SHAPE REPRESENTATION IN COMPUTER VISION

Bryant W. York

COINS Technical Report 81-13

May 1981

This work was supported in part by The National Science Foundation under Grant Number MCS75-16098 A01.

SHAPE REPRESENTATION IN COMPUTER VISION

A Dissertation Presented

by

BRYANT WHITTIER YORK

Submitted to the Graduate School of the
University of Massachusetts in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

May 1981

Computer and Information Sciences

This work was supported in part by

The National Science Foundation

under Grant Number

MCS75-16098 A01

SHAPE REPRESENTATION IN COMPUTER VISION

A Dissertation Presented

by

BRYANT WHITTIER YORK

Approved as to style and content by:

Dr. Edward M. Riseman, Chairman of committee

Dr. Allen R. Hanson, Member

Dr. Michael A. Arbib, Member

William Kilmer

Dr. William L. Kilmer, Member

Dr. Robert N. Moll, Member

Edward M. Riseman, Department Head
Computer and Information Sciences

In memory of my mother, Mabel York

ACKNOWLEDGEMENTS

I am sincerely grateful to Dr. Edward Riseman and Dr. Allen Hanson for directing this dissertation. Both provided useful guidance and support. I am also grateful to the other members of my dissertation committee, Dr. Michael Arbib, Dr. William Kilmer, and Dr. Robert Moll, for many helpful consultations and useful insights.

No acknowledgement would be complete without citing the other graduate students who toiled alongside me and provided substantial intellectual stimulation. It is impossible to remember all of the people who ever worked on the VISIONS project and though I explicitly mention only a small subset, I appreciate the efforts of those who served before my time. I thank Tom Williams for making me aware of the importance of surfaces; John Lowrance for providing the GRASPER language and many discussions of data base organization for scene analysis; Ralf Kohler, Paul Nagin, and John Prager for providing the image segmentations which allowed me to try some of my ideas; Frank Glazer for many helpful discussions of shape and geometry; and the late Cesare Parma for his opinions on spatial processing and matching. Many thanks to Leonard Wesley for LISP and GRASPER support during the most recent re-implementation of this work.

I am especially grateful to Art Karshmer for introducing me to computer graphics and I would also like to thank Daryl Lawton for constantly reminding me of the importance of questioning conventional ideas and approaches to problem solving.

I thank the National Fellowship Fund in Atlanta, Georgia for its financial support during the 1977-78 academic year.

I am indebted to the IBM Research Laboratory in San Jose, California for making available its computing facilities and its document composition system for the production of this thesis. I am particularly grateful to my immediate managers, Eric Carlson and Bob Taylor, for applying the proper amounts of pressure at the appropriate times, to colleagues, Jim King and Kathy Hitchcock, for tremendous help with using the photocomposer and document composition software, and to technical illustrator, Karen Bryan, for the professional art work apparent in some of the figures. I am solely responsible for the drawings of lesser quality.

Finally I wish to express special thanks to Debra Feinman. She has provided a great deal of emotional support throughout this ordeal. In addition, she typed the drafts of this manuscript.

ABSTRACT

Shape Representation in Computer Vision

June 1981

Bryant Whittier York

A.B., Brandeis University

M.S., Massachusetts Institute of Technology

M.S., University of Massachusetts

Ph.D., University of Massachusetts

Directed by: Professor Edward M. Riseman

The recognition ability of a computer vision system is critically dependent upon the quality of its internal representations for the objects it wishes to recognize. In this dissertation the problem of designing a suitable internal representation for three-dimensional(3D) objects is addressed and one of our major goals is to refocus attention upon surface-based representations. In support of this goal a particular surface-based representation is presented and it is compared with existing volume-based representations. It compares favorably with respect to certain demanding criteria -- scope and uniqueness, stability and sensitivity, and accessibility.

In our representation 3D objects are described by structured collections of surface patches and each primitive patch is described in the interpolative patch form first introduced by the late S. A. Coons. Boundary curves and blending functions are represented by cubic

B-spline curves and these curves, in turn, are represented by their defining "vertex polygons". Some of the nice properties of this representation include (1) broad scope -- the ability to represent both polyhedral and curved objects, (2) direct computation of many gestalt shape features, (3) shape classification on both structural and intrinsic features, and (4) economy of storage.

