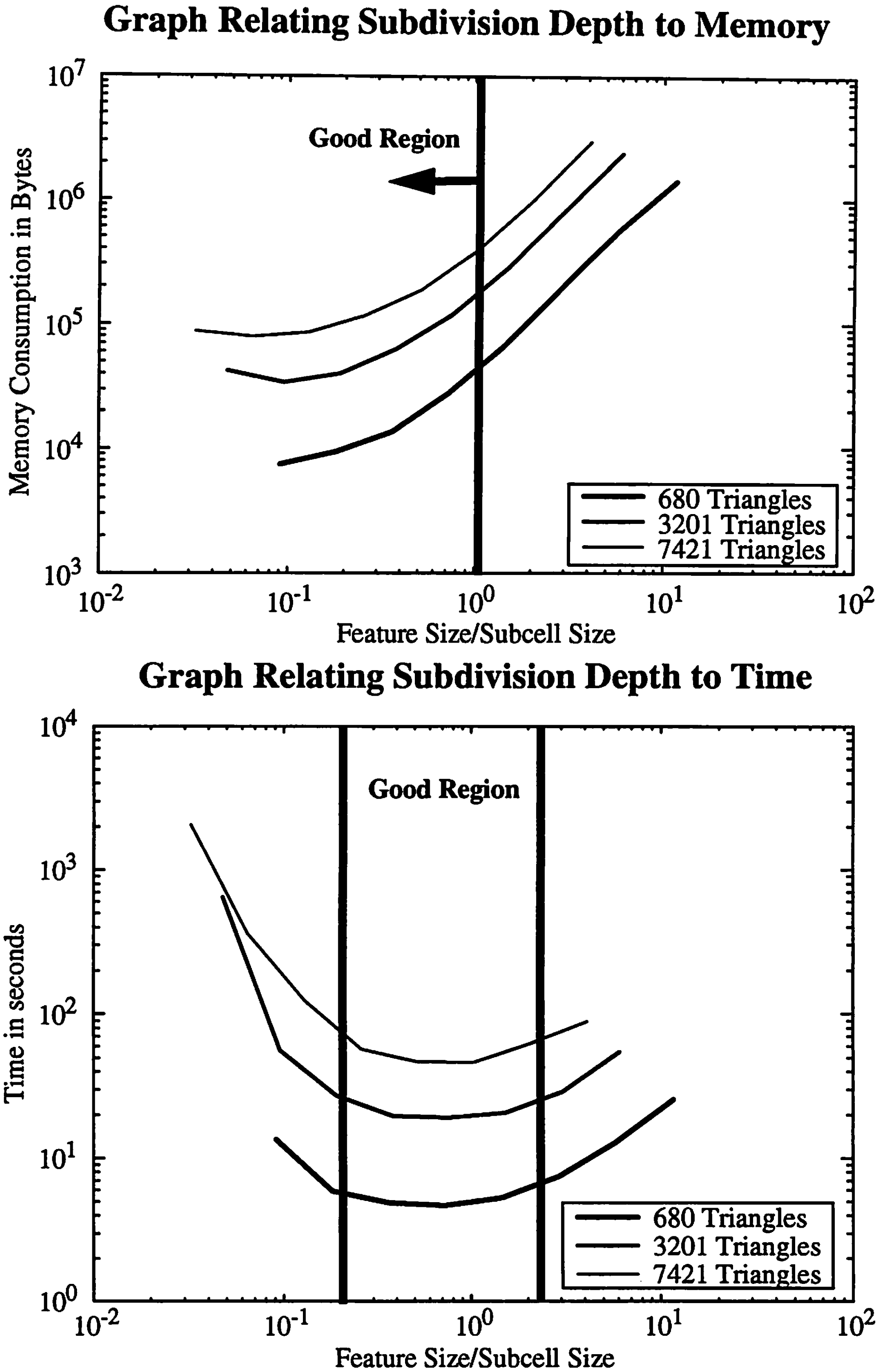
coordinate planes. These two polygons are divided by the second plane so that four polygons are formed, and finally the third plane is employed to form eight polygons. If a polygon formed for a subnode is non-existent, then the triangle does not intersect it. If a subnode has a defined residue polygon, the intersection test can be applied recursively using the new polygon associated with the subnode as the input to recursion. Recursion halts at a predefined depth.

Although the insertion test for a triangle can be applied recursively at any subnode, it is not always desirable to do so. Often, to save memory, recursion down to the preset maximum depth is not performed. In the octtree, if a subnode contains only one triangle, recursive subdivision is not performed. If another triangle is found to intersect the subnode, then recursive division of the subnode is performed, unless the subnode is at a specified maximum depth in the octtree. By employing selectively recursive subdivision, the space that would be consumed if the triangle was inserted into all of the possible subnodes is saved. Experimental tests have demonstrated that the best size for a subnode is about 3 times the average segment length of a triangle. It is recommended that there be 10-20 triangles in the subnode before recursion on the subnode is performed. This is not currently implemented, since the maximum depth criterion is sufficient to optimize CPU time and memory consumption for non-adaptive surface advancement meshes. Should adaptive meshes with widely varying triangle sizes be implemented, then recursion based on a specific number of triangles in each subnode would be practical.

When the limit of recursion is reached, a linked list is formed. This list is the union of all the lists of triangles that intersect that leaf. A new list of triangles is formed, which is the union of all the lists of triangles in the leaves that contain the inserted triangle. An explicit intersection test is performed between the inserted triangle and the candidates to find the actual intersections.

The time that each triangle requires to be inserted into the octtree is a function of the number of intersection tests that must be performed with subnodes of the octtree, and the number of triangle-triangle intersection tests that must be performed. The number of triangle-triangle intersection tests is the number of leaves that the triangle intersects times the average occupancy of the leaves. As the height of the tree increases, the number of leaves of the octtree that the triangle may intersect also increases, since each leaf represents a smaller amount of the simulation region. If the height is increased too far, then the triangle will occupy very many leaves. This is undesirable, since increased accuracy in localizing the triangle, which may ruling out comparison candidates, does not make up for the extra time and memory consumed in forming such a small subdivision. If the tree is too shallow, then the larger leaves may contain too many comparison candidates. The extreme case, where the height of the tree is one, is equivalent to the old method. Therefore, to utilize the octtree to its fullest potential, a method of determining the optimal level of recursion must be found. A sensible trade-off is to force the octtree to stop performing recursion when the leaves of the octtree are similar in size to the triangles. Because the triangles are constrained in their dimension by the ideal segment length, this size is easy to determine. Under this condition, the number of leaves that a triangle may intersect has an average constant value, and the number of leaves in the tree in total is O(N) where N is the number of triangles. The maximal height of this octtree is, therefore, O(logN). The total time for triangle insertion is O(NlogN). Since the time of an algorithm is also related to the size of its output, the number of triangle intersection pairs is also a contributing factor, so the real time required is $O(N\log N + N_i)$ where N_i is the number of intersection pairs, although the latter is never expected to dominate, since most triangles do not take part in any intersection. Likewise, since the number of leaves in the tree is O(N), the memory that must be consumed to form the octtree is expected to be O(N).

Experimental results from an implementation of an octtree confirm the stated expectations (Figure 10). These results were obtained on a DECstation 3000 and are taken from previous work in [15]. The shape of the test meshes that these results were derived from is shown in Figure 11. The three grid sizes employed were 11x11, 21x21 and 31x31. Each yielded triangulations containing 680, 3201 and 7421 triangles respectively. The top chart in Figure 10 shows the memory consumed as the octtree was set at differing levels of recursion. The bottom chart shows the time necessary for each run. The level of recursion was varied, and is expressed as the ratio between ideal segment length and the length of the side of the smallest leaf in the octtree. The good



Graph Relating Subdivision Depth to Memory

Graph Relating Subdivision Depth to Time

Figure 10) Graphs Representing Best Octtree Operation Regions

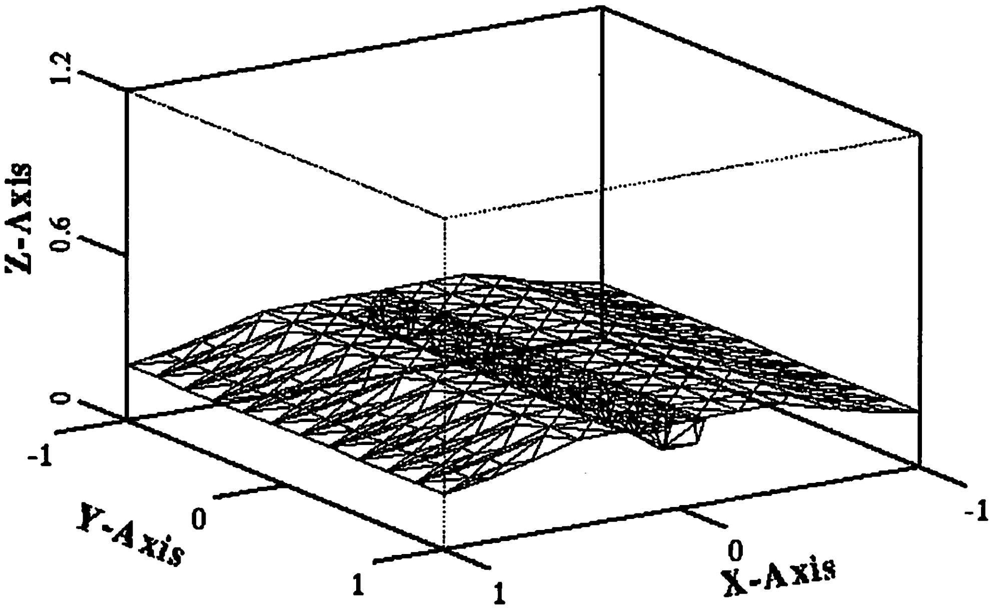


Figure 11) Deloop Timing Test Figure

regions were defined by examining the chart for ratios of feature size to subcell size that yield both small memory consumption and fast execution. It is the authors opinion that a smallest leaf to feature size ratio of 3 is the best ratio.

5.3.3 Application of Spatial Subdivision to other Surface Advancement Issues

Algorithms necessary for modeling other processes may also require spatial subdivision. Processes such as plasma etching and ion-milling often are described by etch models that incorporate visibility and reflection terms. The etch rate depends on parameters like the visible region of a source from each point on the surface, or the amount of flux incident on the surface, both from the source and reflected from other surface locations [3][8]. These terms are globally dependent. Pieces of the surface can influence far removed locations. Because the spatial location of a piece of surface relative to another piece is important, mere understanding of connectivity of the mesh and of local properties at the advancing surface are insufficient. A data structure that allows the surface to be organized based on its spatial relationships is necessary.

In the specific case of plasma etching, it is generally assumed that the mean-free path of a particle is long in comparison to the dimensions of the local topography. This means that the flux of particles during simulation can be approximated as straight lines that travel uninterrupted through non-bulk regions. Therefore, an important part of plasma etching computations is the test to see if a given line segment intersects the surface at any point along the segment. This test is made significantly easier and faster through the use of an octtree, or other method of spatial organization, and may easily be implemented in the same manner as the triangle intersection test.[10] Because the only preprocessing that is required for the line intersection is the insertion of the surface into the octtree, the same information generated during the triangle-triangle intersection tests for loop removal may also be employed in plasma etching computation. Therefore, the octtree is not only a important tool for loop removal, but also a valuable auxiliary data structure that can be usefully employed in all methods of surface based simulation. (Figure 12)

5.3.4 Application of the Octtree to Cell Algorithms

The octtree can also be used for the representation of cells in a cell etching program, although this has not been implemented. Each level of the octtree may contain a binary flag or other information, such as a time stamp. If this flag is used to mark the type of information that cell algorithms employ, then the cell algorithm can be implemented using an octtree. While there is slightly more computational overhead in an octtree implementation than in a grid implementation, this is more than offset by the ability to group large numbers of cells into one octtree node. One undivided octtree node can represent a large volume of cells which all may represent either air or resist.

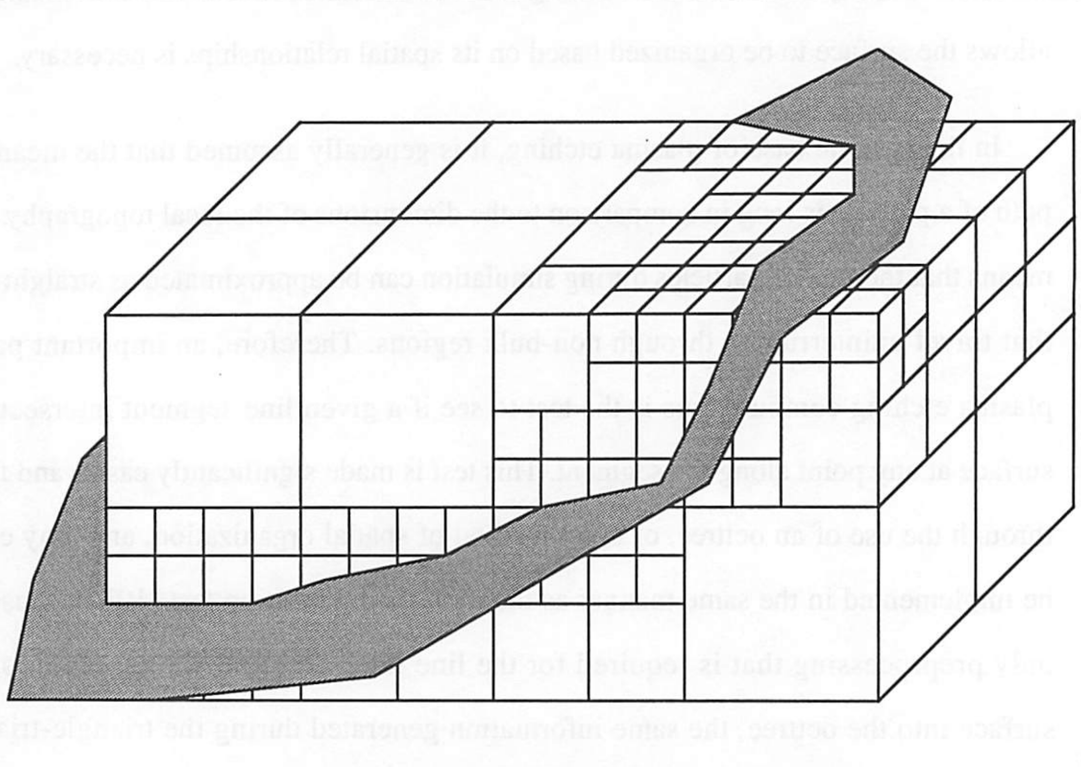


Figure 12) Octtree Subdivision Near a Surface

This allows many cells to be represented by much less memory. The memory consumption without the use of an octtree is $O(N^3)$ where N is the inverse of the size of the cells. With an octtree, the memory consumption becomes $O(N^2 \log N)$ for N as defined above. The memory consumption of the octtree is also proportional to the surface area of the photoresist. For surfaces in general, the memory consumption using an octtree is $O(N^d \log N)$ where d is the fractal dimension of the surface. For most simulated surfaces, however, d is equal to 2.

5.4 Determining the Intersection Lines and Surface Subdivision

5.4.1 Properties of the Triangle Domain Space

The domain space of a mesh is a connected topological space whose connectivity is determined by the connectivity of the points on the triangles, and the connectivity of the triangles with respect to each other. During simulation, the advancement of the

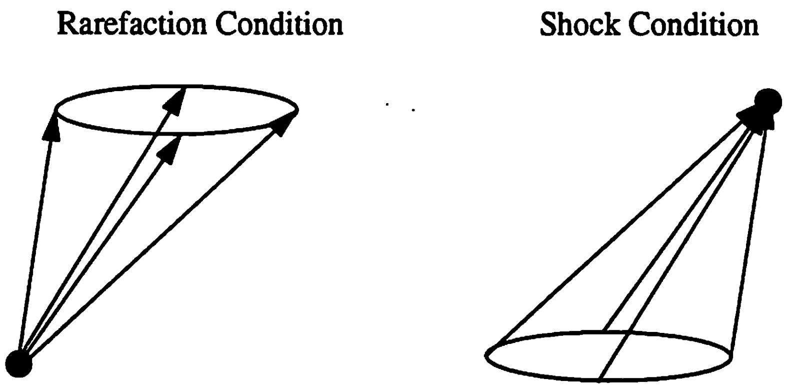


Figure 13) Rarefaction and Shock Point Mappings

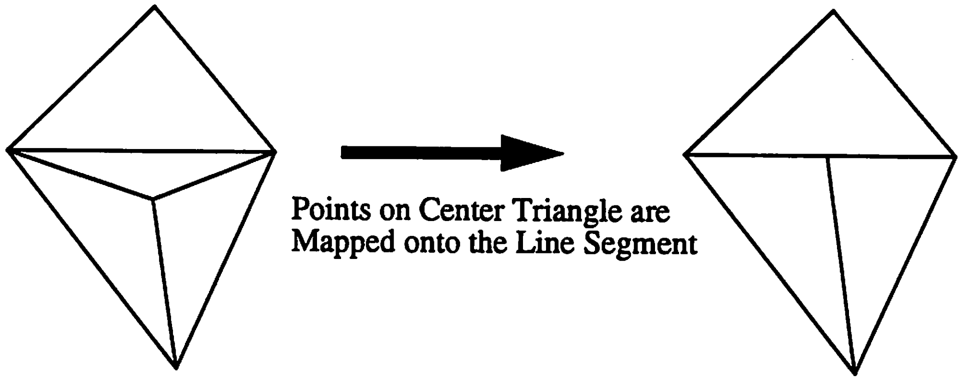


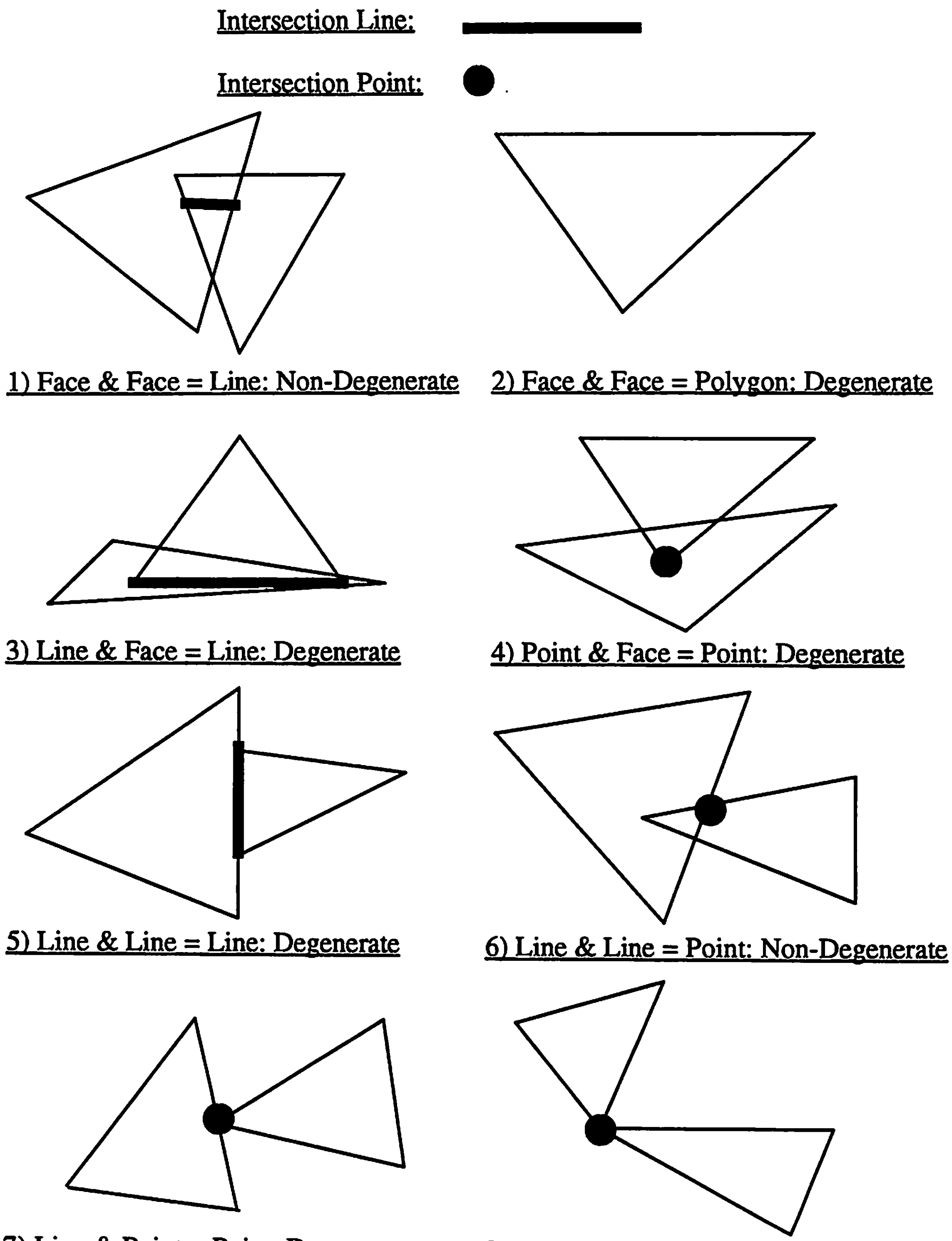
Figure 14) Triangle Undergoing Shock Style Mapping During Removal

surface, and all basic mesh refinement and coarsening operations, this connectivity of the surface is preserved. It has been demonstrated previously that in the rarefaction case, a single point maps to a simply connected set of points, while in the shock condition, a simply connected set of points maps to a single point(Figure 13). In mesh maintenance operations, triangles are deformed in a similar manner (Figure 14). Created triangles are formed so that the points in the original triangle being divided are mapped to the newly formed triangles. Triangles that are removed also have their points mapped to the mesh in a continuous manner. This mapping is typically from a triangle to a line or a specific point. All of these point to point mappings are considered topologically invariant. These mappings preserve the connectivity of the surface. The triangular domain space can be understood, therefore, as a topological space [11] that is described in terms of its connectivity, this connectivity is preserved in all surface advancement and mesh maintenance operations, except for deloop.

5.4.2 Getting Good Intersections

To compute the intersection line, the individual line segments that arise from each pair of intersecting triangles must first be identified. The intersection line segment is the region that is in common between two intersecting triangles. Other than the endpoints, the points on the line defined by each pair are unique. To compute the intersection lines easily, however, the fact that the mesh points are a little 'fuzzy' is employed. (One approach to defensive and efficient programming is to make good use of fuzziness in the application, but the programmer must be very careful.)

Sometimes a triangle-triangle pair will have a coincidence between the vertex of one triangle and the plane of another triangle. A coincidence is assumed when the vertex is within a distance of 10^{-8} times the size of a triangle. Since a triangle in a pair may have a node that is coincident with the plane of another triangle, seven new intersection types arise that are distinct from the basic Face & Face = Line Segment type, as shown in example 1) of Figure 15. To simplify the code and to reduce the number of cases that must be analyzed and handled, undesirable coincidences are removed by small randomized movements of the offending vertices. The undesirable and desirable cases are shown in Figure 15. These desirable cases are 1), 6) and 8) when the two nodes are the same. 6) must be allowed, since this case represents the intersection of the surfaces at the border of the simulation region, and 8) represents the end point of a banana node, which arises from structures similar to Figure 17. Node movement causes the undesirable intersections to be transformed into the desired



1) Face & Face = Line: Non-Degenerate

2) Face & Face = Polygon: Degenerate

3) Line & Face = Line: Degenerate

4) Point & Face = Point: Degenerate

5) Line & Line = Line: Degenerate

6) Line & Line = Point: Non-Degenerate

7) Line & Point = Point: Degenerate

8) Point & Point = Point: Degenerate (Only if not same point)

Figure 15) Types of Pairs of Intersecting Triangles with Degenerate/ Non-Degenerate Status Marked.

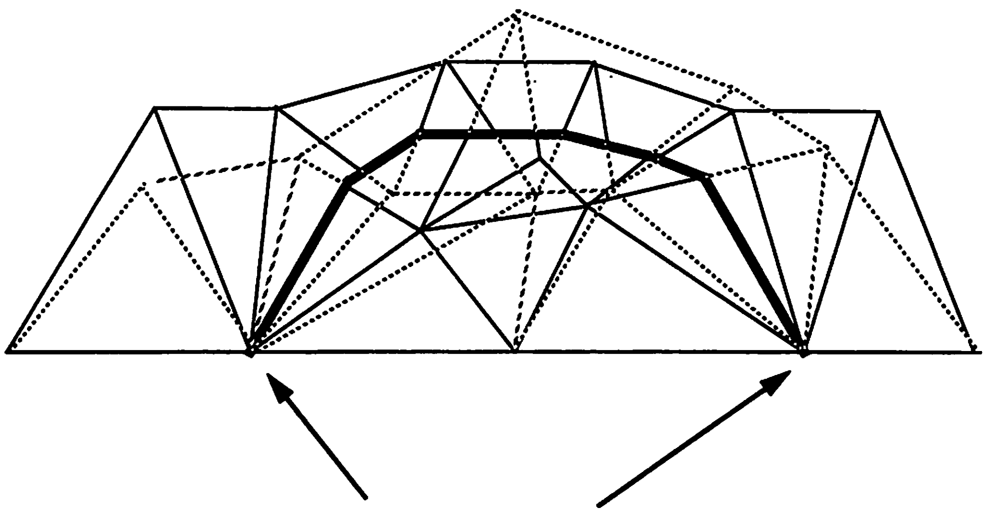


Figure 16) The Banana Loop

cases. The only type of node-plane coincidence that is maintained is the allowable case in 8). To move the nodes, an iterative procedure is employed, where each vertex is moved normal to the plane of the intersecting triangle. The motion is a random distance that ranges between 10^{-8} or 10^{-7} of the size of a triangle. The direction of motion is also random. The additional time that is required by this preprocessing step is $O(N_n \log N)$, where N_n is the number of triangles that contain nodes to be moved. The operation may be performed at this rate by moving the node, removing all triangles that neighbor the node from the octtree, replacing the triangles in their new positions, and recomputing the intersections. Presently when any nodes are moved, the entire set of intersections is recomputed. Although it is only necessary to remove and replace the triangles adjacent to a pushed node, a complete recomputation is performed to simplify programming complexity. Since coincidences are expected to occur infrequently once the symmetry in the initial conditions is broken, this term is omitted from the complexity analysis. Experiment has confirmed this assumption.

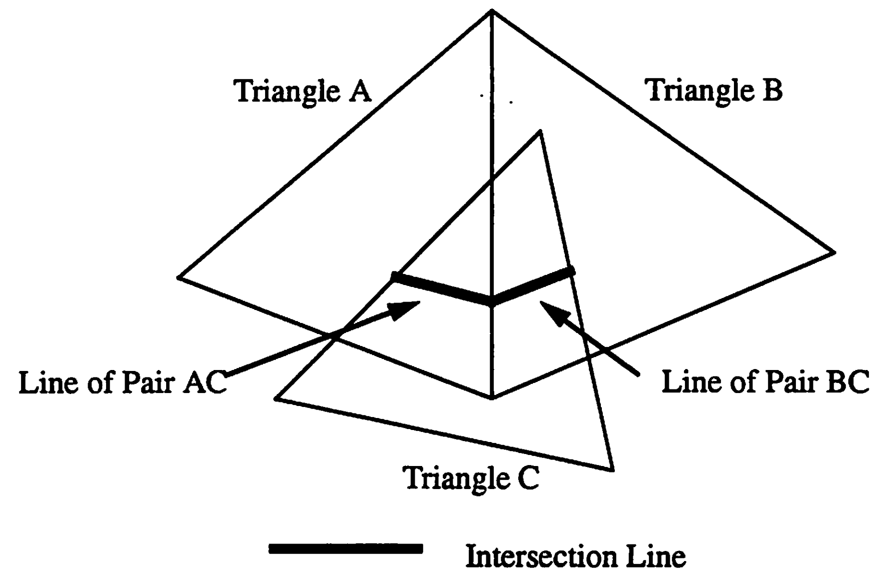
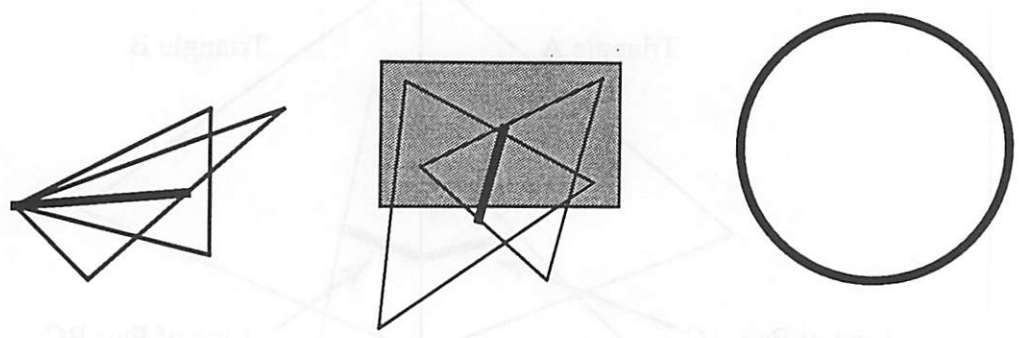


Figure 17) Determining the Next Intersection Pair on an Intersection Line

After the first time step, on examples that represent real structures, the octtree needs to be recomputed at most twice.

5.4.3 Tracking the Intersection Line

Because the intersection lines form the boundaries between parts of the mesh that are actual surface and the parts of the mesh that are loops, it is necessary to subdivide the intersecting triangles along the intersection line. This subdivision will allow an explicit representation of the separate mesh regions, thereby allowing delooping to occur. Since the triangle intersection pairs are generated in an arbitrary order, there is no inherent connectivity of intersection pairs implicit in the order. To properly handle coincidences of the endpoints of intersection line segments and maintain connectivity in the delooped surface, the endpoints of the intersection line segments must be properly matched. Figure 17 shows two intersection line segments that have the same endpoint. This point must be identified as a single point and both intersection line



1) Banana Loop

2) Border Intersection

3) Closed Curve

Figure 18) Termination Conditions for Intersection Lines

segments must explicitly refer to it when the mesh is reconnected after deloop. To perform this operation the connectivity of the mesh is employed. Given the intersection line segment of pair AC, the rightmost endpoints examined. This endpoint is caused by an edge of triangle A intersecting triangle C. To continue the intersection line, the edge of triangle A is examined to determine the other triangle connected to it. This triangle is triangle B. Because triangle C did not terminate on that segment, the next intersection line segment on the intersection line must be the one associated with the pair BC. The endpoint that was determined for AC is now fixed as the starting point of the segment for BC, and connectivity is established. At the end of each intersection line segment, this operation is performed unless one of the three tracking termination conditions is found.

The three termination conditions of the intersection line tracking method are shown in Figure 18. The places where the intersection line terminates in normal space are at the node of a banana loop, the intersection of the line with the simulation boundary and the closed curve where the termination condition is the same point as the starting condition. These three conditions are also the starting conditions for intersection line tracking. To perform intersection line tracking, a pair of intersecting triangles that represent one of the first two termination conditions is found. The line is tracked in the direction opposite of the banana node or boundary intersection via the method shown in Figure 17, until another termination condition is reached. All lines that start with a banana node= or on a boundary will end with one of these two types. Once all intersection lines of this type are tracked, there may still be some untracked intersection line segments of the closed type. An arbitrary triangle pair is chosen and tracked until the same pair is reached again. This establishes connectivity on one closed intersection curve. If there are any other pairs left unconsidered, the process repeats until all closed intersection curves are tracked. The time required for establishing this connectivity on the mesh is $O(N_i \log N_i)$ time where N_i is the number of pairs. Tracking the intersection lines reduces code complexity and enhances robustness, by matching endpoint coordinates for each intersection line segment and by providing a minimum of special cases to be considered. It insures that if some triangles are intersected more than once, either by the same or different intersection lines, the algorithm need only concern itself with a single intersection line at any time

5.4.4 Triangle Subdivision

Once the intersection line has been tracked and the connectivity established, new mesh nodes and edges are created explicitly in the SAMPLE-3D data structure using the coordinates established by the previous step. A new mesh edge is formed for each intersection line segment, and a new node is created at the end of each segment. Each new node that is created represents the intersection of the face of a triangle with a segment that borders two other triangles. Therefore, two new edges are created, between the new node and the nodes of the old edge. In this manner, a framework of edges and nodes is created that represents the intersection line. Once this has been

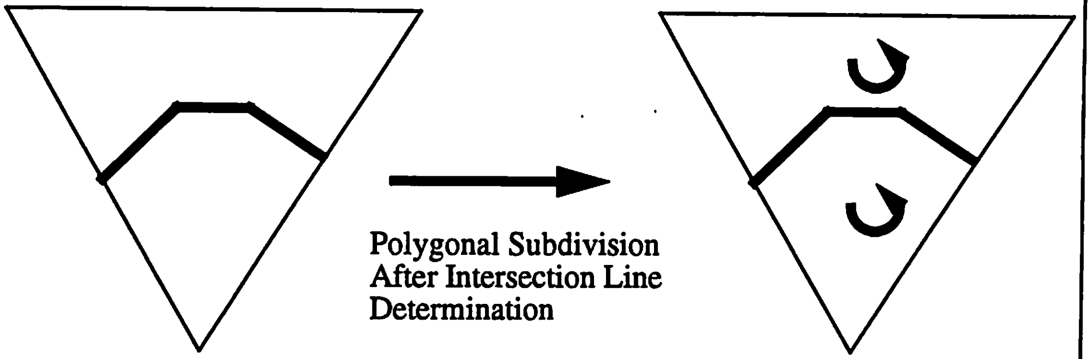


Figure 19) Polygonal Subdivision of Triangles

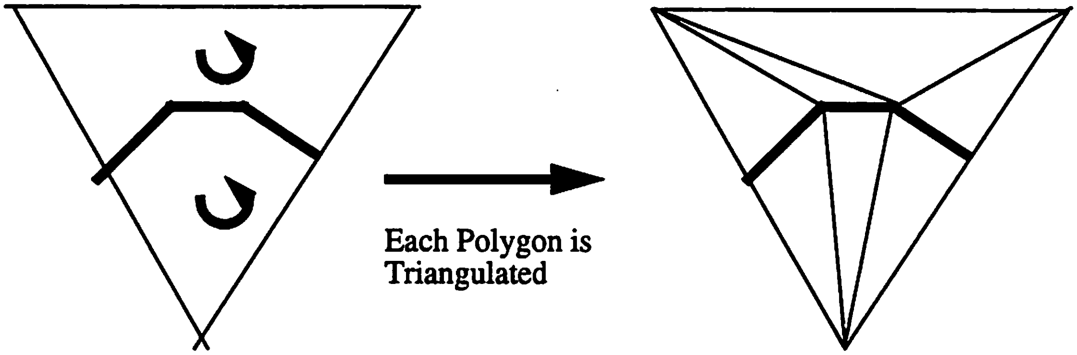


Figure 20) Triangular Division of Triangles

performed, each triangle has associated with it a sequence of intersection lines that traverse it. If the original border of the triangle is traversed in an ordered, non-repeating manner, which can be done in O(N) time, a set of polygons is derived that partition the surface of the triangle (Figure 19). Each of these polygons is then triangulated ((Figure 20) local & (Figure 21) global). This, theoretically, takes $O(N_l \log N_l)$ time [9][12], where N_l is the number of intersection line segments that were contained in that triangle. The old triangle is then removed from the mesh, and the new triangles are inserted.

