1988

A Cellular Algorithm for Data Reduction of Polygon Based Images

Robert James Caesar
University of Central Florida

UNIVERSITY OF CENTRAL FLORIDA

OFFICE OF GRADUATE STUDIES

THESIS APPROVAL

DATE: April 8, 1988

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY Robert James Caesar, Jr.

ENTITLED "A Cellular Algorithm for Data Reduction of Polygon Based Images"

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS OF THE DEGREE OF Master of Science in Engineering

FROM THE COLLEGE OF Engineering

Harley R. Myler
Supervisor of Thesis

Recommendation Concurred In:

Bruce E. Mathews
Coordinator of Degree Program

Committee on Final Examination

Louis M. Trefonas
Dean of Graduate Studies

A CELLULAR ALGORITHM FOR DATA REDUCTION OF POLYGON BASED IMAGES

BY

ROBERT JAMES CAESAR JR.
B.E.E., Georgia Institute of Technology, 1985

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Engineering
in the Graduate Studies Program
of the College of Engineering
University of Central Florida
Orlando, Florida

Spring Term
1988

ABSTRACT

The amount of information contained in an image is often much more than is necessary. Computer generated images will always be constrained by the computer's resources or the time allowed for generation. To reduce the quantity of data in a picture while preserving its apparent quality can require complex filtering of the image data.

This paper presents an algorithm for reducing data in polygon based images, using different filtering techniques that take advantage of a priori knowledge as to the images' content. One technique uses a novel implementation of vertex elimination. By passing the image through a sequence of controllable filtering stages, the image is segmented into homogeneous regions, simplified, then reassembled. The amount of data representing the picture is reduced considerably while a high degree of image quality is maintained.

The effects of the different filtering stages will be analyzed with regard to data reduction and picture quality as it relates to flight simulation imagery. Numerical results are included in the analysis. Further applications of the algorithm are discussed as well.

ACKNOWLEDGMENTS

The research presented in this paper was made possible through the guidance and inspiration given from my coworkers at the Simulation and Control Systems Department of GE in Daytona Beach. Specifically, I would like to thank Barry Fishman, Jon Lin, and Mike Panzitta for their key input during the project. I would also like to thank Dr. Harley Myler of the University of Central Florida Department of Computer Engineering for his insightful suggestions and for keeping me on schedule. Lastly, and most importantly, I want to thank my wife, Grace, for her tolerance of the many hours I had to spend away from home.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

INTRODUCTION

Purpose of the Data Reduction System

The data reduction processes described in this paper are for the creation of simplified data bases used by flight simulator image generators. Image generators used in flight simulation have stringent time constraints in which to perform compute intensive tasks. Some of the major tasks encountered in flight simulation include but are not limited to:

1. Responding to a user's input;
2. Fetching the three-dimensional information from a data base;
3. Culling the fetched data by throwing out portions which would lie completely outside the viewing window;
4. Transforming three-dimensional information into the current viewing coordinate system;
5. Adding special effects; and
6. Transforming the three-dimensional scene to a two-dimensional viewing plane while resolving depth ordering.

The tasks described above are done fast enough to update a picture at least thirty times a second to produce an undetectable latency between the user's inputs and the image's response. Displaying a scene of moderate complexity requires a great deal of special purpose hardware to keep up with the computational demands. Today's high performance flight simulators are executing nearly 200 billion operations per second [Steiner, 1988].

The basic building block of computer generated imagery is the polygon. Three- dimensional models are created by approximating their surfaces with a mesh of polygonal faces. A complex scene can be made up of thousands of models. The image generator in flight simulators processes this vast quantity of data by pipelining processes that operate on a continuous stream of polygons.

To reduce the loading on a flight simulator image generator, scenes must be modeled with as few polygons as possible while maintaining an acceptable level of realism. Special attention must be paid to the representation of terrain. Normally represented as a mesh of polygons fitted to the earth's surface, rolling terrain can quickly use up the processing power of the image generator. Since image generators can only handle a finite number of polygons, a reduced amount of power is available to display significant objects in a scene. To illustrate this point, consider a level view of a town. The town itself may require a thousand polygons for accurate representation, but it may be necessary to process many times more polygons to display the terrain around the town because the terrain polygons are compressed in the area near the horizon. The amount of detail included in the terrain model must be carefully weighed against its relative importance to a scene. For instance, a data base intended for low level helicopter training demands more accurate representation of surface contours than does one for high altitude reconnaissance simulation.

To simplify a scene's terrain, thus allowing more complexity of the objects in a scene, the terrain information is often filtered before polygons are fitted to the surface. The filtering eliminates unnecessary details while retaining enough information for visual cues. Features such as small bumps are unimportant to a pilot who is looking for a city or a mountain. Just as deviations in elevation can be filtered, so can the cultural information layered on top of it. The boundaries of lakes and rivers often have highly convoluted shorelines that require a great number of

vertices to describe their outlines. Smoothing these outlines significantly reduces the amount of information that is sent to an image generator to render a body of water. A pilot will still recognize a water feature by its basic size and shape. The culture information can include features such as forest regions, areas of residential housing, industrial parks, islands, rice paddies, etc. This paper addresses the problem of simplifying the polygonal outlines of such culture features. Although the processing will be from the perspective of flight simulation, the data reduction system can be easily adapted to perform other unique tasks.

Other Uses of the Data Reduction System

Only slight changes in the data structures are needed to allow the data reduction system to perform operations such as the conversion of continuous tone raster images to a set of polygons representing homogeneous regions. The system might also be used to simplify the high contrast boundaries in an image.

Once in polygonal form, analysis routines can assist machines in recognizing features in an image created from sensor data. In an extreme case, feature recognition would allow the plane to navigate itself and leave the pilot more time to concentrate on other items of importance. Computer recognition of surface features through a plane's sensors would be of tremendous value to mission planning. The computer could analyze recognizable features along with mission intent, location, altitude, threats, fuel, and armament, then offer in-flight advice to the pilot, or even take corrective actions automatically. Situation assessment by the computer would be seemingly instantaneous. A program evaluating such a "computer copilot" is currently in progress and is called the Pilot's Associate Program. The

research for this program is being undertaken by numerous universities and private companies.

Structure of This Paper

Chapter I provides an overview of the data reduction system, while subsequent chapters explain the data being manipulated as well as the major processes that make up the system. Chapter II briefly describes the information available through the Defense Mapping Agency. This data is easily obtained and is widely used in simulation based training. The third chapter covers the image filter that is used for the first step in reducing the data. Chapter IV discusses the segmentation algorithm that divides the image by a set of edges. The method of extracting polygons from a segmented image is given in Chapter V. Also in Chapter V, is the technique for simplifying the extracted polygons. Chapter VI provides quantitative results from the data reduction system. Some guidance on additional uses for the system and the conclusions derived from this paper are given in Chapter VII.

CHAPTER I

DESCRIPTION OF A DATA REDUCTION SYSTEM FOR IMAGES COMPOSED OF POLYGONS

The Processing Sequence

The cell oriented data reduction system is made up of simple components. Its basic function is to accept a series of polygons, to process them through a sequence of filtering steps, and then to output the filtered polygons. The system's goal is to reduce the amount of data that must be stored to describe the polygons, but keep enough information so that major polygonal features can be recognized. The amount of reduction allowed depends on the final destination of the polygons. An image requiring detailed visual cues would need less reduction than an image requiring simple outlines intended for computer recognition.

To achieve a simple, recognizable image, the processing must be controllable and must have as much relevant information as possible available to each processing step. This concept is adhered to throughout the system. The Defense Mapping Agency (DMA) analyzes high-altitude photographs to produce detailed descriptions of the earth's surface culture. Much of the information is organized as regions of uniform culture, bounded by a polygon, with a list of attributes attached to it. These polygons are used as input to the system. Using a painter's algorithm, each polygon is converted from boundary line segments to raster format. The first point of control is the raster density. The raster image then passes through a mode filter to produce a somewhat smoother raster image. To allow another degree of control, the radius of the mode filter can be varied. A segmentation process divides the smoothed raster image by forming boundaries between regions of dissimilar culture

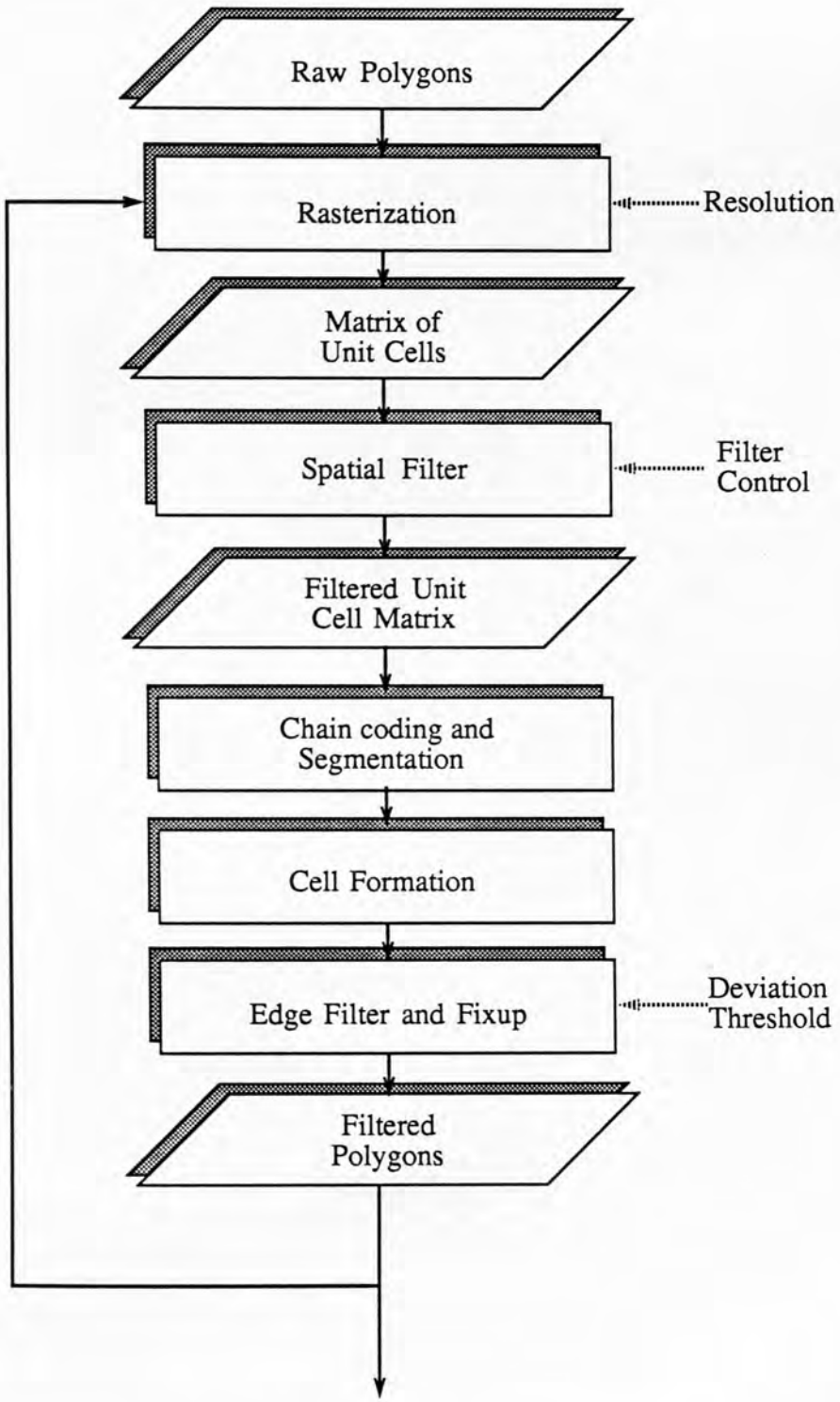


Figure 1. Data Reduction System Flow Chart.

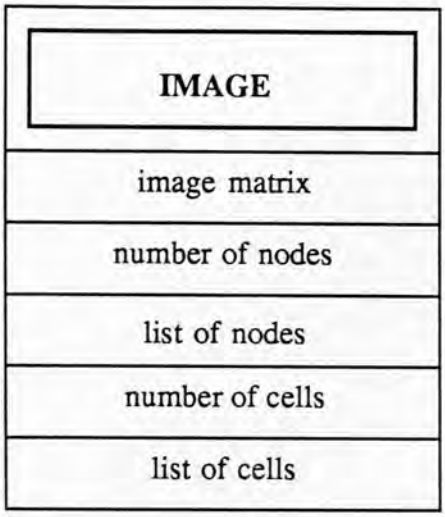
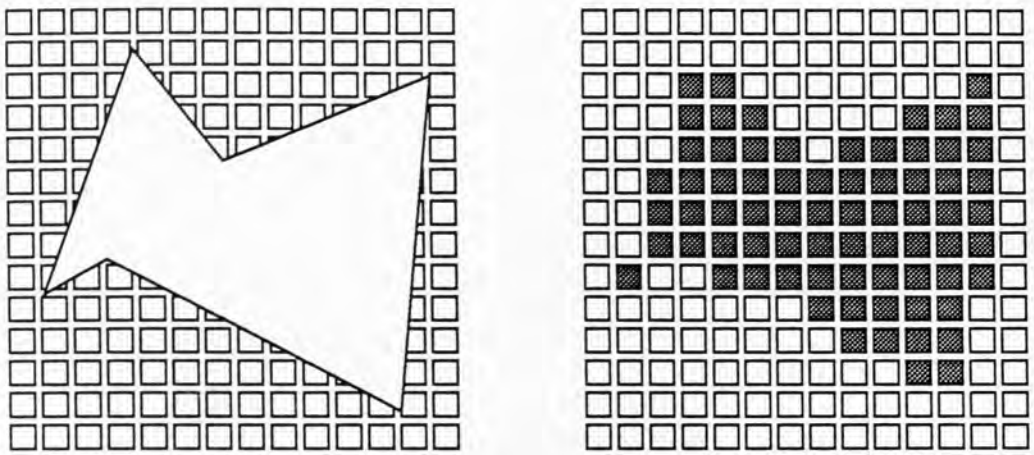


Figure 2. Image Data Structure.

The Image Matrix

The raw polygons entering the system are scan-converted to a matrix format during the rasterization process. Figure 3(a) shows a typical polygon. Figure 3(b) shows how that polygon is likely to be stored in matrix format. This conversion process is the same technique as is used to display filled polygons on a raster display. For efficiency, an active edge table method is used for the scan conversion process [Foley, 1982]. A polygon's edges are sorted in bottom to top order prior to scanning. This allows an active-edge table to be maintained during the scanning process where the table contains only those segments that intersect the scan line being processed. For each scan line, a sorted list of segment intersections becomes a list of start/stop locations. Horizontal lines are filled between alternate start/stop pairs. The size of each of the small squares in Figure 3 is determined by the specified raster density. The small squares are referred to as *unit cells*.

Unit cells represent the smallest unique area of an image. In an environment dealing with DMA data, the resolution of the image can be expressed in terms of



(a) (b)

Figure 3. A Typical Polygon (a) and Its Matrix Format Equivalent (b).