The major part of this thesis is devoted to description of our surface-based representation and analysis of its properties. Several exmples of objects described in this representation are presented and these objects are analyzed with respect to the gestalt shape features which are introduced. Some effort is devoted to the problem of extracting shape descriptions from 2D images and an algorithm for curve fitting 2D cubic B-splines to 4-connected digital curves is presented. Finally, the impact of our representation on matching techniques is explored in the context of 2D scene analysis.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION AND LITERATURE REVIEW

1.1 Background

Since the inception of Artificial Intelligence (AI), representation of information has been an issue of fundamental importance and its significance is no more strongly apparent than in the sub-discipline of computer vision. Computer vision presents us with a wide range of representation problems; however, in this thesis only those which are relevant to shape are addressed. In particular, we study that subclass of computer vision systems which store internal models of three dimensional (3D) objects and which interpret two dimensional (2D) images of scenes containing such objects.

The problems of shape representation and recognition in this class of systems are enormously complex and general solutions are not available. The prevailing approach to 3D shape representation encodes object descriptions as collections of atomic units of identical shape. Usually a simple geometric volume such as a cylinder, cone, or sphere is the atomic unit, and the range of shapes which can be realized in a particular representation is determined by the structuring mechanisms for putting together atomic units to form larger units.

Although this approach has a certain appeal, it has two serious drawbacks:

(i) The atomic unit is a volume. Volumes may be compactly represented; however, they do not afford easy access to the surface information which is so often required for shape analysis. For instance, the shape of the surface of a human torso is not easily determined from a collection of spheres and their spatial dispositions with respect to one another.

(ii) All atomic units have identical shape. This is an unnecessary restriction required only to facilitate implementation.

In this thesis a 3D shape representation is presented which overcomes these two difficulties. The primitive or atomic unit is a Coons surface patch. All surface patches have identical internal structure and thus may be manipulated in a uniform manner by feature extraction and recognition algorithms; however, a uniform internal structure does not imply a single shape. The shape of a surface patch depends on the shapes of some of its subatomic components. These components are the boundary curves and the boundary tangent vector curves of the patch. In addition there are components which do not have an intrinsic shape, but which help to determine the shape. They are the corner position vectors, the boundary tangent vectors, and the corner twist vectors.

It may seem that this representation is more complex than those in which the primitive is a simple volume such as the sphere. However, there is a tradeoff between complexity of the primitive and the number of primitives required to describe a complex object; generally the simpler the primitive, the more primitives are required. Our representation is a suitable compromise between extreme simplicity and extreme complexity of the primitive. For sufficiently complex objects such as those that appear in natural scenes, our representation

compares well with the generalized sphere and generalized cylinder representations as is demonstrated in Chapter II.

In addition our representation has certain nice properties not offered by other representations. First, both polyhedral and curved objects are captured in a single representation, an extremely useful feature for representing objects which have both polyhedral and curved components. With our representation it is no longer necessary to decompose the object into curved pieces and polyhedral pieces; an object may be partitioned along natural boundaries regardless of the shapes of its parts. Secondly, both global and local shape features may be easily computed from the internal representation. Many of these features have 2D analogues which were discovered to be useful measures in studies of human visual perception of form. Finally, our representation has a significant impact on the design of recognition algorithms. Algorithms for matching 3D generalized cylinder models are faced with the problem of finding the principal axis of the cylinder in the image of the object. This problem becomes combinatorily explosive without additional information to constrain the possible positions and orientations for the projection of the axis in the image. Our representation shows promise for matching of projections of surface patches to regions of images and the matching process should be more highly constrained by the relationships between the components of the patch. Possible algorithms for matching surface patch models are outlined in Chapter V.

The remainder of this chapter gives an overview of the thesis and discusses the relevant literature.

I.2 Overview of Thesis

This chapter introduces the problems of 3D shape representation and 2D scene analysis. The emphasis is on the broad nature of the problems and the wide range of approaches. Related work is reviewed and specific representations are briefly described.