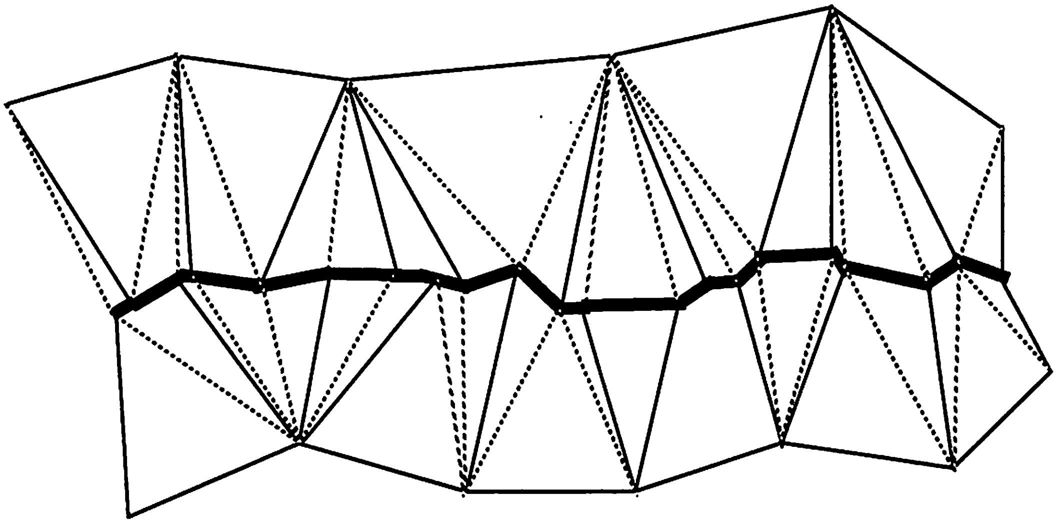


Figure 21) Subdivision of Triangles from (Figure 6)

Since each intersection pair contributes one line segment to each triangle, the total time for this algorithm is $O(N_i \log N_i)$. Three, four or n-way mutual triangle intersections may be handled by checking for intersections of intersection line segments. New nodes would be formed at the points where the segments cross, and the segments would be divided into smaller segments with the new nodes as endpoints. In some cases, such as when the N triangles in the mesh are mutually intersecting, the time complexity would be $O(N^2)$. For most geometries encountered in practical lithography tasks, these higher order intersections are so rare that they can be neglected in the complexity analysis.

Once all of the polygons that were formed by the intersection line have been triangulated, the intersection line now lies along the edges of the new triangles (Figure 20) & (Figure 21). Each intersection line segment now has four triangles connected to it, two triangles in each of two planes.

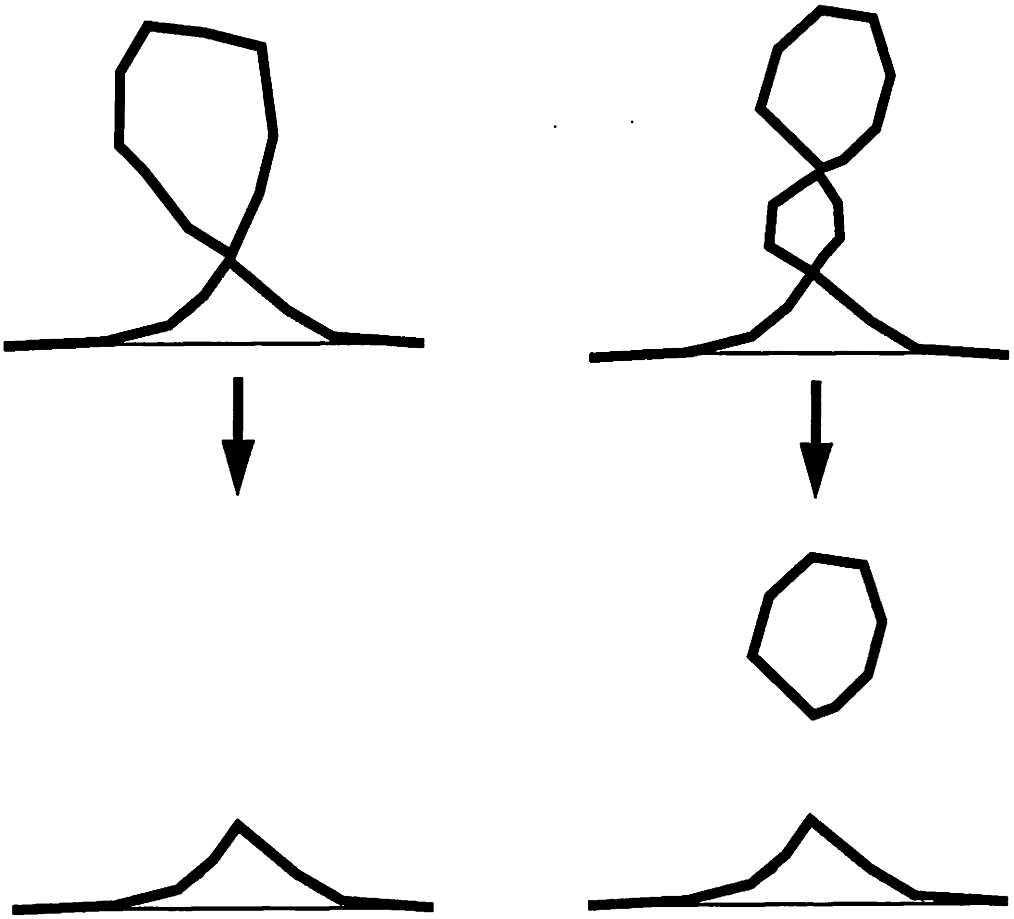
Another method was previously tried for triangle subdivision. This method subdivided each triangle into smaller triangles simultaneously with intersection line traversal. As the intersection line was observed to leave a triangle, the triangle would be divided into smaller triangles that had the intersection line along their edges. The triangles were reinserted into the octtree and rechecked for intersections with other triangles. The motivation for this method was that divisions of triangles could be simplified to a single line crossing from one side of a triangle to another. This would remove the difficulties that are inherent when many intersection lines cross a single triangle. Although this approach was implemented on some simple examples of triangle subdivision, it was found to be extremely ungainly when applied more complex examples, such as intersection lines that enter and leave the same side of a triangle. The code required was excessive in length and hard to maintain. In many cases, the need of this method to recompute intersections with extremely small triangles also led to significant numerical inaccuracy. The approach in the above paragraphs did away with many of these complications.

5.5 Loop Identification and Removal

5.5.1 Basic Loop Removal

Now that the mesh has been tessellated so that the intersection line only appears on mesh segments, it is still necessary to determine what parts of the mesh are surface and what are loops. Considering the four triangles meeting at each intersection line segment, it is clear that no two surface triangles or two loop triangles can both exist in the same local plane on opposite sides of the intersection line. Also, each of the intersection lines forms a closed loop in the domain space (the closure may occur outside the simulation boundary). Therefore the mesh is divided into two distinct groups of triangles by the intersection lines. The Jordan Curve Theorem [11][12] can be used to identify the triangles forming the loops. This theorem states that, given a closed curve in Euclidean space, an escape ray from a point in the region bounded by the curve will cross the curve an odd number of times. An escape ray from a point external to the curve will cross it an even number of times. The same situation holds one dimension higher, where the role of the curve is assumed by the surface mesh, and the Euclidean space is the simulation space, then any path on the surface can be considered to be part of an escape ray. A path along the mesh that connects any two triangles that crosses the intersection line an even number of times is therefore equivalent to saying that the two triangles are of the same type. A connecting path between two triangles that crosses an odd number of times means that the triangles are of a different type. Any path between two triangles will give the same results since the intersection curves are closed. Different paths will have the same parity of their number of crossings, although they may have a different number of absolute crossings. The mesh triangles can therefore be labeled in O(N) time. Each triangle is selected and labeled either "surface" or "loop". It is then put into a list of triangles that possibly have adjacent triangles that are not yet labeled. This triangle is then removed from the list, its adjacent triangles are labeled and these are placed into the list if they are not yet labeled. This process repeats itself until all triangles are labeled. The parts that are labeled as loop triangles are then removed from the mesh. After all triangles of type "loop" have been removed from the mesh, the data structure needs to be restored to its canonical form. Since at each intersection line segment, there are two "loop" triangles and two "surface" triangles that contain it as an edge, mesh integrity is restored by replacing the intersection line segment with a normal edge that connects the two "surface" triangles. A cross sectional view of this example on the two basic types of loops is shown (Figure 22).

In the preceding paragraph, the algorithm assumed that there exists reliable knowledge of the surface or loop condition for a single triangle. For topography modeling tasks that are primarily two-dimensional in nature, such as IC wafer



Simple Loop

Bridge/Pinchoff Loop

Figure 22) Cross Sectional View of Loop Removal

processing, it is appropriate to assume that corners of the simulation area, sufficiently far away from areas where loops might be generated, represent the actual surface. For more complicated three-dimensional tasks, such as pieces of silicon that are floating free in a wet etch process, the type of a starting triangle can be determined by counting the number of surface intersections on an escape ray leaving the triangle in the direction of its outward surface normal (Figure 23) If the escape ray crosses an equal number of surface pieces in the direction of their outward surface normals as in the direction of their inward surface normals, then the ray has emanated from a valid

Escape Ray Crosses Surface Once In Each Direction Giving Initial Condition of 0.

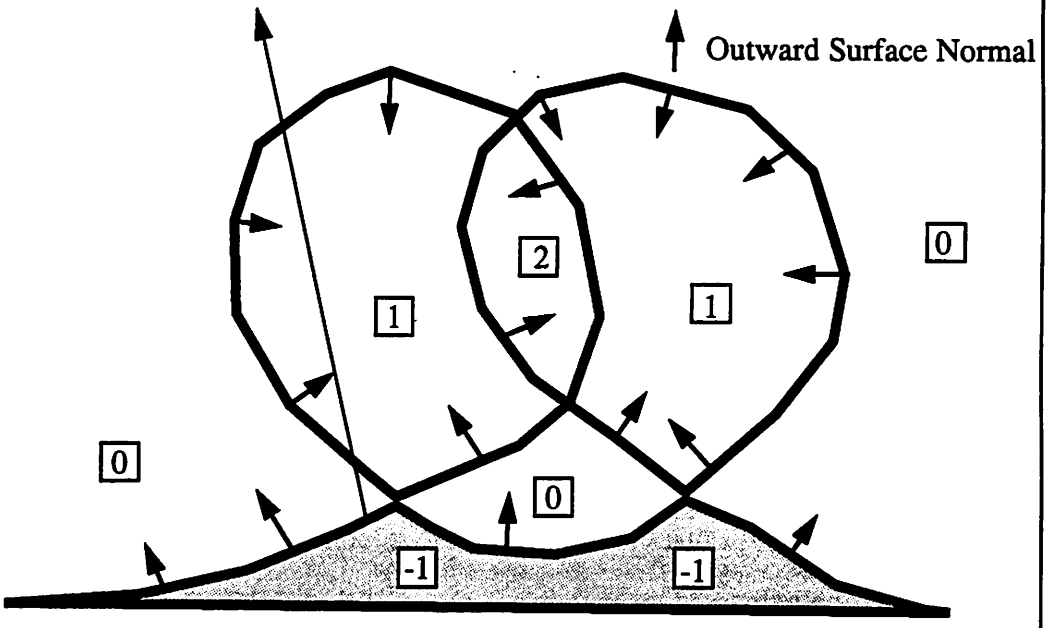


Figure 23) Escape Ray Surface Determination with Winding Numbers Marked

surface triangle. This triangle may now be used in the loop labeling algorithm as a valid starting point.

5.5.2 Identification of Detached Parts

Parts of the surface may become separated during the loop removal; they represent pieces of the bulk that have become detached during processing. These pieces can be detected with the same marking algorithm that marks triangles as "loop" or as "surface". If the labeling is initiated at a triangle that clearly belongs to the bulk of the device, all other "bulk" triangles will be connected to it through a path along other "surface" triangles. "Surface" triangles that cannot be reached in this manner are on pieces that have become detached from the surface. To identify the discrete pieces that have become detached, a list of all the "non-bulk surface" triangles is made. One of the triangles in the list is chosen, and all triangles that can be connected to it along "surface" triangles are located. These triangles represent one piece. If the "non-bulk surface" triangle list is non-empty, repeat the process to locate other discrete pieces until the list is empty.

Nested loops, i.e. areas where two loops intersect, have the ability to create a 'false' surface (Figure 23). These structures can be detected by a variant of the loop marking scheme that is dependent on orientation. This method is a three-dimensional variant on the two dimensional winding number. Instead of using a two-state value to distinguish between "loop" and "surface" regions, an integer is used. At the true surface, the integer flag is set to 0. The mesh is marked in the same manner as the original scheme with one exception. As the intersection segments are crossed, the plane that is being passed through is examined. If the marking path is proceeding in a direction opposite to that of the outward surface normal, then the counter is decremented by 1. If the marking path is proceeding in the direction of the surface normal, then the counter is incremented by 1. If no intersection line was crossed, no change is made to the counter. After labeling, all 0's are actual surface. Both the "bulk" and detached pieces that represent actual objects that have become removed from the bulk during processing will be labeled with 1's. Pieces that have been labeled with other numbers are "non-surface". This detection method is not necessary for most applications that SAMPLE-3D performs, however. Only the binary loop removal has been implemented in order to keep the code simple.

5.6 Resist Deloop

5.6.1 Why Another Deloop Algorithm?

As discussed before in the introduction to this chapter, ray advancement can be used to track the shock condition in photoresist dissolution simulation. It is desirable for a simulator to exploit this property to simplify the advancement of the mesh. Ray-

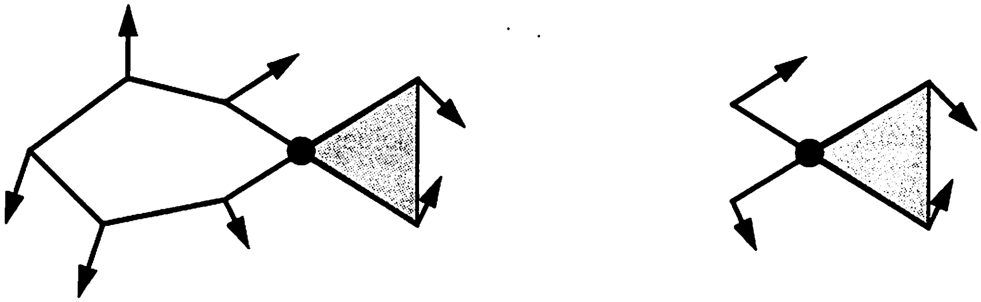


Figure 24) Need for Resist Deloop

trace advancement employs the fact that the surface of the resist can, at each time step, be defined as the outer envelope of all rays extended from the initial surface. The outer envelope of the rays are the rays that are on the border of the volume made of the rays and all points that have been traversed by them. During advancement some rays lose their outer envelope status by forming loops, as seen in Figure 24. These rays must be removed so that CPU resources are not consumed by advancing them. This operation is valid, since the rays that have become invalid do not affect the results of the simulation. The drawback of removing these rays is that the mesh will become disconnected and/or bounded at other places than the simulation boundary. Since the original need for deloop has not disappeared, and the extra requirement now exists for removing the trailing ends of the mesh around the shock conditions, a clear need exists for a new mesh operation related to deloop.

The first deloop method was implemented with the intention of removing the loops from the photoresist problem, but after it was implemented, it was not clear how to advance the surface after a deloop operation. The points of the mesh that represent the shock had been generated by the triangle splitting method at the intersection line.

Because the first deloop method results in the formation of a continuous surface, it is not clear how to interpolate new rays for the shock conditions. The previous implementation of photoresist dissolution in SAMPLE in two dimensions addressed this difficulty by breaking the surface into two distinct sections, as seen in Figure 24, and advancing each independently. The first deloop method, though it could easily be modified to not stitch the shock line together, could not remove loops from non-continuous surfaces. Therefore, although the first method could handle the initial deloop in photoresist dissolution with minor modifications, it could not handle the second. Other methods were considered to avoid the difficulty of a new deloop implementation, such as explicit tracking of the shock, but such methods are difficult to implement, since many shocks in photoresist development are saddle points. As evidenced by [13], explicit shock tracking is significantly more difficult to implement in 3-dimensions than the new loop removal method described below. It was also recognized that a method of explicit shock tracking would require a full treatment of the mesh maintenance methods in Chapter 6 to operate effectively.

5.6.2 Theoretical Basis for Resist Deloop

Surface advancement is considered to be a mapping of an initial surface in 3-D space to another surface in 3-D space. This mapping is dependent on time (i.e. for different simulation times, different mappings occur) When these mappings are performed successively for small time intervals, this represents the advancement of the surface over successive time steps. For photolithography simulation, only mappings that preserve the connectivity of the rays topologically are valid mappings. Of the set of all mapped rays, the outer envelope of rays, which represent the surface, exists as a set of connected subsets in the ray domain space. The outer envelope of rays is defined as the set of rays whose advanced endpoints have coordinates in space that no other

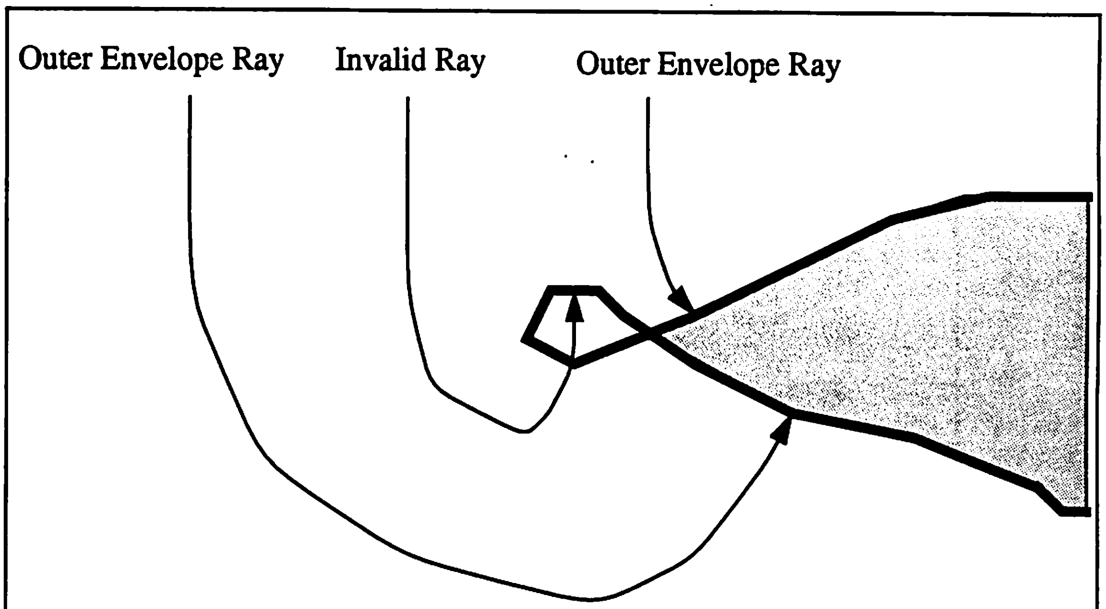


Figure 25) Outer Envelope and Occluded Rays

ray has passed through at any previous time (Figure 26). Invalid rays are those that have been overlapped by outer envelope rays and have formed loops. The subsets of rays that represent the outer envelope may be distinct subsets after sufficient advancement. For this reason it will be important that the subsets maintain some sense of connectivity through the domain space, although the explicit triangle-segment-node connectivity of the mesh maintained in the data structure could be split into two or more disconnected sets. An example of how this disconnection may occur is given in Figure 26, Figure 26 and Figure 27. Figure 26 is a delooped contact cut that is radially symmetric. It is shown with the looped surface included in Figure 26. The loop that has formed is also radially symmetric, therefore the removal of the looped mesh by the method shown in Figure 23 will result in two detached mesh parts. The domain space representation of the rays is shown in Figure 27. This figure is a representation of the connectivity of the rays. It can also be considered to represent the initial planar surface that the rays have propagated from, since the set of the endpoints of all rays form a topological space with the same connectivity as the initial condition no matter how

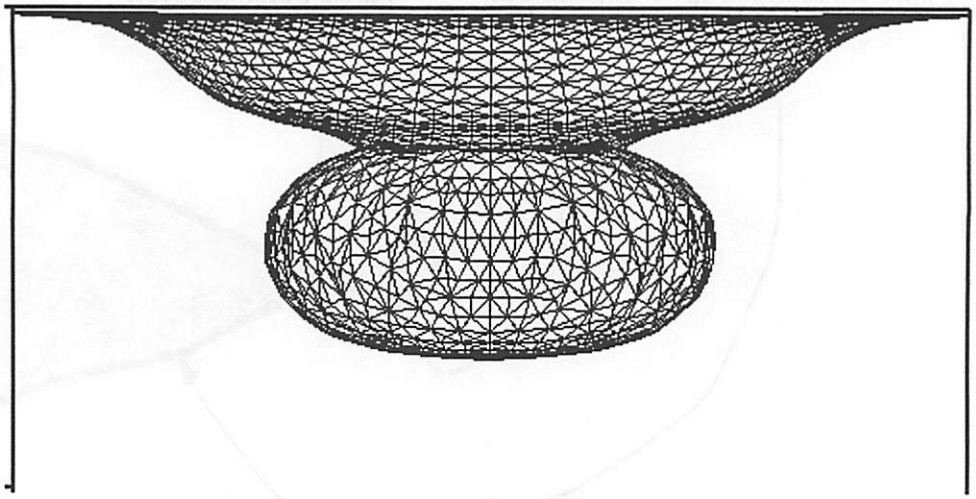


Figure 26) Outer Envelope of Rays for an Analytic Etch Function

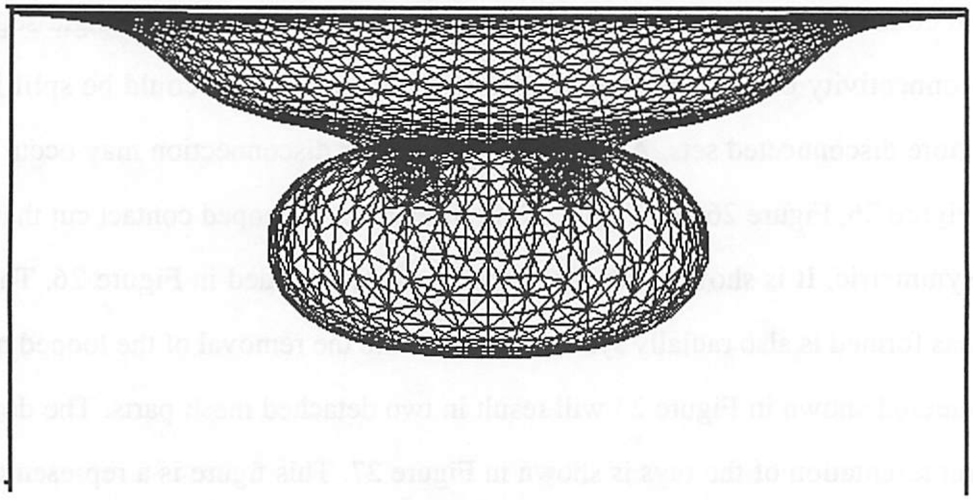


Figure 26) Full Set of Rays for an Analytic Etch Function

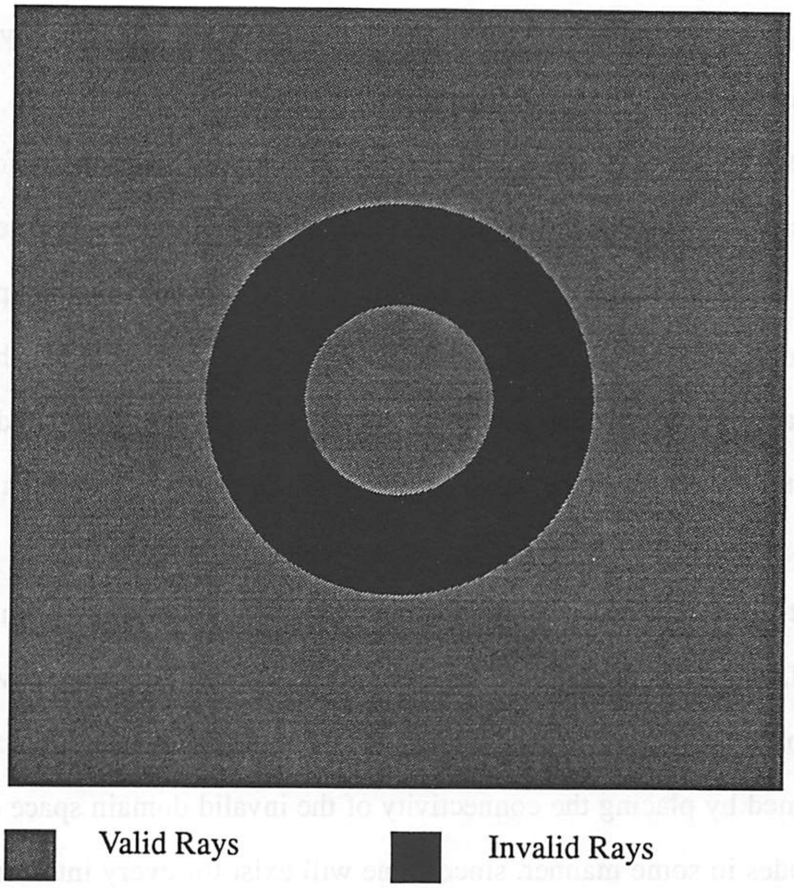


Figure 27) Representation of Ray Domain Space for Above Example

many advancements have occurred. The rays that are still valid after advancement are shown in gray and those that have become invalid are shown in black. The grey parts have become disconnected into two separate regions, representing the two separate pieces of mesh that now describe the outer envelope. Because the invalid rays are removed, a winding number method of finding the real surface over successive deloops becomes impractical. It is more appropriate to employ the connectivity properties of the domain space in identifying the proper resist surface. Methods for doing so are described below.

5.6.3 Properties of the Domain Space

The most important property of the domain space that is employed in ray-trace loop removal is that rays that are not on the outer envelope do not affect the advancement of the rays that are on the outer envelope. This property follows from the least time path argument in Chapter 3. Some of the nodes that represent invalid rays may have to be left in the mesh due to triangles and segments that are partially invalid and partially valid (i.e. the intersection line crosses them) so that the shock conditions may be tracked properly. Because there are few of these rays compared to the number on the outer envelope, however, they do not significantly affect the amount of CPU resources consumed. Therefore the advancement of invalid rays is fundamentally irrelevant to the accuracy of advancement, or to the amount of time or resources consumed, as long as the rays are removed often enough. The connectivity through the domain space must be maintained, however, even though rays are removed. This may be performed by placing the connectivity of the invalid domain space on the leftover invalid nodes in some manner, since some will exist for every intersection line after every deloop operation, due to the removal of node-triangle coincidences during the triangle intersection tests.