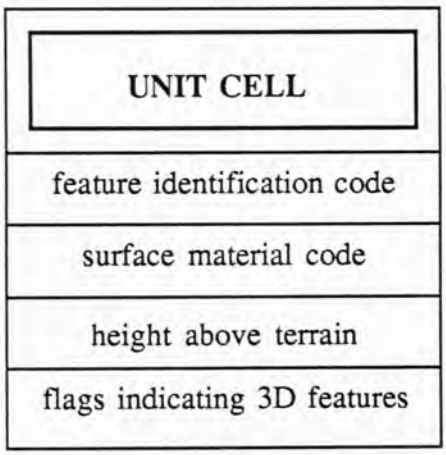


Figure 4. Unit Cell Data Structure.

TABLE 1

SURFACE MATERIAL CATEGORIES.

CATEGORY	SURFACE TYPE
1	Metal
2	Part Metal
3	Stone / Brick
4	Composition
5	Earthen Works
6	Water
7	Desert / Sand
8	Rock
9	Concrete
10	Soil
11	Marsh
12	Trees
13	Snow / Ice
14	Asphalt

unit cells per degree. No polygon output from processing can be smaller than one unit cell in size. The important attributes of a polygon are stored in each unit cell that the polygon covers. Figure 4 gives the data structure of a unit cell. Within a unit cell structure, the feature identification code indicates a category of unique feature types (e.g., trees, water). The surface material codes are listed in Table 1. A unit cell always stores the cultural surface that last covered the unit cell.[1] To

[1] This technique is commonly referred to as a "painter's algorithm."

allow more intelligent processing by subsequent stages, a set of flags is included to signal the existence of objects located within the unit cell. Point features and linear features are small features, discussed in Chapter II, that have no effect on the unit cell's contents other than the setting of flags.

Flags are set during the rasterization step, but are not used until the final processing of cell formation and fixup. The image matrix structure is available to all subsequent processing steps. Complex processing, beyond that which is covered here, can take advantage of the abundance of information that the matrix provides.

CHAPTER II

POLYGONAL IMAGES FROM THE DEFENSE MAPPING AGENCY

Description of Data

The Defense Mapping Agency (DMA) assembles detailed descriptions of the earth's topology and its surface culture. The elevation data are configured as a matrix of elevation posts for various rectangular latitude-longitude regions. The culture data, however, are divided into several levels of detail. Both sets of data contain values that have been extracted from satellite and aerial photographs by automated processes and careful digitization by hand. Errors are introduced by the extraction processes, but they are usually quite small. For example, the relative accuracy of culture data is a circular error of less than 26 meters. This paper addresses only the reduction of surface culture data. The data are divided into three types of features: point, linear, and areal.

The first of these features is the *point feature*. A point feature is used to describe relatively small objects such as buildings, monuments, or storage tanks. Associated with a point feature are its location in latitude-longitude coordinates, its height , length, width, orientation, surface material, and a unique feature

identification code. A small building might be described by the Defense Mapping Agency as a point feature as follows:

Feature type:	Flat roof building
Height:	40 meters
Length:	60 meters
Width:	120 meters
Orientation:	22.5 degrees
Surface material:	Concrete
Location:	-76:23:33.0 latitude 36:23:01.3 longitude

The second type of feature is the *linear feature.* A linear feature is used to describe structures with a high aspect ratio such as bridges, roads, or power lines. Some of the characteristics associated with a linear feature are a series of latitude-longitude coordinates, its height, width, surface material, and unique feature identification number. An example of a line feature representing a steel bridge is:

Feature type:	Bridge
Height:	30 meters
Width:	50 meters
Surface material:	Metal
Center line coordinates:	
	-76:23:33.0 latitude 36:23:01.3 longitude
	-76:23:33.0 latitude 36:23:01.5 longitude
	-76:23:33.1 latitude 36:23:01.7 longitude

Areal features are the third type of features; they are large regions whose boundaries are described by a series of coordinates which form a simple closed polygon. The attributes of an areal feature include height, surface material, number of features per square kilometer, percent roof cover, and percent tree cover. An areal feature can be a grouping of small features where the description of the individual features is not necessary. Areas of single family residential housing are common

areal features occurring in DMA data. An example of a specific areal feature, a collection of buildings that make up a chemical processing plant, is as follows:

Feature type: Chemical Processing Plant
Height: 15 meters
Surface material: Composition
Number of structures per kilometer: 87
Percent roof cover: 60
Percent tree cover: 0
Polygonal boundary coordinates:
-76:23:33.0 latitude 36:23:01.3 longitude
-76:23:32.9 latitude 36:23:05.7 longitude
-76:23:29.5 latitude 36:23:05.6 longitude
-76:23:29.5 latitude 36:23:01.0 longitude
-76:23:31.0 latitude 36:23:02.4 longitude

Use of DMA Data in Flight Simulation

Flight simulator data bases depend heavily on DMA data to provide accurate images with a high degree of realism. Large areas of the earth's surface can be represented as polygons to be stored in simulator data bases. To train a pilot for a mission in a certain region of the world, a data base resembling that region must be created. Many training data bases are originally based on data that are extracted from DMA data, then reformatted so that they can be used by a specific simulator without exceeding any of the image generator's limitations. The conversion process can include fitting polygons to the surface described by the elevation data and layering the surface culture on top of it. The conversion is generally complex, requiring numerous hours of computer time. As stated earlier, some of the original data contain inaccuracies. These inaccuracies, such as a lake that lies on a mountain side rather than in a neighboring valley, must be corrected by the conversion algorithms or by interactive modification by data base designers.

Shown in Figure 5 is a typical conversion. The process shows that the culture data are divided into two-dimensional (2D) and three-dimensional features (3D). The 2D features are areas such as water, soil, or asphalt. The 3D objects are items such as buildings, power pylons, or cooling towers. Three-dimensional features are generally selected from a library of models or are synthesized when they are placed on the terrain.

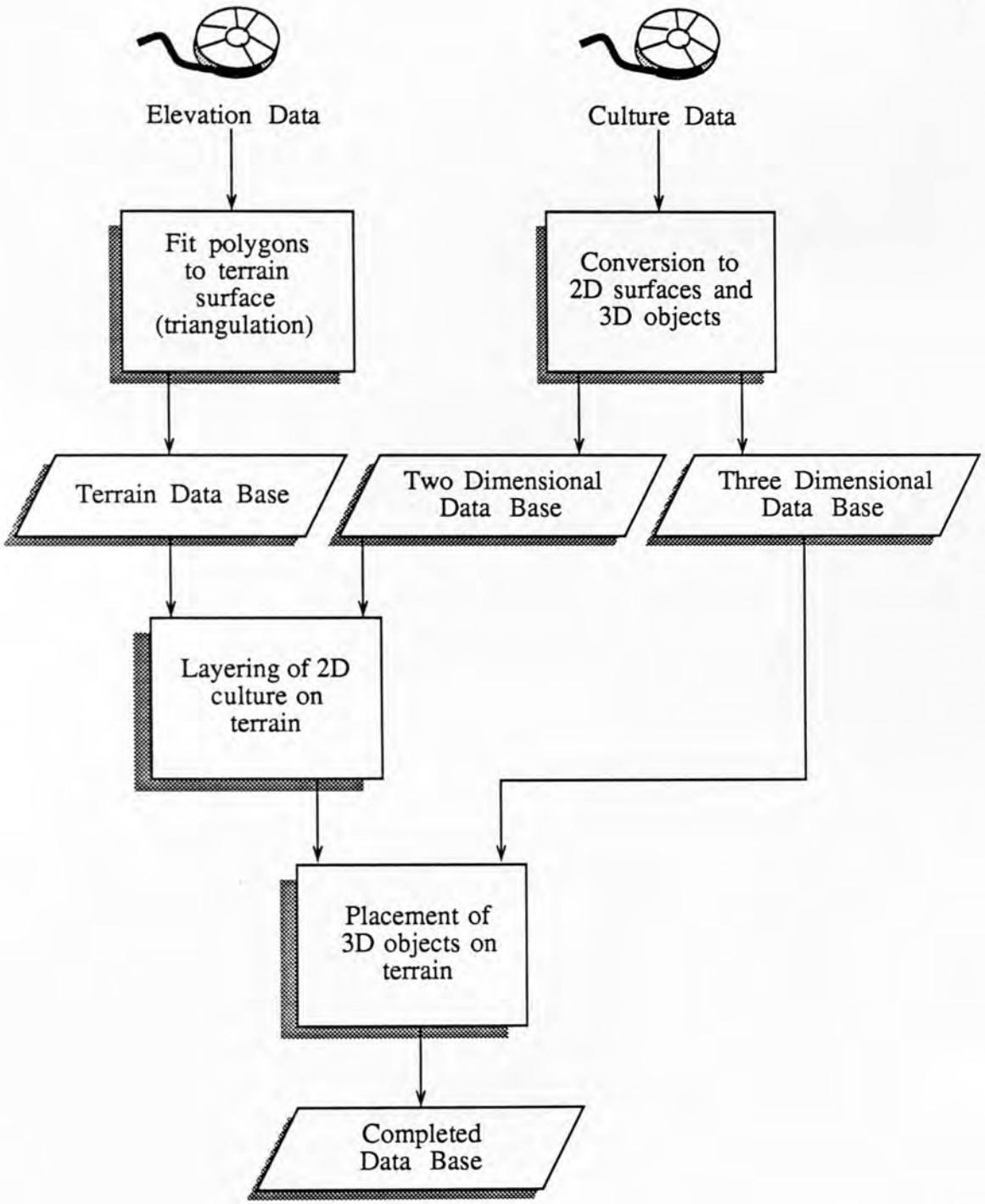


Figure 5. Typical Conversion Process From Defense Mapping Agency Data to a Flight Simulator Data Base.

CHAPTER III

SPATIAL FILTERS

Description of Filter Used for DMA Data

The first step in simplifying the data is the filtering of the image matrix. The picture is smoothed to reduce the high frequency changes in the area boundaries. DMA data are discrete and unordered. The unit cells have unique values that cannot be averaged or adjusted. That is, a unit cell representing trees cannot be averaged with a unit cell of water type to get some intermediate value. For this reason, a mode filter, described below, is used. The mode filter is applied to the image matrix to produce a second image matrix with smoother boundaries. The smoothed matrix will be used by subsequent processing.

The mode filter steps through each unit cell in the original image matrix. The set of unit cells surrounding the original are divided into groups of similar type. The unit cell type of the group with the greatest number of elements is the mode of the set. A unit cell in the corresponding position in a second image matrix is loaded with the mode value. The user has control over the two dimensional filtering by specifying the radius of the filter. The radius is given in unit cells. An example of mode filtering for one unit cell location is illustrated in Figure 6.

To show the effects of increasing the radius of the mode filter, Figures 7 through 10 show an original image of 512 x 512 unit cells. The culture regions have been pseudocolored to better show the effects of filtering.

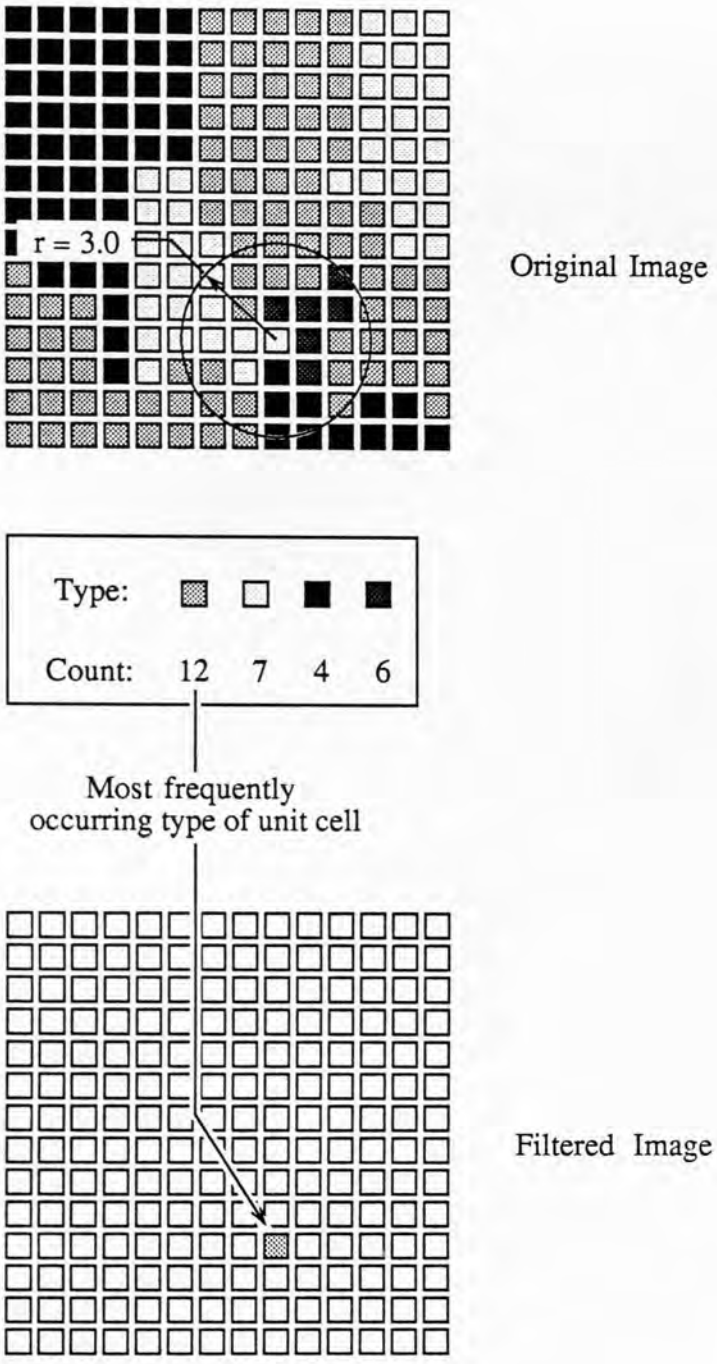


Figure 6. Mode Filter Processing.

Other Image Filters

Although the mode filter works well with the discrete DMA data, other image filters are appropriate for other data types. A median filter would be a good choice for continuous tone pictures. The colors in a continuous tone image have order and thus relative values to one another, so the term median is applicable to a set of colors. A convolution filter can be used to perform an averaging function over a localized set of unit cells. The image output, H(i,j), will contain the same basic information as the input image, F(i,j), with the high frequency noise removed given

$$H(i,j) = F * G = \sum_{m}\sum_{n} F(m,n)G(i-m,\ j-n), \qquad (3\text{-}1)$$

where G(i,j) is a symmetric pulse centered about the origin [Castleman, 1979]. Discrete low-pass filters can also be used when the data are continuous and ordered.

The purpose of the two dimensional filtering is to reduce the amount of data needed to describe an area. Therefore, high-pass and band-pass filters are not good choices for this step in the reduction process. Low-pass filters will have the effect of blurring the image. The blurring reduces the frequency content of the boundaries between dissimilar region types. Line segments can be fitted to the smoother boundaries using fewer vertices.