Chapter II delves into the problem of 3D shape representation in great detail and several generally accepted criteria for the design of a good shape representation are presented. Our representation is described, measured against those criteria, and compared to the popular generalized cylinder representation. Significant shape features are presented together with an analysis of those features for a small collection of objects.

Chapter III is primarily concerned with the structure of knowledge about shape and access to that knowledge. It is shown how 3D object models are represented and accessed in the long term memory of the VISIONS system. The representation of shape features and the indexing mechanism are described in great detail. Certain experiments are performed on a small data base to demonstrate the logical connection structure and certain performance considerations.

Chapter IV deals with the representation of 2D shape and the extraction of shape features from a 2D image. An algorithm for curve fitting of 2D cubic B-splines to 4-connected digital curves is presented. The algorithm is applied to certain standard 2D geometrical shapes as well as to actual data.

Chapter V proposes an as yet untested approach to object matching based upon our representation. Chapter VI summarizes the contributions of this thesis and makes suggestions future work.

The thesis is supplemented by three appendices containing material which would disrupt the continuity of the text if it were included in the chapters; however, this material is essential to the development presented here. Appendix I is a brief introduction to splines and Appendix II describes surface patch interpolation, with particular emphasis on the Coons surface patch interpolation formula. Appendix III contains a significant portion of the network which represents stored knowledge about objects and their shapes; the complete network is too voluminous to include in this document.

I.3 Terminology and Environment

Within this dissertation the terms computer vision, machine vision, and scene analysis are used synonymously to denote any computer vision system whose purpose is the analysis and interpretation of visual information. In our research environment a scene is a collection of 3D real world objects in close proximity to one another. An image or picture is a projection of a scene onto some 2D medium. Perspective projection is the mode of projection and the 2D medium is photographic film. The photographic image is digitized to produce three 2D arrays of numbers, one for each of the primary colors (red, green, blue). The size of the 2D digitized image is generally 256x256 picture elements, or "pixels". Six bits of intensity are encoded for each of the primary colors, thus allowing 64 discrete intensity levels for each primary color at each pixel. Figure 1 is a gray scale reproduction of a sample image depicting a common house scene.

The scene analysis problem involves the transformation of these image arrays into symbolic descriptions of the depicted scene. In order to define an interpretation of a scene, we must first define the language of the symbolic descriptions, and secondly the transformation process. For our purposes an interpretation, or high level description of a scene, consists of the names of the objects in the scene, the association of certain physical attributes such as color, shape, and size, and the locations of objects in 3D space relative to the viewer or camera. Figure 2 shows an example of a desired (partial) symbolic interpretation of the image in Figure 1. Given this restricted definition of interpretation, the problem of interpretation reduces to the specification of transformations which map a digital image onto a high level description.