To understand how loop removal may be accomplished, three other self evident properties of the domain space are given. First, if a path can be drawn between two points which does not cross a shock condition (i.e. intersection line), and one of these points is known to be on the outer envelope, then it is clear that the other point and all points on the path also are on the outer envelope (Figure 28). Second, since the outer envelope forms a continuous surface, therefore, if there exists a shock condition, then there must be another part of the mesh that attaches at that shock condition that is also part of the outer envelope. This part will also be connected to the same intersection segment as the first part. Third, because a region in the domain space of rays that has

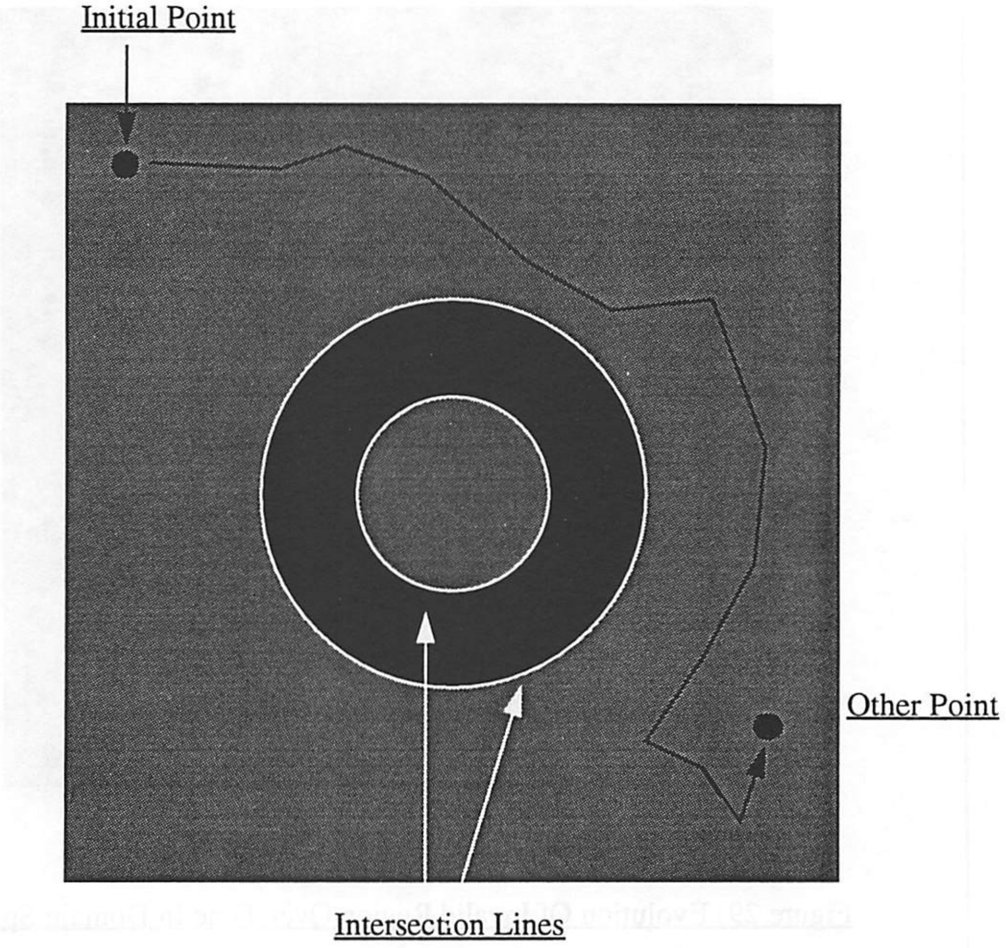


Figure 28) Valid and Invalid Region Boundaries are Defined By Intersection Lines

become invalid will not become valid again, the loop area may be understood as 'growing' outward to cover more of the domain. (Figure 29)

The intersection lines in the domain space form the following types of structures. First, the banana loop forms a single closed path in the domain space. This path encloses a region, the interior of that is invalid surface. A closed curve forms a ring made of two nested curves in the domain space. The area between the curves is invalid. Topological alterations, such as tunnels, form two separate rings. The area enclosed by each ring is invalid. (Figure 30) All other types of self intersection consist

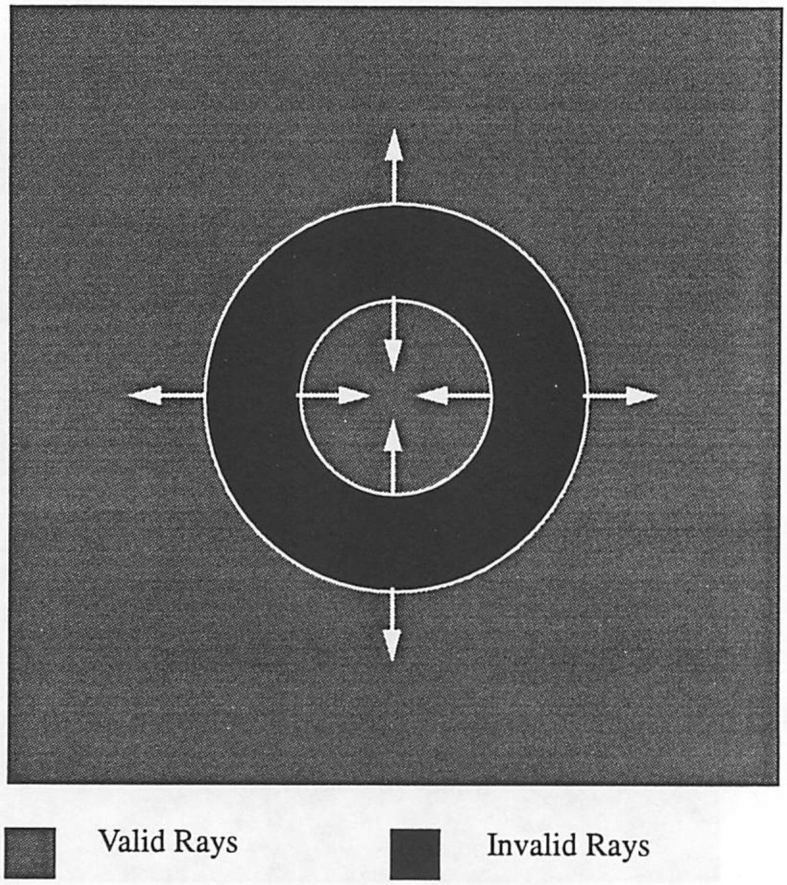


Figure 29) Evolution Of Invalid Region Over Time In Domain Space

of combinations of these three basic curves. When combinations of these curves occur, then the full set of invalid rays is the union of the sets described by each curve individually. As an example, consider a three plane intersection (Figure 31). Each pair of tunnel ends creates one pair of topological curves, giving 3 pairs of topological curves in total that are organized as shown in the bottom half of Figure 31. Intersection lines that are formed entirely in invalid regions can be ignored, since no new rays are made invalid by them. It is now possible to determine the status of any ray given the patterns that intersection lines form in the domain space, and a method of determining all relevant intersection lines from the mesh exists from in the previous deloop method. It is trivial to remove the extraneous parts of the mesh defined by banana

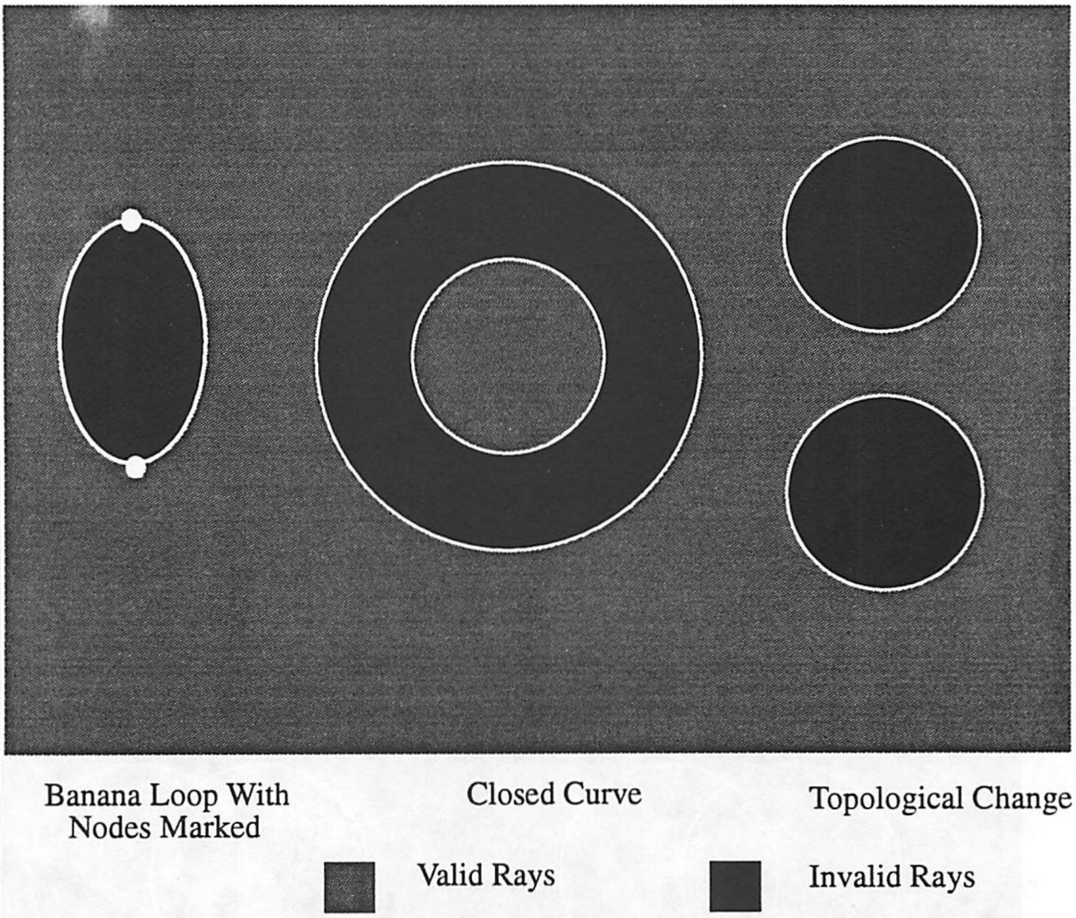
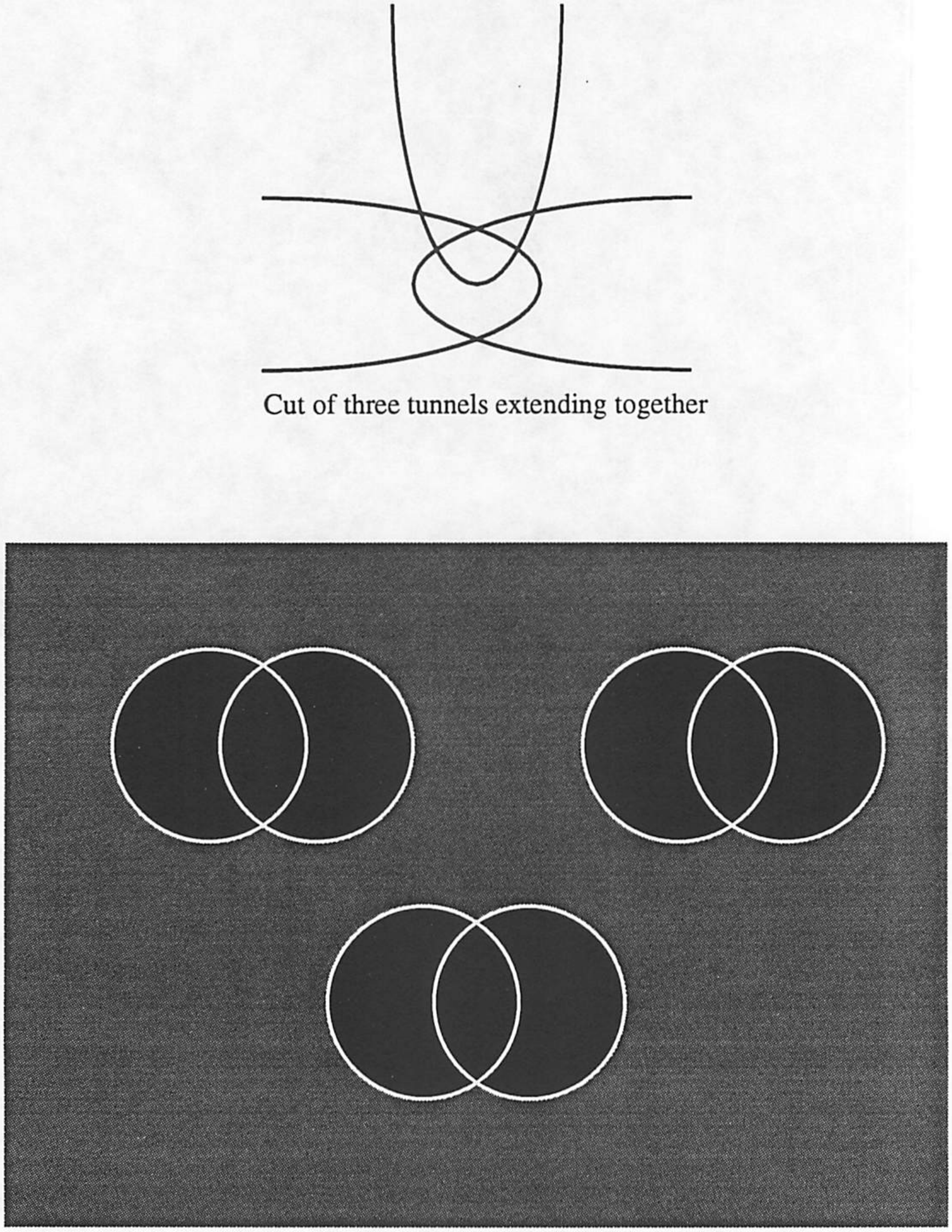


Figure 30) Basic Types of Invalid Regions in Ray Domain Space

loops and topological alterations, but the removal of loops that are formed by closed curves is not so simple. It is still necessary for the method to properly determine where the valid region is inside a closed curve, and for the appropriate connectivity to be maintained. Two methods are described below.

5.6.4 The Algorithm

To exploit this additional surface motion knowledge for photoresist development, it is first assumed that there is an area in the domain space that can be assumed to be actual surface. This is not as easy to determine explicitly as normal deloop, since the escape ray test can't be used. It is assumed that a certain part of the mesh is always real surface, or that there is some other property that can be exploited, such as most slowly



Cut of three tunnels extending together

Figure 31) Domain Space Representation of a Three Tunnel Intersection

Code Section	Level of Implementation
Intersection Location	Handles all cases due to node pushing Has problem with triangles with angles < 10^{-3} degrees
Intersection Line Tracking	Handles all cases, except 3-plane intersections actually on intersection line
Triangle Subdivision	Handles all cases, except 3-plane intersections actually on intersection line
Removal	Integer Labeling initiated at corner of simulation region

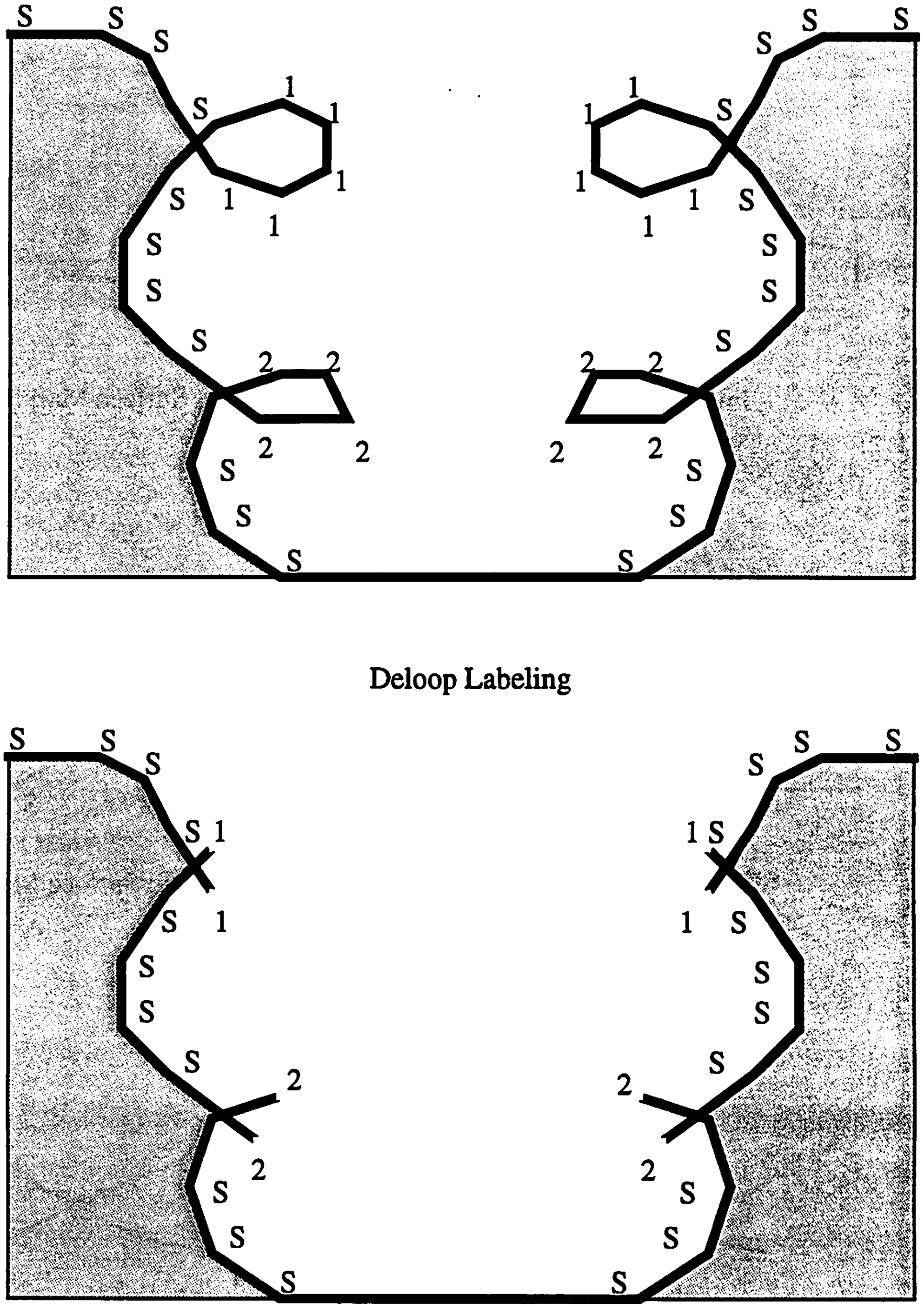
Figure 32) Resist Deloop Routines That Have Been Implemented

moving part or largest area. Once this particular location is determined, it is clear that all points in the domain space that are connected to it by a path that does not cross any intersection lines are also on the surface. The intersection lines that can be reached mark the boundary of a region of the mesh that is known to be part of the surface. This marking is represented in Figure 31 as the marking of the first set of 'S' at the top of the top picture, and in Figure 34 in the top right hand corner. These intersection lines that bound this region of the mesh that has been marked with 'S' are marked 'M'. Each intersection line that is marked 'M' is part of a shock front that is part of the real surface. To continue determining the outer envelope of rays, it is necessary to determine the mesh piece that connects to the opposite side of the shock. This other piece that connects to the shock will also be adjacent to the intersection line. Therefore every portion of the surface that connects to the shock will be marked 'M' even if it wasn't originally marked 'S' in the first pass.

The loop portions of the surface may now be labeled. The marked intersection lines now completely enclose an invalid region of the domain space, since each single intersection line forms the boundary of an invalid region. All triangles that are completely enclosed in these sections of the domain space must be removed. Therefore, since the boundary of this space has been completely marked by intersection lines labeled 'M', a labeling sweep is performed within the interior of the region bounded by 'M'. These nodes are marked with the integer '1'. An example of this marking is shown in the middle left part of Figure 34.

After this marking of the loop nodes has been completed, not every node in the mesh has necessarily been marked as a surface or loop node. An incomplete marking will occur if one the loops is a closed curve loop. To mark these unmarked nodes, the nodes that are adjacent to intersection lines marked 'M' are marked 'S' and are used as the initial nodes in a new 'S' marking. This is valid, since of the four pieces of surface next to an intersection line marked 'M', one was the initial piece of surface marked 'S' that led to the marking of 'M', and two others were marked with an integer to determine the attached loop. Therefore the unmarked surface is the proper continuation of the surface from the shock front. The continued surface marking is shown in the right middle portion of Figure 34. If intersection lines are reached by the continued marking, these lines are marked 'M' and the loop marking is initiated again. This time, however, the integer that is used to mark the loops is incremented by 1 from the last iteration. (Figure 31)

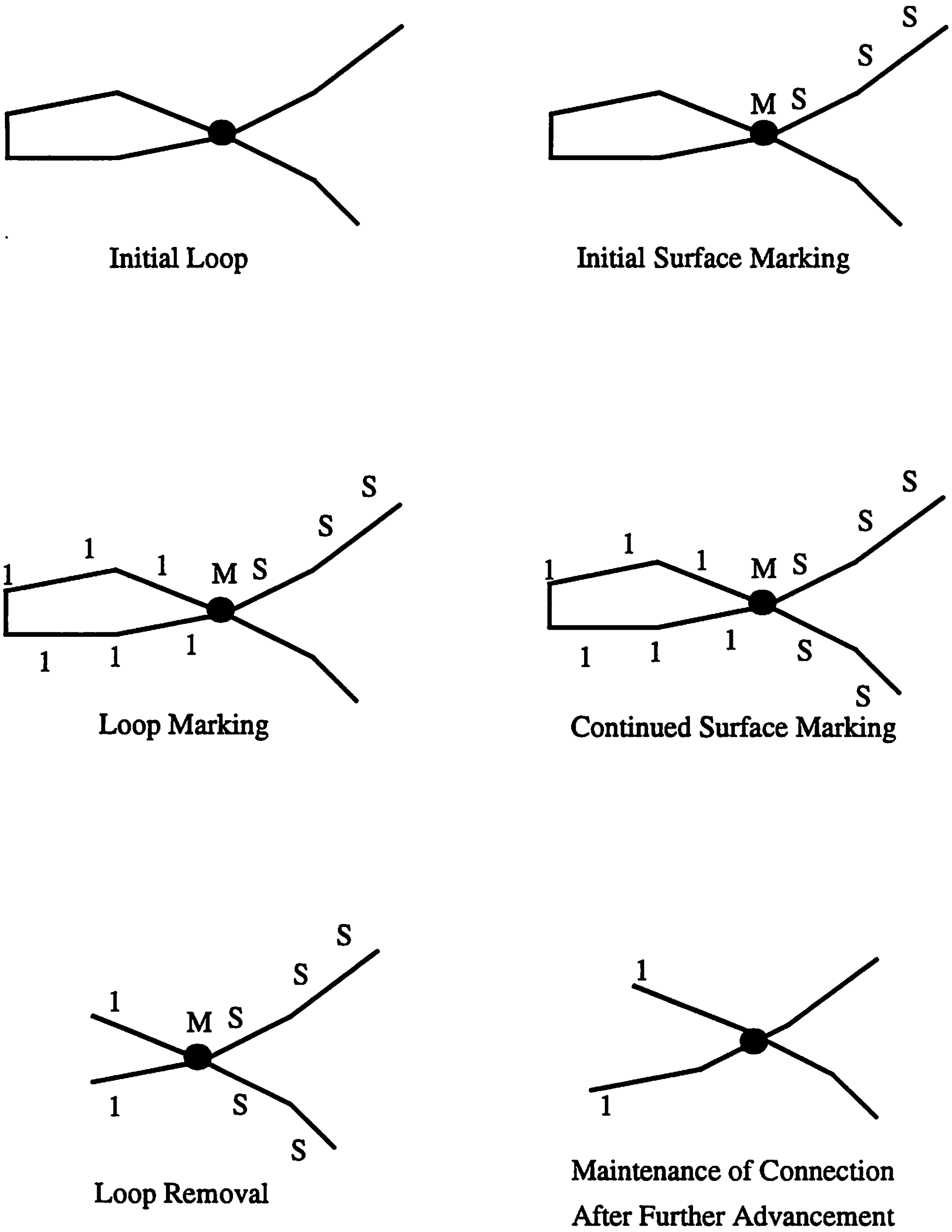
Once the regions of invalid mesh have been removed, it is possible that the mesh has become explicitly disconnected as in Figure 31. The integers that were marked on



Deloop Labeling

Deloop Removal

Figure 33) Resist Deloop Mesh Markings Before and After Removal



Initial Loop

Initial Surface Marking

Loop Marking

Continued Surface Marking

Loop Removal

Maintenance of Connection
After Further Advancement

Figure 34) Resist Deloop

the nodes at the previous timestep are used to maintain connectivity over further time steps. When sweeping out the invalid region during further deloops, whenever a node with an integer marking is located, all other integer nodes with that marking are located and considered to be connected to that node. This allows the tracking of invalid surface to continue across regions of the ray domain space where the invalid rays were removed from the simulation mesh. There is no possibility of confusing marking nodes on the outer envelope with these integers, since nodes that are marked with integers were invalid rays in previous time steps, and invalid rays do not regain their validity.

5.6.5 Normal Labeling / An Alternate Method

While integer marking has been found to be fast and efficient, a complete proof of its correctness is difficult. It is particularly well suited to standard photolithography analysis due to the nature of loop formation in these conditions, since the loops tend to occur along standing wave nulls, which are generally coplanar. Another method of loop identification, however, is clearly robust. This method analyzes the surface normals at the intersection line, and is called 'normal labeling'. It tracks the outer envelope of the rays, by using the ray directions, which are equivalent to the inner surface normals, to determine the proper pieces of the outer envelope at shock conditions. An example is shown in Figure 35. This removal method was implemented in a partial form in Kenny Toh's previous work on SAMPLE-3D [1].

The 'normal labeling' removal method proceeds as follows. Assuming that the initial surface guess is correct, then all other points that can be reached without crossing a shock are also surface. At each shock corner, the marking of the outer envelope must continue on the intersecting piece of surface as shown in Figure 35. The proper direction to take on the intersecting piece of surface to continue the outer

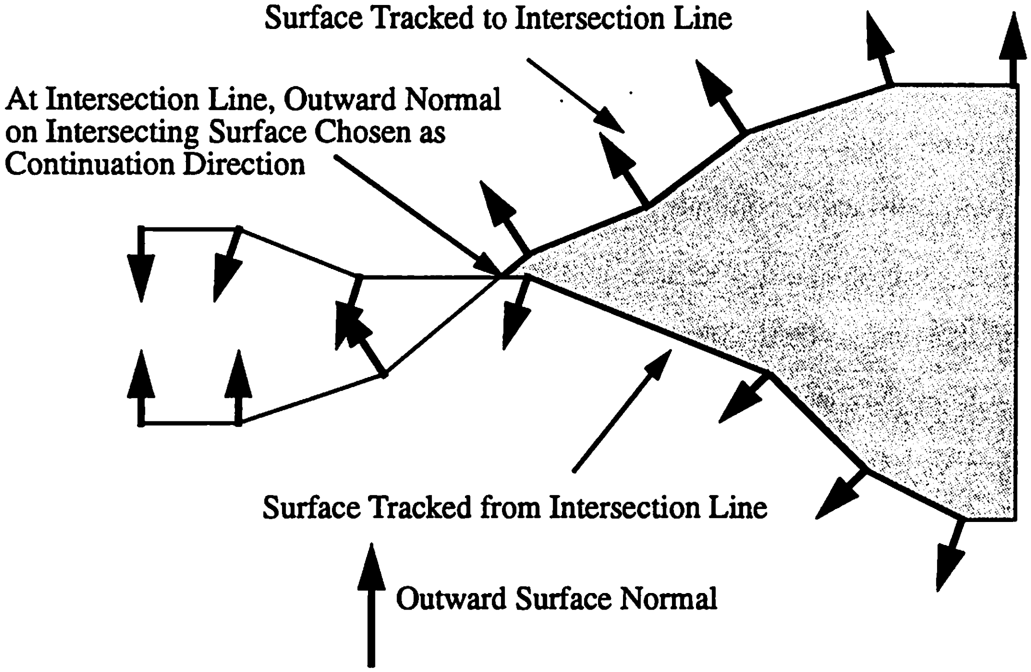

Surface Tracked to Intersection Line
At Intersection Line, Outward Normal on Intersecting Surface Chosen as Continuation Direction
Surface Tracked from Intersection Line
Outward Surface Normal

Figure 35) Second Loop Identification Method

envelope marking is in the direction of the inward surface normal of the original piece that met the intersection line. The process of marking continues until no more pieces of the outer envelope can be reached.

By the normal labeling method, each intersection line test chooses exactly one way of connecting the surfaces. This allows a consistent, non-intersecting and orientable surface to be constructed. Normal labeling has not been implemented, because integer labeling has been found to be satisfactory and probably equivalent to normal labeling. Normal labeling is worth mentioning because it equivalent to the concept of an 'outer envelope' and completely proving the validity of integer labeling has been extremely difficult. Integer labeling, however, has never been found to fail, so far, in any real or theoretical example.

5.6.6 Boundary Conditions

Two types of boundary conditions must be considered in the resist deloop routine. Both of these conditions can be considered to be reflective. For resist deloop, however, there is a difference. The first type of boundary condition is the condition on the sidewalls of the simulation region. The surface is considered to be symmetrically reflected across the sidewall planes. This means that a reflection of the same structure exists on the other side of the planes that bound the simulation region and are perpendicular to the initial condition plane and the substrate plane. This symmetry is shown in Figure 36 for a simple two-dimensional object that has Laplacian conditions imposed on a square boundary. In this case surfaces that have been reflected across the boundaries are mathematically equivalent to an infinitely repeating mesh that embodies the symmetry. Therefore, clipping the mesh at the boundary is completely satisfactory for loop removal purposes.

For the intersection of the surface mesh with the bottom of the simulation region, i.e. the interface of the resist with the substrate, the conditions for normal deloop and resist deloop must be handled separately. The substrate is assumed to be an infinitely unetchable material. In etching, a material with 0 etch rate is equivalent to a symmetric boundary condition as discussed before. For standard deloop, simple clipping against the boundary will suffice. For resist deloop, however, a special boundary condition is required, since pieces of resist can become separated from one another by etching processes that form trenches and other structures. For the mask shown in Figure 37, loops occur on the sidewalls on either side of the small central square. These loops occur even after significant post-exposure baking and are representative of the type of large scale formations, other than standing waves, that demand deloop. Clipping at the substrate boundary during simulation creates problems for deloops that occur after the clipping operation. If clipping occurs, traversal of the mesh can not be performed

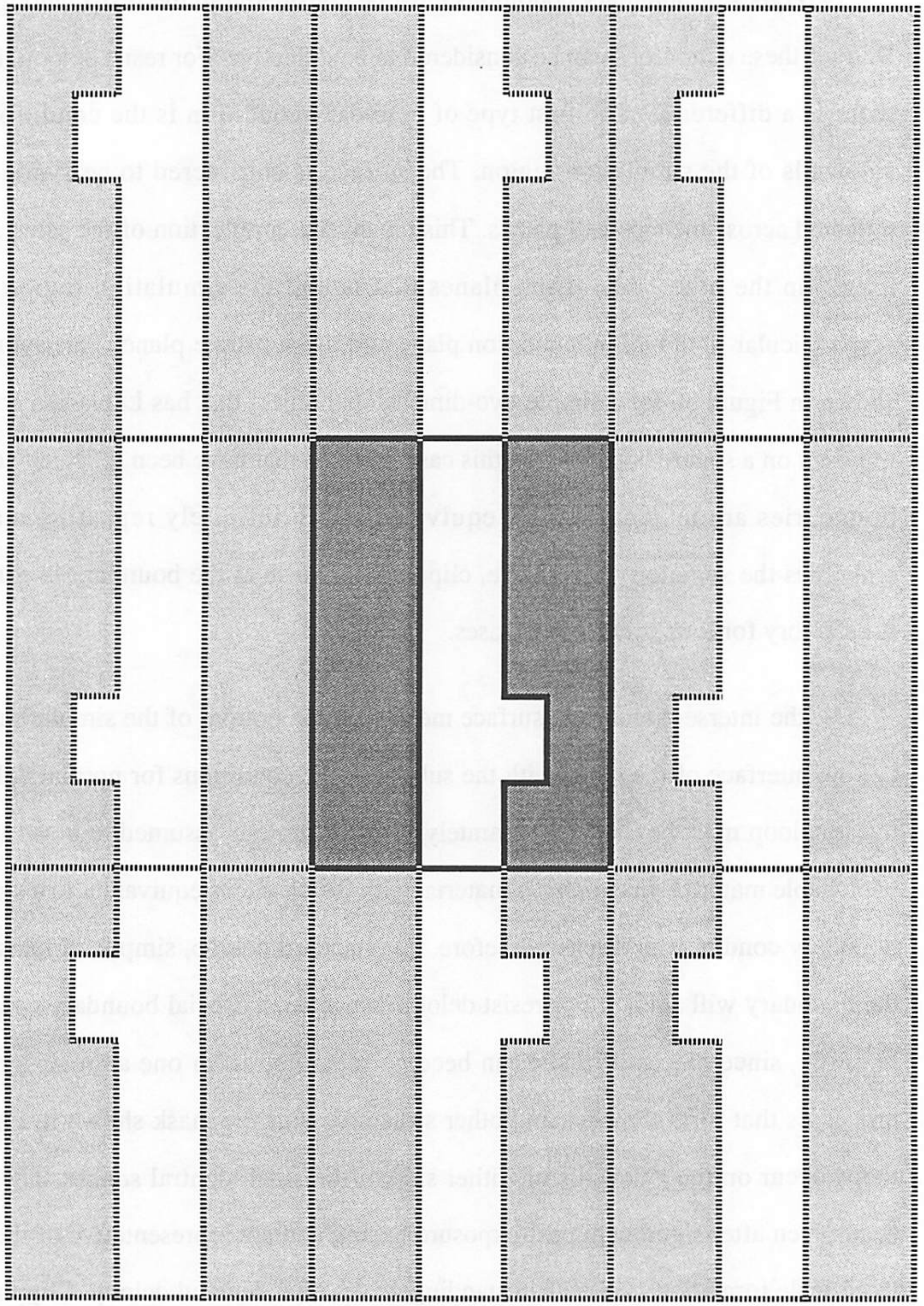


Figure 36) Reflecting Boundary Conditions

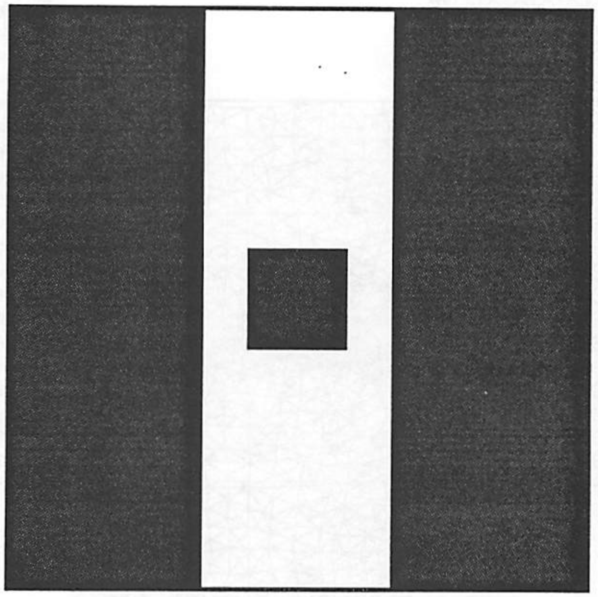
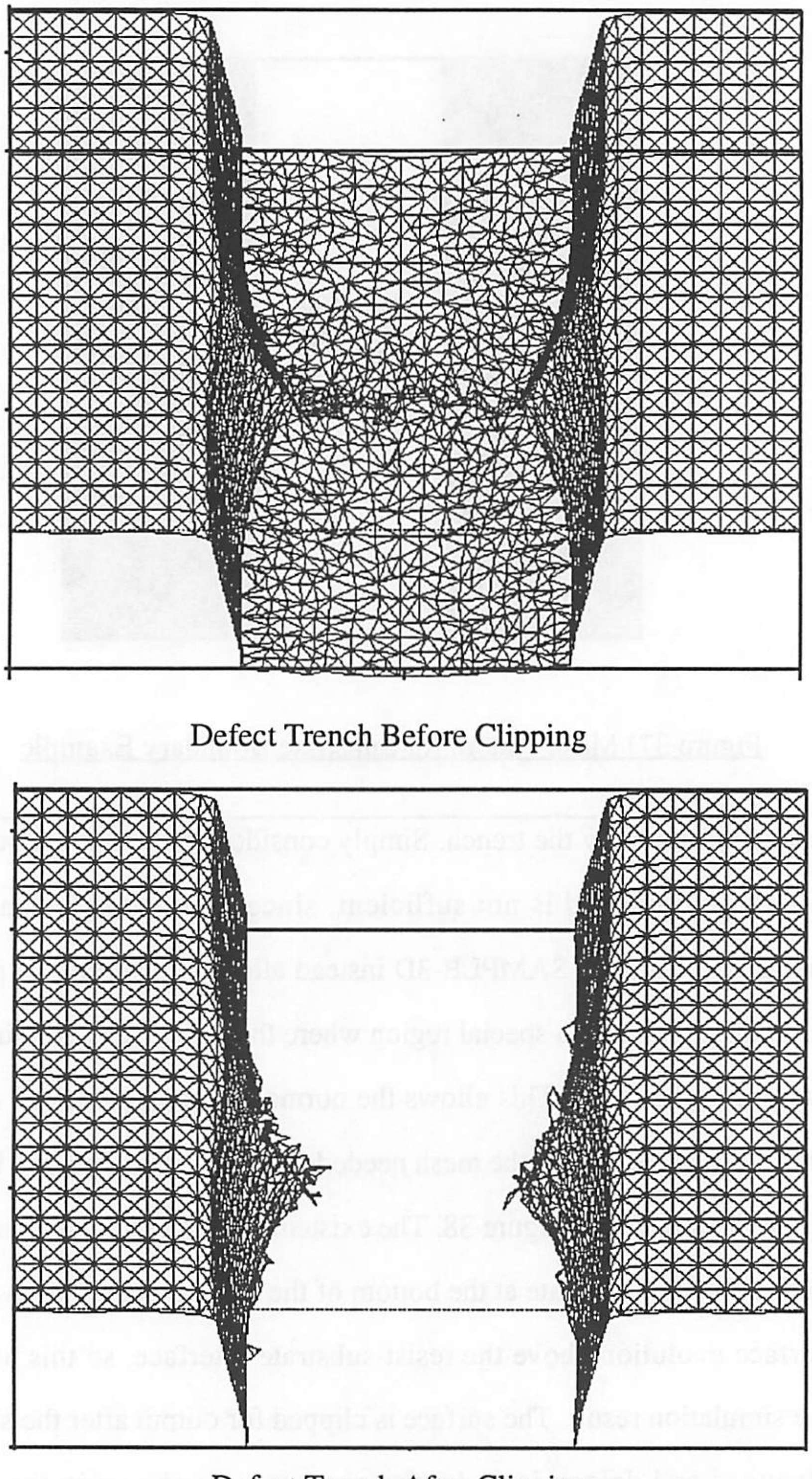


Figure 37) Mask Pattern for Substrate Boundary Example

across the gap formed by the trench. Simply considering the bottom surface as part of the mesh to be traversed is not sufficient, since loop triangles can intersect the boundary as noted above. SAMPLE-3D instead allows the surface to pass underneath the simulation region into a special region where the etch rate is 100 times higher than at the bottom of the resist. This allows the normal deloop operations to be employed on the mesh, since the part of the mesh needed for traversal is retained by the program, as shown in the top half of Figure 38. The existence of an extremely slow etch rate that is proportional to the etch rate at the bottom of the standard simulation region does not affect surface evolution above the resist-substrate interface, so this method does not affect the simulation result. The surface is clipped for output after the surface has been fully advanced and deloop is no longer necessary, as shown in the bottom half of Figure 38.