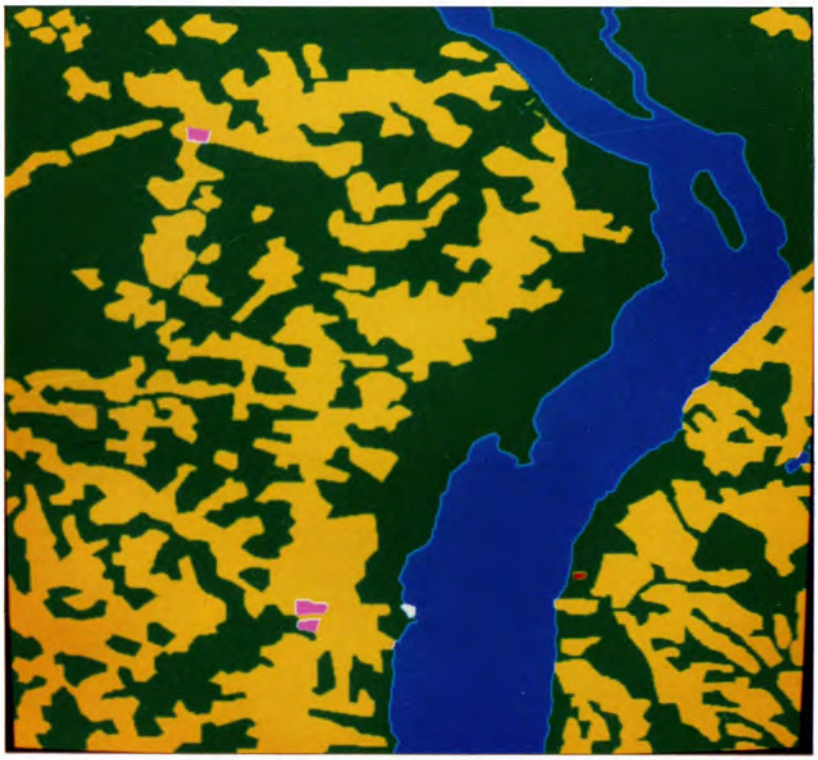


Figure 7. Original River Image - 512 x 512 Unit Cells.

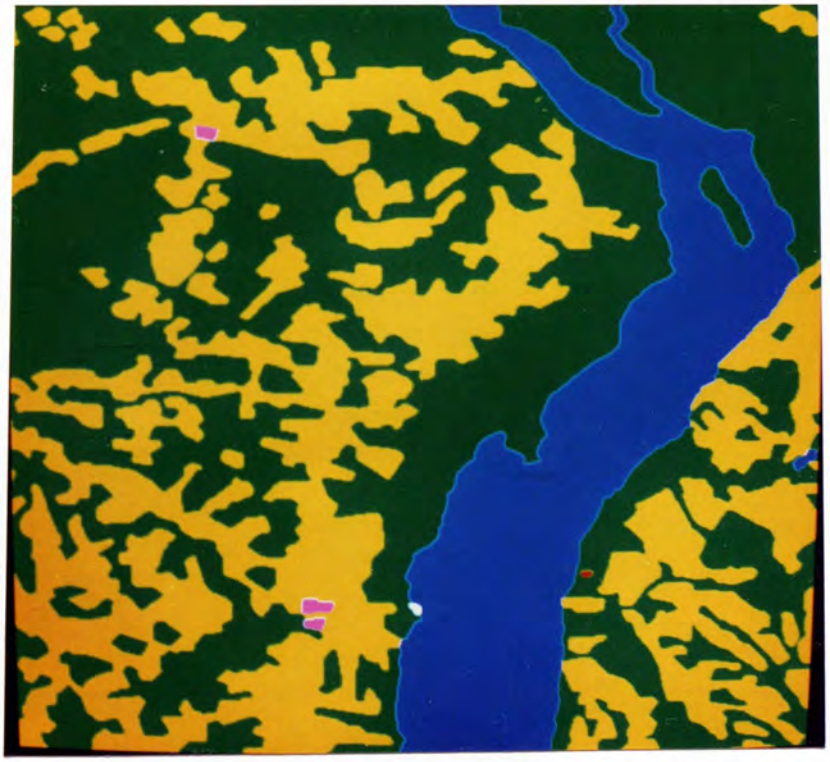


Figure 8. Mode Filtered Image - Radius = 1.5 Unit Cells.

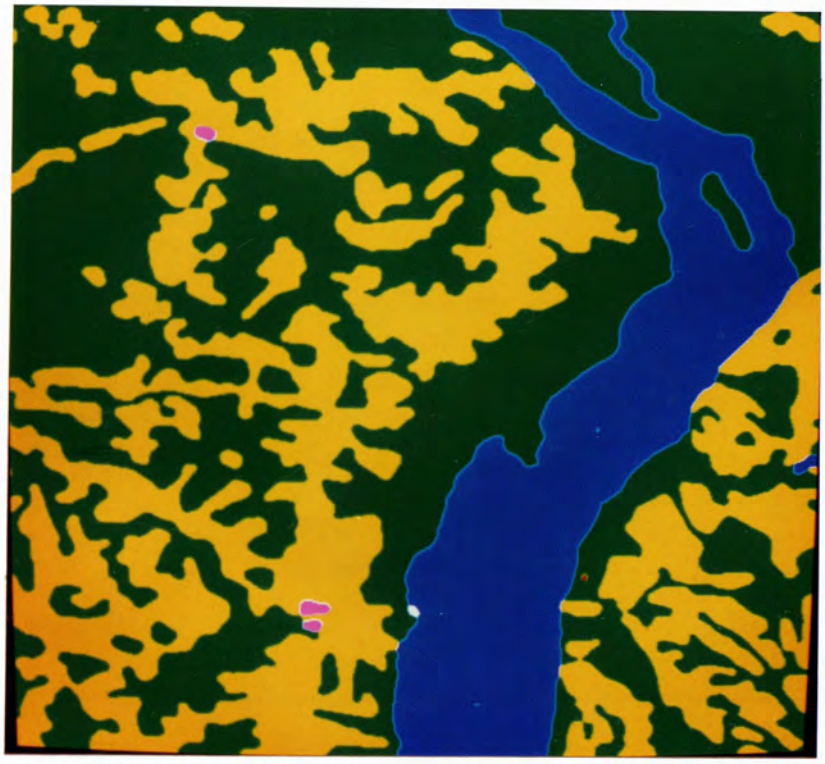


Figure 9. Mode Filtered Image - Radius = 4.0 Unit Cells.

Figure 10. Mode Filtered Image - Radius = 15.0 Unit Cells.

CHAPTER IV

SEGMENTATION

A Brief Description

The segmentation process accepts an image matrix and produces a mesh of boundaries that divide the matrix into regions containing homogeneous members. The boundary lines are composed of chain codes. A connected length of chain codes will be referred to as a *fragment.* Boundaries are defined by any number of concatenated fragments. Where one boundary touches or crosses another boundary, a *node* is created. The process of creating the lists of nodes and fragments will be termed *segmentation* in the context of this work.

Chain Codes

A chain code is a vector that connects a picture element to one of its nearest neighbors. The chain codes used in this paper are illustrated in Figure 11. Other implementations of chain codes use eight directions [Pavilidis, 1982], but only four are used in this data reduction system to simplify processing. Limiting the number of possible directions has an added advantage that will be explained in the segmentation processing portion of this chapter. The purpose of using chain codes is to reduce the amount of data that must be stored to represent a boundary whose vertices must be located at quantum intervals. Image matrices are quantized by the size of a unit cell. Chain codes also provide an easy way to calculate areas, curve lengths, etc. [Freeman, 1960].

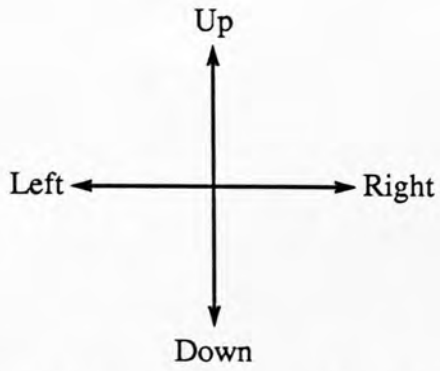


Figure 11. Four Direction Chain Codes.

Fragments

Fragments are a series of chain codes that create a boundary between unit cells. The end points of fragments terminate either at a node or at the end of another fragment. The structure of a fragment is given in Figure 12. Previous and next

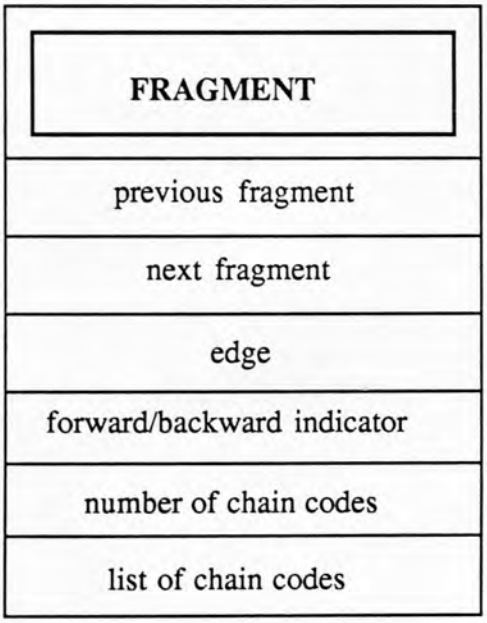


Figure 12. Fragment Data Structure.

fields in the fragment structure allow the fragments to be implemented as doubly linked lists. Linked lists simplify fragment traversal. The edge field is left empty during segmentation, but later it will point to a list of vertices that make up the final polygon edge. A list of chain codes is the main part of the structure. This list can grow in either a forward or backward direction. The purpose of allowing forward or backward fragments will be made clear in the segmentation processing section of this chapter.

Figure 13 demonstrates how fragments are used. Two regions are separated by a pair of linked fragments. The fragments may dynamically allocate the list of chain codes during processing to handle variable length fragments or allocate a fixed length list and link several fragments. If the images to be reduced are simple with

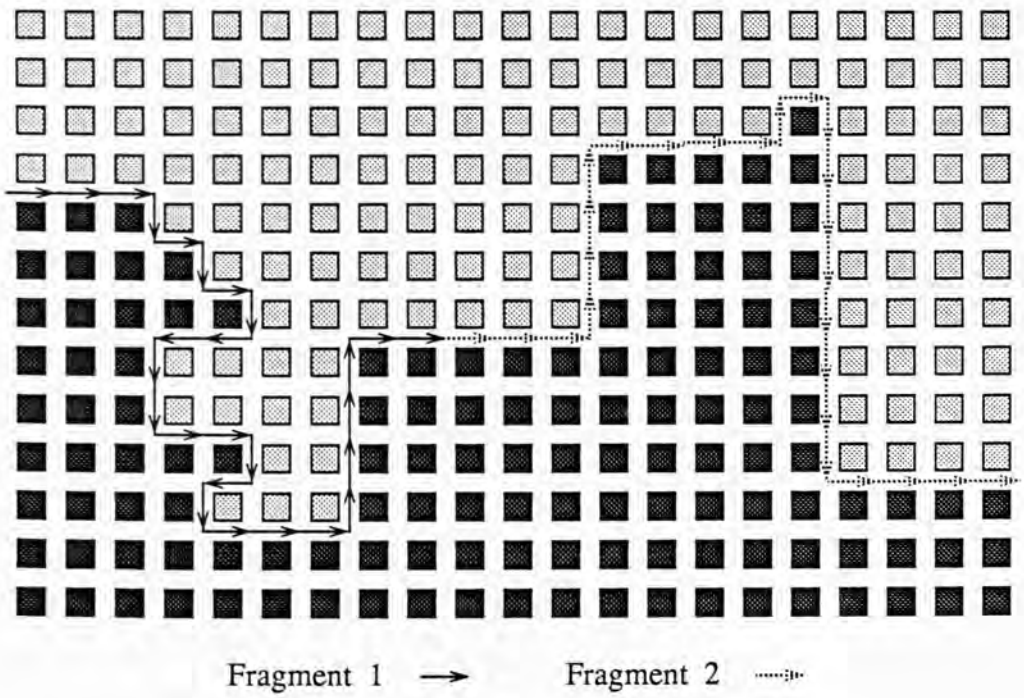


Figure 13. Using Chain Codes to Define Boundaries.

very few but long boundaries, then dynamic allocation is the best choice of allocation. However, images derived from DMA data tend to be complex and contain a great number of boundaries, so many short, fixed length chain code lists are used to speed processing.

Nodes

Whereas a fragment is a boundary between two different types of unit cells, a node is defined at the point where three or four different types of unit cells touch. Figure 14 gives some examples of node points in a matrix of unit cells. Four types of culture converge at node "a" and three types of culture converge at node "b." Node "c" is a special case; its reason for existence will be explained later in this chapter. The data structure of a node is given in Figure 15.

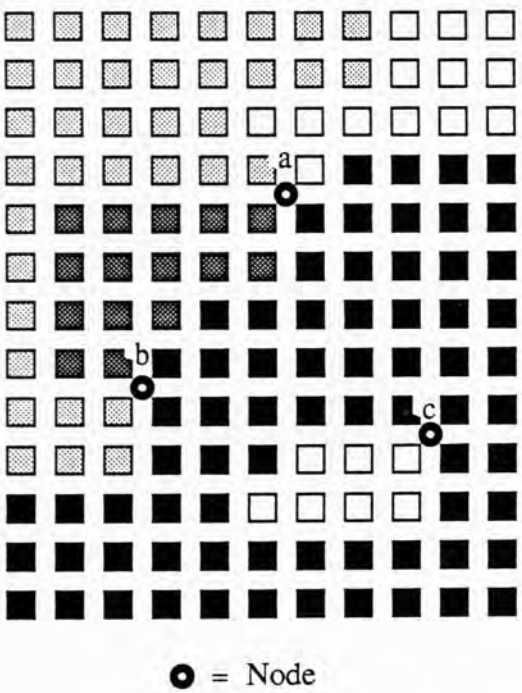


Figure 14. Examples of Nodes.

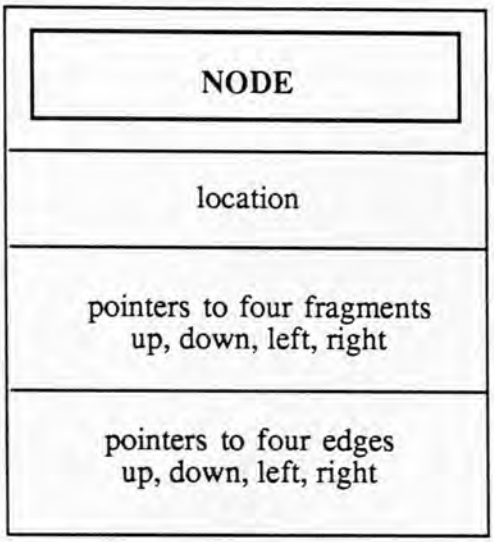


Figure 15. Node Data Structure.

The node structure is simple. It contains only an (X,Y) location, given in unit cell coordinates, and pointers to four boundary structures. The edge structure will be filled in by subsequent processing. The edges contain a vertex list, as opposed to the fragments that contain a chain code list. Nodes are key elements in providing connectivity between the different data structures. Fragments can be joined through this structure, a matrix location can locate the nearest node, and edges can reference the fragments from which they are derived. These are just a few of the uses of the node structure. Additional uses will be explained later.

Nodes are also key anchoring points for recognition of an area. Moving a boundary between two regions usually is not very noticeable, but moving the location where three or more regions come together can make an image unrecognizable. For this reason, nodes will remain at a fixed location once they are placed.

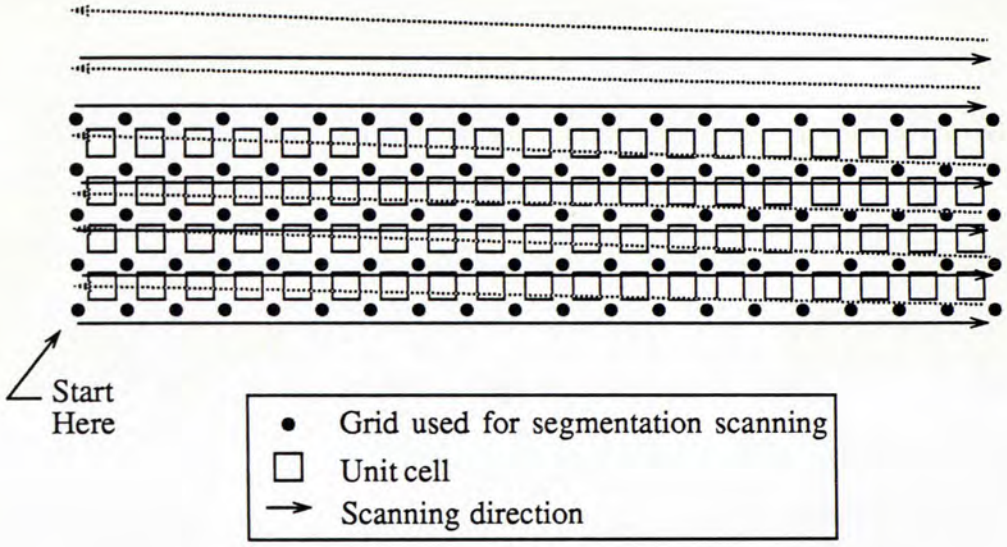


Figure 16. Scanning Sequence for Segmentation.

Segmentation Process

The image matrix is scanned left to right and bottom to top while searching for boundaries and nodes. Segmentation scanning follows a grid that lies between the unit cells and around the matrix boundary. Figure 16 illustrates the scanning process. Fragments are built up during this scanning process to mark the boundaries. Each time a node is found, it is added to the node list of the image structure.

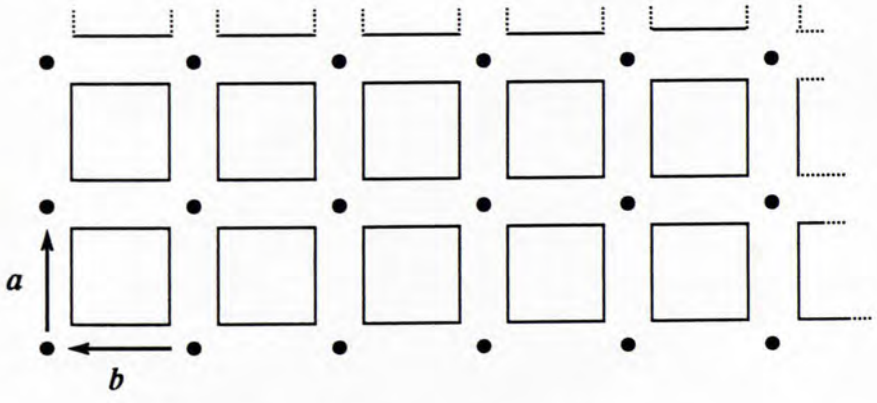


Figure 17. Formation of Initial Fragments.

Beginning at the lower left corner point, the segmentation process is started by creating two initial fragments, each containing one chain code. These fragments will grow and new fragments will be formed as segmentation progresses. In Figure 17, fragment ***a*** is list of chain codes being built in the forward direction. A *forward* fragment is one in which the next chain code added will have its tail connected to the head of the last chain code. Forward refers to processing with the scan direction, backward as against it. Fragment ***b*** is built as a backward chain. *Backward* fragments are formed by connecting the new chain code's head to the tail of the last existing chain code. The forward/backward fragment types allow the joining of two fragments so that they have the appearance of one continuous series of chain codes. Two fragments are joined through their *previous* and *next* links for the subsequent traversal of region boundaries. Occasionally, the chain codes in a fragment must be reversed to allow fragments growing in opposite directions to be joined when they meet. Chain coding is easily reversed by adding 2 (modulo 4) to each code in the fragment's list [Freeman, 1969].

From each grid point, there exists only four possible directions that a chain code can take to separate two different unit cells. Thus, there are four pairwise comparisons that must be performed on the unit cells for each grid point. When comparisons are performed at a grid location that lies on the image matrix border, the area outside the border is considered a unique unit cell type that is always different from any of the cells in the image. Making the border a special case of comparison insures that the entire border is linked by fragments and that nodes will be placed where an internal boundary intersects an image border. Table 2 lists the possible outcomes of testing the unit cells surrounding a single grid point. It is important to note that it is impossible for just one chain code to pass through a

TABLE 2

POSSIBLE RESULTS OF COMPARING FOUR ADJACENT UNIT CELLS.

NUMBER OF DIFFERENCES FOUND	DEFINITION
0	No boundary passes through this grid point.
1	Impossible.
2	A boundary passes through this grid point.
3	Node. Three regions touch at this grid point.
4	Node. Four regions touch at this grid point.

given grid location. This means that all fragments must eventually end at a node. A set of linked fragments will always begin and end at node points.

As the image matrix is segmented, many distinct fragments can be growing at the same time. The set of active fragments is divided into two sets: a current fragment and an array of fragments that are growing upwards from the previous scan line. The current fragment is the boundary that is propagating horizontally along the row of grid points being scanned. If the previous grid point test found no boundary extending to the right, then no current fragment exists for the current grid point. The current fragment is flagged to be nonexistent prior to scanning each new line. Fragments extending upward from the current scan line are stored in a linear array. As the next scan line is processed, the array can be accessed to continue building the active fragments. Figure 18 shows the usage of an upward fragment

TABLE 2

POSSIBLE RESULTS OF COMPARING FOUR ADJACENT UNIT CELLS.

NUMBER OF DIFFERENCES FOUND	DEFINITION
0	No boundary passes through this grid point.
1	Impossible.
2	A boundary passes through this grid point.
3	Node. Three regions touch at this grid point.
4	Node. Four regions touch at this grid point.

given grid location. This means that all fragments must eventually end at a node. A set of linked fragments will always begin and end at node points.

As the image matrix is segmented, many distinct fragments can be growing at the same time. The set of active fragments is divided into two sets: a current fragment and an array of fragments that are growing upwards from the previous scan line. The current fragment is the boundary that is propagating horizontally along the row of grid points being scanned. If the previous grid point test found no boundary extending to the right, then no current fragment exists for the current grid point. The current fragment is flagged to be nonexistent prior to scanning each new line. Fragments extending upward from the current scan line are stored in a linear array. As the next scan line is processed, the array can be accessed to continue building the active fragments. Figure 18 shows the usage of an upward fragment

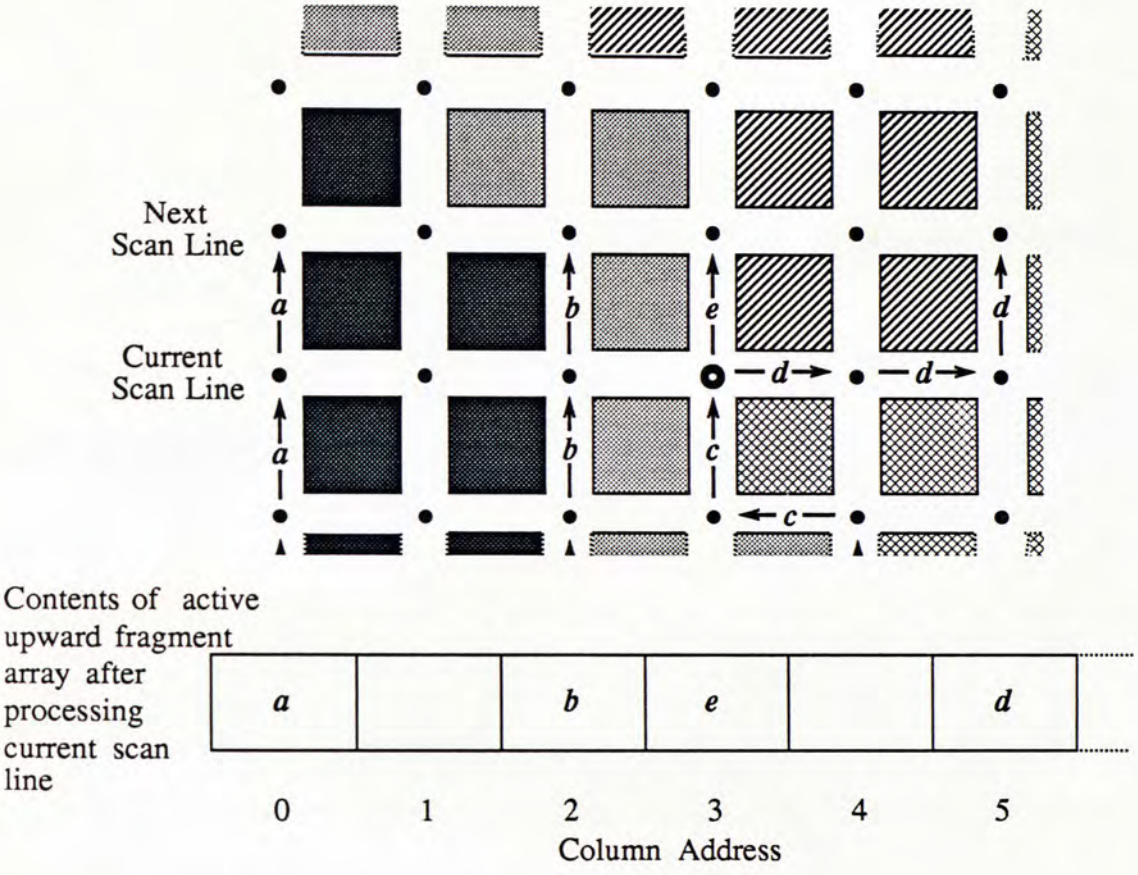


Figure 18. Usage of Active Fragment Array.

list. Since the array is used to connect between the previous scan line and the next, the array is reused for every scan line.

A special case arises when the boundary test for the current grid point indicates that fragments should only exist left and down from the point. Normally, the segmentation algorithm would link the current fragment (continued from the previous grid point) with the fragment stored in the active upward fragment array for the current column. However, if these two fragments are the same fragment or if they are linked together by other fragments with no nodes along their path, then the

current point is forced to be a node. Figure 19 demonstrates this situation. A node is placed because the node structure contains pointers to the fragments attached to them. The node list is later traversed in sequence to build edges from the fragments. If a group of fragments connected back to itself without touching a node, then the subsequent node list traversal would never know the fragments existed and boundary information would be lost. The set of unit cells contained within a boundary of fragments attached to only one node is call an *island* cell. The term island is used because the region is completely contained within another region, thus resembling an island.

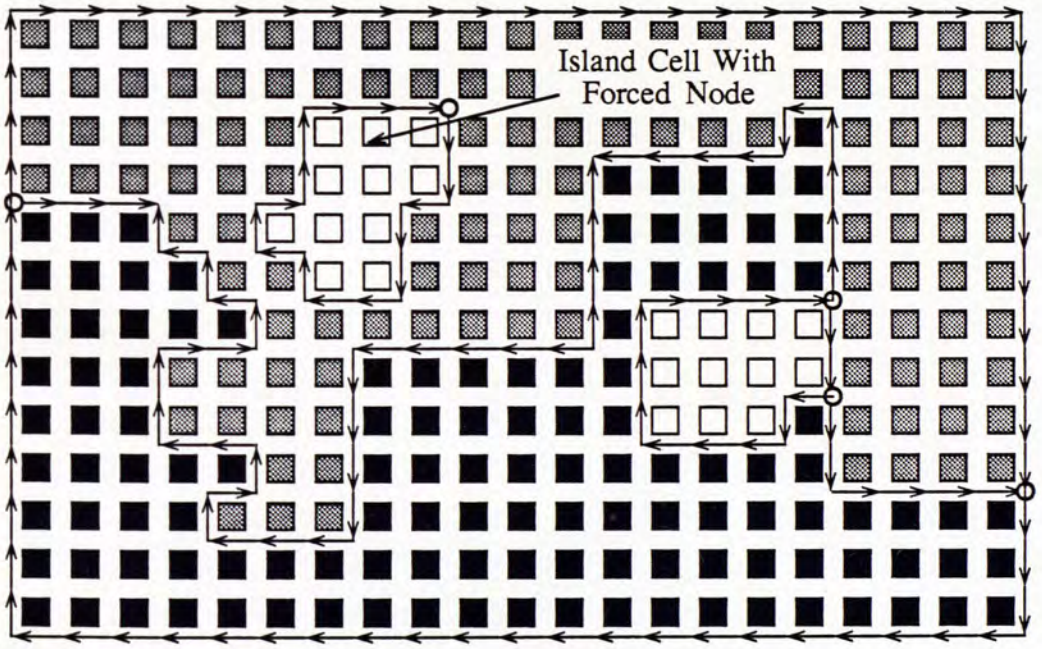


Figure 19. Forcing a Node for Island Cell.

CHAPTER V

CELL FORMATION AND EDGE FILTERING

Description of a Cell

Regions of uniform surface culture surrounded by a simple boundary are called *cells*. The data structure of a cell is shown in Figure 20. A cell structure contains all of the information contained by the unit cells making up the region. Thus, each cell is assigned a surface material code, a feature identification code, and a predominant height above terrain. The boundaries of a cell are defined by a series of edges where each edge is a list of vertices. A two-dimensional cover and the cell's area are calculated from the vertex lists and stored in the cell structure as well. The cover is defined to be the lower-left and upper-right corners of the

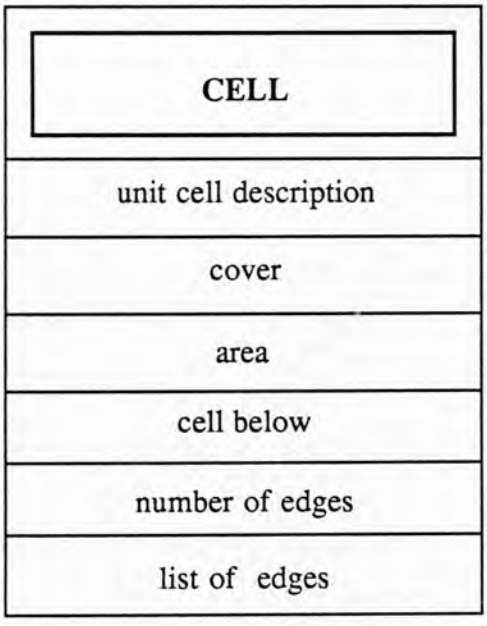


Figure 20. Cell Data Structure.

smallest rectangle that completely contains the cell. The last piece of information stored in a cell structure is an index of the cell below a given cell. This index is used to order the cells during output to avoid covering a small cell by a subsequent larger cell. A cell structure is meant to contain, or have access to, all of the information necessary to produce a closed polygon with the attributes of the region it represents. In fact, as cells are constructed, the processing routine can take into account all of the information stored in the data structures built previously. These structures are the image matrix, the node list, and the fragment chains.