Defect Trench Before Clipping

Defect Trench After Clipping

Figure 38) Loop Removal Condition for Substrate Boundary

5.7 Set Operations Based on Deloop

The basic deloop engine has also been modified to allow set operations to be performed in SAMPLE-3D. This modification was performed by J. Sefler and was partially based upon the ideas presented in this section. This modification was performed to assist the operation of multi-layer plasma etching operations that were developed by E. Scheckler [8]. E. Scheckler's facet motion algorithm is capable of advancing a surface that represents the interface of the air with more than one material. To perform capacitance extraction, or other simulation techniques, on only one particular layer of material, the entire surface of the material must be identified. To perform this operation, the intersection of the volume of the original material with the air-material interface surface must be performed. The implementation that was performed by J. Sefler was based upon the upon the techniques in the following section and is called the CUT-3D program. Because the author of this manuscript only contributed existing code and advice to the CUT-3D effort, only the general theory of set operations that was employed is presented here.

5.7.1 Theory of Deloop Based Set Operations

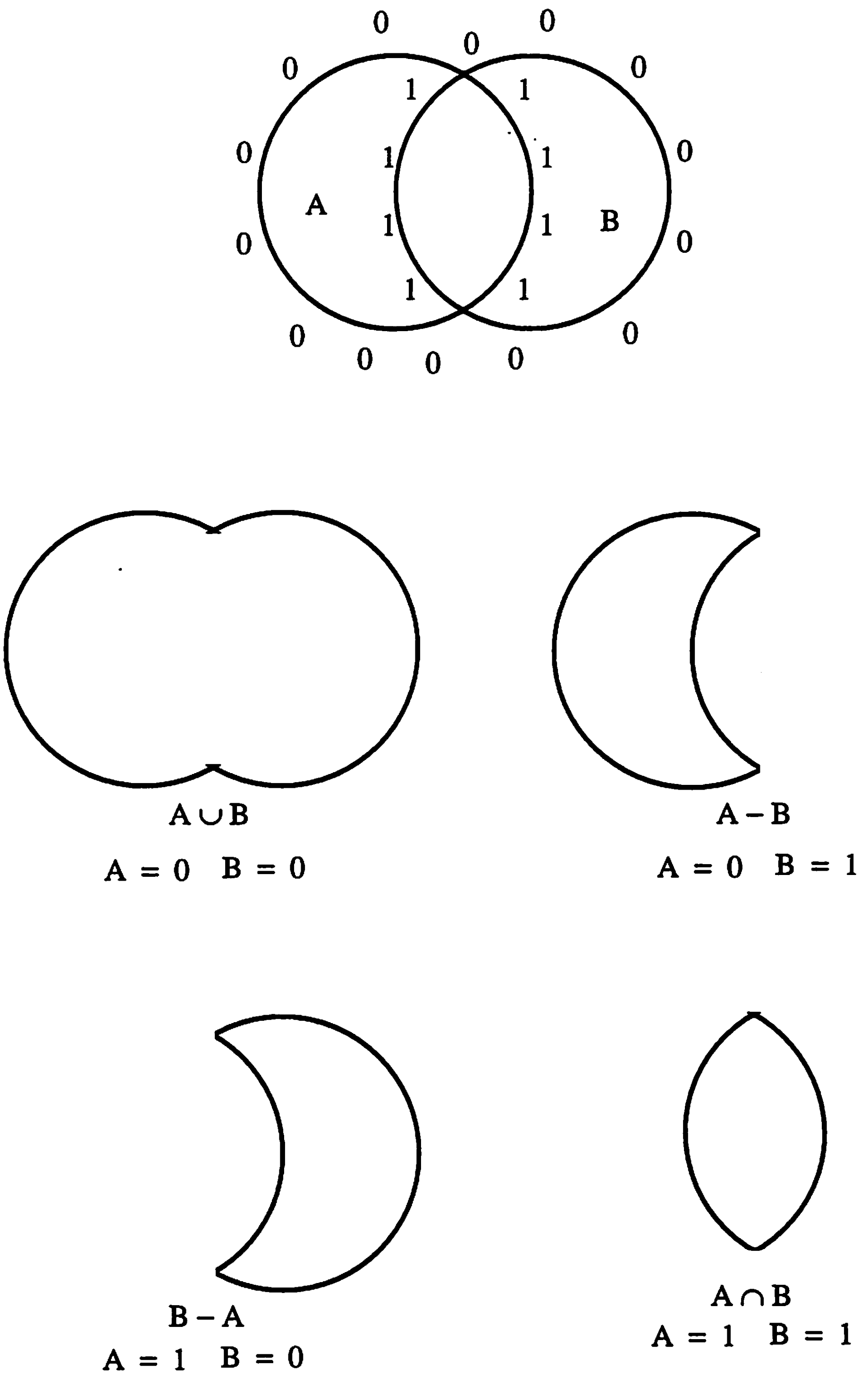
The deloop operation is a unitary operation performed on a surface in order to remove distortions created by a topological mapping. The same operation may likewise be performed on N surfaces in such a manner as to perform set operations. First, consider the special case of two surfaces. Surface subdivision occurs in the usual manner, with all triangles under consideration inserted into the octtree. The surfaces are now divided, such that the intersection line represents the common region between the two surfaces. The subdivision of triangles proceeds normally as before. The major difference now appears in the removal step. Both surfaces are first self-delooped. The winding number of each surface element is then computed with respect to the other surface. A winding number is found in the same manner as in Figure 23, except that the numbers are incremented and decremented if they cross the other surface. If the surfaces actually intersect, it is not necessary to extrude an escape ray, since the surfaces have already been self-delooped. Intersecting surfaces that have been self-delooped, may have their proper winding numbers determined by examining the orientation of the surfaces at the points of intersection. The escape ray may be necessary if the surfaces do not intersect, since the containment of one surface by another is possible.

The set operation that is performed is based on the particular set of values, for each surface, which are maintained by the remove step. For instance, if all nodes marked 0 on surface A are maintained, and all nodes marked 0 on surface B are maintained, then the operation is $A \cup B$. If all nodes marked 1 on surface A are maintained and all nodes marked 1 on surface B are maintained, then the operation is $A \cap B$. There are four possibilities for the two body situation (Figure 39).

The 2 body deloop-set operation problem can be represented by a 2x2 set of operators $D_{ij}(w)$, whereD_{ij} is winding number test of i in relation to j, and 'w' representing the winding number that is preserved. Simple deloop on body i can then be notated as $D_{ii}(0)$ and the operation that generates the basic loops of the surface is $D_{ii}(1)$. For the set operation formulation, the union of multiple deloop operations is used. The set operation $A \cup B$, can therefore be described as $D_{AB}(0) \cup D_{BA}(0)$. (Figure 39) The subscripts represent that the winding number on the surface of the first subscript is being considered in relation to the surface of the second subscript. Therefore the two body operations are:

Set Operations Table

D Operator	Set Operation
$D_{AB}(0) \cup D_{BA}(0)$	$A \cup B$



A ∪ B
A = 0 B = 0

A − B
A = 0 B = 1

B − A
A = 1 B = 0

A ∩ B
A = 1 B = 1

Figure 39) The Four Basic Set Operations Using Winding Number Labeling

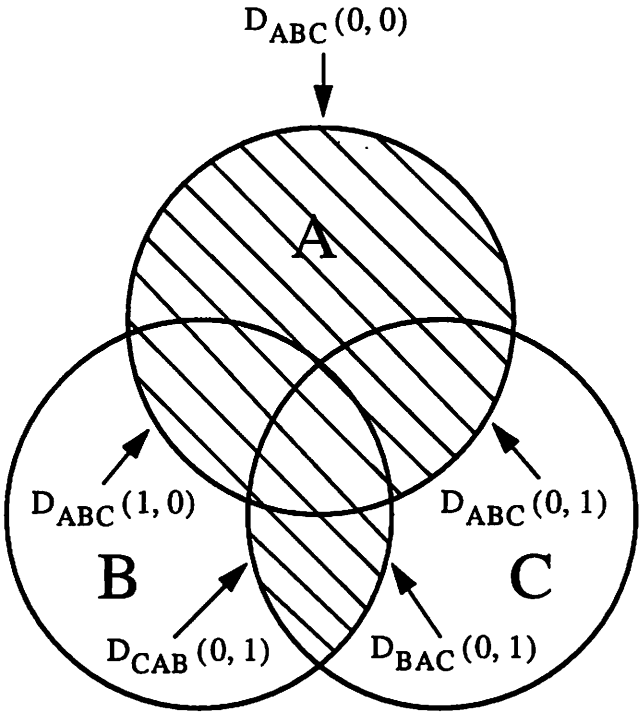
Set Operations Table

D Operator	Set Operation
$D_{AB}(0) \cup D_{BA}(1)$	$A - B$
$D_{AB}(1) \cup D_{BA}(0)$	$B - A$
$D_{AB}(1) \cup D_{BA}(1)$	$A \cap B$

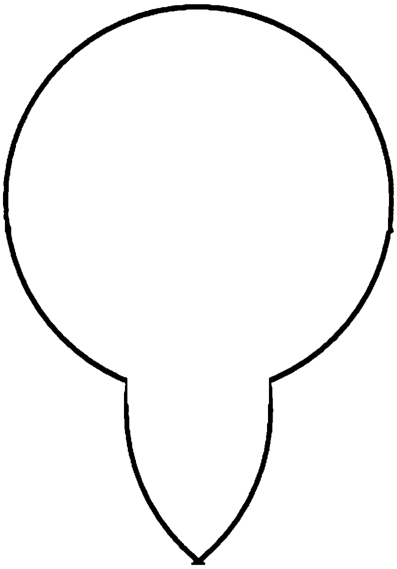
For the N body problem, the operations must be performed with a remove step for each level in the operation. Unfortunately, the expression may have to be evaluated as a large collection of minterms. For instance, there is no set of single integers to describe what parts should be kept of A, B and C for the operation $A \cup (B \cap C)$ (Figure 40). This function evaluates to the union of the terms $D_{ABC}(0, 0)$, $D_{ABC}(1, 0)$, $D_{ABC}(0, 1)$, $D_{CAB}(0, 1)$ and $D_{BAC}(0, 1)$, where the first letter in the subscript is the body containing the surface part, and the arguments are the winding numbers according to the other subscripted surfaces. Because two-argument set operations are easier to compute, since only one winding number is required for a surface, it may be preferable to only perform these operations, and break down all desired high level operations into this form. Such a breakdown can be precomputed using logic minimization techniques such as the MIS system [14].

5.8 Comparison of Resist Deloop with Toh's Loop Removal Method

A technique with $O(N\log N)$ time behavior for removing negative volume regions from three dimensional meshes has been designed and implemented. This method is significantly faster than the method designed by K. Toh [1], which has $O(N^2)$ execution time. The new deloop method, on a DECstation 5000, performed at a loop removal step in 12 seconds for 3000 triangles. Two octtree computations were performed, each requiring 5 seconds, at a rate of 600 triangles/sec. 2 seconds were required for labeling and removal. The old deloop method required 110 seconds for



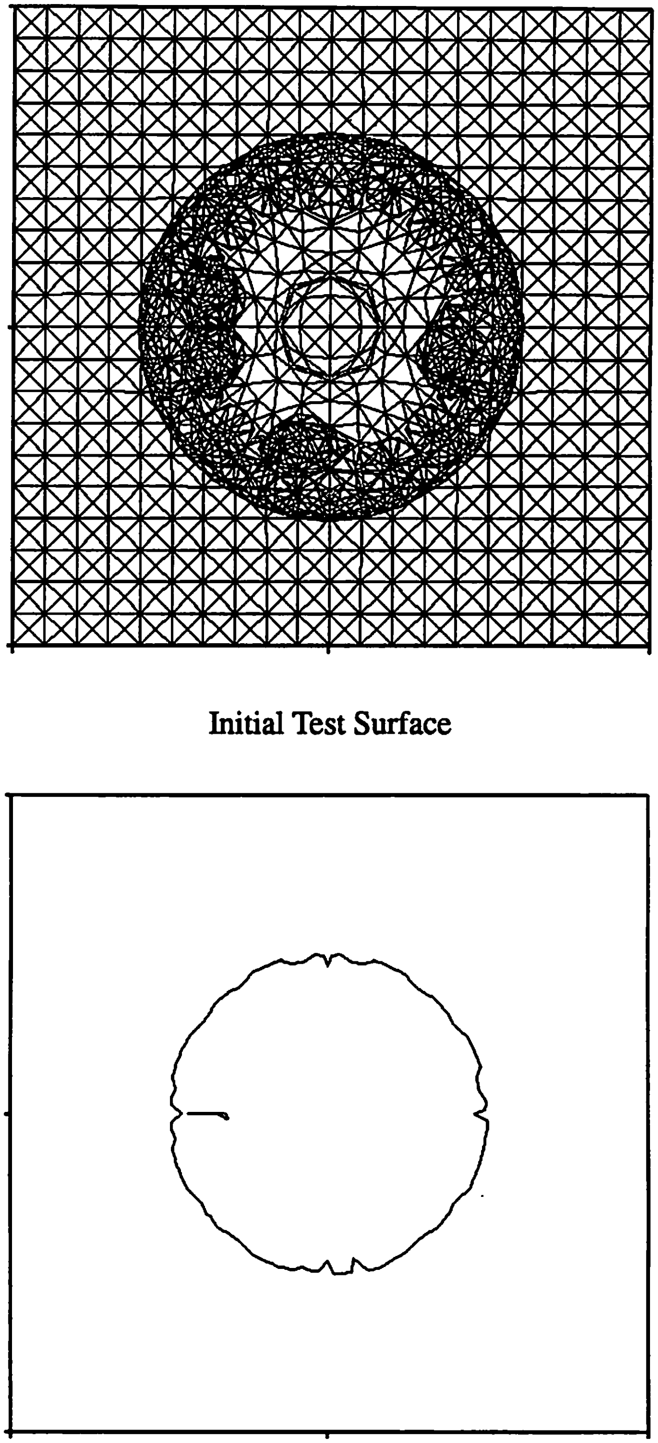
Initial Sets with Deloop Expressions Marked



Extracted surface of $A \cup (B \cap C)$

Figure 40) A Three Surface Set Operation the same problem. The total time necessary to advance the surface to this operation was 40 seconds. The surface is shown as a top view in Figure 41 along with its intersection line. The result of applying old deloop to the test case is shown in Figure 42, along with the result of applying the new deloop method. As well as being faster, the new resist deloop technique successfully identified the actual surface and removed the loops.

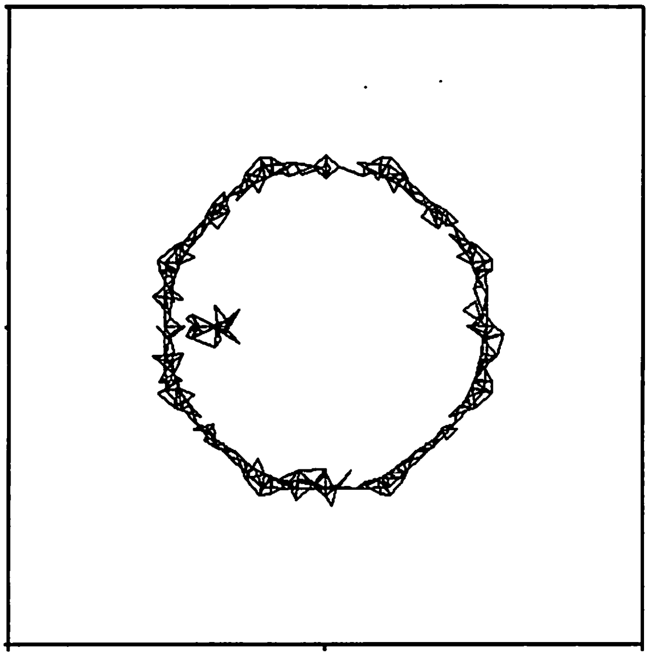
In summary, the most important loop removal technique for general application is the basic loop removal technique that constructs a closed non-intersecting surface. This method has wide application in entire surface advancement field. For the purposes of simulating photoresist development via the ray-trace method, the resist deloop technique is preferable, although it is unknown at this time whether integer labeling or normal labeling is the best method to be employed. Implementing resist deloop has added significant functionality to ray-trace, but it doesn't allow ray-trace to operate fully as a general photoresist dissolution simulator. Once resist deloop was implemented, a new difficulty with ray-trace was discovered that could not be detected earlier. This is the phenomena of ray scattering. It is discussed in Chapter 6.



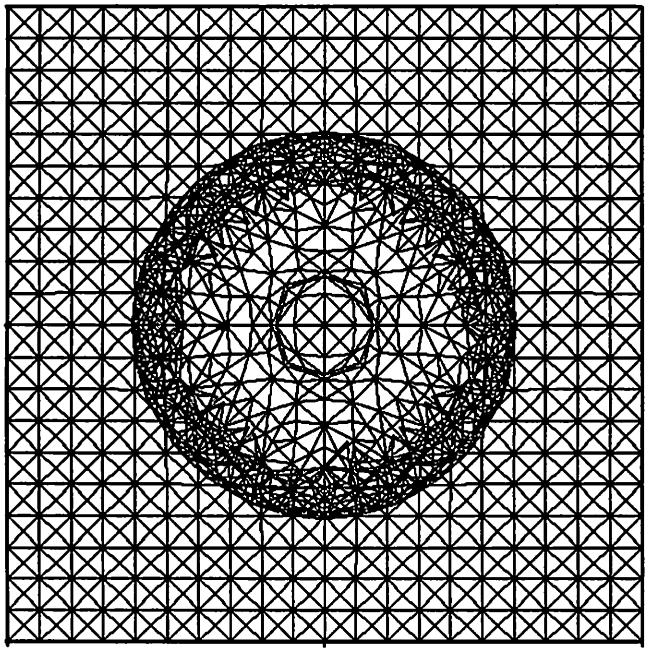
Initial Test Surface

Intersection Line of Test Surface

Figure 41) Test Surface for Deloop Comparison



Old Loop Removal



Resist Deloop

Figure 42) Results of Deloop Comparison

Reference for Chapter 5

[1] K.K.H. Toh, *Ph.D. Dissertation*, University of California, Berkeley, Dec. 1990.

[2] E.W. Scheckler, K.K.H. Toh, D.M. Hoffstetter and A.R. Neureuther, "3D Lithography, Etching and Deposition Simulation," *Symposium on VLSI Technology*, pp. 97-98, (Oiso, Japan), May 28-30, 1991.

[3] S. Hamaguchi, M. Dalvie, R. T. Farouki and S. Sethuraman, "A Shock-Tracking Algorithm for Surface Evolution Under Reactive-Ion Etching," *IBM Research Report*, RC 18283 (80168), Aug., 1992

[4] J. A. Sethian, "Numerical Algorithms for Propagating Interfaces: Hamilton-Jacobi Equations and Conservation Laws," *Journal of Differential Geometry*, vol. 31, pp. 131-161, 1990.

[5] P.I. Hagouel, "X-Ray Lithographic Fabrication of Blazed Diffraction Gratings", *Ph.D. Dissertation*, University of California, Berkeley, 1976.

[6] L. Jia, W. Jian-kun, and W. Shao-jun, "Three-Dimensional Development of Electron Beam Exposed Resist Patterns Simulated by Using Ray Tracing Model," *Microelectronic Engineering*, vol. 6, pp. 147-151, 1987.

[7] E. Barouch, B. Bradie, H. Fowler, and S.V. Babu, "Three-Dimensional Modeling of Optical Lithography for Positive Photoresists," *Interface' 89: Proceedings of KTI Microelectronics Seminar*, pp. 123-136, Nov. 1989.

[8] E. Scheckler, *Ph.D. Dissertation*, University of California, Berkeley, Nov. 1991.

[9] D. Meagher, "Geometric Modeling Octtree Using Encoding", *Computer Graphics and Image Processing*, p. 192, June 1982.

[10] J. Foley, A. van Dam, S. Feiner, and J. Hughes, *Computer Graphics: Principles and Practice 2nd Edition*, p. 706, Addison-Wesley, Reading, MA, 1990.

[11] J. Munkres, *Topology: A First Course*, Prentice-Hall, Englewood Cliffs, NJ, 1975.

[12] F.P. Preparata and M.I. Shamos, *Computational Geometry*, Springer Verlag, New York, 1985.

[13] B. Foote, *M.S. Thesis*, University of California, Berkeley, Sept. 1990

[14] R.K. Brayton, R.Rudell and A.L. Sangiovanni-Vincentelli, "MIS: A Multiple-Level Logic Optimization", *IEEE Transactions of Computer Aided Design*, pp. 1062-1081, November 1987.

[15] J. Helmsen, E. Scheckler, A. R. Neureuther and C. Sequin, "An Efficient Loop Detection and Removal Algorithm for 3D Surface-Based Lithography Simulation," *NUPAD IV*, May, 1992.

Chapter 6 Surface Mesh Maintenance

6.1 Introduction

SAMPLE-3D requires that its meshes be as regular as possible for a variety of reasons. As previously stated in Chapter 4, it is necessary for points to be well spaced for a properly accurate representation of important surface properties, such as the surface normal. In addition, the existence of a regular mesh assists in alleviating difficulties that occur during surface advancement. The two major difficulties that are encountered during surface advancement, other than improper segment lengths, are the existence of thin triangles and crenulations in the surface. If these features can be removed, or made less severe, during advancement, then advancement of the surface mesh can become easier and more accurate. The most desirable way to remove these difficulties is through the use of easily computed local heuristics, since the effect of these operations on the accuracy of the overall topography is generally not severe, except in the beneficial sense, and other attempts at regularizing the mesh [1], which were based on computing global approximations, are typically found to involve significant computation time.

Attempts at heuristic regularization of a triangular mesh that does not have an associated volumetric data structure for the purpose of surface advancement have not been found in the literature. Some methods that involve advancement by conjugate gradient techniques have been discussed [10][11], but these are slower than would be desired for photolithography and topography simulation purposes. Therefore, an approach is outlined in this chapter, which is based upon two dimensional observations and the desire to remove crenulation from a three dimensional surface. The heuristic techniques must, of course, be considered tentative. This approach is partially implemented and encouraging examples are given.

6.1.1 Grid Generation

Grid generation is an important consideration for any method to numerically solve partial differential equations that are embedded in a continuous spatial domain. To represent the domain for the purpose of solving partial differential equations, it is necessary to divide the space into a finite number of connected patches. Typical two dimensional examples of patches are squares, rectangles and triangles. To represent important simulation parameters, i.e. the variables and other important terms in the differential equations, shape functions are normally employed. A shape function is a parameter or set of parameters with an associated interpolation function that is defined over a single spatial patch [2]. This function is used to determine the value of a variable or other parameter at any point contained in the patch. For this reason, numerical methods depend on shape functions to represent the parameters being manipulated. Many types of shape functions exist, though the one that is used most often is a single parameter value that is defined over an entire patch.

Because the shape functions are interpolation methods, they contain error terms that are dependent on the size and shape of any particular spatial patch. Therefore, the ability of a shape function to represent important simulation values is strongly dependent on the size and shape of individual patches. The patches must be small enough to allow the error that arises from the interpolation of shape functions to be minimized. In the case of rectangular or triangular spatial patches, it may also be necessary to enforce particular conditions upon the dimensions of these shapes, such as keeping the length to width ratio of rectangles below a specified maximum value, or keeping every angle in each triangle above a certain value. More complex conditions upon the possible range of shapes may also be enforced. In addition to constraints on patches that arise due to simulation error, other limitations may be required in order to minimize the amount of memory and CPU time that is consumed during simulation. Limitations of this form are normally expressed as restrictions on the minimum size of patches and as limits on the number of patches that may be employed.

6.1.2 Dynamic Grids

Many methods that employ grids are concerned only with the initial generation of the grid. Programs such as FASTCAP [3] that do not contain time dependent variables have no need to use more than one spatial subdivision. Other simulation algorithms, such as ADVECT [4] or CRATER [5] in SAMPLE-3D (Chapter 4), that do simulate time dependent processes, simplify both programming and algorithmic complexity by employing only one spatial grid over all time steps. Finally, there are methods that do dynamically alter the grid over successive time steps. One class of grids that alter over successive time steps are 'perturbational grids'. These grids simulate physical systems by deforming themselves to conform to the motion of a particular physical parameter or object. The SAMPLE-3D string algorithm and associated winged-edge data structure is one example of this method, since the elements s of the grid distort themselves to track the motion of the evolving surface. Examples also include the work of H. Trease [6] with Free-Lagrangian grids and the Surface Evolver of K. Brakke [11]. The Surface Evolver also models points with triangles in 3-dimensions, but it focuses exclusively on minimum energy surfaces.

6.1.3 Triangulation of a Moving Interface

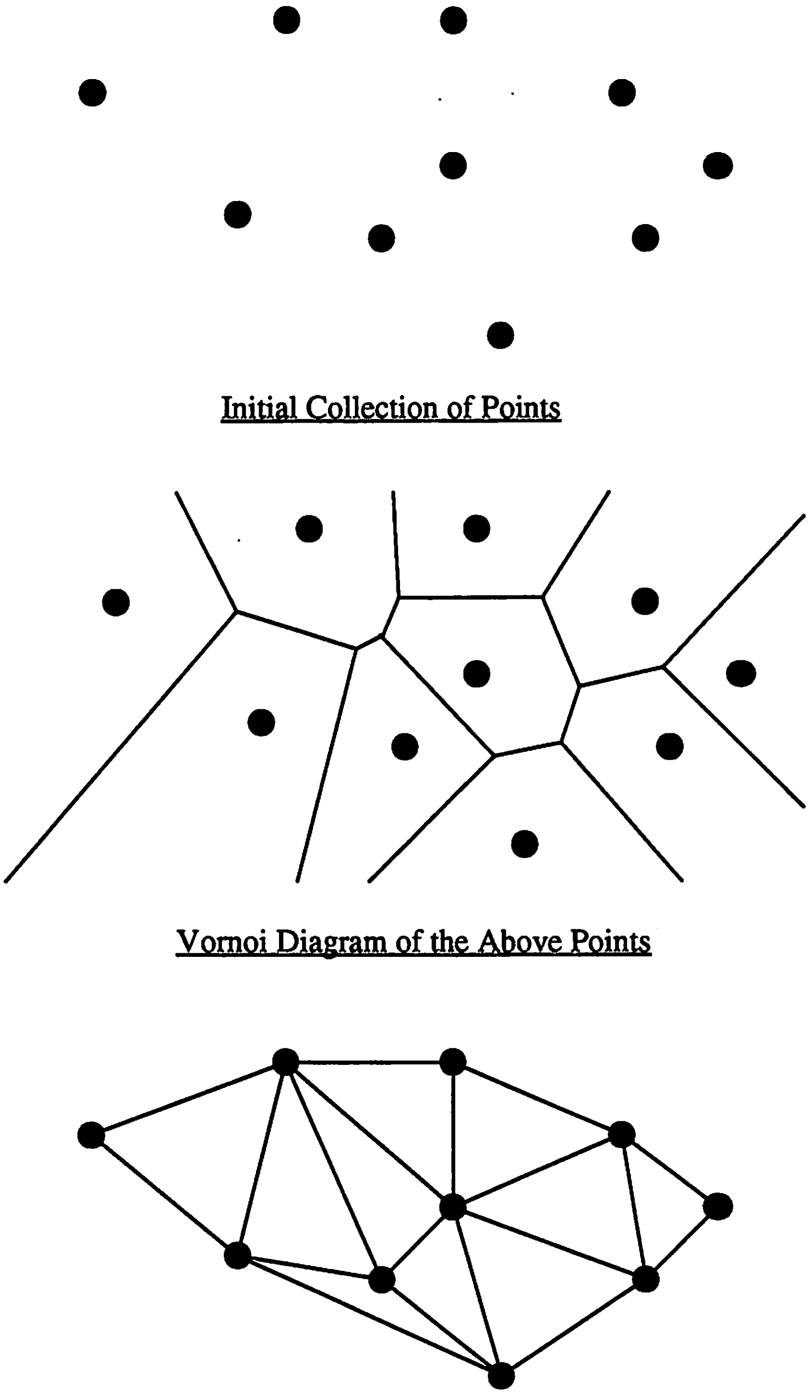
SAMPLE-3D concerns itself specifically with a two-dimensional grid of triangles whose motion are determined by the advancement of the points in a perturbational manner that depends on the Lagrangian equations of motion (Chapter 3). The triangles represent the connectivity of the mesh points and are used to determine when particular points must be removed, so that the point trajectories don't get too dense, and when new points must be created so that new rays may be interpolated. For the purposes of accuracy, it is necessary that the triangles be reasonably close to equilateral. A uniform triangle size also allows for a reasonably large time step to be used, which is approximately 15% of the ideal segment length divided by the maximum movement rate in SAMPLE-3D. This uniformity of triangle sizes is especially important for ray-trace, since the points do not use information from nearby points to determine the advancement rate and direction. It is, therefore, extremely important that high frequency distortions in the mesh be damped, while low frequency distortions that represent topography be maintained and allowed to form. If this is not done, then extremely small loops may form that are not appropriate for the delooper to handle. Therefore, we see from experience that there are four important conditions that should be maintained in order to properly represent the topographical surface and assist the advancement method in maintaining accuracy. 1) Small segments must be removed, since points that are too close are redundant and can cause non-meaningful loops to form. 2) Large segments must be broken into smaller segments new points must be interpolated. 3) Thin triangles must be removed for three reasons. First, non-meaningful loops may form at these locations. Second, interpolating the direction vector when new rays are introduced is extremely inaccurate if the surrounding surface contains a thin triangle. Third, and most important, large time steps can be taken without turning the triangle inside out. 4) High order variations of the surface normal must be removed. Removal of these variations by local operations that do not move points significantly, will insure that important topographical features are not affected adversely.

6.2 Triangulations in Two Dimensions

Surfaces that represent topography often contain large planar areas. Therefore, most operations that preserve a proper triangulation on topographical structures are built from existing methods developed for triangulations in two-dimensions. An explanation of the most popular planar triangulation method, Delaunay Triangulation [7], is given and its close relationship to altitude-based triangulation. A specific heuristic method of enforcing a minimum altitude that has been developed for SAMPLE-3D is described.