Cell Formation Process

Cells are formed by creating vertex lists from a segmented image. Each node in the previously built list of nodes is visited in sequence. Each fragment emanating from the node is followed through successive fragments and nodes while taking the most clockwise path possible. The path is followed until the original node is encountered again. The nature of the segmentation process guarantees that the original node will be found again. Using the segmented image from Figure 19, Figure 21 identifies the path taken from a node during cell formation. The path begins at the node in the lower right portion of the image and follows the fragments. Only three other nodes occur along the path before arriving at the original node. In all cases, the path chooses to follow the fragment exiting the nodes that are next, in a counterclockwise direction, from where the last fragment entered the node. The path around the cell must still be checked to see if it does indeed form a clockwise path. Figure 22 shows an example of how a clockwise path is not necessarily built by taking the most clockwise path possible. If the path taken happens to be counterclockwise, then it is discarded.

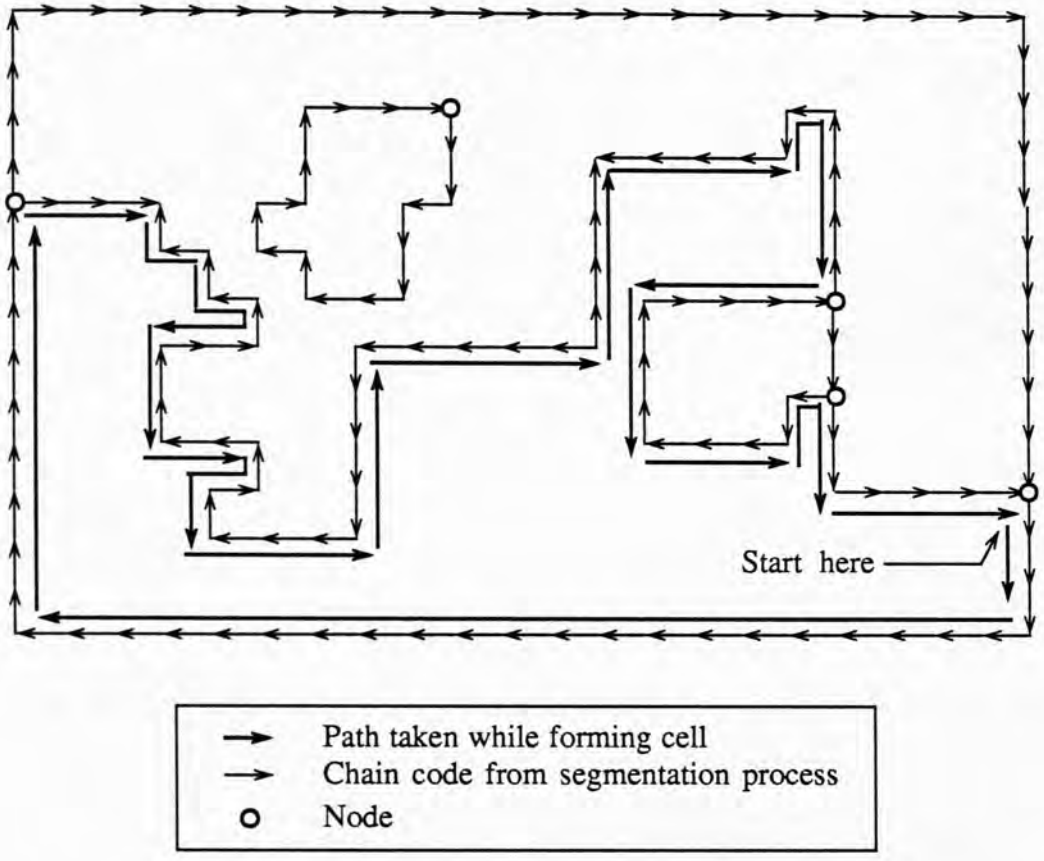


Figure 21. Clockwise Path Taken During Cell Formation.

While traversing the fragments to form a cell, edge structures are created and loaded with boundary vertex information. The edges are a key interconnection between the cell structures and the structures built by segmentation. Edges contain cell adjacency information because each edge represents a shared boundary between two cells. It is this sharing of boundaries that makes the data reduction system possible. The vertex information for an edge is stored only once, whereas the original polygons contain no adjacency information, making boundary sharing impossible. Also, adjusting an edge has a complementary effect on the cell that shares the edge, so overlap removal and gap elimination are an intrinsic part of processing and require no post processing. The original data from the Defense

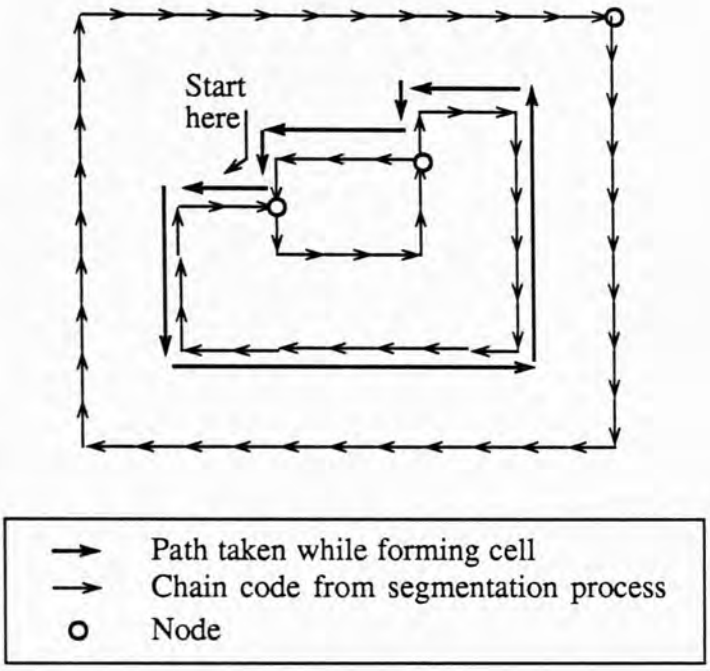


Figure 22. Illegal Counterclockwise Path Taken During Cell Formation.

Mapping Agency are guaranteed to overlap in an attempt to avoid leaving an area uncovered. This overlap accounts for much of the superfluous vertex information contained in the original data. The edge structure which allows these advantages mentioned above is shown in Figure 23.

The segmentation process also insures that two consecutive chain codes will not double back on themselves. Stated in another way, the difference between two adjacent chain codes has only three possible results,

$$(a_{i+1} - a_i) = \begin{cases} -1 & \text{Counterclockwise turn} \\ 0 & \text{Continue in same direction} \\ 1 & \text{Clockwise turn} \end{cases} \qquad (5\text{-}1)$$

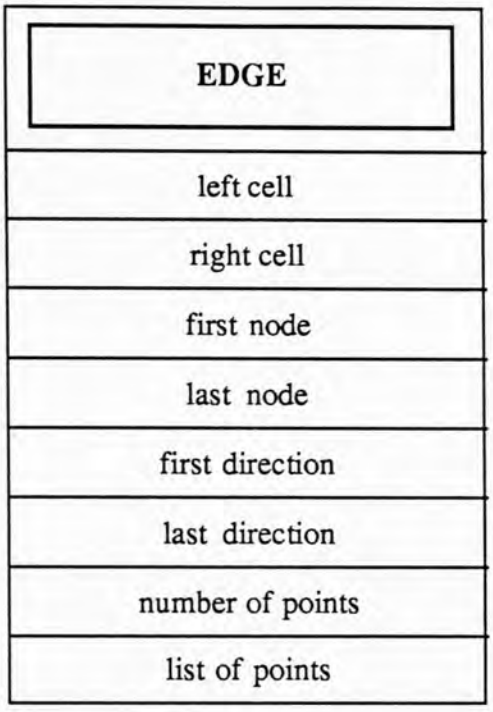


Figure 23. Edge Data Structure.

where a_i represents the ith chain code value. Given n chain codes, the orientation of a closed loop can be checked by using

$$\sum_{i=0}^{n} (a_{i+1} - a_i) = \begin{cases} -4 & \text{Counterclockwise path} \\ 4 & \text{Clockwise path.} \end{cases} \qquad (5\text{-}2)$$

Each edge corresponds to a set of linked fragments between two nodes with no intervening nodes. The edge is a boundary between two adjoining cells. Thus, the number of edges in a cell directly reflects the number of neighboring cells. The initial vertices in an edge are the locations where the fragments change directions. Left unfiltered, this set of vertices represents the most accurate approximation of the boundary data subjected to the rasterization process. The grid structuring arti-

ficially adds angular details to the boundary and can appear unnatural when scaled to a larger grid. Therefore, some edge filtering should be done in order to smooth out the unnecessary details.

Edge Filtering Using Vertex Elimination

The edge filter used in the cellular reduction system is a simple vertex elimination scheme. The two end points of an edge, which are nodes, are always included in the edge's final vertex list. The interior vertices of an edge, on the other hand, are removed from the vertex list by a recursive maximum deviation algorithm. The basic operation can be stated as: given two end points in a sequence of points and a line segment connecting the end points, the intermediate point furthest from the line segment is added as a middle point only if its distance from the line segment is greater than a specified *threshold.* This threshold is denoted as δ_t. The threshold envelope surrounding a pair of end points is illustrated in Figure 24.

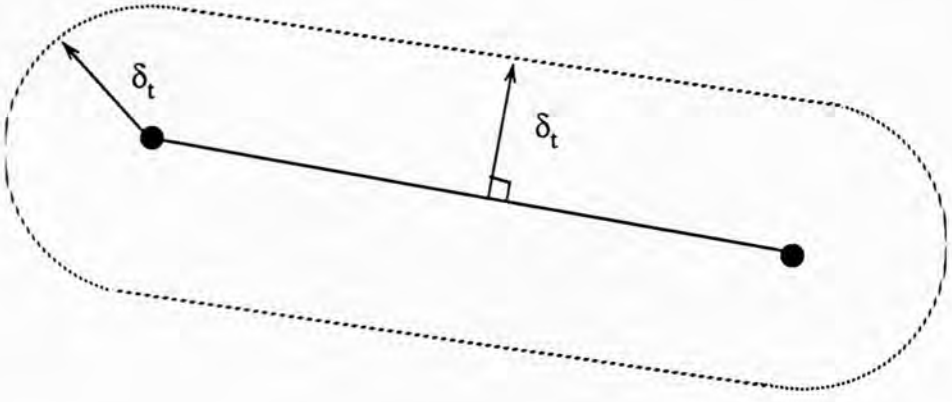


Figure 24. Illustration of Threshold Envelope.

Several distance calculations are needed to apply this vertex elimination algorithm. Let point **P**, located at Cartesian coordinates (x, y), be an intermediate point between the end points $\mathbf{P}_0$ and $\mathbf{P}_1$. The distance from **P** to $\mathbf{P}_0$ and from **P** to $\mathbf{P}_1$ is

$$d_0 = |\mathbf{P}_0\mathbf{P}| = \sqrt{(x - x_0)^2 + (y - y_0)^2} \tag{5-1}$$

and

$$d_1 = |\mathbf{P}_1\mathbf{P}| = \sqrt{(x - x_1)^2 + (y - y_1)^2} \tag{5-2}$$

respectively. The perpendicular distance from a line passing through $\mathbf{P}_0$ and $\mathbf{P}_1$ to the point **P** is

$$d_\perp = \frac{|\mathbf{P}_0\mathbf{P} \times \mathbf{P}_0\mathbf{P}_1|}{|\mathbf{P}_0\mathbf{P}_1|} = \frac{(x - x_0)(y_1 - y_0) - (y - y_0)(x_1 - x_0)}{\sqrt{(x_0 - x_1)^2 + (y_0 - y_1)^2}} . \tag{5-3}$$

The true distance, *d*, of the point **P** to the line segment $\mathbf{P}_0\mathbf{P}_1$ is calculated by one of the three equations above depending on where **P** is relative to the two end points. Expressed mathematically, the true distance between the point and the line segment is

$$d = \begin{cases} d_0 & (d_{\parallel} \le 0) \\ d_1 & (d_{\parallel} \ge |\mathbf{P}_0\mathbf{P}_1|) \\ d_\perp & (0 < d_{\parallel} < |\mathbf{P}_0\mathbf{P}_1|) , \end{cases} \tag{5-4}$$

where

$$d_{\parallel} = \frac{|\mathbf{P}_0\mathbf{P} \cdot \mathbf{P}_0\mathbf{P}_1|}{|\mathbf{P}_0\mathbf{P}_1|} = \frac{(x - x_0)(x_1 - x_0) + (y - y_0)(y_1 - y_0)}{\sqrt{(x_0 - x_1)^2 + (y_0 - y_1)^2}} . \tag{5-5}$$

The distance $d_{\|}$ indicates whether point **P** lies in the region between the two end points or lies closer to one of the end points. Figure 25 shows the relationship between the d_0, d_1, $d_{\perp}$, and $d_{\|}$ distances.

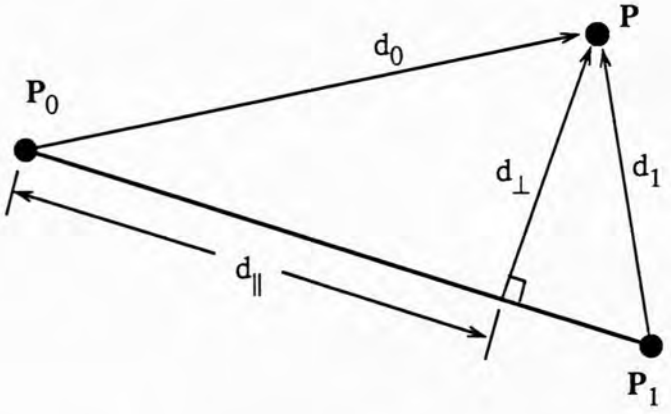


Figure 25. Measurement of Error Distances.

The vertex elimination algorithm makes use of Equations 5-1 through 5-5 in the following recursive steps.

Step 1: Set greatest error, δ, to a threshold, δ_t.

Step 2: For each interior vertex, calculate $d_{\|}$ to determine whether Equation 5-1, 5-2, or 5-3 should be used for calculating the distance between interior point, **P**, and the line segment $\mathbf{P}_0\mathbf{P}_1$.

Step 3: Calculate distance, *d*, to the line segment $\mathbf{P}_0\mathbf{P}_1$.

Step 4: If *d* is greater than δ, then let $\delta = d$ and save the current interior point by letting $\mathbf{P}_{\delta} = \mathbf{P}$.

Step 5: Repeat steps 2 through 5 for all interior points in the list.

Step 6: If δ is greater than δ_t, then add the vertex $\mathbf{P}_\delta$ between the two end points, otherwise the filtering between points $\mathbf{P}_0$ and $\mathbf{P}_1$ is complete.

Step 7: Repeat steps 1 through 7 for the two new edge segments, $\mathbf{P}_0\mathbf{P}_\delta$ and $\mathbf{P}_\delta\mathbf{P}_1$, until all interior vertices lie within δ_t of an edge segment.

The recursive algorithm described above is easily implemented in a program. The original sequence of points is extracted from chain-coded fragments created by the segmentation process and the resulting filtered vertices are stored in the cell's edge structures. This algorithm insures that the original sequence of points is obtained by lowering the threshold to zero.

A minimum threshold distance, however, exists for rasterized images. Specifying a threshold lower than this absolute minimum would add undesirable angularity due to undersampling. Figure 26 illustrates the approximation of a diagonal line by a staircased set of line segments. The steps are a by-product of the

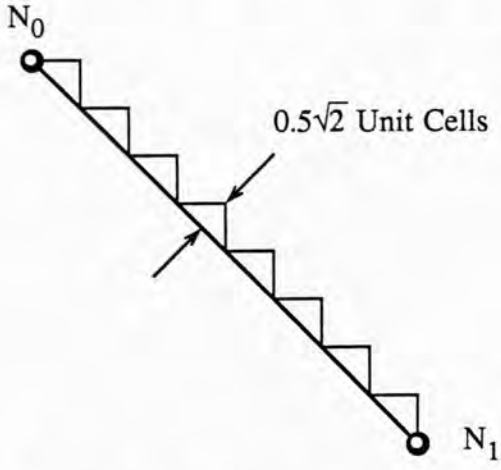


Figure 26. Absolute Minimum Edge Filter Threshold.

rasterization process. It is quite likely that the original data connected the two nodes, N_0 and N_1, by a single line segment. If an edge filtering threshold is less than $0.5\sqrt{2}$ unit cells, then the edge filter would include many of the intermediate vertices.

The data reduction system in this paper sets the deviation threshold to a fixed value at the beginning of edge filtering. There is no reason that this threshold cannot be dependent on the available information. For example, the threshold could be dependent on the frequency content of the original fragment list or be dependent on the type of cells on either side of the edge. An example of this data dependent filtering might be a small threshold for water boundaries to create detailed shorelines while tree/soil boundaries might use a larger threshold to eliminate unnecessary complexity.

Edge Filtering Anomalies

The recursive, greatest error, edge filtering technique described above is relatively simple and performs the task of vertex elimination efficiently. Some anomalies, however, arise during the edge filtering process. There are three major categories of anomalies: non-simple cells, collapsed cells, and encroachment violations. The data reduction system detects and corrects these problems after the edge filtering process has been completed. Fixing the errors as a post process provides a higher degree of modularity and greater flexibility for the system.

The first category of anomalies are non-simple cells. A cell is defined by a closed set of line segments. A non-simple cell is one in which two or more of the segments cross over one another. Figure 27(a) shows a cell before edge filtering. Filtering the edges between the nodes N_0 and N_1 might produce the cell shown in Figure 27(b). The simplicity check compares each line segment to every

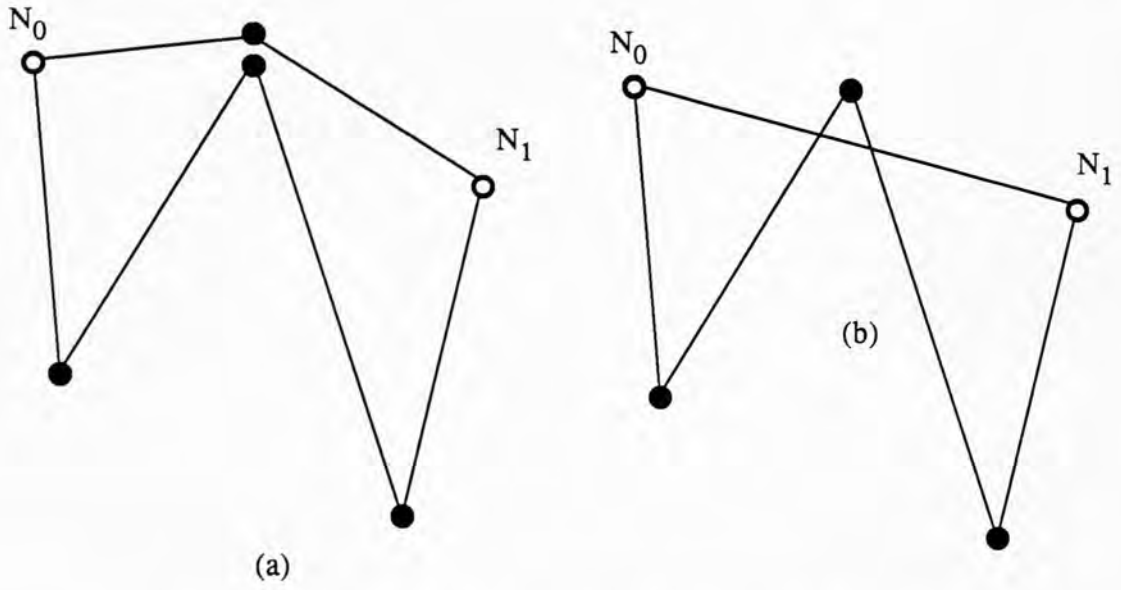


Figure 27. Non-Simplicity Caused by Edge Filtering.

other segment in the cell to evaluate whether or not they intersect. The segments cross if and only if the end points of each segment lie in opposite half-planes where the other segment defines the line where the plane is divided. Mathematically, the crossing test can be expressed as follows. Two segments cross if

$$[\mathrm{SGN}(\mathbf{V}_{02}\cdot\mathbf{N}_{23}) = -\mathrm{SGN}(\mathbf{V}_{03}\cdot\mathbf{N}_{23})]$$

and

$$[\mathrm{SGN}(\mathbf{V}_{20}\cdot\mathbf{N}_{01}) = -\mathrm{SGN}(\mathbf{V}_{21}\cdot\mathbf{N}_{01})], \qquad (5\text{-}6)$$

where

$\mathbf{V}_{ij}$ is the vector formed by $\mathbf{P}_j - \mathbf{P}_i$,

$\mathbf{N}_{ij}$ is the vector normal to $\mathbf{V}_{ij}$,

and

SGN() is the arithmetic sign of its argument.

A pair of typical line segments are depicted in Figure 28 to show some of the components of Equation 5-6.

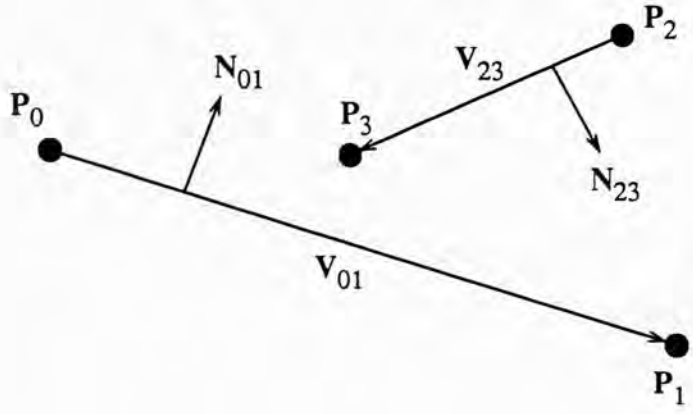


Figure 28. Components of Line Segment Crossing Test.

The comparison of line segment pairs stops immediately to take corrective action once a crossed pair is detected. The problems are corrected as they are encountered. This way, a cell is fixed only once for each pair of crossing edges. Once the corrections for the entire cells are complete, the cell is checked again to be sure the corrections themselves have not caused a simplicity problem.

Fixing an edge crossing error takes advantage of earlier processing methods. The segmentation process assures that all cells are simple before they are processed by the edge filter. This guarantee stems from the process of taking the most clockwise path possible; smaller cells would be created rather than allowing non-simple cells to be formed. The action taken to correct the non-simplicity is refiltering the cell's crossed edges using a lower threshold. Lowering the error threshold has the effect of adding back into an edge vertices that were eliminated in the earlier edge filtering step. The refiltering and re-checking, performed as a recursive process, is repeated until the error is resolved.

A second type of anomaly arising from edge filtering is collapsed cells. Collapsed cells are defined as cells which have zero area. Such cells are created by the

edge filter when all internal vertices of a cell's edges are eliminated and the remaining nodes cannot form a polygon. A simple example of a collapsed cell is a cell with two nodes where all points forming the two edges lie within the threshold envelope around the two nodes (see Figure 24). The data reduction system discards collapsed cells.

The third type of anomaly caused by edge filtering is encroachment errors. An encroachment error is defined as the illegal displacement of surface culture from beneath an above-ground feature. One instance of such an error can occur when an airport runway's surroundings are changed from a clearing to a forest. A more common problem is when buildings, originally placed on the soil side of a shoreline, are out in a body of water after filtering. Either the mode filter or the edge filter can cause the soil/water boundary of a shoreline to move and cause this error. The errors are detected by scanning the image matrix within the bounds of the filtered cells. For each unit cell, the feature flags (indicating buildings, pylons, etc.) associated with the unit cell are compared to the current surface type in that cell. The legality of feature/surface combinations are evaluated against a set of rules. Simple rules, such as buildings cannot exist in a water cell, bridges must touch non-water cells at each end, and storage tanks cannot occur in a forest, are included in the data reduction system to signal encroachment errors. The complex nature of encroachment errors makes automated corrections difficult. Signaling such errors allows an operator to locate and correct the problem interactively. Much research is currently being directed toward intelligent, automatic corrections of map data, but until such algorithms are developed, flagging errors followed by interactive adjustments remains the most expedient solution.

CHAPTER VI

RESULTS

Data Reduction

The three control values, raster density, mode filter radius, and edge filter threshold, can be adjusted independently to affect quality of the picture and the quantity of data necessary to describe that picture. This chapter will investigate the relationships among these control values and their contributions to the overall filtering of an image. In order to quantify data, the figures of merit are chosen to be the number of cells and number of vertices. These numbers will be plotted as a function of the filtering parameters to better understand the effects of filtering.

A region covering the city of Norfolk, Virginia was selected to test the data reduction system because it includes a varied mix of forest, shorelines, dense city area, and residential areas. High contrast colors are used in the following photographs to differentiate the types of surface culture. Table 3 lists the colors for the different culture categories.

The data presented here are based on the original data shown in Figure 29. This section of the Norfolk area ranges from 76°27'00" West to 76°24'00" West and from 36°58'00" North to 37°01'00" North. A total of 4613 vertices are used to describe the 308 polygons comprising this image. The amount of data reduction for this area is typical, and other areas of similar complexity will produce similar results.

The percentage of original vertices is shown as a function of mode filter radius in Figure 30. The density of the raster is 512 by 512 unit cells and the edge filter threshold is fixed at 2.0 unit cells.

TABLE 3

COLOR ASSIGNMENTS FOR SURFACE CULTURE.

COLOR	SURFACE TYPE
Black	Earthen works and rocks
White	Metallic structures (e.g., bridges and railroad yards)
Magenta	Residential housing
Blue	Salt water and fresh water
Red	Composite structures (e.g., brick and concrete buildings)
Green	Forests and marshes
Yellow	Soil and clay
Cyan	Asphalt

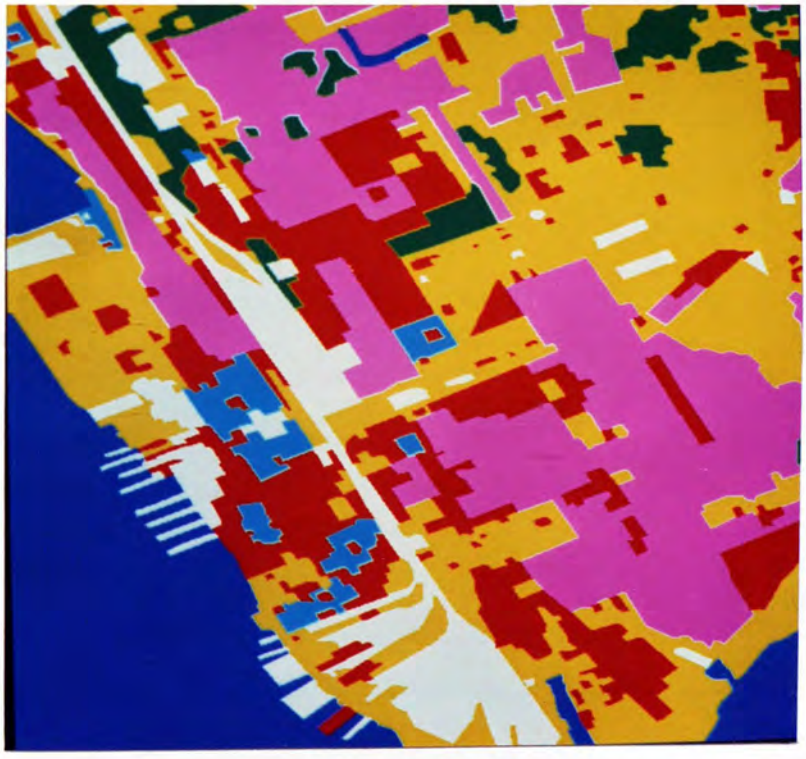


Figure 29. Original City Image.

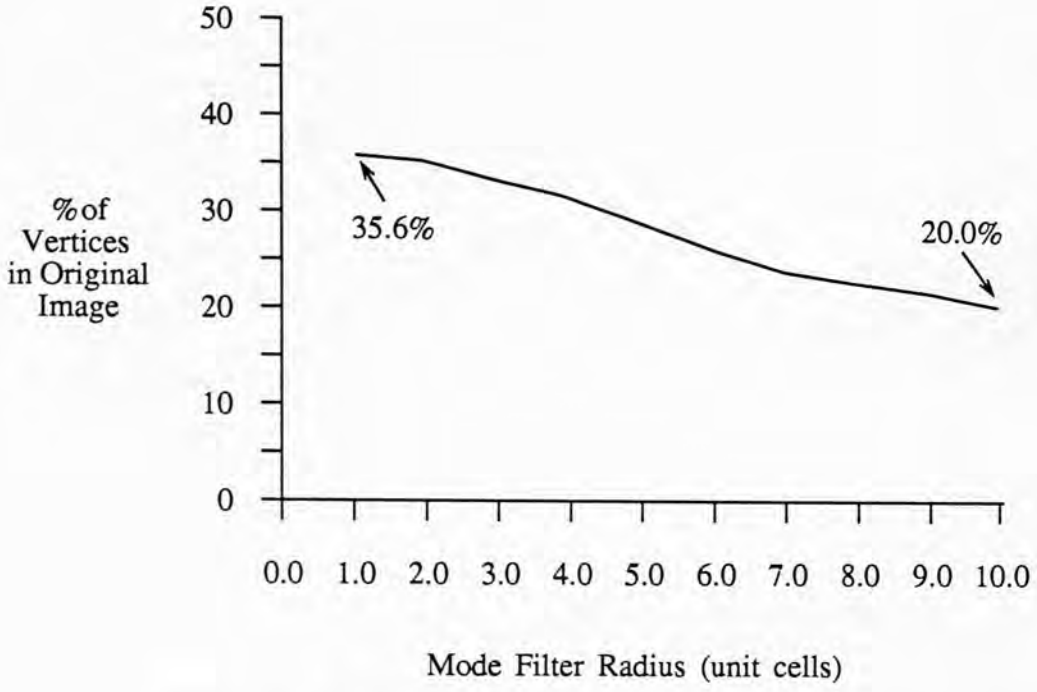


Figure 30. Mode Filter Effects on Vertex Reduction of City Image.