6.2.1 Delaunay Triangulations

The Vornoi diagram [7] of a set of mesh points in the plane is a collection of regions that partition the plane. The Vornoi region of any mesh point is the set of all points in the plane that are closer to that mesh point than any other mesh point. Exactly one region is associated with each mesh point (Figure 1). The Delaunay method of constructing a triangulation of some set of mesh points, is to construct the dual of the Vornoi diagram of that set of mesh points. This procedure constructs a unique triangulation that has several important properties. First, the circle of circumcision for any triangle does not enclose any points other than those in the triangle. This is known as the 'Delaunay condition'. Also, the minimum angle of all the triangles in the mesh is maximal for all possible triangulations. These properties mean that the triangulation contains a minimum number of 'thin' triangles. In general, the triangles in a Delaunay triangulation will be as equilateral as possible. Since it is desirable for the triangles in the SAMPLE-3D surface mesh to be as equilateral as possible, methods that are used to maintain Delaunay-like conditions have direct relevance to SAMPLE-3D mesh maintenance problems.

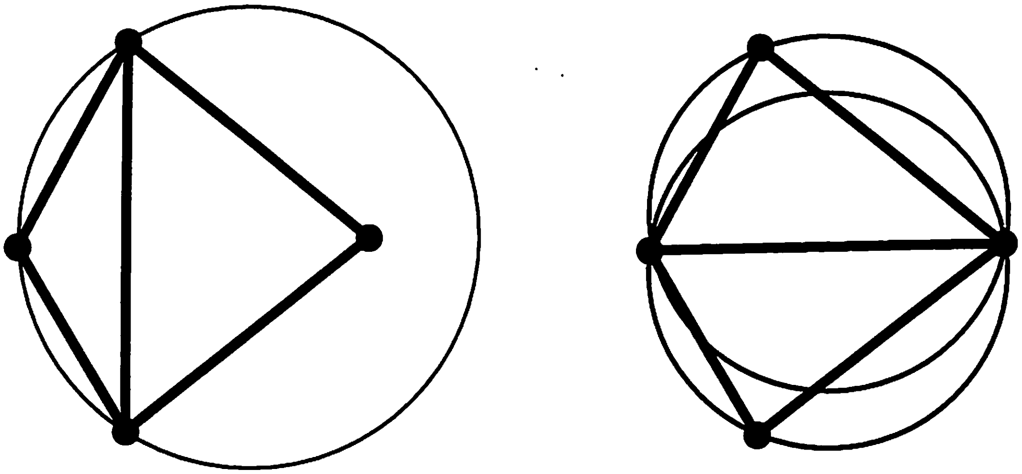


Initial Collection of Points

Vornoi Diagram of the Above Points

Delaunay Triangulation Formed from the Vornoi Diagram

Figure 1) The Delaunay Triangulation



Before Delaunay Flip

After Delaunay Flip

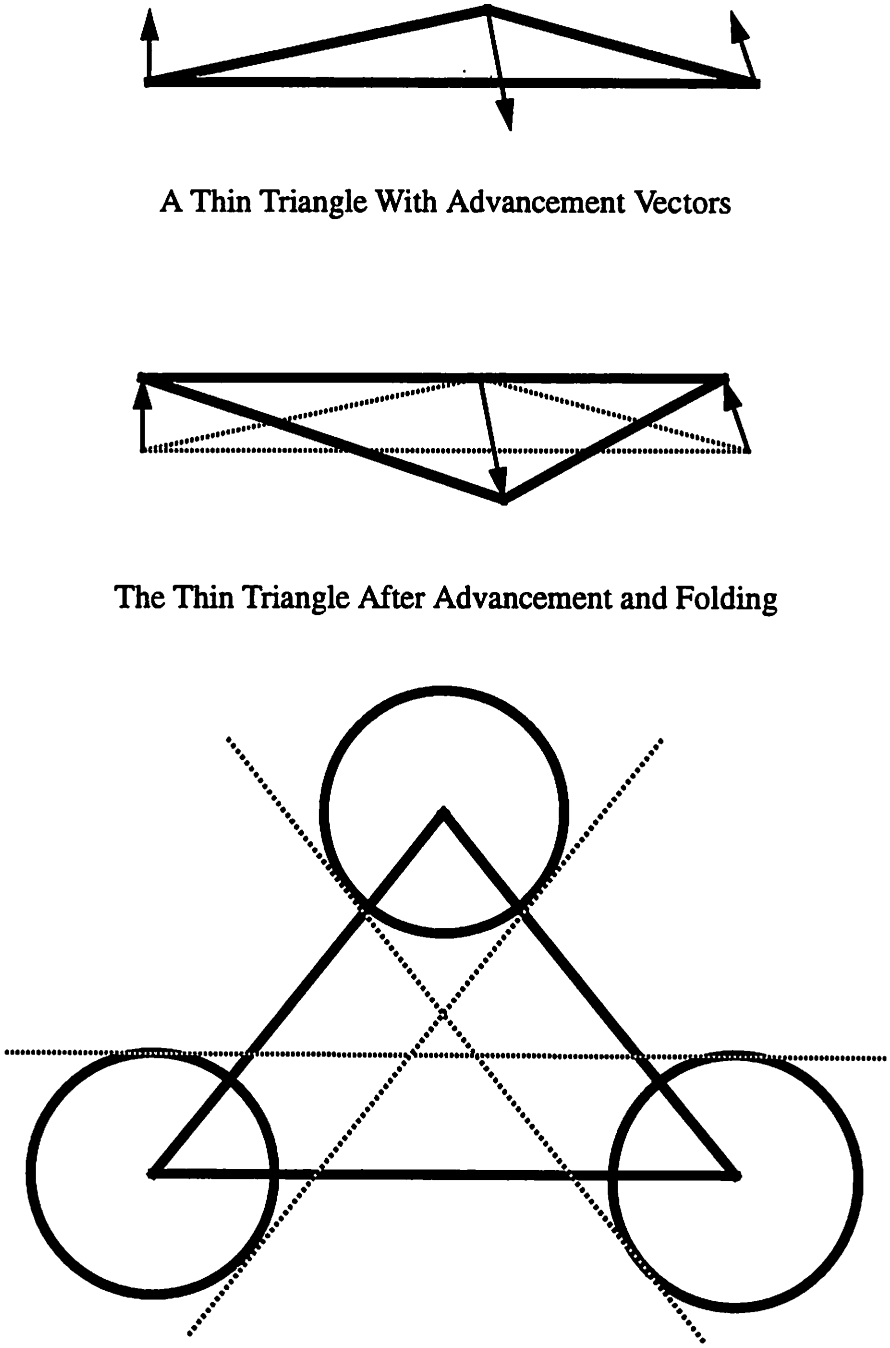
Figure 2) The Delaunay Flip

6.2.2 Delaunay Triangulations with Moving Points

If a point is moved in a Delaunay type mesh without affecting the surrounding connectivity, it is possible that some of the resulting triangles may no longer satisfy the Delaunay Condition. If this should occur, it is desirable to use the existing mesh as a guide for a new triangulation of the points that satisfies the Delaunay condition, instead of completely regenerating the entire triangulation by generating a new Vornoi Diagram. Fortunately, such a method exists. An example for the three-dimensional tetrahedral case was developed by J. Painter [12]. This method, called the 'Delaunay Flip', allows the original triangulation to serve as an initial condition for a new triangulation, and constructs the new triangulation through a series of local operations. Given two triangles that share a common edge, where the circle of circumcision encloses all four points, the two triangles may be replaced by two different triangles by 'flipping the edge' as seen in Figure 2. The important circles of circumcision are included in the figure. While the first triangulation does not satisfy the Delaunay condition, the two new triangles do satisfy it, in regard to the original four points. Successive application of this operation, starting with the thinnest triangles, leads to a new mesh that satisfies the Delaunay condition in a few operations. This suggests that a proper triangulation can be maintained for a set of points in motion via local operations, and, if the motion of the points is not extreme, the updating procedure will be faster than a completely regenerating the planar triangulation. The existence of this method is dependent, however, on the existence of a set of well-defined conditions, like the Delaunay condition, that can be used to inform the design and application of the local operations.

6.2.3 Altitude and Segment Length Condition with Moving Points

The planar part of the SAMPLE-3D mesh refinement scheme is now described. To satisfy the first three of the four requirements for a well triangulated mesh, two conditions are defined that must be maintained on the triangles. First, the lengths of all the segments in the mesh must be between a lower and an upper bound. This condition was proposed and implemented by Kenny Toh [8]. The second condition is that all the altitudes of the triangles must be greater than a minimum length [9]. The altitude is the distance of a vertex from the opposite triangle side. The second condition is used to remove thin triangles from the mesh and to keep triangles from 'folding' themselves. Folding is described as a point, moving through a segment, in-between updates in triangulation. An example of this phenomenon is shown in Figure 3. The folding phenomena can be prevented from occurring, as shown in the bottom of Figure 3, because the motion of the points in SAMPLE-3D is limited by an advancement length, as discussed in Chapter 4. If the distance that the points move between thin triangle removal steps is less than half of the minimum altitude, it is impossible for a point to pass through the opposite side of a triangle during advancement. By enforcing a minimum altitude condition, as shown in Figure 4, in conjunction with a minimum

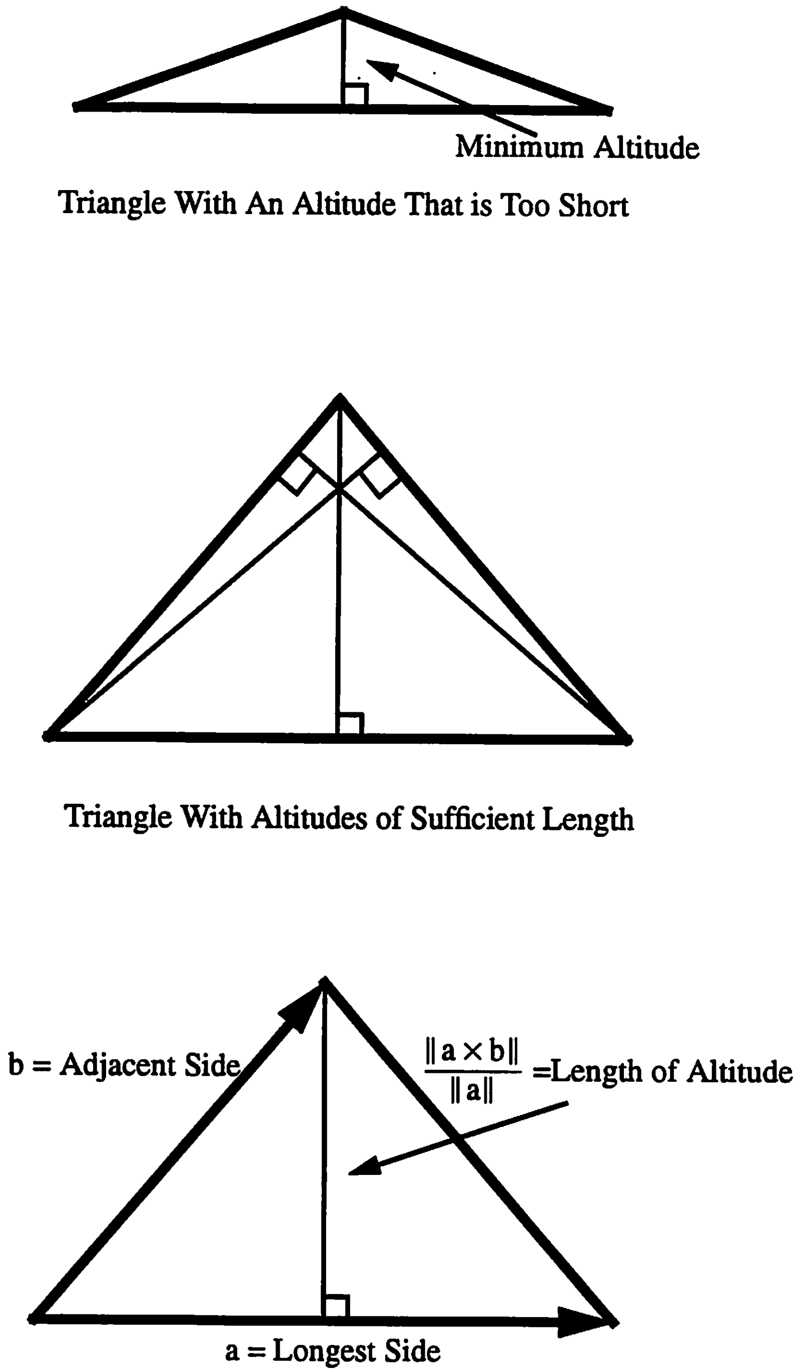


A Thin Triangle With Advancement Vectors

The Thin Triangle After Advancement and Folding

A Triangle That Is Impossible To Fold

Figure 3) Triangle Folding



Triangle With An Altitude That is Too Short

Triangle With Altitudes of Sufficient Length

Computing the Minimum Altitude

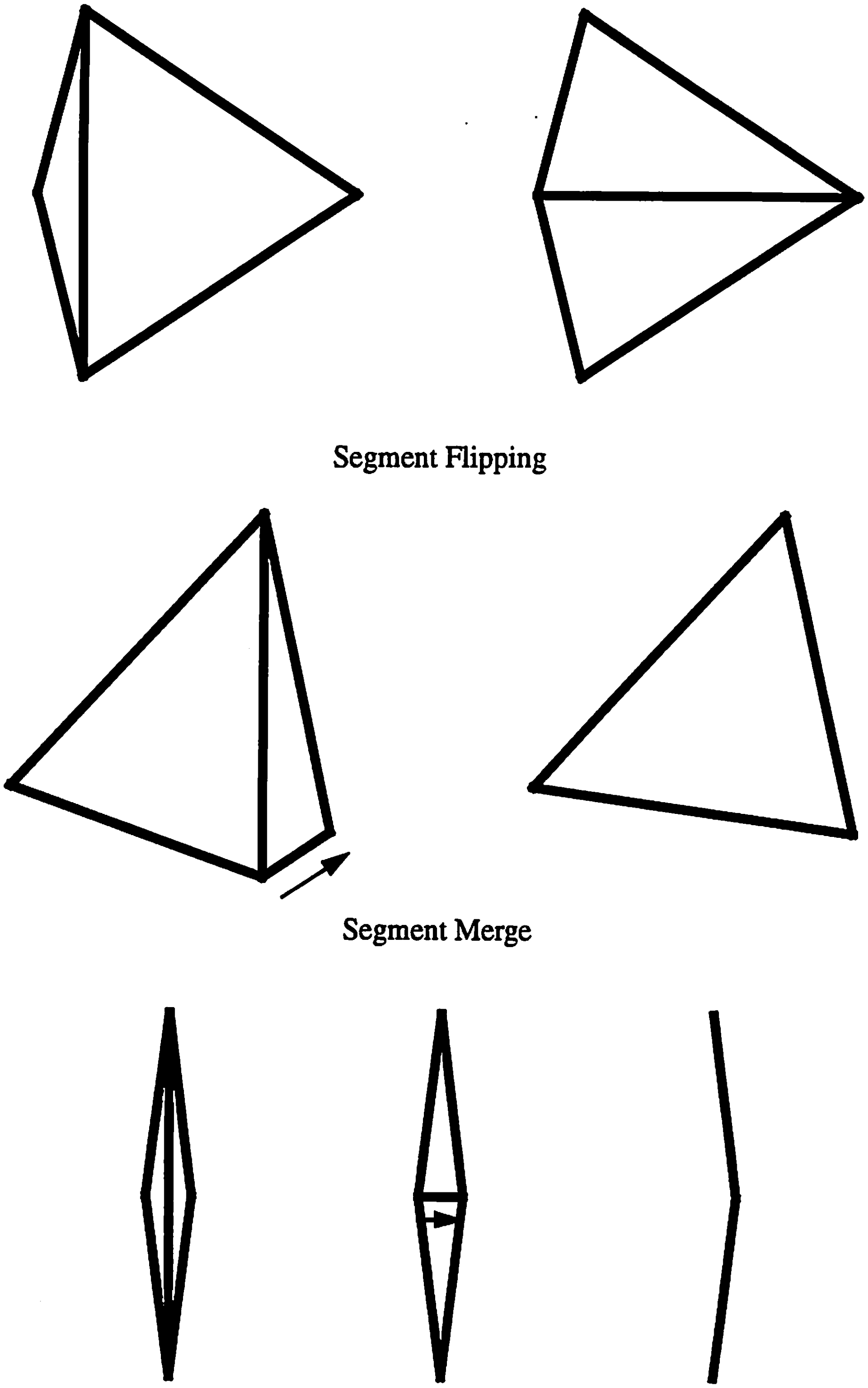
Figure 4) Minimum Altitude Condition

segment length condition, a triangulation is generated that cannot contain folding after advancement. This method has often been employed in triangular and tetrahedral free-lagrangian grids [6][12]. The smallest altitude is associated with the longest side of the triangle, since the side length times the corresponding altitude equals twice the triangle area, and can be computed by the formula:

$$\frac{\|a \times b\|}{\|a\|} \qquad \text{[EQ: 1]}$$

where the vectors a and b are two sides of the triangle as given in Figure 4.

To maintain these triangulation requirements, local mesh modification strategies are employed to remove thin triangles. These methods are shown in Figure 5. The first strategy is the Delaunay Flip. It is performed if it's application will not form another thin triangle. In SAMPLE-3D meshes, this strategy is used the most. If it is detected that the application of a Delaunay Flip will form a single thin triangle, then the segment in common between the original thin triangle, and the new thin triangle that would be formed, is merged. The node that is opposite the longest side in the thin triangle is left stationary, and the other node of the segment is moved so that the two nodes occupy the same coordinates. The connectivity of the mesh connected to the two nodes is then combined to form a single node as in a normal segment merging step. If a Delaunay Flip would form two thin triangles, or if the diagonal segment that would be formed is less than the minimum segment length, then the diagonal segment is generated and the two nodes are merged into one node. At the border of the simulation region, it may occur that a thin triangle has formed where the long segment is coincident with the edge. Since flipping is impossible in this case, the opposite node of the thin triangle is moved to the location where the altitude intersects the long segment, thereby removing the thin triangle. Figure 6 shows this procedure. These methods strongly tend towards convergence, since points and segments that are



Segment Flipping

Segment Merge

Diagonal Segment Merge

Figure 5) Thin Triangle Removal

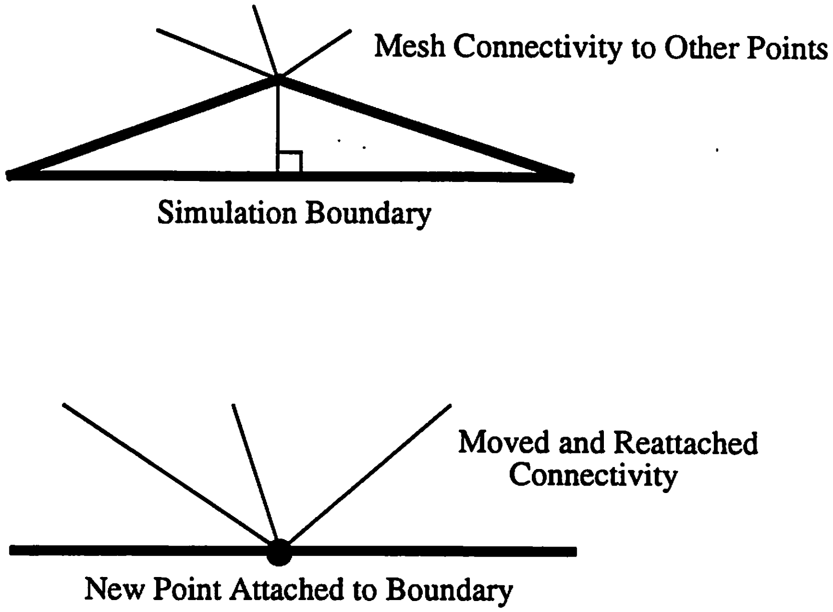
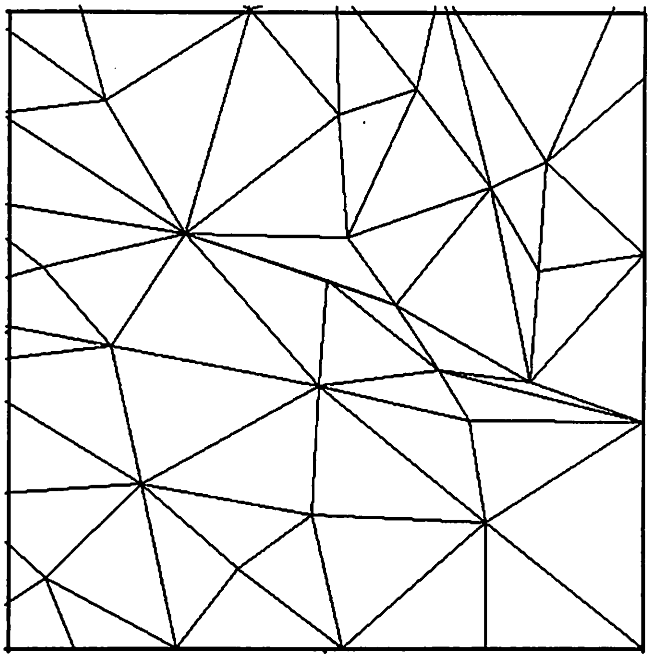
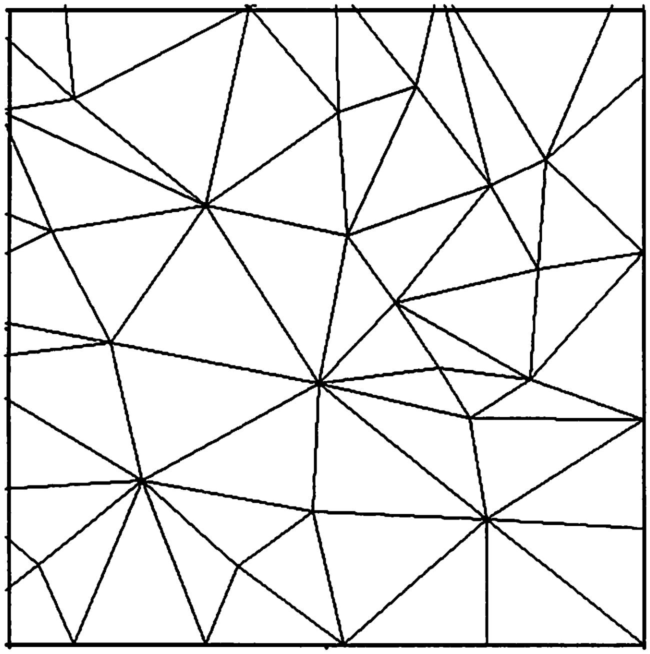

Mesh Connectivity to Other Points
Simulation Boundary
Moved and Reattached Connectivity
New Point Attached to Boundary

Figure 6) Removal Of Thin Triangle At Boundary

problematic in forming proper triangulations are removed. The actual minimum altitude used is half of the minimum segment length. (Although advancement of rays is normally limited to half the segment length, and the altitude needed to prevent folding for this advancement distance in any direction would be the minimum segment length, the points move mostly in the direction of the normal of the triangle during photolithography simulation. Therefore, the altitude requirement can be less strict.) While this altitude may seem small, it contributes significantly to the robustness of the triangulation method, since this value is large enough to catch problematic triangles, while being small enough that significant mesh alteration is not necessary to remove these triangles. An example of thin triangle removal in the SAMPLE-3D mesh is shown in close up in a somewhat planar region in Figure 7.



Mesh Without Thin Triangles Removed



Mesh With Thin Triangles Removed

Figure 7) Thin Triangle Removal

6.3 Surface Triangulations in Three Dimensions

In three dimensions, issues arise that did not appear in the planar triangulations. Besides keeping the triangles from folding and interpolating new points in appropriate places, it is necessary to adjust the triangulation to reduce the overall sum of the dihedral angles of the segments of the mesh, and to prevent inadvertent topological changes from occurring. Because of the ability for the mesh to be non-planar, segment merging and thin triangle removal are affected. Crenulation, defined as unnecessary folding that occurs during advancement, needs to be damped out as well, since thin triangle methods help reduce it but cannot prevent it. Reducing crenulation is also important, since it makes advancement methods that employ many triangles meeting at a point much more stable and accurate than they otherwise would be.

6.3.1 Segment Merging and Subdivision

Basic segment merging and subdivision with ray interpolation were covered in Chapter 4. Segment subdivision has no great significance for mesh maintenance in this chapter, since the shape of the surface is undistorted. The tendency of segment subdivision to form thin triangles is small. Segment merging has other properties, however, that can cause difficulties to appear when applied in a three-dimensional setting. If segment merging is considered in the domain space of triangles, as in Figure 8, it is seen that the removal of a segment 'compresses' the triangular regions next to the segment. These regions may contain more than a single triangle and still satisfy other mesh conditions, since the surface pieces can extend into the third dimension. Techniques related to segment merging have also been

Two merging difficulties may occur, but they are dependent on an identical condition occurring in the mesh. As seen in Figure 9, the existence of mesh connectivity within the region to be merged, can lead to serious distortions in the

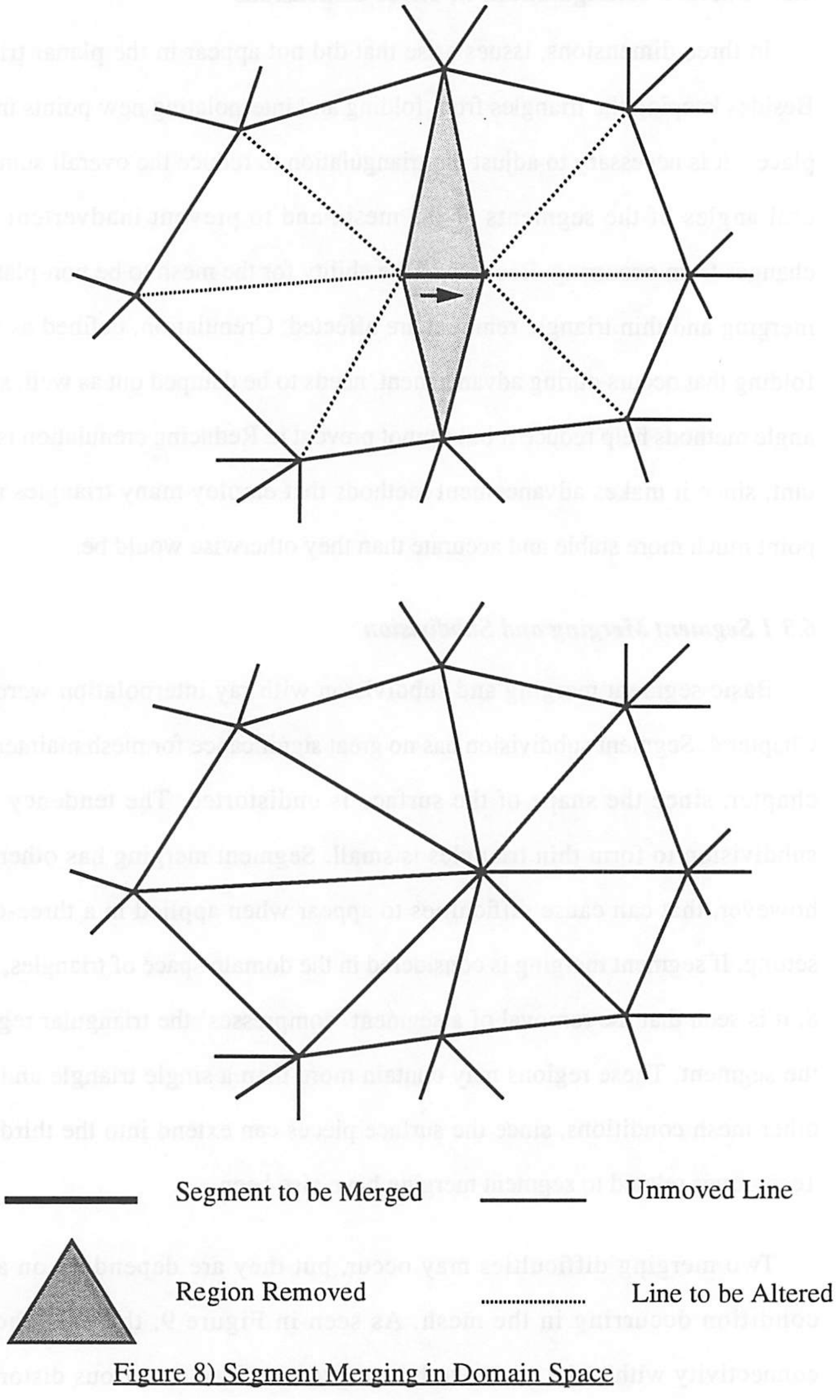
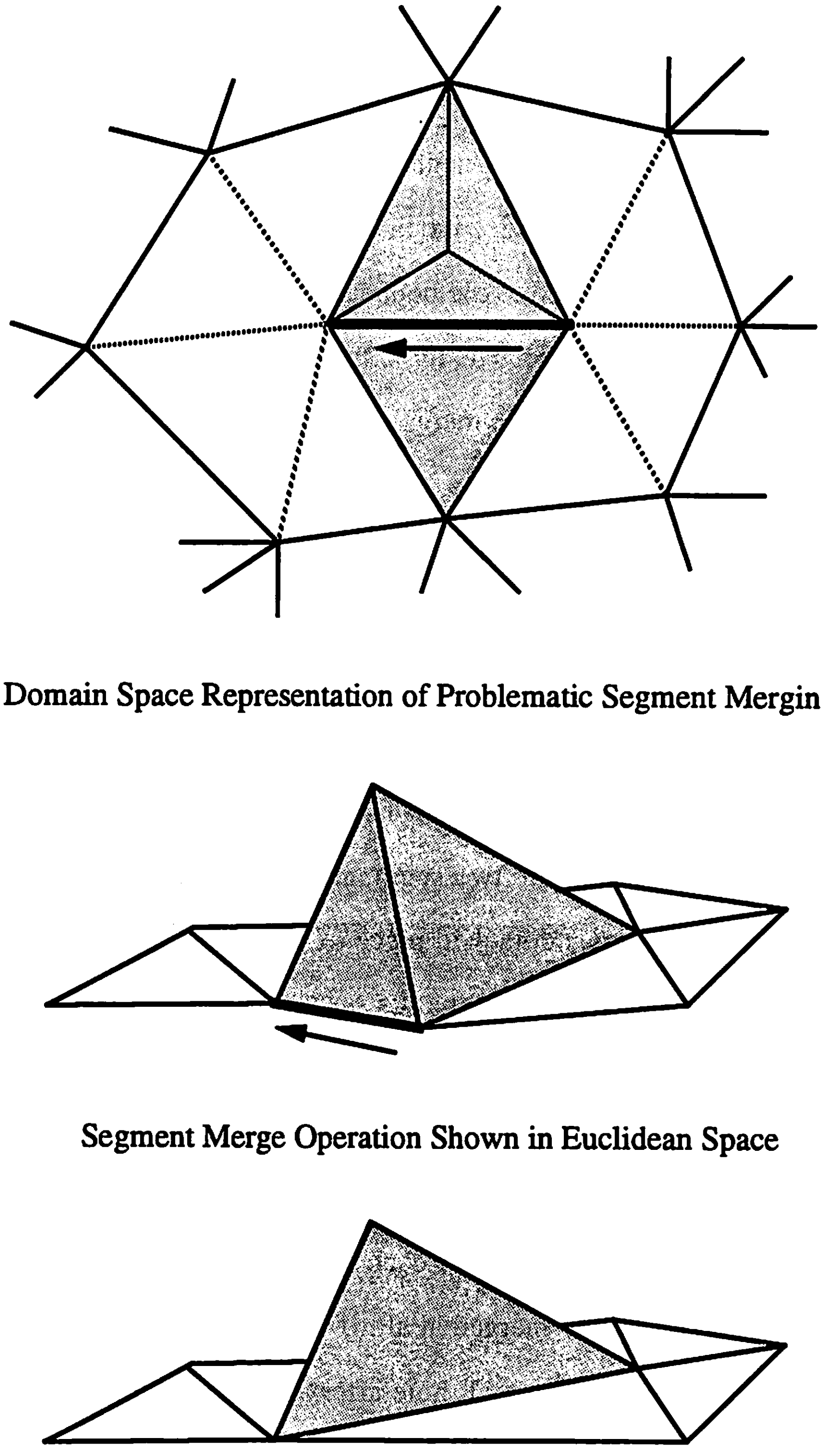


Figure 8) Segment Merging in Domain Space

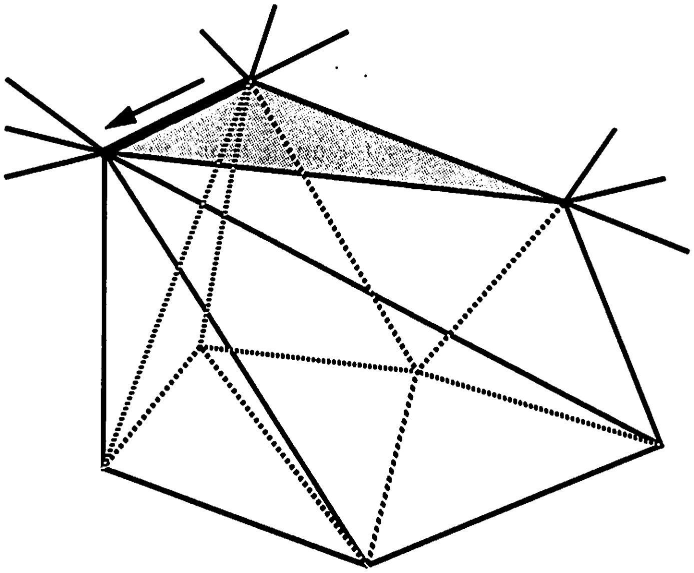


Domain Space Representation of Problematic Segment Merging

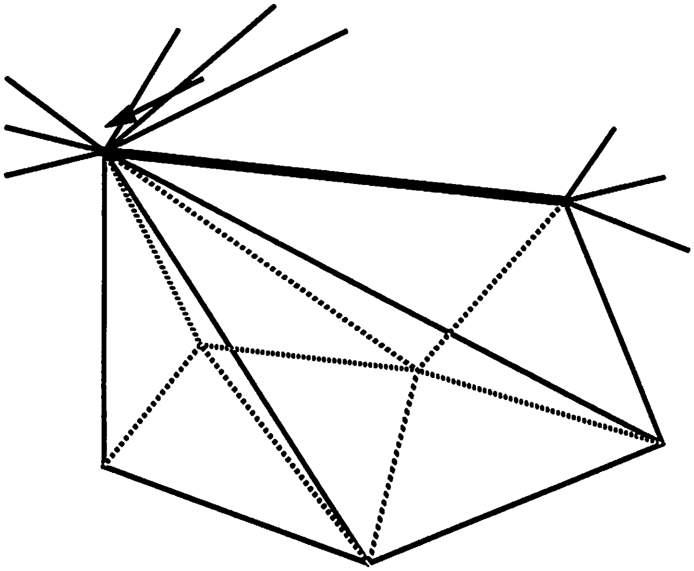
Segment Merge Operation Shown in Euclidean Space

Coplanar Triangles Formed as a Result of Segment Merging

Figure 9) Coplanar Triangles Formed by Segment Merging mesh. The particular example below, that of a pyramid collapsing to form two collinear triangles, was originally described by Kenny Toh [8] and Ed Scheckler[5]. Each implementation tried to overcome this difficulty by performing a recursive segment merge on one of the segments connected to the pyramid vertex. Both of these methods, however, were not effective in all cases, since they did not specifically cause the vertex point to be moved to the boundary of the collapse region. This method has been improved by the author. The three triangles are removed entirely and replaced with a single triangle before the merge operation takes place. Topological alterations, which have been implemented by J. Sefler in CUT-3D, are also important to consider during segment merging. They may occur if, in the last advancement step, two parts of the surface passed through one another to form a small hole that represents a topological alteration. They may also occur if the mesh will separate into two sections at the next time step. Figure 10 shows a topological alteration that occurs when a segment is merged. The grey triangle is not an actual triangle in the mesh. It represents the planar space between the two segments that will be collapsed. This particular merge operation causes a detached piece of mesh to form that encloses a volume. Proper handling of this case depends on the application, so either removal or detachment may be valid. If removal is valid, then the offending mesh is labeled and removed. If detachment is appropriate, the connectivity of the mesh is altered before the merge operation, so that the grey region in Figure 10 would represent two congruent triangles, one triangle for each separate mesh. It may also be the case that the segment should not be merged at all. If two surfaces are advancing towards one another, and have formed a small hole after a deloop operation, the hole should be allowed to grow, and neither detachment nor removal is valid. Either operation would undo the topological alteration just performed by the delooper. Topological alteration



Topological Alteration Before Merge



Topological Alteration After Merge

Figure 10) Merge Operation Creates a Topological Alteration consideration has not been implemented by the author, but has been implemented by K. Brakke [11] and J. Sefler.