The gradual change in vertex count with respect to the mode filter radius indicates that the mode filter does not have a significant impact on the final number of vertices. The purpose of the mode filtering step is to improve the quality of the final image by blurring the image slightly before edge filtering. Blurring the image removes some of high frequency changes along region boundaries [Rosenfeld, 1969]. The subsequent edge filter usually produces a better fit to the smoother boundaries.

The edge filter threshold has a much more dramatic effect on the reduction of vertices. Figure 31 shows how a slight increase in the edge filter threshold causes a large drop in the vertex count for small values. The density of the raster is 512 by 512 unit cells and the mode filter radius is fixed at 1.0 unit cell. To achieve maximum reduction, the threshold value should be made as large as possible. The desired

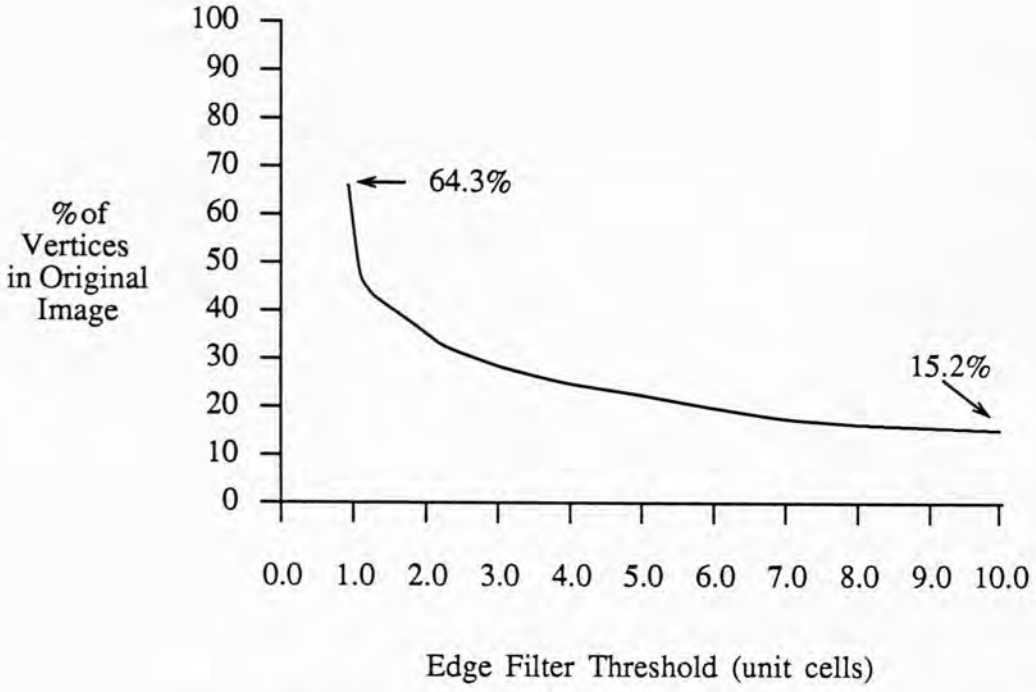


Figure 31. Edge Filter Effects on Vertex Reduction of City Image.

quality of the output image is the limiting factor in how large this threshold value can be.

Number of Cells

The number of areal features, or cells, in an image is also reduced. Figures 32 and 33 show how the mode filter radius and edge filter threshold affect the number of cells. Figure 32 is the 512 by 512 city image where the edge filter threshold is set to 2.0 unit cells and the number of cells is plotted as a function of mode filter radius. Figure 33 is the same image, but the mode filter radius is set to 1.0 unit cell and the number of cells is given as a function of the edge filter threshold. Cells are eliminated at a slower rate than the vertices within the image. Therefore, features from the original image will be retained as they pass through processing, but will be represented by fewer vertices.

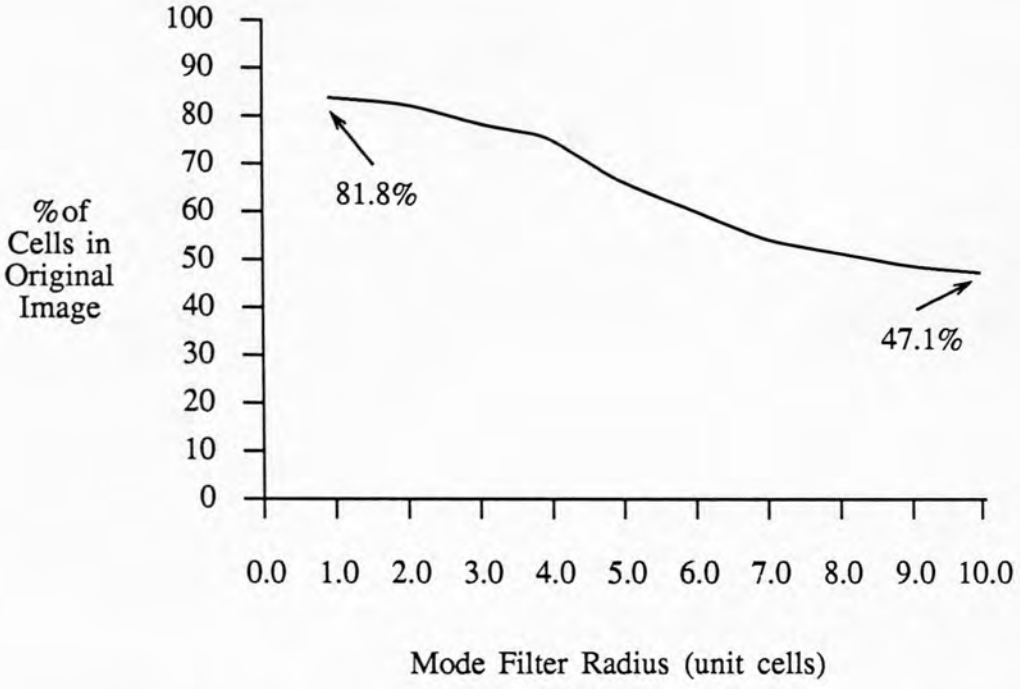


Figure 32. Mode Filter Effects on Cell Reduction of City Image.

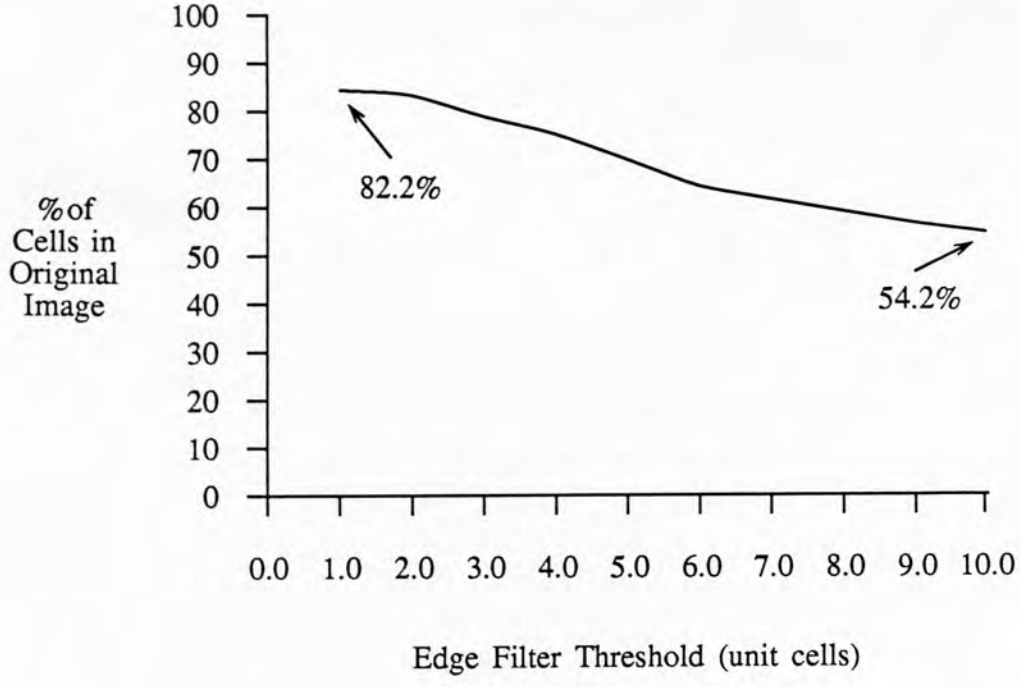


Figure 33. Edge Filter Effects on Cell Reduction of City Image.

Ideally, all features would be retained by the filtering process to preserve the original information content. The data reduction system, however, eliminates cells when their area is too small. Flight simulation data bases assume that small features are either unimportant and thus can be eliminated or that they are significant enough to bypass the general surface culture processing to be merged directly into the final data base. A small patch of trees, for example, can be discarded, but a small railroad yard might be added directly to the final data base. Small features that bypass processing might even have synthesized detail added to them.

Picture Quality

Figure 34 and Figure 35 show how the raster density affects the final image quality. For both runs, the mode filter radius is 3.0 unit cells and the edge filter threshold is 2.0 unit cells. Figure 34 places the original image into a 512 by 512 unit cell raster. Figure 35 painted the same image into a 128 by 128 unit cell raster. Using a lower density raster allows the data to be reduced early in the filtering process. Choosing an appropriate raster density depends on the smallest feature size in the original data and the resolution of the original data. For example, if a complex image is painted into a low density raster, then that image will be undersampled and small features will be lost. This undersampling also produces a rough looking image. At the other extreme, rasterizing the image at a resolution greater than provided by the original data produces no additional benefits. Defense Mapping Agency data typically has a resolution of 0.1 arc seconds. Depending on the area's complexity, DMA data can be rasterized at 6 to 40 percent of the original resolution and produce acceptable surface culture polygons for flight simulation data bases. The reason these percentages are so low is that DMA culture features generally do not contain vertices whose separation distances are near its smallest resolution.

Figure 34. City Image 512 by 512 Unit Cells.

Figure 35. City Image 128 by 128 Unit Cells.

Applying a lower resolution raster has the effect of moving the original vertices slightly rather than eliminating vertices by merging them.

Figures 36 and 37 show how the mode filter effects the final image quality. For both runs, the edge filter threshold is 3.0 unit cells and the raster density is 512 unit cells by 512 unit cells. A mode filter with a radius of 1.0 unit cell was applied during processing to produce the image in Figure 36. Increasing this radius to 10.0 unit cells yields the image in Figure 37. Several observations can be made by comparing the two results:

1. The smaller features have been eliminated by the larger radius filter;
2. The boundaries of the larger features have significantly less variation in them as well; and
3. Some of the larger features have merged with equivalent regions as a result of eliminating intervening features.

If the objective of filtering is simply to reduce data, the observations above do not present a problem. On the other hand, if a one-to-one correlation to the original set of features is necessary (e.g., for feature recognition), the elimination of small features and merging of large features may make correlation difficult. A small, or zero, mode filter radius may be necessary for improved correlation.

Figures 38 and 39 show how the edge filter affects the final image quality. For both runs, the mode filter radius is 3.0 unit cells and the raster density is 512 by 512 unit cells. Figure 38 is the result of using an edge filter threshold of 1.5 unit

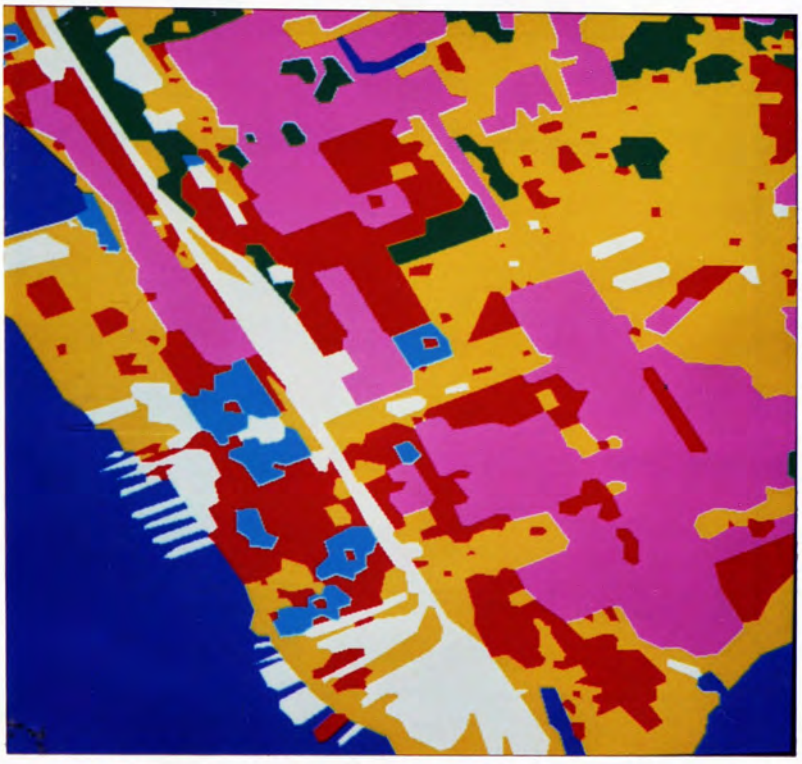


Figure 36. City Image Using Mode Filter Radius of 1.0 Unit Cell.

Figure 37. City Image Using Mode Filter Radius of 10.0 Unit Cells.

cells. Figure 39 illustrates the effect of increasing this threshold to 8.0 unit cells. The key points to be made about changing the edge threshold values are:

1. The smaller edge features are eliminated by a larger edge filter threshold; and
2. Increasing the edge filter threshold increases the angularity of features.

The filtering parameters of raster density, mode filter radius, and edge filter threshold are best chosen through a trial and error process. Aesthetic quality and vertex count are traded off against one another until an acceptable result is obtained. Acceptability depends on the limits and capabilities of the image generator intending to use the processed data. The reduction system's filtering parameters must be chosen with the intended application in mind. Through continued use and evaluation on a given application, these parameters can be chosen quickly with predictable results.