6.3.2 Non-Planar Thin Triangle Removal

When removing thin triangles, the angle along the long segment of the thin triangle between the thin triangle and the neighboring triangle must be examined. The dark line in Figure 11 shows where the non-planarity test is performed. If the dot product of the unit surface normals on either side of the segment is less than 0.8, then the non-planar thin triangle removal routines are employed, otherwise normal planar removal of the thin triangle occurs using the techniques discussed before. When removing a non-planar thin triangle, it is necessary to move one of the points. The point is moved to the location where the altitude intersects the long side. The opposite triangle is subdivided into two triangles as shown in Figure 12. This point motion does not affect the accuracy significantly, since the motion of the point is always an amount less than the minimum altitude value. It is assumed that the inaccuracy incurred by moving the point such a small amount is less than the inaccuracy that would occur if the thin triangle is not removed. Other non-planar thin triangle cases occur as a result of the segment length conditions as seen in Figure 12. The first removal case shown is the standard removal case, where the point is moved to the original diagonal. The second case occurs if the motion of the point to the base of the altitude would leave a segment that is too small. This is handled by moving the point directly to the node that would form the other end of the small segment. If both nodes are candidates, then the segment that is smallest is chosen. The third case is when the new diagonal that is formed as a result of the flip and move operation is shorter than the minimum segment length. In this case both points on the diagonal are moved to the base of the altitude.

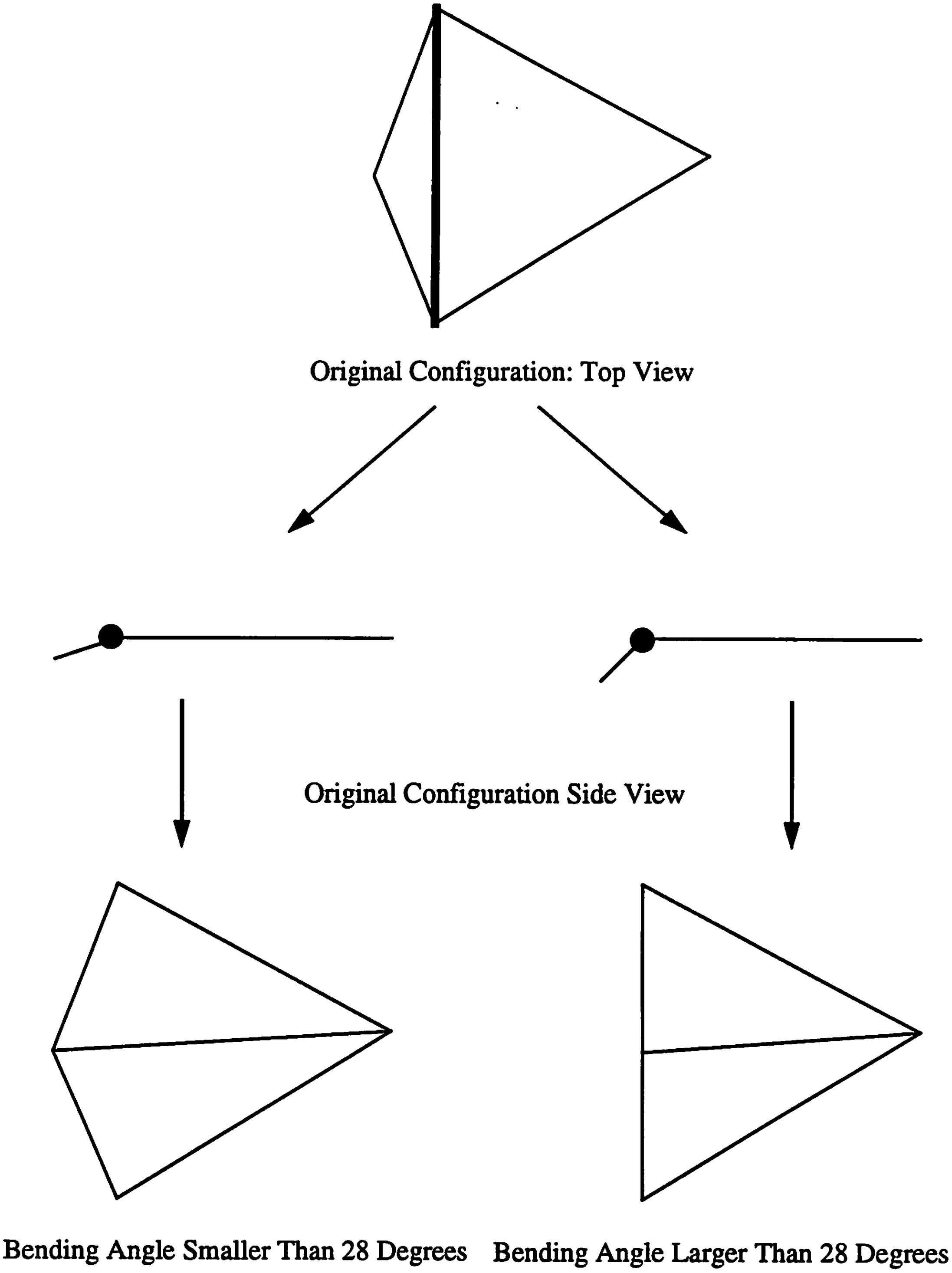
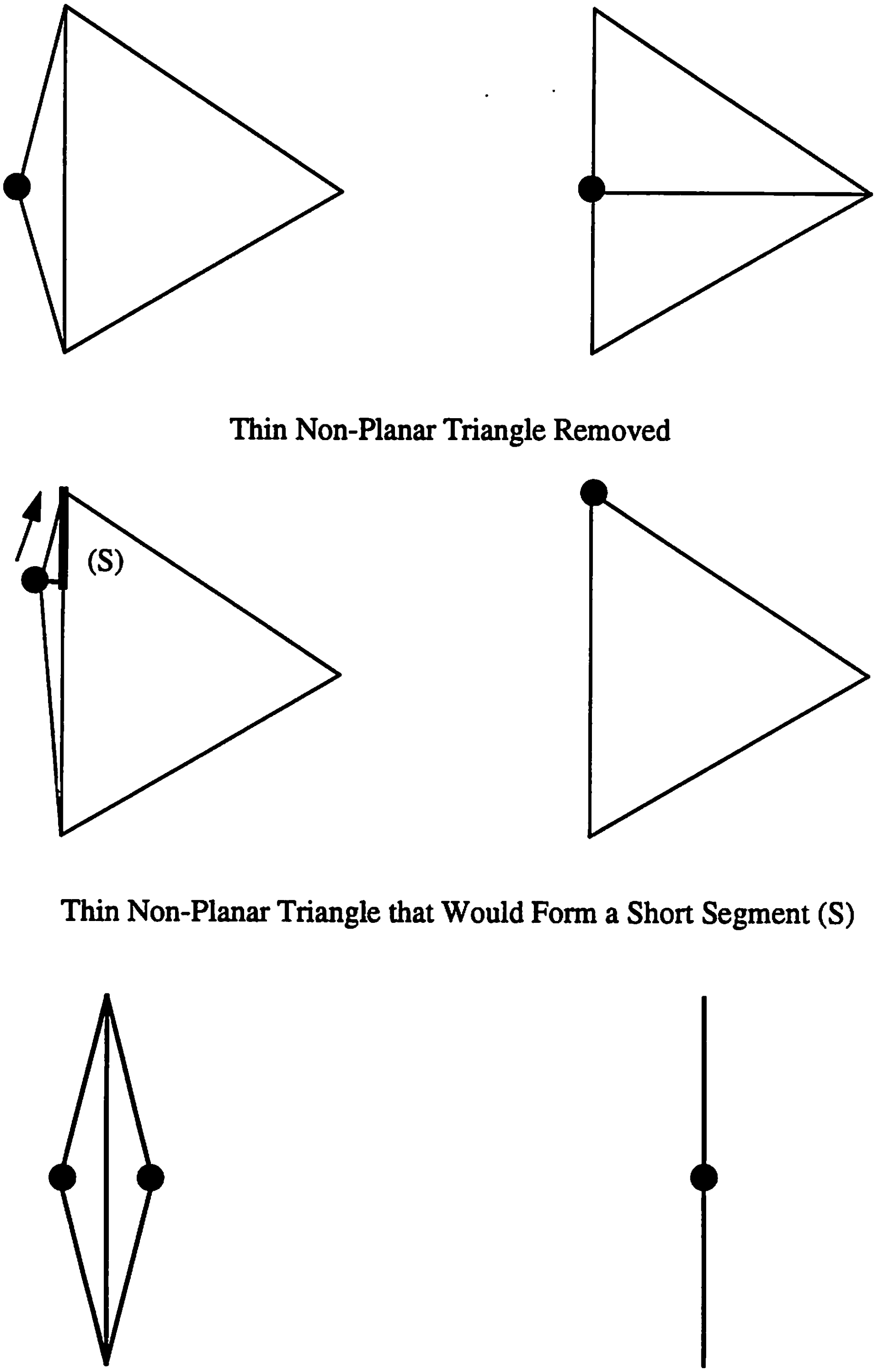

Original Configuration: Top View
Original Configuration Side View
Bending Angle Smaller Than 28 Degrees
Bending Angle Larger Than 28 Degrees

Figure 11) Non-Planar Thin Triangle Removal



Thin Non-Planar Triangle Removed

Thin Non-Planar Triangle that Would Form a Short Segment (S)

Two Thin Non-Planar Triangles that Do Not Form Short Segments

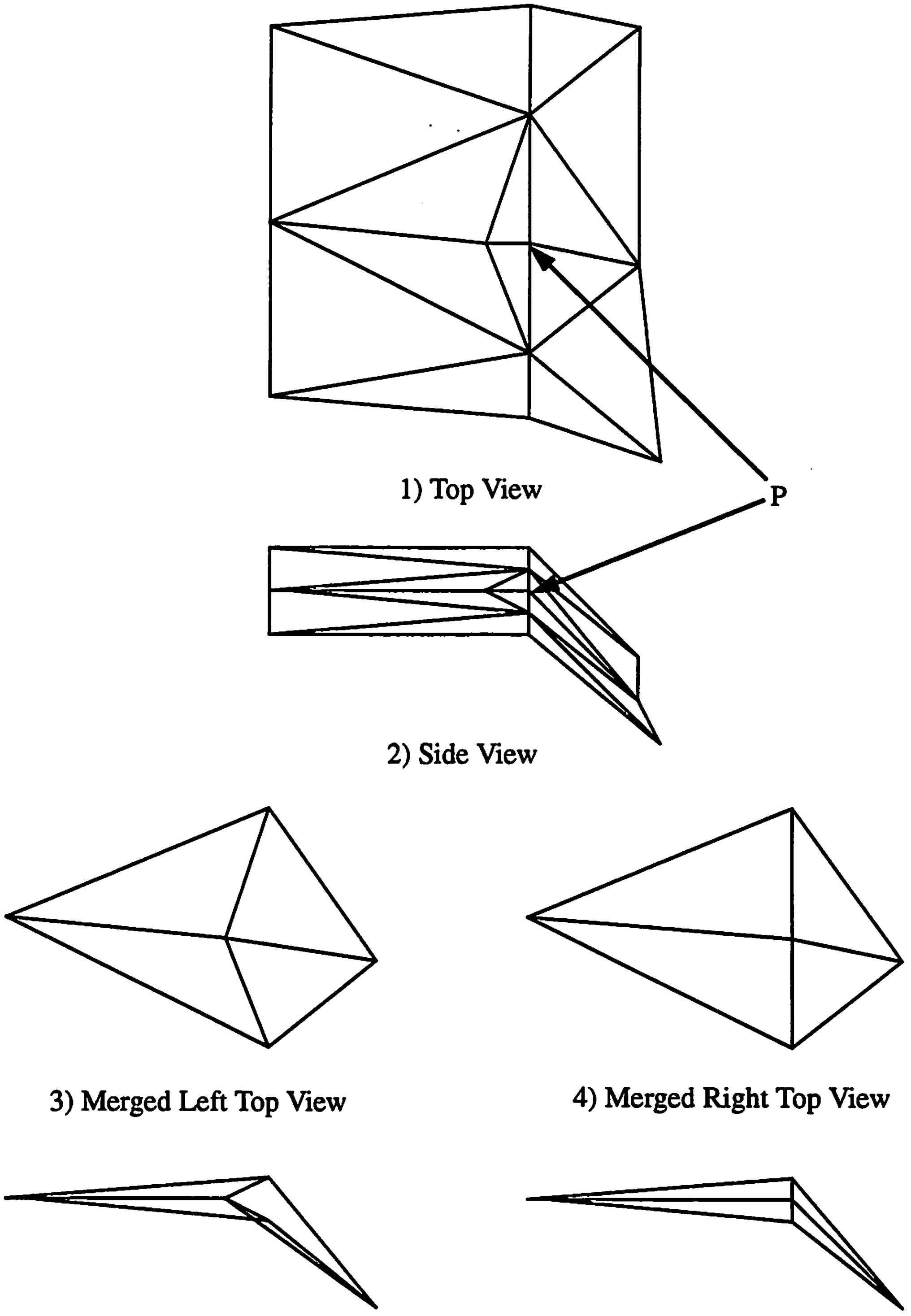
Figure 12) Non-Planar Thin Triangle Removal Cases

6.4 Reducing Non-Planarity

Another important consideration in mesh maintenance is the reduction of the non-planarity of the mesh. The non-planarity of a mesh is defined as the sum of the bending angles between triangle surface normals at each segment. This a different concept than surface curvature minimization as expressed by K. Brakke [11], since it is more important in etching simulation to sharpen ridges between planar regions than to smooth them out or 'crenulate' them. It is very desirable that a mesh that represent a topography or photolithography surface has a small non-planarity, since this seems to improve the accuracy of advancement and the appearance of the result. In addition, surface advancement in regions of homogeneous etch rate tends to dampen significant variations in planarity in real resists, so minimizing non-planarity is physically reasonable. Reducing non-planarity is not the primary consideration in the construction of a good mesh, however, since carrying the concept to its logical extreme would destroy the topography represented by the mesh in favor of a flat plane. Therefore, it is best if any operations to increase planarity operate in a local manner and avoid moving mesh points by any significant distance. The other three desirable properties of a good mesh (i.e. small segments, large segments, and thin triangles) must also be taken into account, but methods designed to enforce these conditions tend to improve the planarity of the mesh as well. Specific planarity improvement methods are not implemented in SAMPLE-3D at present.

Planarity improvement, in the case of small segment merging, can be enhanced by properly selecting which point of the segment will be removed. Figure 13 represents a segment merge taking place where two generally planar regions of the mesh meet at an angle. Two options exist for removing the small segment. Both are shown at the bottom of Figure 13. The removal of the right hand point P, as shown in section 5, results in a piece of mesh that is significantly non-planar, since there are large differences in the surface normals of the mesh triangles across five segments of the mesh. In the case of 2, there are only two segments that have significant changes in the surface normal. The non-planar thin triangle removal methods, that have been previously discussed, can also be used to flatten the mesh. Cross sections of how thin triangle removal can achieve this goal are shown in Figure 14. The figure in 1 represents a thin triangle and surrounding mesh from the domain space perspective. Removal of the thin triangle, assuming the triangle is non-planar, improves or maintains the planarity of the mesh in all three cases. These cases are typical of those encountered in SAMPLE-3D. The cross sections also demonstrate that small segment removal may improve planarity.

Planarity may also be improved by the removal of crenulation. Crenulation is defined as a condition where local alterations in the connectivity of triangles between the mesh points can result in an improvement of the planarity. An example of crenulation in an actual mesh is shown in Figure 15. The crenulations can be observed along the top ridges of the elbow structures and along the sides of the trenches. Crenulation removal represents a departure from minimum curvature surface advancement concepts. Minimum curvature surface simulation often tries to increase crenulation [11]. The manner in which crenulation tightens ridges is shown in Figure 16. Two planar meshes, which are colored with different shades of gray, meet at a ridge line that is represented by the solid line. The two triangles in the center of the picture have surface normals that point towards each other, thereby forming a 'crenulation' in the ridge line. To reduce the non-planarity of the mesh, it is clearly advantageous to perform a Delaunay like flip on the two triangles that form the crenulation. It is recommended that after advancement, each segment be evaluated to see if a flip of the segment will improve the planarity of the mesh over some preset factor. If this can be performed, then the segment is flipped. There may be more than



1) Top View

2) Side View

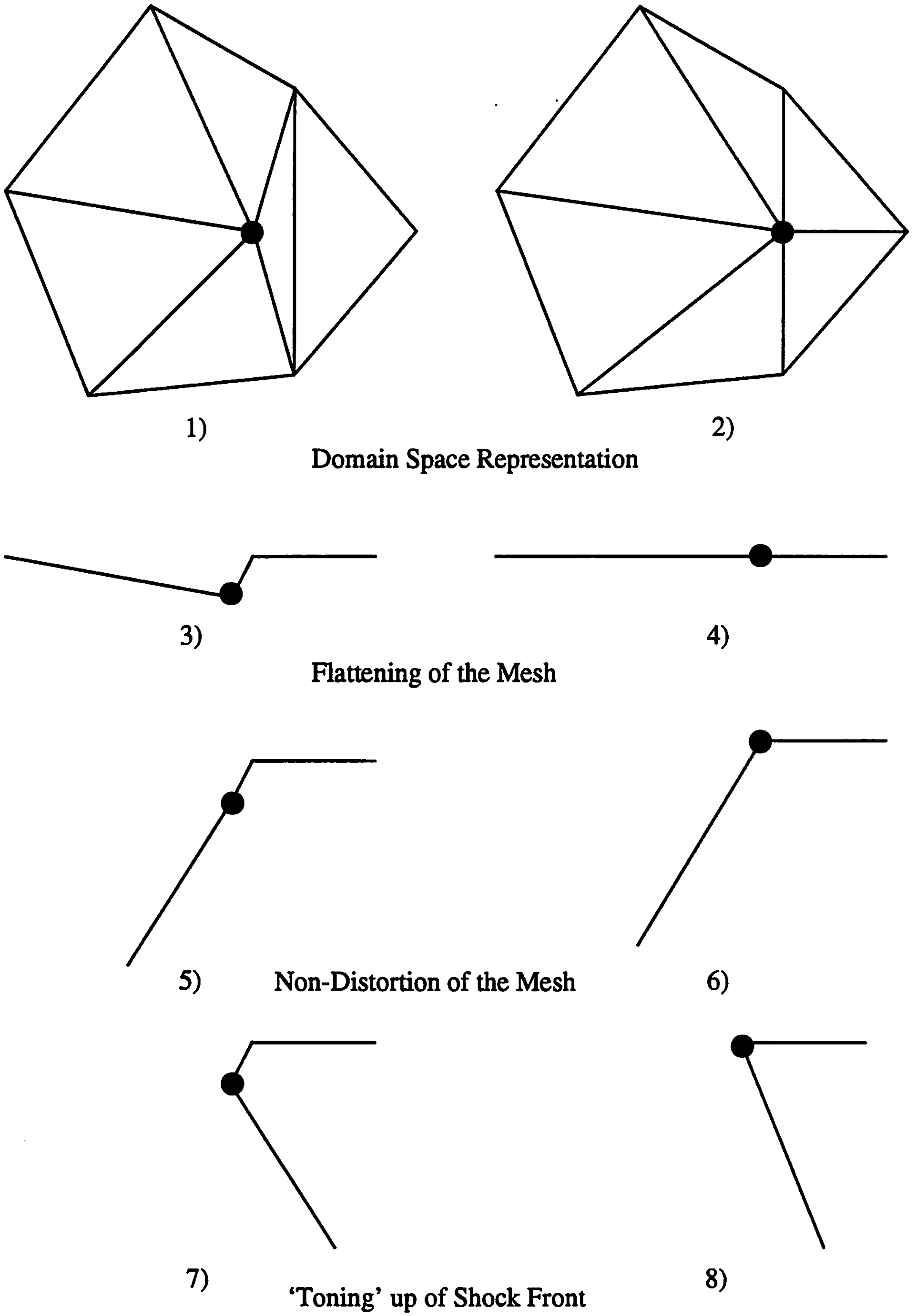
3) Merged Left Top View

4) Merged Right Top View

5) Merged Left Bottom View

6) Merged Right Bottom View

Figure 13) Segment Merging Cases



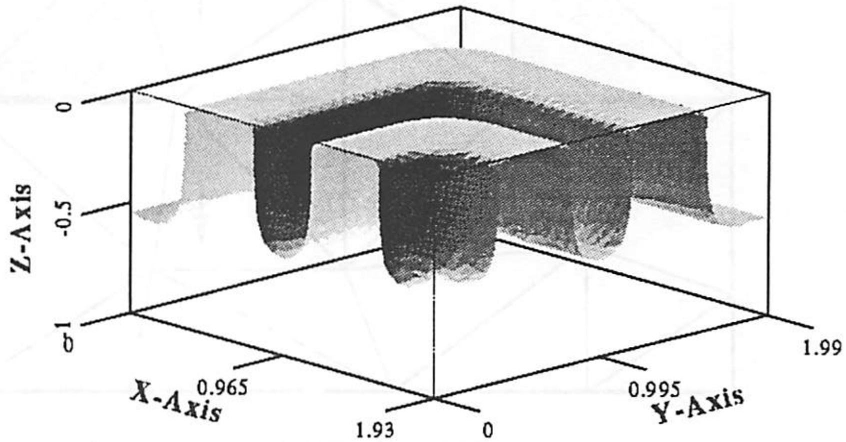
Domain Space Representation

Flattening of the Mesh

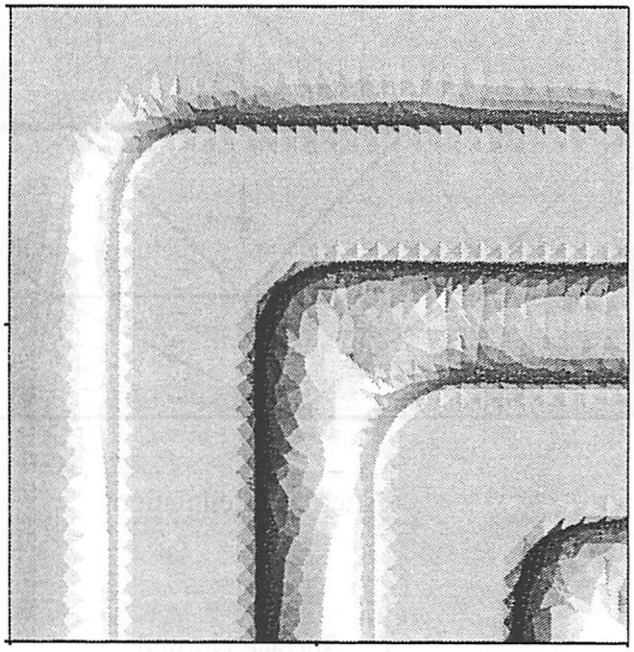
Non-Distortion of the Mesh

‘Toning’ up of Shock Front

Figure 14) Planarity Preservation via Non-Planar Thin Triangle Removal

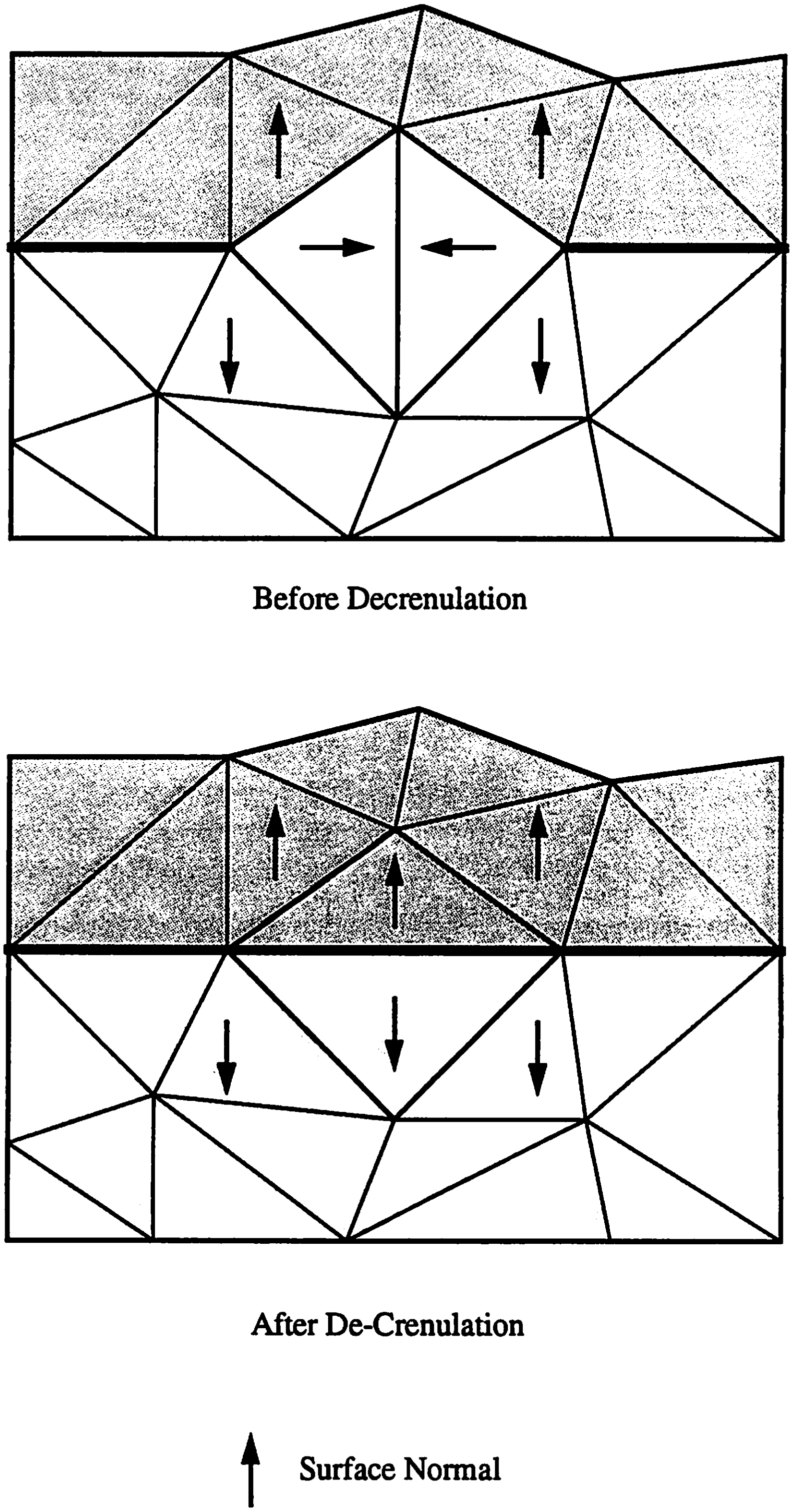


Side View



Top View

Figure 15) The Crenulation Example



Before Decrenulation

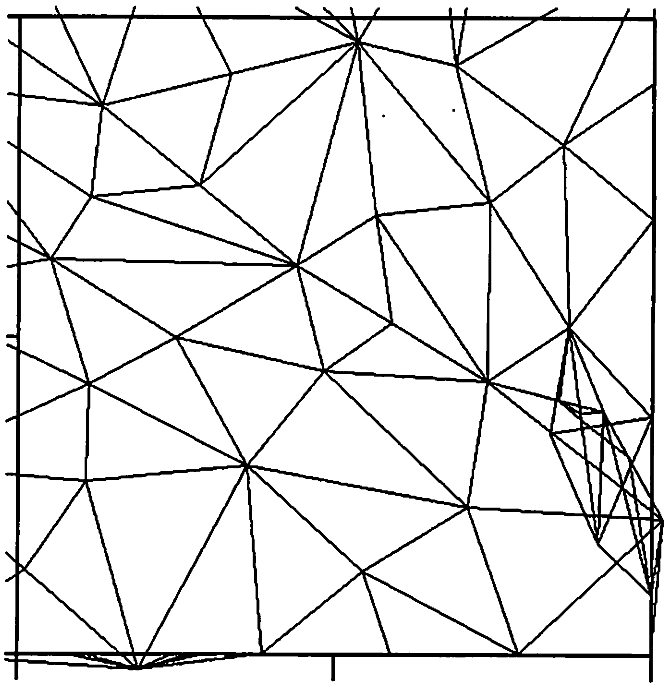
After De-Crenulation

Figure 16) The Decrenulation Operation

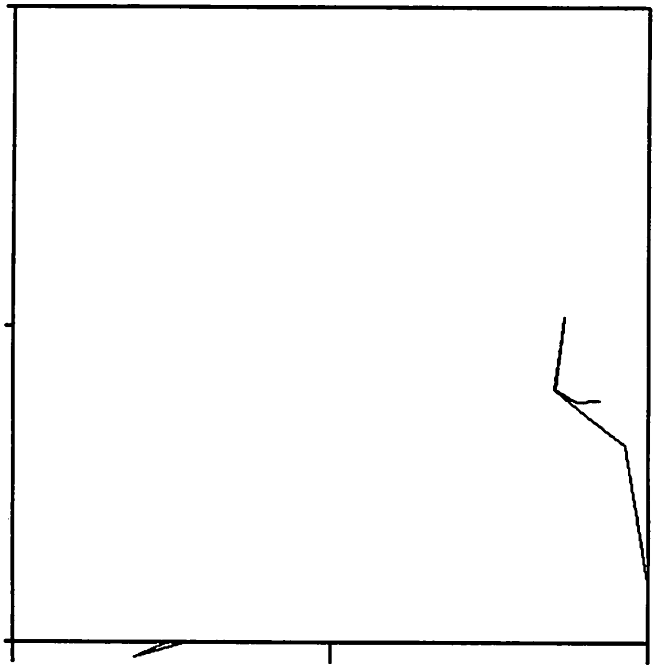
one method of flipping though, since the flip operation may cause small segments, large segments or thin triangles to form. This contingency probably can be dealt with satisfactorily by the addition of only a few special cases. Performing the decrelunation operation will have two advantages. First, the accuracy of mesh advancement in SAMPLE-3D, both for photolithography and other processes, such as plasma etching and ion-milling, is significantly improved, since the error that arises from point advancement routines is strongly related to the level of non-planarity in the mesh. Second, the possibility for formation of loops is significant without decrelunation, since increasing surface roughness allows more opportunities for points and triangles to pass through one another without being detected by small segment and thin triangle routines.