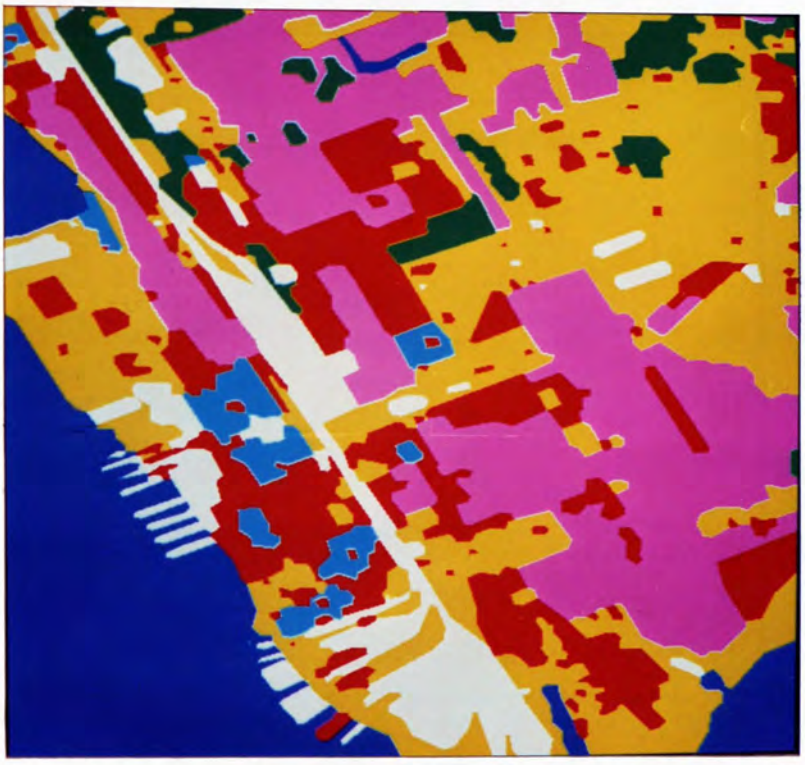


Figure 38. City Image Using Edge Filter Threshold of 1.5 Unit Cells.

Figure 39. City Image Using Edge Filter Threshold of 8.0 Unit Cells.

CHAPTER VII

USAGE AND CONCLUSIONS

Processing Continuous Color Images

Although this paper is based on very structured input data, there is nothing to prevent the algorithm from processing a continuous tone image. The only change needed is the definition of a new difference function for the segmenter and the bypassing of the rasterizer. Making these two changes creates a processing system that produces a stream of polygons from an input picture.

The difference function is used to test for an edge passing between two unit cells. The discrete nature of DMA data simplifies the difference function to be nothing more than a series of comparisons between the unit cell description fields. For example, two unit cells have an edge between them if they have different surface materials or if their feature identification codes are different. A continuous tone image, however, is made up of color-valued unit cells (or pixels). Testing for an edge in this case depends solely on the color components in a pair of unit cells. A common method of edge detection in pictures is to use a contrast detecting function which compares a unit cell to those surrounding it on a relative scale. Since the segmenter defines the boundaries around a homogeneous set of unit cells, the comparison of two unit cells must be done on an absolute scale rather than relative. A relative scale might cause edges to be lost where color and intensity values are changing slowly, thus creating cells that are made up of very different unit cell types. An absolute scale would force boundaries around a common set of unit cells, even in areas of low contrast. The process would be similar to the creation of contour maps from relief data.

Of course, the choice of appropriate threshold values is key to the usefulness of processing an image in this manner. Hall [1979] describes a method of threshold calculation based on a two-dimensional histogram of hue and intensity. The height at each hue / intensity pair is the number of pixels in the picture having that characteristic. A gradient following function can be used to locate thresholds by finding troughs between the histogram peaks. An automated threshold calculator such as this could be developed if, or when, the data reduction processing is applied to continuous tone images.

Possibilities for Feature Recognition

Given a difference function that can discriminate objects from the background scene, the segmentation and filtering processes could also be used to identify objects. Once the segmenter extracts a feature's outline from an image, the filters can reduce the outline to a simpler shape. Simplifying the shape reduces the sensitivity to slight variations in the detected boundary. This filtered shape can then be analyzed to determine characteristics such as area, perimeter, aspect ratio, center of mass, central moments of inertia, and axis of minimum moment of inertia. If rules can be established that classify certain objects in terms of the characteristics given above, then automated feature recognition is possible.

Gonzalez develops formulas for calculating the central moments of a region in a digital image and gives a list of seven moment invariants. These moment invariants are equations of normalized central moments and are constant for a given region regardless of its rotation, translation, or scaling [Gonzalez, 1977]. Eskenazi [1979] provides equations for calculating some of the central moments from chain coded outlines. The equations for calculating the centroid and axis of minimum

moment of inertia are given below. Equations 7-1 through 7-10 are based on Freeman chain codes that can take on one of eight possible values. The assignment of chain codes is shown in Figure 40. Each chain code, C_i, is a fixed length vector from the vertex (x_{i-1}, y_{i-1}) to the next vertex (x_i, y_i). A boundary outline is made up of $n + 1$ chain codes, where $i = 0, 1, \ldots n$. The change in the horizontal and vertical position is

$$\Delta x_i = x_i - x_{i-1} = \begin{cases} 1, \text{ if } C_i \in \{ 0, 1, 7 \} \\ 0, \text{ if } C_i \in \{ 2, 6 \} \\ -1, \text{ if } C_i \in \{ 3, 4, 5 \} \end{cases} \quad (7\text{-}1)$$

and

$$\Delta y_i = y_i - y_{i-1} = \begin{cases} 1, \text{ if } C_i \in \{ 5, 6,, 7 \} \\ 0, \text{ if } C_i \in \{ 0, 4 \} \\ -1, \text{ if } C_i \in \{ 1, 2, 3 \}. \end{cases} \quad (7\text{-}2)$$

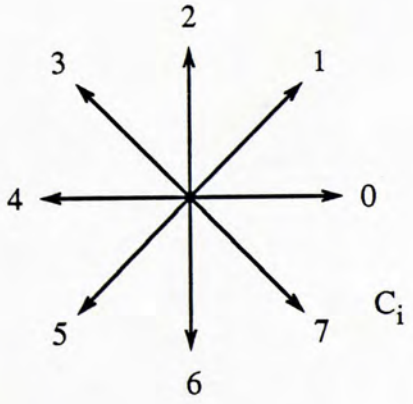


Figure 40. Eight Direction Chain Codes.

The area of a feature, A, or 0,0th order moment of inertia, is

$$A = M_{00} = \sum_{i=1}^{n} \Delta y_i x_i - \frac{1}{2} \Delta y_i \Delta x_i. \tag{7-3}$$

The higher order moments of inertia are

$$M_{10} = \frac{1}{2} \sum_{i=1}^{n} \Delta y_i x_i^2 - \Delta y_i \Delta x_i x_i + \frac{1}{3} \Delta x_i^2 \Delta y_i, \tag{7-4}$$

$$M_{01} = -\frac{1}{2} \sum_{i=1}^{n} \Delta x_i y_i^2 - \Delta x_i \Delta y_i y_i + \frac{1}{3} \Delta y_i^2 \Delta x_i, \tag{7-5}$$

$$M_{02} = -\frac{1}{3} \sum_{i=1}^{n} \Delta x_i y_i^3 - \frac{3}{2} \Delta x_i \Delta y_i y_i^2 + \Delta y_i^2 y_i \Delta x_i - \frac{1}{4} \Delta y_i^3 \Delta x_i, \tag{7-6}$$

$$M_{20} = \frac{1}{3} \sum_{i=1}^{n} \Delta y_i x_i^3 - \frac{3}{2} \Delta x_i x_i^2 \Delta y_i + \Delta x_i^2 x_i \Delta y_i - \frac{1}{4} \Delta x_i^3 \Delta y_i, \tag{7-7}$$

and

$$M_{11} = \frac{1}{2} \sum_{i=1}^{n} \Delta y_i x_i^2 y_i - \Delta x_i \Delta y_i x_i y_i + \frac{1}{3} \Delta x_i^2 \Delta y_i y_i - \frac{1}{4} x_i^2 \Delta y_i^2 - \frac{1}{2} \Delta y_i^2 x_i^2 + \frac{1}{3} \Delta x_i \Delta y_i^2 x_i \; . \tag{7-8}$$

The location of a feature's center of mass, (x_{cm}, y_{cm}), is

$$(x_{cm}, y_{cm}) = \left(\frac{M_{10}}{M_{00}}, \frac{M_{01}}{M_{00}} \right). \tag{7-9}$$

There is an axis that minimizes the moments of inertia. The angle, θ, between this axis and the x-axis is

$$\theta = \frac{1}{2} \tan^{-1}\left(\frac{2\,(M_{00}M_{11} - M_{10}M_{01})}{M_{00}M_{20} - M_{10}^2 - M_{00}M_{02} - M_{01}^2} \right). \tag{7-10}$$

The chain code scheme used by the segmenter has only four chain code directions (see Figure 11). Limited to the four directions, a single chain code can change the x position or the y position but not both. Therefore, $\Delta y_i \Delta x_i$ will always be zero. Equations 7-3 through 7-8 are reduced to

$$A = M_{00} = \sum_{i=1}^{n} \Delta y_i x_i, \tag{7-11}$$

$$M_{10} = \frac{1}{2} \sum_{i=1}^{n} \Delta y_i x_i^2, \tag{7-12}$$

$$M_{01} = -\frac{1}{2} \sum_{i=1}^{n} \Delta x_i y_i^2, \tag{7-13}$$

$$M_{02} = -\frac{1}{3} \sum_{i=1}^{n} \Delta x_i y_i^3, \tag{7-14}$$

$$M_{20} = \frac{1}{3} \sum_{i=1}^{n} \Delta y_i x_i^3, \tag{7-15}$$

and

$$M_{11} = \frac{1}{2} \sum_{i=1}^{n} \Delta y_i x_i^2 y_i - \frac{1}{2} \Delta y_i^2 x_i^2 - \frac{1}{4} x_i^2 \Delta y_i^2 . \tag{7-16}$$

The moments above can be used to describe a particular feature being sought. It is a simple matter to calculate these moments for each cell to check for a possible match. Some tolerance should be allowed to compensate for errors introduced by the filtering process.

Conclusions

The system presented meets the goal of reducing the amount of data required to describe areas of homogeneous surface culture. Each stage of processing is composed of simple operations, yet when combined, the system performs remarkably well. Reasonable image quality is maintained while reducing the number of polygon vertices by approximately seventy percent. The system also offers a high degree of controllability. One can specify the sampling density in the rasterization step, the mode filter radius, and the edge filtering threshold to achieve varied effects.

The data reduction system is arranged as a sequence of modules. Each module creates specific data structures, such as the image matrix, the node list and the cell list, from the information assembled up to that point. Informed decisions are made possible during processing by tightly coupling these data structures. In addition, the data structures allow for future enhancements by providing a great deal of information and making that information available to the other processing steps. The system is flexible and modular so that it is easily adapted to various types of image data.

Processing time is fairly short due to the simplicity of each of the processing steps. On a Digital Equipment Corporation MicroVAX^{TM}II system with 9 megabytes of physical memory, the time needed to filter an average 10 minute by 10 minute area as a 512 x 512 image, with a mode filter radius of 2 and an edge thresh-

old of 2, is less than 11 minutes. The processing time is most sensitive to the mode filter radius. Under the same conditions, but with a radius of 10, the processing time is just under 157 minutes. Rasterization time is not included as part of the times given because rasterization time depends solely on the amount of input information.

Further research could improve the performance of this system and find new applications for it. The improvements can be divided into two categories, module replacement and data analysis. Module replacement would be the implementation of different filtering techniques and use of different rules for decision making. Data analysis research would include the extraction or reformatting of the structured data to provide useful information.

A likely candidate for module replacement is the mode filter. The mode filter was used in this system because of its simplicity and ease of implementation. Other filters should be analyzed to test their effects and performance. Possible alternate filters include location weighted filters and data dependent filters.

The edge filtering module also warrants additional investigation. The maximum error technique used in this system was chosen for its simplicity. The filter's tendency to create angular boundaries might be corrected somewhat by an edge filter that chooses a line of best fit. If this improvement is made, however, special care must be taken to check cell simplicity. Edge crossing tests must be done not only on edges of a given cell, but also on the edges of surrounding, even nonadjacent, cells. In any event, the filter must produce the original edge, defined by the chain of fragments formed during segmentation, as the worst case filter; this insures the replication of the original data if it is not possible to filter the edge without causing a rule violation.

More work in the area of data analysis could lead to new applications. The discussion above concerning the possibilities for feature recognition is a good

example of future applications. Such research might produce results such as the automatic correlation of aircraft sensor data to a data base model containing target information.

REFERENCES

Caesar, Robert J. and Jonathon L. Lin. "Data Base Generation from DMA Data." Daytona Beach, Florida: General Electric Simulation and Control Systems Department, February 11, 1986.

Castleman, Kenneth R., *Digital Image Processing.* Englewood Cliffs, New Jersey : Prentice-Hall Incorporated, 1979.

Defense Mapping Agency Aerospace Center. "Product Specifications for Digital Landmass System (DLMS) Data Base." Second Edition. Defense Mapping Agency, St. Louis, Missouri, April, 1983.

Eskenazi, R., and J. M. Wilf. "Low-Level Processing for Real-Time Image Analysis." National Aeronautics and Space Administration. Jet Propulsion Laboratory. Pasadena, California, September, 1979.

Foley, James D., and Andries Van Dam. *Fundamentals of Interactive Computer Graphics.* Reading, Massachusetts: Addison-Wesley Publishing Company, 1982.

Freeman, Herbert. "On the Encoding of Arbitrary Geometric Configurations." IRE Transactions on Electronic Computers. The Institute of Radio Engineers, Inc. New York : June, 1961. Pages 260-268.

Freeman, Herbert. "A Review of Relevant Problems in the Processing of Line-Drawing Data." *Automatic Interpretation and Classification of Images.* Edited by A Graselli. New York : Academic Press, Inc., 1969.

Gonzalez, Rafael C., and Paul Wintz. *Digital Image Processing.* Reading, Massachusetts : Addison-Wesley Publishing Company, 1977.

Hall, Ernest L. *Computer Image Processing and Recognition.* New York: Academic Press, Inc. 1979.

Kernighan, Brian W., and Dennis M. Ritchie. *The C Programming Language.* Englewood Cliffs, New Jersey: Prentice-Hall Incorporated, 1978.

Pavlidis, Theodosios. *Algorithms for Graphics and Image Processing.* Rockville, Maryland : Computer Science Press, Inc. 1982.

Rosenfeld, Azriel. "Figure Extraction." *Automatic Interpretation and Classification of Images.* Edited by A Graselli. New York : Academic Press, Inc. 1969.

Steiner, Walter R. General Electric Company, Daytona Beach, Florida. Personal communication, 1988.

Thomas, George B. Jr., and Ross L. Finney. *Calculus and Analytic Geometry*. Chapter 11, "Vectors and Parametric Equations." Reading, Massachusetts: Addison-Wesley Publishing Company, 1981.