6.5 Ray Scattering

Ray trace advancement has been shown to accurately represent large scale shock fronts by employing resist deloop (Chapter 5). Resist deloop works best, however, on large shock fronts. Non-planarities in the surface can introduce loops that resist deloop is not well adapted for, because they are too small to efficiently remove. An example of such a loop is shown in Figure 17. This figure is a close up of the lower right hand corner of the bottom picture in Figure 15 after advancement with thin triangle removal turned off. This figure also represents what happens to the thin triangles in Figure 7 when advancement proceeds. Notice, also, the rays that are departing the simulation region due to the lack of thin triangle removal at the boundaries. The loop that has been formed in Figure 17 is rather complex and small. The resist delooper cannot remove it, however, since a part of each triangle is valid mesh. Thin triangle removal assists significantly, however, in keeping the loop from forming. Figure 18 shows the same region after the same advancement time with thin triangle removal turned on.

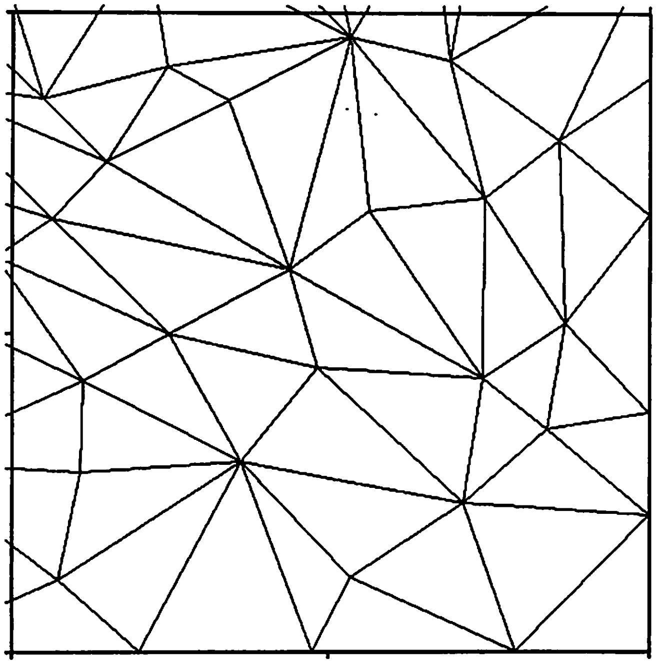


Small Loop Formation



Intersection Line

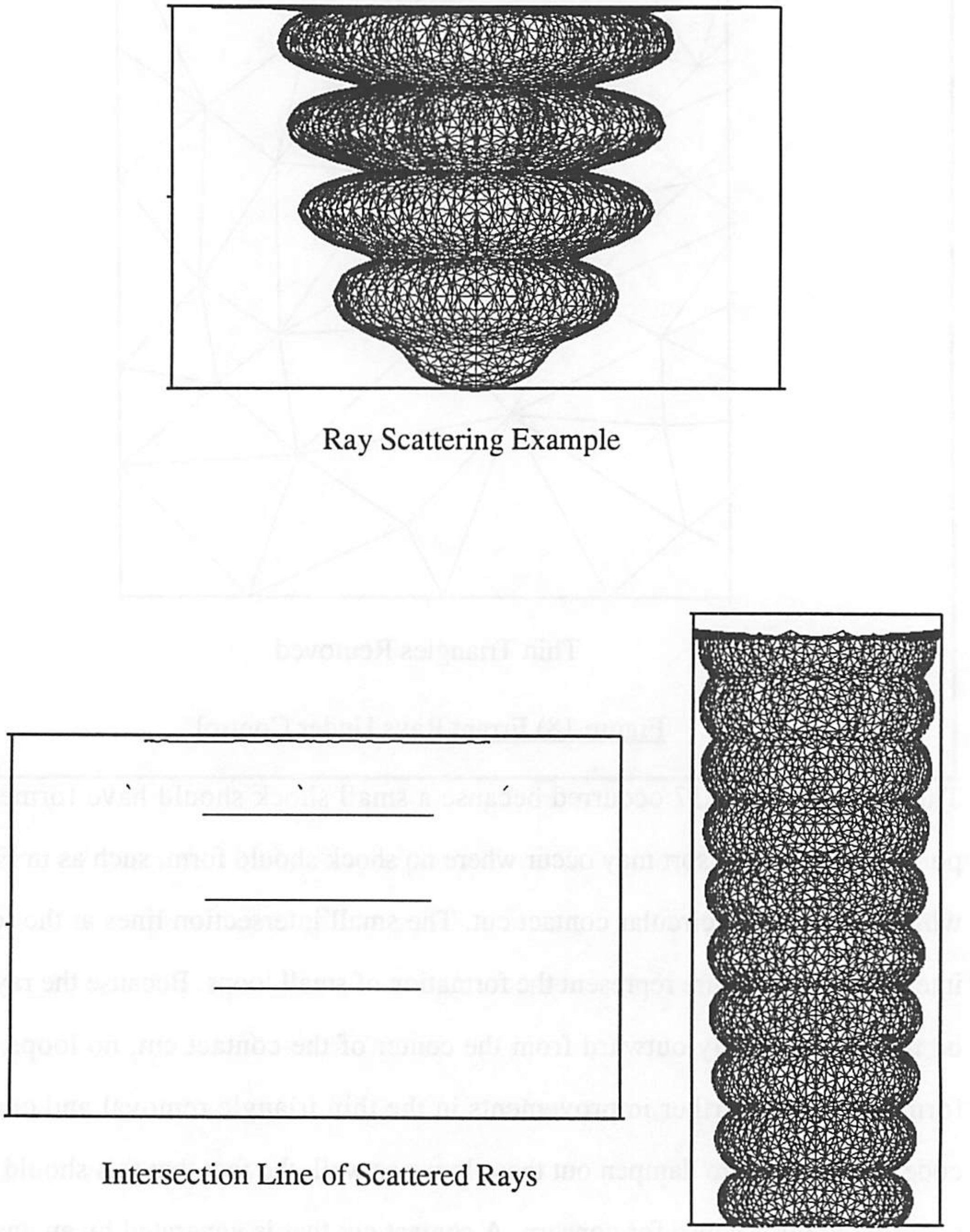
Figure 17) Errant Rays



Thin Triangles Removed

Figure 18) Errant Rays Under Control

The loop in Figure 17 occurred because a small shock should have formed at that point. Loops of this sort may occur where no shock should form, such as in Figure 19, which represents a circular contact cut. The small intersection lines at the top of the intersection line figure represent the formation of small loops. Because the rays should be traveling radially outward from the center of the contact cut, no loops ought to form. Although further improvements in the thin triangle removal and crenulation code should serve to dampen out these loops as well, the fact that this should not have occurred at all is cause for concern. A contact cut that is generated by an analytically computed etch rate and gradient does not exhibit any ray scattering behavior. This suggests that the scattering may be occurring as a result of errors in the etch rate interpolation function or the generation of the gradient from the etch rate data. This phenomena demands further investigation.



Ray Scattering Example

Intersection Line of Scattered Rays

Unscattered 10 Standing Wave Example

Figure 19) Ray Scattering

6.6 Conclusions

Four conditions for a proper mesh in SAMPLE-3D have been described. The importance of maintaining each condition has been explained. The conditions are small segment removal, large segment subdivision, thin triangle removal, and planarity improving operations. While the first three have been implemented in SAMPLE-3D and have contributed to its robustness, a full mesh maintenance method that employs planarity methods still remains to be designed. It must also be noted that the maintenance of planarity as the surface advances is also contingent on the advancement algorithm. In photolithography, the planarity of the surface should be reinforced by the advancement algorithm, since regions with high frequency variations and a reasonably homogeneous etch rate tend to become flatter according to the mathematics. An automatic dampening of the singularities ought to occur. This does not occur in ray-trace at present due to the lack of implemented crenulation removal techniques, and due to the phenomena of ray scattering.

Reference for Chapter 6

[1] E. Barouch, B. Bradie, H. Fowler and S. Babu, "Three-Dimensional Modeling of Optical Lithography for Positive Photoresists," *Interface '89: Proceedings of KTI Microelectronics Seminar*, pp. 123-136, Nov. 1989.

[2] M. D. Giles, D. S. Boning, G. R. Chin, W. C. Dietrich Jr., and others, "Semiconductor Wafer Representation for TCAD," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 13, pp. 82-95, Jan. 1994

[3] K. Nabors, J. White, "FASTCAP: A Multipole Accelerated 3-D Capacitance Extraction Program" IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 10, pp. 1447-59, Nov. 1991

[4] J. Sethian, "An Analysis of Flame Propagation", *Ph.D. Dissertation*, University of California at Berkeley, 1982.

[5] E.W. Scheckler, *Ph.D. Dissertation*, University of California, Berkeley, Nov. 1991.

[6] H.E. Trease, M.S. Sahota, "Massively Parallel Hydrodynamics on Unstructured Grids" Proceedings of the 1993 Simulation Multiconference on the High Performance Computing Symposium, pp. 123-6, (Arlington, VA), March 29 - April 1, 1993.

[7] F.P. Preparata and M.I. Shamos, *Computational Geometry*, Springer Verlag, New York, 1985.

[8] K.K.H. Toh, *Ph.D. Dissertation*, University of California, Berkeley, Dec. 1990.

[9] R. LaBarre, *Computational Geometry Techniques for 2D and 3D Unstructured Mesh Generation with Application to the Solution of Divergence Form Partial Difference Equations,* Ph.D. Dissertation, University of Connecticut, 1992

[10] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, W. Stuetzle, "Mesh Optimization", *SIGGRAPH '93 Proceedings*, pp. 19-26, Anaheim, CA, August 1-6, 1993

[11] K. Brakke, "The Surface Evolver" Experimental Mathematics, vol. 1, pp. 141-165, 1992.

[12] J. Painter, J. Marshall, "Massively Parallel Hydrodynamics on Unstructured Grids" *Proceedings of the Next Free-Lagrange Conference*, pp. 123-6, (Moran, WY), June 3-7, 1990.

Chapter 7 Comparison of Methods

7.1 Introduction

In this chapter, the three techniques for simulating photolithography dissolution will be directly compared. The performance improvements of ray interpolation and loop removal have been included in the ray-trace method for the comparison. The iterative level-set technique has also been implemented on the benchmark. The merits of each approach will be discussed. Recommendations that are based on these comparisons for further research into simulator development will be discussed.

7.2 General Criterion for Comparison

To create any product suited for a specific task, a basis for judgements must be set forth so that the attributes of the final product can be appropriately selected. The judgement system that is appropriate for designing physical products is, of course, also suitable for designing algorithms and programs. These criteria are the product's ability to perform the task, efficiency of resource consumption while performing the task, and the capital expenditure of other resources required for its creation, which includes human labor costs. For a computer program that performs simulation of physical systems that are represented by continuous variables, these criteria take on more specific forms. One crucial measure of the ability of a simulators perform its task is its accuracy. Resource consumption is measured as the memory required to execute the program and the number of CPU cycles necessary to complete it. Finally, the capital expenditure, since no unusual hardware is necessary, is purely in terms of programmer-hours. These programmer hours will also serve as an estimate of the difficulty of program maintenance.

7.3 Specific Comparison

The three programs that were described in Chapter 4 (ray-trace, level-sets and the cell method) were run using a benchmark example. This example is a square contact cut that has standing waves. Because the size of the square contact is approximately the length of the wavelength, the intensity contours of the contact are nearly circular. This means that the rate function that will be generated will also have radial symmetry. Therefore, the shape that is will be formed by the simulators ought to be radially symmetric. The parameters that were employed in the construction of this example are shown in Table 1. The rate file that served as input to each of the dissolution

Table 1: Test Case Exposure and Development Parameters

Parameter	Value
Contact Edge Size	0.45 μm
NA	0.7
Partial Coherence	0.5
Wavelength	0.365 μm
Focal Plane	0.0 μm
Dill Parameter A	0.74
Dill Parameter B	0.2
Dill Parameter C	0.012
Refractive Index	1.68
Thickness	0.5 μm
Dose	180 mJ/cm^2
Kim Parameter R1	0.062
Kim Parameter R2	0.0001
Kim Parameter R3	8.5
Advancement Time	13 seconds

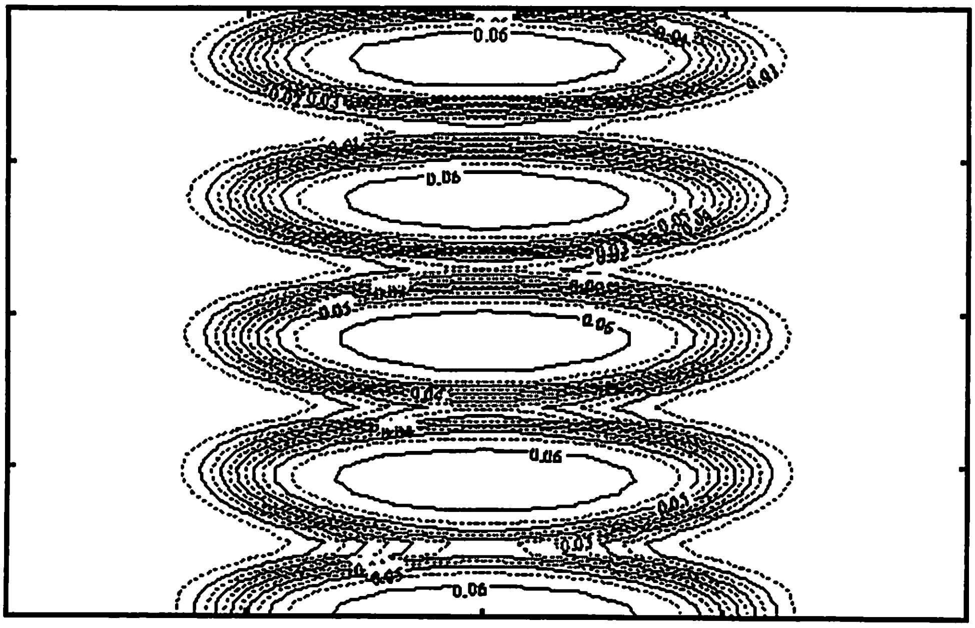


Figure 1) Test Case Rate Contour

simulators was generated by High-NA SPLAT [1]. High-NA SPLAT was developed by M. Yeung and D. Lee.

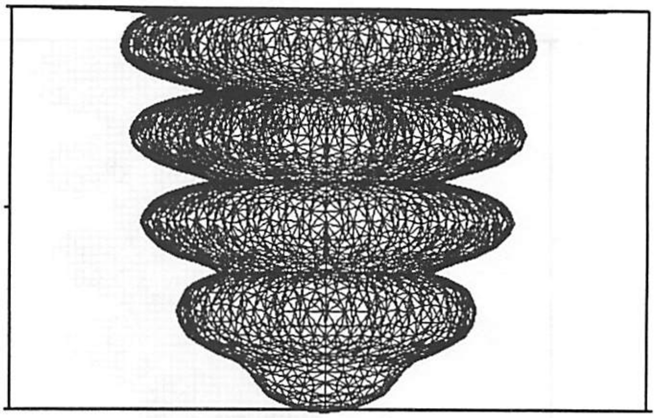
A contour plot of a cross section that contains the central axis of the contact cut is shown in Figure 1. The rate function, in three dimensions, is nearly radially symmetric about a vertical line constructed in the middle of Figure 1. This example demonstrates many of the circumstances that are encountered in real photolithography situations. The most significant characteristics in this example are the standing waves. The ratio between the maximum etch rate and the minimum etch rate along the vertical center line is approximately 50:1. The initial surface is the top plane of the simulation region. The surface was advanced for a period of 13 seconds. The ray-trace method initiated simulation with an initial grid of points in a 41x41 face centered configuration. The ideal segment length remained fixed during execution at one fortieth of the size of the square that represented the initial surface. The cell method and the level-set method employed an 80x80 grid of cells at regular intervals to represent the different horizontal planes. The simulation was performed with various numbers of cells in the vertical direction so that the effect of different grid sizes on accuracy could be measured. The levels of discretization were 50, 100 and 200 cells. The results of the simulations are summarized in Table 2. Specific details of the comparisons are given in the rest of this section.

Table 2: Lithography Development Methods

	CM	RT	Ad
Accuracy	Anisotropic	Good	Low Order
Time	40 Min	20 Min	>64 Hours
Memory	10 MB	10 MB	>80 MB
Coding Time	2 Months	2 Years	2 Months

7.3.1 Time Consumption

In order to fairly determine the amount of CPU time that each program requires, it is necessary to demand a similar level of accuracy from each program. It is possible to get low CPU times that do not represent the real advancement time if the required accuracy is reduced. Accuracy is determined by the distance of the bottom of the surface from the bottom of the simulation region. The value of 13 seconds was chosen, since the distance of the resist surface from the bottom of the simulation region is very close to 0 for the ray-trace advancement method at an evolution time of 13 seconds. This near intersection allows for easier comparisons of varying development rates between methods, since deviation from this condition is easy to test. It will be assumed that the ray-trace advancement method, as shown in Figure 2, gives an answer that is sufficiently accurate. This is a reasonable assessment, not only because the ray-trace method contains adaptive methods for advancement in critical regions that were developed by K. Toh [2], but the other methods also tend towards the ray-trace solution as the accuracy is improved. Accuracy improves for the cell and level-set

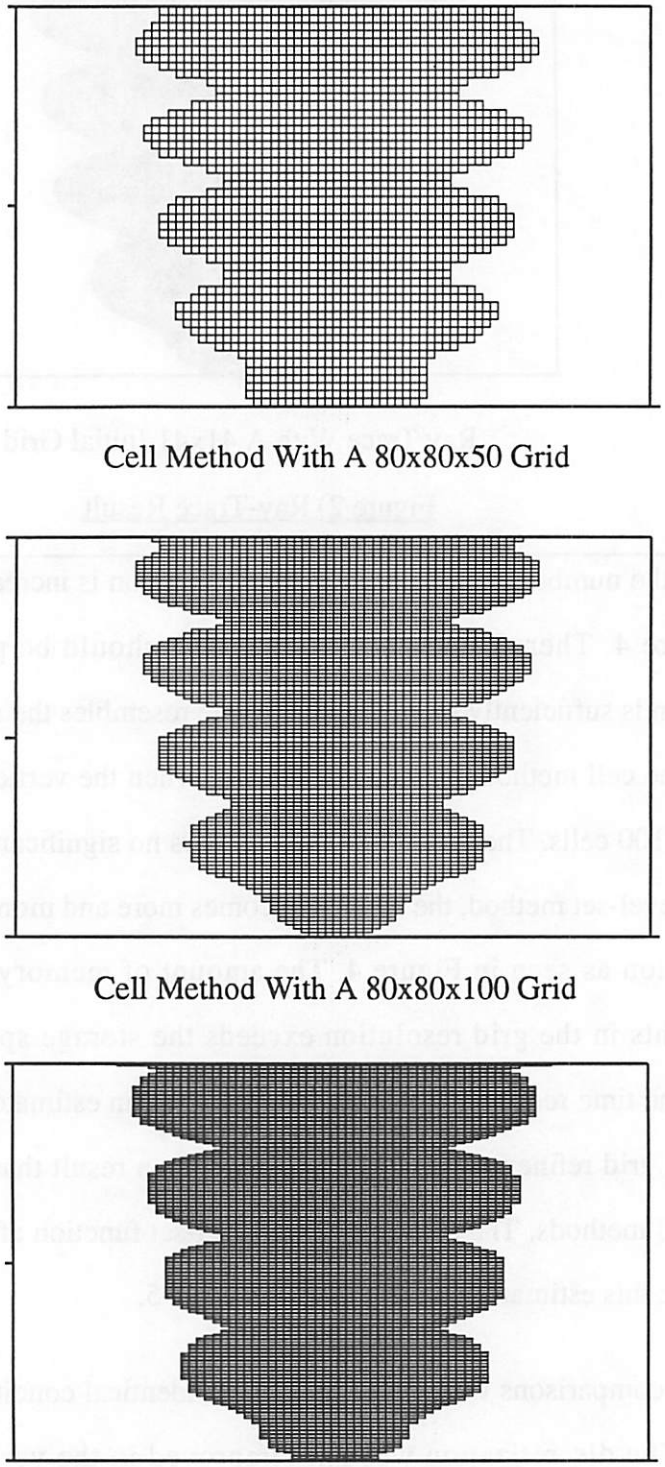


Ray Trace With A 41x41 Initial Grid

Figure 2) Ray-Trace Result

methods as the number of cells in the vertical direction is increased, as seen in Figure 3 and Figure 4. Therefore, a time comparison should be performed when the discretization is sufficiently small that the profile resembles the ray-trace method. This occurs for the cell method, as seen in Figure 3, when the vertical discretization is on the order of 100 cells. The use of 200 cells renders no significant improvement. In the case of the level-set method, the method becomes more and more accurate with further grid resolution as seen in Figure 4. The amount of memory required for further improvements in the grid resolution exceeds the storage space of the computer. Therefore, the time required, as shown in Table 2, is an estimate based on the amount of estimated grid refinement necessary to generate a result that is similar to the ray-trace and cell methods. The contours of the level-set function after advancement were used to make this estimate and are shown in Figure 5.

All time comparisons were performed under identical conditions on a DECstation 5000/240. The discretization was only improved in the vertical direction, since 80x80x200 cells was the maximum amount of available memory on the machine. It was also expected that the most error would occur in this direction as a result of the

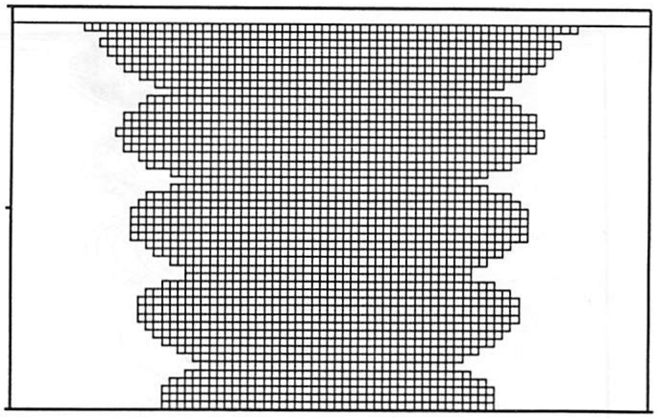


Cell Method With A 80x80x50 Grid

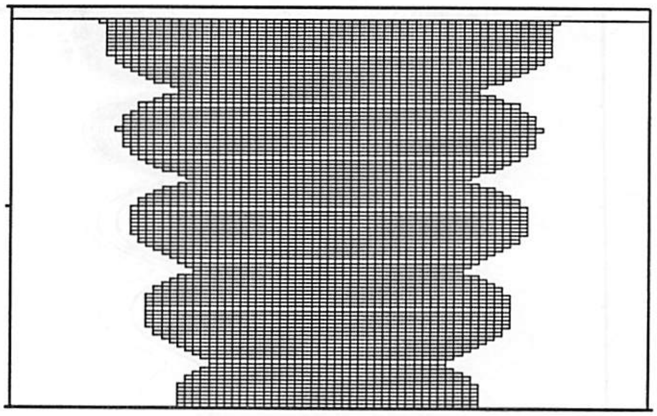
Cell Method With A 80x80x100 Grid

Cell Method With A 80x80x200 Grid

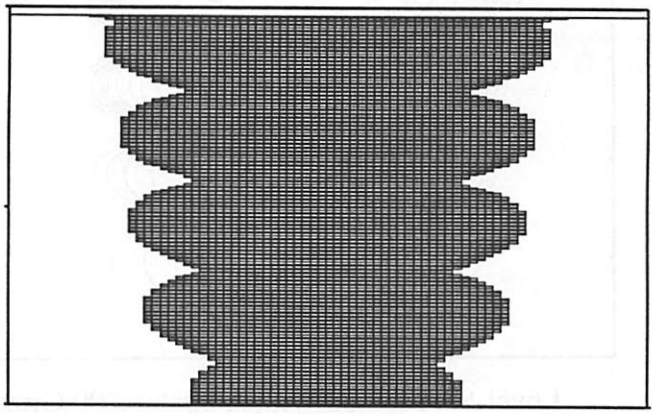
Figure 3) Cell Method Results



Level-Set Method With A 80x80x50 Grid

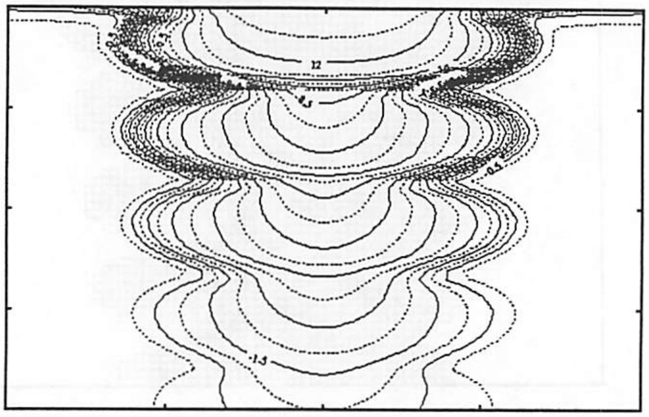


Level-Set Method With A 80x80x100 Grid

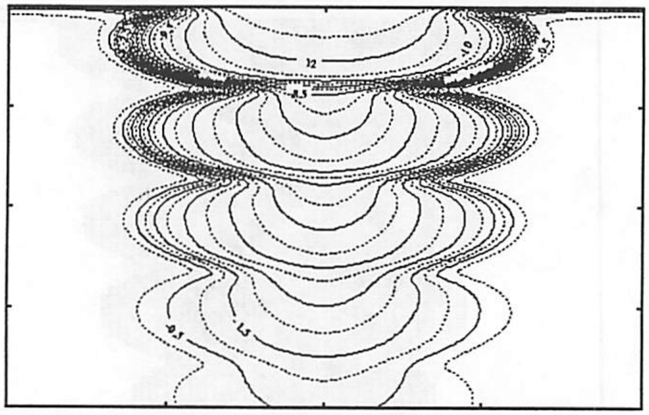


Level-Set Method With A 80x80x200 Grid

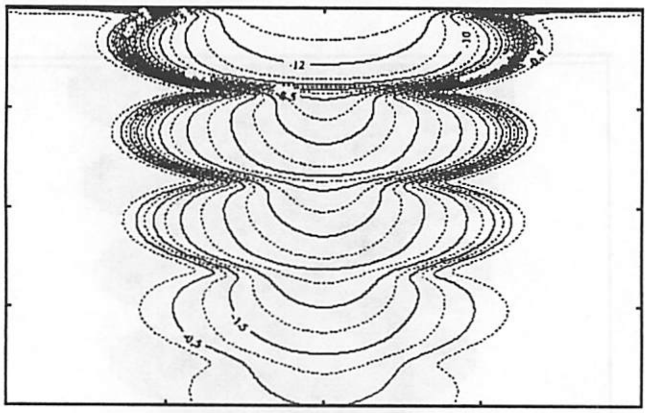
Figure 4) Level-Set Method Results



Level-Set Method With A 80x80x50 Grid



Level-Set Method With A 80x80x100 Grid



Level-Set Method With A 80x80x200 Grid

Figure 5) Level-Set Method Contour Plots high second derivative of the etch rate in the vertical direction. A high second derivative in the advancement is already known to be a significant source of error in existing fluid mechanical level-set methods [9]. The ray-trace method required an advancement time of 20 minutes to develop the profile in Figure 2 including deloop steps. The cell method requires 20 minutes to develop the first profile in Figure 3, but this profile is not as accurate as the ray-trace method. The second profile in Figure 3 is sufficiently accurate, and requires 40 minutes for advancement. The doubling of the advancement time is directly related to the doubling of the number of cells in this example. The third profile results in no improvement in accuracy over the second profile and required 80 minutes for advancement. It is clear that a linear increase in accuracy for the cell method in one direction generates a linear increase in the amount of time consumed. This is to be expected, since the number of cells that must be removed to compute the simulation result has increased linearly as well. The removal of a cell is a constant time operation. Finally, the level-set scheme required 15 minutes to generate the top figure in Figure 4, but it required 1 hour for the second result and 4 hours for the third result. This represents a quadratic increase in time as accuracy in the vertical direction increases. The reason for this increase is twofold. First, it is necessary to double the number of cells with each doubling of grid resolution, but it is also necessary to double the number of time steps taken, since there is a limitation on the size of the timestep that is related to the shortest dimension of each cell. Because Figure 5 suggests that the level-set method requires at least another two improvements in grid resolution before the method approaches the accuracy of ray-trace and cells, the quadratic behavior suggests that 64 hours are required to generate the desired result. This is the basis of the computation time estimate in Table 2 for the level-set method.

7.3.2 Memory Consumption

Memory consumption statistics are determined in Table 2 by the same method as the time consumption statistics. The methods are normalized for accuracy first, and then the memory consumption is determined. The ray-trace method consumes 10 MB for both the final mesh and the associated octtree. Approximately 25000 triangles appear in the final figure, giving 400 bytes per triangle. This is a reasonable number, considering that each triangle also requires segment, node and octtree information. The cell method, for the 80x80x100 cell configuration, consumes approximately 10 MB, or about 20 bytes per cell. The level-set method, for the assumed 80x80x800 configuration, would employ about 5 million cells at a present 40 bytes per cell. 40 bytes are necessary for each location in the level-set scheme, since there are 3 floating point numbers that are stored for the level-set method, as opposed to the cell method, which employs 1 floating point number and an address that points to specific information regarding surface cells.

7.3.3 Coding Difficulty

Estimates for coding time are for programming a simple commercial version of the routines by one person, provided that the methods being applied are well known. The estimates are based on the assumption that the theory is well understood. Writing the code for the cell and the level-set methods is quite simple. No delooping or mesh maintenance routines are required for the methods. Since boundary conditions are reflective, these are also quite simple to implement. Therefore, since these methods are relatively straightforward, a time of 2 months has been assigned. Ray-trace requires significantly more coding support to be implemented. A delooper and a mesh maintenance method, as given in Chapters 5 and 6, must be implemented. The advancement method is as sophisticated as the cell or level-set schemes as seen in

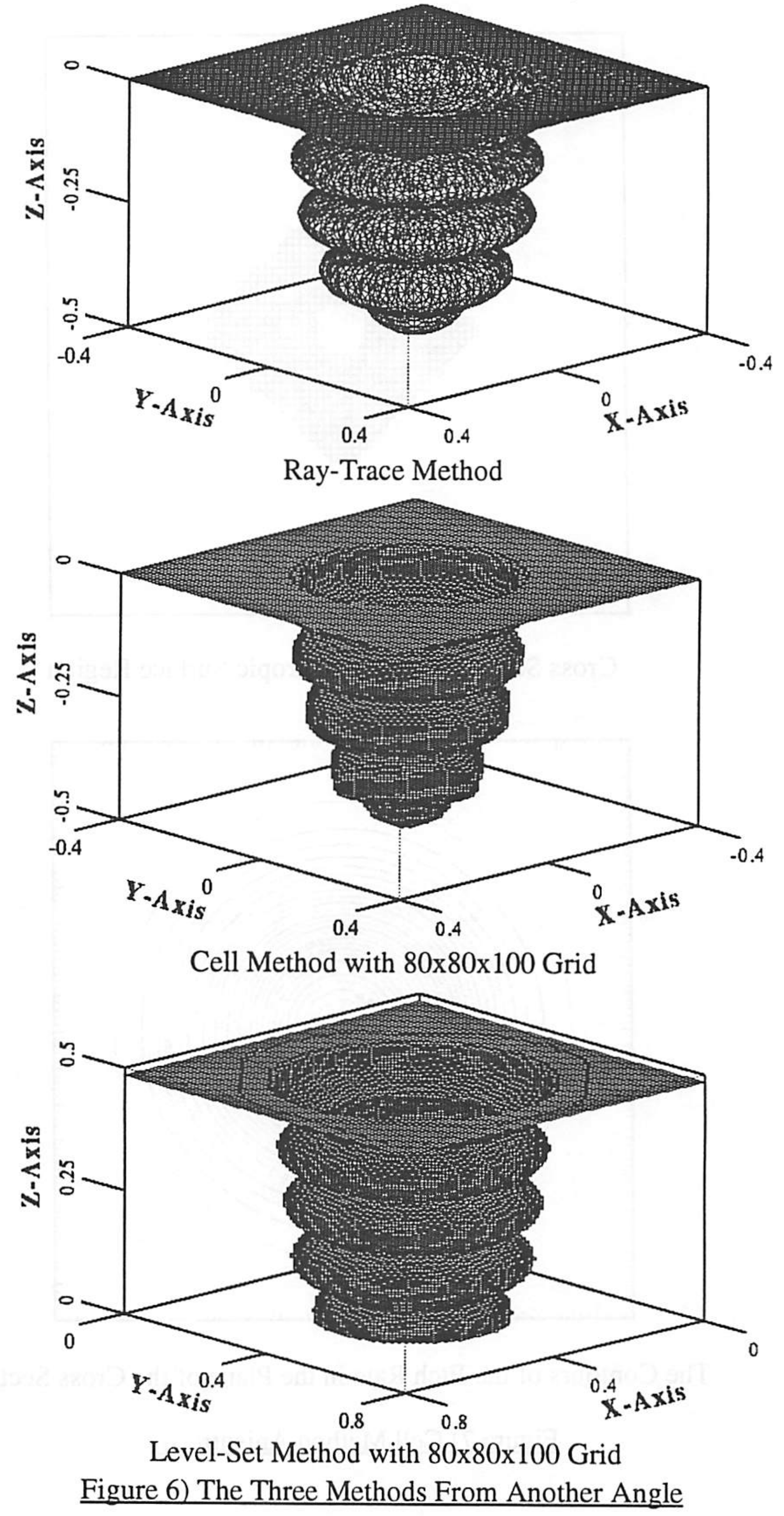
Chapter 4. Finally, boundary conditions must be handled in a more detailed manner as shown in Chapters 3 and 5. For these reasons, the estimate is 2 years.

7.3.4 Notes on Accuracy

The standing waves in the test case, as noted earlier, are an excellent test for accuracy. At the null of each wave, where the etch rate is small, the etch rate decreases rapidly as the null is approached from either the top or the bottom. The second derivative of the etch rate in the vertical direction becomes extremely large. This derivative is one of the two main sources of error that may be encountered by a first order advancement method. The ray-trace and cell methods gain their accuracy from specific properties of their advancement methods. Both methods are known as 'Lagrangian' methods [5], as opposed to the level-set technique, which is an 'Eulerian' method [5]. Lagrangian methods are known to be more resistant to variations in the etch rate than Eulerian methods. Therefore, standing wave nulls do not affect the accuracy of ray-trace or cells as much as the level-set method. The ray-trace method also has adaptive time step control, which allows it to take many small steps through the standing wave nulls, and thereby resolve the features more accurately. The recursive adaptive advancement scheme also assists in this function, since it effectively forces each ray to independently take smaller time steps in areas where significant error may be accumulating. Therefore, these methods can traverse strong standing wave nulls accurately. The level-set method has no such defense to protect itself against the effects of standing wave nulls. As seen in Figure 4, the grid must be refined to resolve the advancement of the surface across the null accurately. If a level-set method could be developed that only refines the grid in troublesome areas, the error of the method might be reduced without increasing the memory and CPU resource consumption as much. Adaptive gridding been shown to increase the accuracy of the cell method [11].

The cell method has demonstrated an additional accuracy difficulty that is not present in the other two methods. Because the cell method was not developed with an accurate method of computing the surface normal at the surface, and implemented with a method of removing cells that can make effective use of an accurate surface normal computation, 'facets' appear. As seen in Chapter 3, faceting can be removed through the use of integer cell labeling combined with Huygen spheres [2][11], although employing this technique has been shown to significantly increase execution time [2]. Faceting may also be removed by using more sophisticated methods of interpolating the surface [8]. The cell method in the comparison appears accurate in Figure 3 because the advancement algorithm has been optimized to advance properly in all three coordinate directions of the grid. Since the surface has been optimized to advance properly in the direction of the z-axis, this explains the observed accuracy. When the surface is viewed at a 45 degree angle, as in Figure 6, the grid dependence of the cell method becomes apparent. Figure 6 shows that the bottom of the cell method surface is not circular as in the ray-trace and level-set cases. Instead it takes the shape of a square. The top half of Figure 7 shows the cross section of the bottom part of the surface, and the bottom half of Figure 7 shows the contour of the rate function in the same location. It is clear from the rate function that the cross section of the surface ought to be circular, but instead it is not. Anisotropy is not an issue for level-sets, since the level-set technique has a good method of computing the surface normal that does not show grid dependence. Ray-trace contains no grid dependence, because the ray vectors can represent any surface normal at full floating point accuracy.

Another possible source of error may arise from the curvature of the surface. This contribution is small in these examples, since the grid has been sufficiently discretized to resolve the surface curvature that does arise. The ray-trace method has 6 grid points per standing wave null. The cell and level-set methods contain 12, 24 and 48 grid

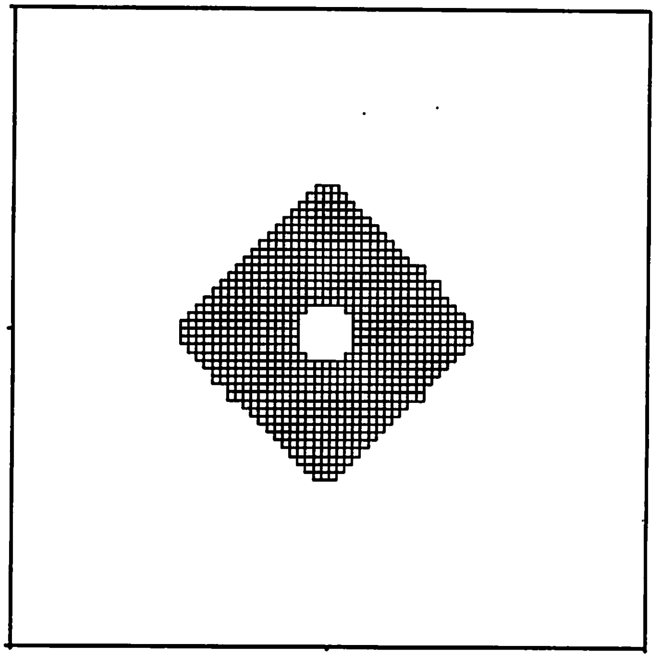


Ray-Trace Method

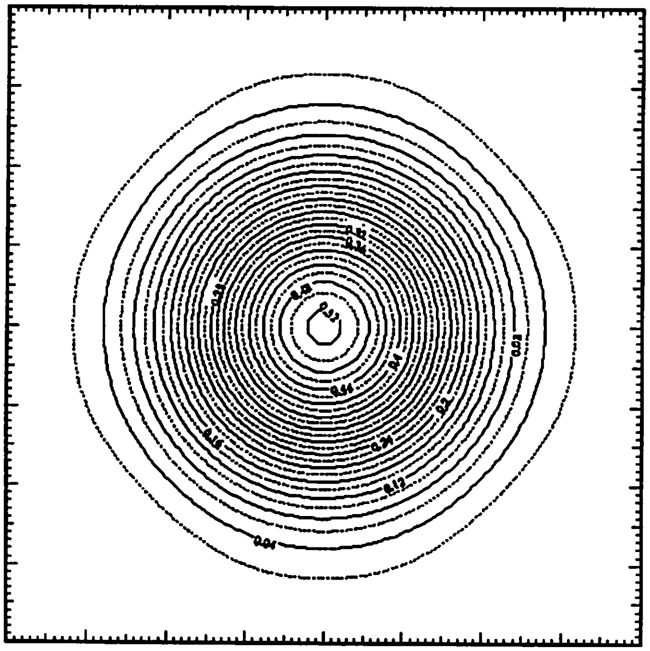
Cell Method with 80x80x100 Grid

Level-Set Method with 80x80x100 Grid

Figure 6) The Three Methods From Another Angle



Cross Section of the Anisotropic Surface Region



The Contours of the Etch Rate in the Plane of the Cross Section

Figure 7) Cell Method Anisotropy

points per standing wave null for the 50, 100 and 200 vertical cell discretizations. It has been noted by K. Toh [2] that ray interpolations when large segments are subdivide that do not take curvature into account can cause errors. The interpolation method described in Chapter 4, and used in this version of ray-trace was of this form. While such error was shown in [2] to appear for curved surfaces with even finer griddings than those employed here, this effect of large grid size does not seem to appear in the benchmark example. A proposed solution to this difficulty was suggested by K. Toh, but this method was found to be unworkable when integrated with resist deloop in three dimensions, since the rays were used as the basis of the interpolation instead of the geometry. (See Chapter 4 for further discussion.) Further investigation of the effect of interpolation methods in real examples for ray-trace may be necessary.

Linewidth is of great concern in photoresist simulation. The width of device features significantly affects their operation. If the distances across the profiles in Figure 2, Figure 3 and Figure 4 are measured at the center of the standing wave maxima, it is clear the ray-trace method and the cell method yield similar linewidths. The linewidth generated by the level-set method is larger than that of the cell method by two cells in either direction, however. To improve this criterion, grid refinement must be performed in the other two grid directions as well. This is not a particularly desirable method of solving the problem, since a halving of the grid size in the horizontal directions (assuming that the finest refinement has occurred in the vertical direction already) will increase the memory consumption and the execution time four fold. This causes the already large values for the level-set technique in Table 2 to become even worse.

7.3.5 Summary of Comparison

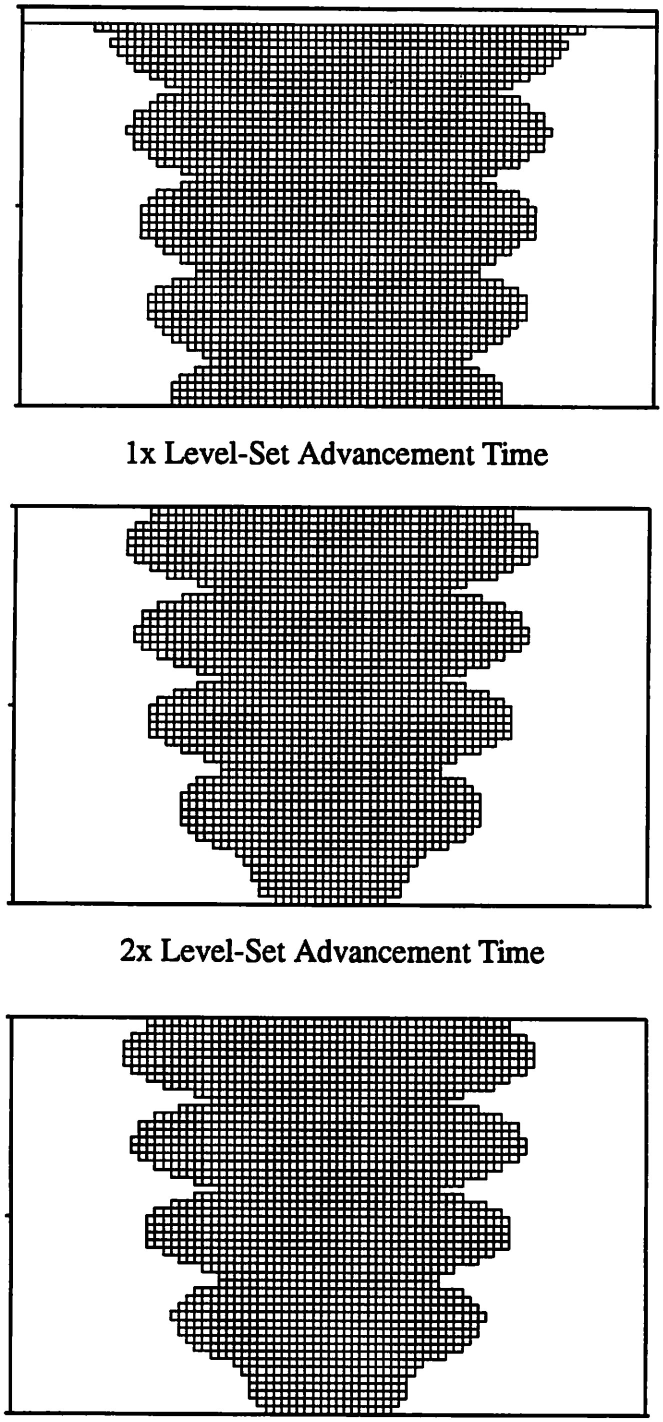
In conclusion, it should be noted that each method has both advantages and disadvantages. When the methods are normalized for vertical accuracy, the ray-trace and cell methods require the same amount of time, while the level-set method is significantly slower. Level-Set approaches require $O(1/h^4)$ time where h is the average element size. Both the cell method and ray-trace were found to require $O(1/h^3)$ time to advance, where h is the average element size. Cell methods have significant difficulties in performing advancement in directions that are not in the coordinate directions of the underlying, and this error cannot be removed in this particular implementation by improving the grid resolution. Some methods for removing anisotropy have been implemented in photolithography simulators [2][11], but require significant computational time. A cell method has not, however, been implemented for photolithography problems that performs an adequate calculation of the surface normal for each cell. It is suggested before further work on cell methods is performed, the work of E. Puckett should be examined [7][8]. Ray-trace is both fast and accurate, but if it is to be fully robust, more work must be performed in the area of mesh maintenance and techniques for correcting scattering rays. The decrease in speed that would result from the inclusion of these techniques would probably not be more than a factor of two, since it would probably not be necessary to implement ray fixing every time step, and thin triangle removal does not slow advancement significantly. Improved implementation techniques for thin triangle removal would also increase the speed of the ray method.

In the area of memory consumption, both the cell and level-set methods grid the entire space, although only the part near the surface is really necessary for the purposes of advancement. The cell and level-set methods therefore require $O(1/h^3)$ elements to be contained in memory. The ray-trace method, since it only represents the surface at any time, requires $O(S/h^2)$ memory, where S is the surface area of the resist at any point in time and h is the average segment length. Therefore, the ray-trace method requires less memory overall for fine grid sizes. Techniques for decreasing the memory consumption of the cell method by using the octtree should be investigated.

In general, attempts to reduce both memory consumption and time consumption simultaneously lead to significant increases in programming complexity. Ray-trace tries to improve the accuracy of the method by representing the surface explicitly from time step to time step instead of reinterpolating it as the cell method does. Ray-trace also minimizes memory consumption by only representing the surface instead of embedding the surface in another mathematical structure as the level-set technique does. By doing this, the issues of delooping and mesh-maintenance are, at some level, unavoidable. Therefore, attempts to improve the performance of either the cell-method or the level-set method in the areas of accuracy, memory or time consumption will lead to increases in programming complexity. The trade-off between improvements in performance at the cost of increasing code complexity is typical of algorithms from both computational geometry and partial differential equations. It is clear that surface advancement, which is derived from both fields, is no different. Therefore, each of the three simulation methods presented, should be seen as a 'vanilla' method that represents extremes of simulator behavior. These extremes also represent possible trade-offs that can be used to design simulators with the best mix of properties for photolithography simulation.

7.4 The Iterative Level-Set Method

A tentative new technique for improving level-set based advancement has been developed. The details of its implementation are described in Chapter 4. This tech-



1x Level-Set Advancement Time

2x Level-Set Advancement Time

4x Level-Set Advancement Time

Figure 8) Iterative Level-Set Methods

nique is based on grid generation techniques developed by J. Sethian in [10]. By advancing the level-set field for a period of time before actually advancing the surface contour, good initial conditions seem to be created. The results of the application of this method are shown in Figure 8. Continued iterations improve results, although it is clear that after two iterations, no significant further improvement is forthcoming. It is not clear whether this method iterates towards second order, or a very good first order implementation. The run times for the three simulations shown in Figure 8 are 15 minutes, 30 minutes and 1 hour. The 30 minute example is almost as accurate as the 20 minute ray-trace example and has no anisotropic behavior. The iterative level-set technique, however, still requires $O(1/h^4)$ time to advance the surface where h is the average cell side length. Therefore, although a smaller grid size than the normal level-set technique may be used for most problems, for problems that require very fine grids, this method still will not be as fast as ray-trace or cells.

Reference for Chapter 7

[1] J. Helmsen, M. Yeung. D. Lee and A. Neureuther, "SAMPLE-3D Benchmarks Including High NA and Thin Film Effects" SPIE Optical/Laser Microlithography VII, vol. 2197, pp. 478-88, 1994.

[2] K. Toh, *Ph.D. Dissertation*, University of California, Berkeley, Dec. 1990.

[3] E. Scheckler, "Algorithms for Three-Dimensional Simulation of Etching and Deposition Processes in Integrated Circuit Fabrication", *Ph.D. Dissertation*, University of California, Berkeley, Nov. 1991.

[4] J. Sethian, "An Analysis of Flame Propagation", *Ph.D. Dissertation*, University of California, Berkeley, 1982.

[5] J. Sethian, "Numerical Algorithms for Propagating Interfaces: Hamilton-Jacobi Equations and Conservation Laws", *Journal of Differential Geometry*, 1990, pp. 131-161.

[6] W. Noh and P. Woodward, in *Lecture Notes in Physics; 59*, ed. A. van der Vooren and P. Zandbergen, pp. 330-340, Springer, New York, 1976.

[7] E. Puckett, in *Proceedings of the 4th International Symposium on Computational Fluid Dynamics*, ed. H. Dwyer, pp. 933-938, U. C. Davis, 1991.

[8] J. Pilliod and E. Puckett, (unpublished).

[9] C. Hirsch, *Numerical Computation of Internal and External Flows*, Wiley, New York, 1988.

[10] J. Sethian, "Curvature Flow and Entropy Conditions Applied to Grid Generation", *Journal of Computational Physics*, Dec. 1994.

[11] J. Pelka, "SOLID: Comprehensive Three Dimensional Simulation Program for Optical Microlithography", *Information Brochure, Fraunhofer-Institut fur Mikros-*

trukturtechnic, May 1990.

Chapter 8 Survey of Contributions and Future Work

8.1 Survey of Contributions

Chapter 1 and Chapter 2 were basic introductions to the field of three dimensional photolithography dissolution simulation.

Chapter 3 assembled into one location the mathematics of the Hamilton-Jacobi equation and three popular methods of computing solutions to the equation in the field of photolithography dissolution simulation. The least time path formulation was used, based on methods of geometrical optics, to categorize the types of shock fronts that may occur during ray advancement.

Chapter 4 provided the exact details of the ray-trace, level-set and cell method implementations. Prof. Sethian's gradient operators and level set advancement techniques were implemented in the level-set method. A boundary condition that is well made for photolithography was implemented. A new method of interpolating the rays of the ray-trace method was introduced. A new iterative method of performing level-set solutions is described and implemented.

Chapter 5 introduced two loop removal methods. Both methods were based on octtree triangle intersection techniques. The octtree was also identified as a useful data structure for many problems in surface advancement. A general deloop algorithm for removing negative volume regions from a mesh was implemented. Important parts of this deloop method were also used to perform set operation techniques by J. Sefler. Methods for using deloop to perform set operations were described. A loop remover that was specially modified for resist problems was implemented. This loop remover was shown to improve the robustness of the ray-trace scheme significantly.

Chapter 6 demonstrated mesh maintenance techniques, both implemented and unimplemented, to reduce the number of thin triangles in the surface and improve the planarity of the surface. These techniques were created to reduce the level of ray scattering in the ray-trace method. An example was shown where thin triangle removal reduced ray scattering and prevented a loop from occurring that would have been difficult for the resist delooper to remove.

Chapter 7 presented the results of the comparison of the three methods. Advantages and disadvantages were found for each approach. Ray-trace requires more mesh maintenance technique development and a fix for ray-scattering. The cell method requires a good surface normal approximator and an advancement method that is based on good surface normal approximations. Promising results that require further investigation were found with the iterative level-set method. Both level-set methods could be improved if a method can be created that only computes the solution near the 0.0 contour. It was noted that difficulties with both the cell and the level-set method may be solved, if a cell method is developed that is based on level-set mathematics.

8.2 Future Implications in Other Fields

Finally, it is important to note that the advancement of triangular meshes that are unsupported by an assisting volume mesh of simplices is a relatively uninvestigated simulation technique, and is in a stage of infancy. The concepts of loop removal and mesh maintenance set forth here are fundamental concepts for the application of this technique. These methods are also important for properly advancing unstructured volume meshes in three dimensions with moving triangular boundaries. These arise in many applications, such as the Free-Lagrangian method [1] for determining the evolution of an interface between two fluids, and the oxidation process in integrated circuit manufacturing [2]. Finally, it is reasonable to assume that the explicit mapping of points from the surface at an earlier time step to a new surface at a later time step allows for better enforcement of conservation laws on the surface than would be achieved with cell or level-set methods. Some examples of fields where this may be applied are the oxidation process, where it is important to track of the number of reaction sites, simulating fabrics and sails by keeping track of surface area and other properties during deformation, and conserving catalysts in chemical reactions on surfaces. This research might also be applied to computing surface tension in fluid mechanics and other methods involving conjugant gradients.

Reference for Chapter 8

[1] H.Trease, M. Sahota, "Massively Parallel Hydrodynamics on Unstructured Grids" Proceedings of the 1993 Simulation Multiconference on the High Performance Computing Symposium, pp. 123-6, (Arlington, VA), March 29 - April 1, 1993.

[2] C. Rafferty, *Stress Effects in Silicon Oxidation - Simulation and Experiments*, Ph.D. Dissertation, Stanford University, 1989

Bibliography

E. Barouch, B. Bradie, S. Babu, "Resist Development Described by Least Action Principle-Line Profile Prediction", *Journal of Vacuum Science & Technology B*, vol. 6, no. 6, pp. 2234-7, Nov. 1988.

E. Barouch, B. Bradie, H. Fowler and S. Babu, "Three-Dimensional Modeling of Optical Lithography for Positive Photoresists," *Interface '89: Proceedings of KTI Microelectronics Seminar*, pp. 123-136, Nov. 1989.

E. Barouch, J. Cahn, U. Hollerbach and S. Orszag, "Numerical Simulation of Submicron Photolithographic Processing," *Journal of Scientific Computing*, vol. 6, no. 3, pp. 229-50, 1991.

J. Bauer, "Modelle fuer den fotolithografischen Prozess," *Feingeraetetechnik*, vol. 29. pp. 127ff, 1980.

M. Born, E.Wolf, *Principles of Optics, Sixth Edition,* Pergammon Press, London 1980.

K. Brakke, "The Surface Evolver" *Experimental Mathematics*, vol. 1, pp. 141-165, 1992.

R. Brayton, R. Rudell and A. Sangiovanni-Vincentelli, "MIS: A Multiple-Level Logic Optimization", *IEEE Transactions of Computer Aided Design*, pp. 1062-1081, November 1987.

L. Carll, *A Treatise on the Calculus of Variations*, pp. 335-344, John Wiley & Sons, New York, 1881.

A. Charlesby, *Atomic Radiation and Polymers*, Pergammon Press, London, 1960.

A. Chorin, "Flame Advection and Propogation Algorithms," *Journal of Computational Physics*, vol. 35, pp. 1-11, 1980.

F. Dill, A. Neureuther, J. Tuttle and E. Walker, "Modeling Projection Printing of Positive Photoresists", *IEEE Transactions on Electron Devices*, vol. ED-22, no. 7, July 1975.

H. Dill, W. Hornberger, P. Hauge, and J. Shaw, "Characterization of Positive Photoresist," *IEEE Transactions on Electron Devices,* vol. ED-22, pp. 456-464, July 1975.

R. Ferguson, J. M. Hutchinson, C.A. Spence, and A.R. Neureuther, "Modeling and Simulation of a Deep-UV Acid Hardening Resist," J*ournal of Vacuum Science and Technology B*, vol 8, pp. 1423-1427.

J. Foley, A. van Dam, S. Feiner, and J. Hughes, *Computer Graphics: Principles and Practice 2nd Edition*, p. 706, Addison-Wesley, Reading, MA, 1990.

B. Foote, M.S. Thesis, University of California, Berkeley, Sept. 1990.

M. Giles, D. Boning, G. Chin, W. Dietrich Jr., and others, "Semiconductor Wafer Representation for TCAD," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 13, pp. 82-95, Jan. 1994

J. Greeneich, "Developer Characteristics of Poly(methyl methacrylate) electron resist:", *Journal of the Electrochemical Society*, vol. 122, 970-976, 1975.

P. Hagouel and A. Neureuther, "Modeling of X-ray Resists for High Resolution Lithography", *American Chemical Society 170th Meeting*, vol. 35, no. 2, pp. 298-305, Aug. 1975.

P. Hagouel, *X-ray Lithographic Fabrication of Blazed Diffraction Gratings*, Ph.D. Dissertation, University of California, Berkeley, 1976.

S. Hamaguchi, M. Dalvie, R. T. Farouki and S. Sethuraman, "A Shock-Tracking Algorithm for Surface Evolution Under Reactive-Ion Etching," *IBM Research Report*, RC 18283 (80168), Aug., 1992.

J. Helmsen, E. Scheckler, A. R. Neureuther and C. Sequin, "An Efficient Loop Detection and Removal Algorithm for 3D Surface-Based Lithography Simulation," *NUPAD IV*, May, 1992.

J. Helmsen, M. Yeung, D. Lee and A. Neureuther, "SAMPLE-3D Benchmarks Including High-NA and Thin Film Effects", *SPIE Optical/Laser Microlithography VII*, vol. 2197, pp. 478-88, 1994.

W. Henke, D. Mewes, M. Weiss, G, Czech and R. Schiessl-Hoyler, "Simulation of Defects in 3-Dimensional Resist Profiles in Optical Lithography," *Microelectronic Engineering*, vol. 13, pp. 497-501, 1991.

W. Henke, D. Mewes, M. Weiss, G. Czech, and R-Schiessl-Hoyler, "A Study of Rectile Defects Imaged into Three-Dimensional Developed Profiles of Positive Photoresist Using the SOLID Lithography Simulator", *Microelectronic Engineering*, vol. 14, pp. 283-297, 1991.

Y. Hirai, M. Sasugo, M. Endo, K. Ikeda, S. Tomida and S. Hayama, "Three Dimensional Process Simulation for Photo and Electron Beam Lithography and Estimations of Proximity Effects," *Symposium on VLSI Technology, Digest of Technical Papers*, p. 15, 1987.

Y. Hirai, S. Tomida, K. Ikeda, M. Sasago, M. Endo, S. Hayama, and N. Nomura, "Three-Dimensional Resist Process Simulator PEACE (Photo and Electron Beam Lithography Analyzing Computer Engineering System)", *IEEE Trans. on CAD*, vol. 10, pp 802-807, 1991.

C. Hirsch, *Numerical Computation of Internal and External Flows*, Wiley, New York, 1988.

H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, W. Stuetzle, "Mesh Optimization", *SIGGRAPH '93 Proceedings*, pp. 19-26, Anaheim, CA, August 1-6, 1993

L. Jia, W. Jian-kun and W. Shao-jun, "Three-Dimensional Development of Electron Beam Exposed Resist Patterns Simulated by Using Ray Tracing Model," *Microelectronic Engineering*, vol. 6, pp. 147-151, 1987.

R. Jewett, P. Hagouel, A. Neureuther and T. Van Duzer, "Line-Profile Resist Development Simulation Techniques", *Polymer Engineering and Science*, vol. 17, no. 6, June 1977.

F. Jones and J. Paraszczak, "RD3D (Computer Simulation of Resist Development in Three Dimensions)," *IEEE Transactions on Electron Devices*, vol. ED-28, no. 12, pp. 1544-1552, Dec. 1981.

Y. Karafyllidis, P. Hagouel, "Simulation of Multiple Etch Fronts," *Microelectronics Journal*, vol. 22, pp. 97-104, 1991

D. Kim, W. Oldham and A. Neureuther, "Development of Positive Photoresist," *IEEE Transactions on Electron Devices*, vol. ED-31, no. 12, pp.1730-1735, Dec. 1984.

M. Komatsu, "Three Dimensional Resist Profile Simulation", *SPIE Optical/Laser Microlithography VI*, vol. 1927, pp. 413-26, 1993.

R. LaBarre, *Computational Geometry Techniques for 2D and 3D Unstructured Mesh Generation with Application to the Solution of Divergence Form Partial Difference Equations,* Ph.D. Dissertation, University of Connecticut, 1992.

K. Lee, Y. Kim and C. Hwang, "New Three-Dimensional Simulator for Electron Beam Lithography," *1991 International Workshop on VLSI Process and Device Modeling*, pp. 44-45, Oiso, Japan, May 26-27, 1991.

C. Mack, "Development of Positive Photoresists," *Journal of Electro-Chemical Society*, vol. 134, no. 1, pp. 148-152, 1987.

T. Matsuzawa, T. Ito and H. Sunami, "Three-dimensional Photoresist Image Simulation on Flat Surfaces," *IEEE Transactions on Electron Devices*, vol. ED-

32, no. 9, pp. 1781-1783, Sep. 1985.

D. Meagher, "Geometric Modeling Octtree Using Encoding", *Computer Graphics and Image Processing*, p. 192, June 1982.

A. Moniwa, T. Matsuzawa, T. Ito and H. Sunami, "A Three-Dimensional Photoresist Imaging Process Simulator for Strong Standing Wave Effect Environment", *IEEE Transactions on CAD*, vol CAD-6, no. 3, May. 1987.

J. Munkres, *Topology: A First Course*, Prentice-Hall, Englewood Cliffs, NJ, 1975.

K. Nabors, J. White, "FASTCAP: A Multipole Accelerated 3-D Capacitance Extraction Program" *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 10, pp. 1447-59, Nov. 1991

A. Neureuther, *Simulation of Semiconductor Lithography and Topography*, (unpublished).

W. Noh and P. Woodward, in *Lecture Notes in Physics; 59*, ed. A. van der Vooren and P. Zandbergen, pp. 330-340, Springer, New York, 1976.

S. Osher and J. Sethian, "Fronts Propagating with Curvature-Dependent Speed: Algorithms Based on Hamilton-Jacobi Formulations," *Journal of Computational Physics*, vol. 79, pp. 12-49, 1988.

J. Painter, J. Marshall, "Massively Parallel Hydrodynamics on Unstructured Grids" *Proceedings of the Next Free-Lagrange Conference*, pp. 123-6, (Moran, WY), June 3-7, 1990.

Parmex 1.0 User Guide, Electronics Research Laboratory, University of California, Berkeley 1989.

J. Pelka, "SOLID: Comprehensive Three Dimensional Simulation Program for Optical Microlithography", *Information Brochure, Fraunhofer-Institut fur Mikrostrukturtechnik*, May 1990.

J. Pelka, Simulation of Ion-Enhanced Dry-Etch Processes, *Proceedings of the SPIE*, vol. 1392, pp. 55-66, 1991.

F. Preparata and M. Shamos, *Computational Geometry*, Springer Verlag, New York, 1985.

E. Puckett, "A Volume-of-Fluid Interface Tracking Algorithm with Applications to Computing Shock Wave Refraction", *Proceedings of the 4th International Symposium on Computational Fluid Dynamics*, pp. 933-938, 1991.

E. Puckett, in *Proceedings of the 4th International Symposium on Computational Fluid Dynamics*, ed. H. Dwyer, pp. 933-938, U. C. Davis, 1991.

J. Pilliod and E. Puckett, (unpublished).

C. Rafferty, *Stress Effects in Silicon Oxidation - Simulation and Experiments*, Ph.D. Dissertation, Stanford University, 1989

N. Tam, R*esist mechanisms and models in electron-beam lithography*, Ph.D. Dissertation, University of California at Berkeley, 1991.

L Thompson, C. Willson and M. Bowden, *Introduction to Microlithography, Second Edition*, American Chemical Society, Washington D.C., 1994.

K. Toh and A. Neureuther, "Identifying and Monitoring Effects of Lens Abberations in Projections", *SPIE Optical Microlithography VI*, vol. 772, pp. 202-209, 1987.

K. Toh, *Algorithms for Three-Dimensional Simulation of Photoresist Development*, Ph.D. Dissertation, University of California at Berkeley, 1990.

H. Trease, M. Sahota, "Massively Parallel Hydrodynamics on Unstructured Grids" *Proceedings of the 1993 Simulation Multiconference on the High Performance Computing Symposium*, pp. 123-6, (Arlington, VA), March 29 - April 1, 1993

E. Scheckler, *Algorithms for Three-Dimensional Simulation of Etching and Deposition Processes in Integrated Circuit Fabrication*, Ph.D. Thesis, University of California, Berkeley, 1991.

E. Scheckler, K. Toh, D. Hoffstetter and A. Neureuther, "3D Lithography, Etching and Deposition Simulation," *Symposium on VLSI Technology*, pp. 97-98, (Oiso, Japan), May 28-30, 1991.

J. Sethian, *An Analysis of Flame Propagation*, Ph.D. Dissertation, University of California at Berkeley, 1982.

J. Sethian, "Curvature and the Evolution of Fronts", *Communications in Mathematical Physics*, v. 101, pp. 487-499, 1985

J. Sethian, "Numerical Algorithms for Propagating Interfaces: Hamilton-Jacobi Equations and Conservation Laws", *Journal of Differential Geometry*, pp.131-161, 1990.

J. Sethian and J. Strain, "Crystal Growth and Dendritic Solidification", *Journal of Computational Physics*, v. 98, pp. 231-253, Feb. 1992

J. Sethian, "Curvature Flow and Entropy Conditions Applied to Grid Generation", *Journal of Computational Physics*, Dec. 1994

E. Walker, "Reduction of Photoresist Standing-Wave Effects by Post Exposure Bake", *IEEE Transactions on Electron Devices*, vol. ED-22, no. 7, pp. 464-466. July 1975.