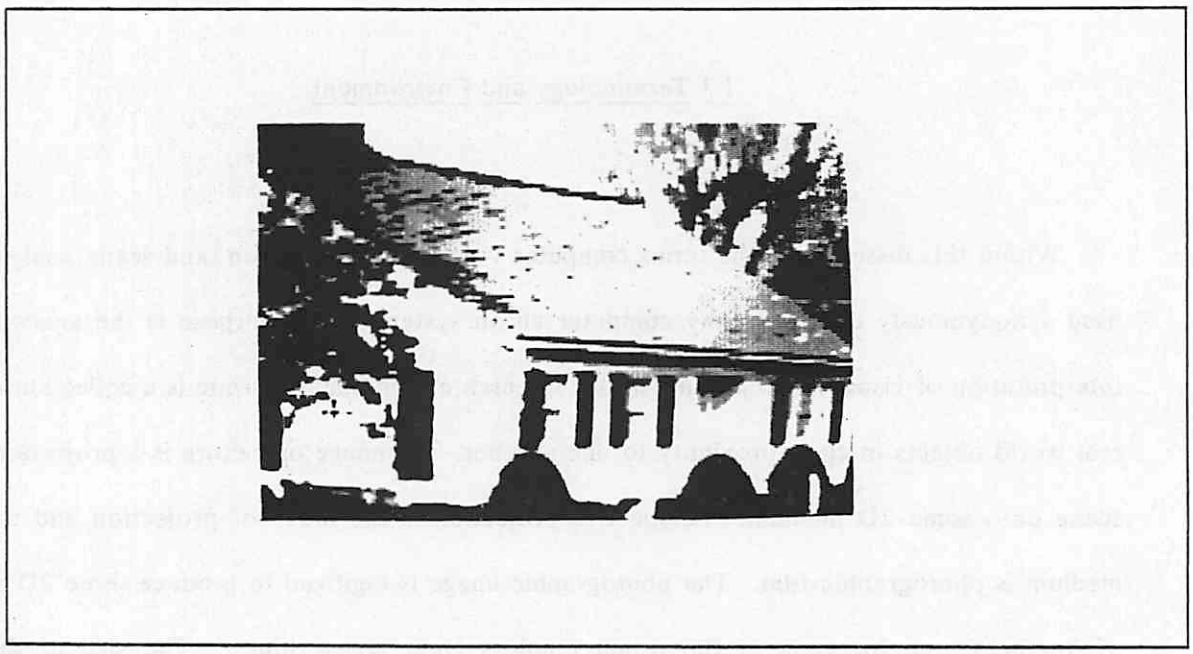

Figure 1: Image of a common house scene

There are a variety of models for computer visual perception; however, the most common model consists of the three steps, sensing, segmentation into regions or surfaces, and interpre-

tation. The VISIONS system (Hanson and Riseman 1978a,b) is a particular architecture for this paradigm and it provides the specific context within which this research was conducted.

```
OBJECTS
    HOUSE
        FRONT-WALL (COLOR WHITE) (NUMBER 1) ...
        FRONT-WINDOWS (NUMBER 3) ...
        FRONT-WINDOW-SHUTTERS (NUMBER 6) (COLOR BLACK)
            (SIZE 3) (AR 1:3) ...
        SIDE-WALL (COLOR WHITE) ...
        ROOF :SHAPE PITCHED) (COLOR GRAY) ...
        SIDE-WINDOW-SHUTTER (NUMBER 2) (COLOR BLACK)
            (SIZE 6) (AR 1.5:4) ...
    GARAGE
        ROOF (COLOR GRAY) (SHAPE PITCHED) ... RELATIONS
            APPROX VIEW FROM LEFT OF FRONT-DOOR
            (HEIGHT 5) (DISTANCE 40) ...
            TREE-1 PARTIALLY OCCLUDES HOUSE-SIDE-WALL

    ...
```

Figure 2: A symbolic partial interpretation of a scene

The salient feature of the VISIONS architecture is the use of intermediate levels of representation in order to decompose the interpretation process into many, better-understood sub-transformations. A brief overview of the VISIONS architecture follows.

I.3.1 VISIONS Architecture.

The VISIONS system embodies two of the stages of the visual perception process -- segmentation and interpretation. For this reason the system may be thought of as consisting

of two major subsystems, a low-level system for segmentation and a high-level system for interpretation. The terms low-level and high-level have no special significance other than the indication of our arbitrary subdivision of the perceptual process.

The obvious input to the low-level subsystem is the set of three arrays of color intensities. Less obvious are the outputs and the processes by which inputs are transformed into output. The form of the output of the VISIONS low-level system is a list of regions and an associated set of features for each region. Each region has a boundary curve which is expressed as the list of x-y coordinates of the digital curve which surrounds the region. Adjacent regions share common portions of their respective boundaries; thus, depending upon the number of regions adjacent to a given region, the boundary of the given region will be divided into a varying number of segments. A region's boundary may be represented as a list of segment identifiers and associated with each segment identifier is the the list of x-y coordinates for that portion of the region's boundary which the segment comprises. Adjacent segments meet at a point and these points are referred to as vertices (or picture junctions).

The output of the low-level system is represented as a graph with three kinds of nodes -- region nodes, segment nodes, and vertex nodes. Figure 3 is a sample portion of the graph structure generated by the low level system. It is referred to as RSV for Regions, Segments, and Vertices. Each region node is connected by arcs in the graph to the segment nodes which comprise its boundary and each segment node is connected to the two vertex nodes which represent the endpoints of the corresponding segment. Associated with each region node is a set of primitive visual features, such as color, centroid, and area. Color is actually a complex feature containing distributions of hue, saturation, and intensity; however, color information is not currently used in determination of shape.

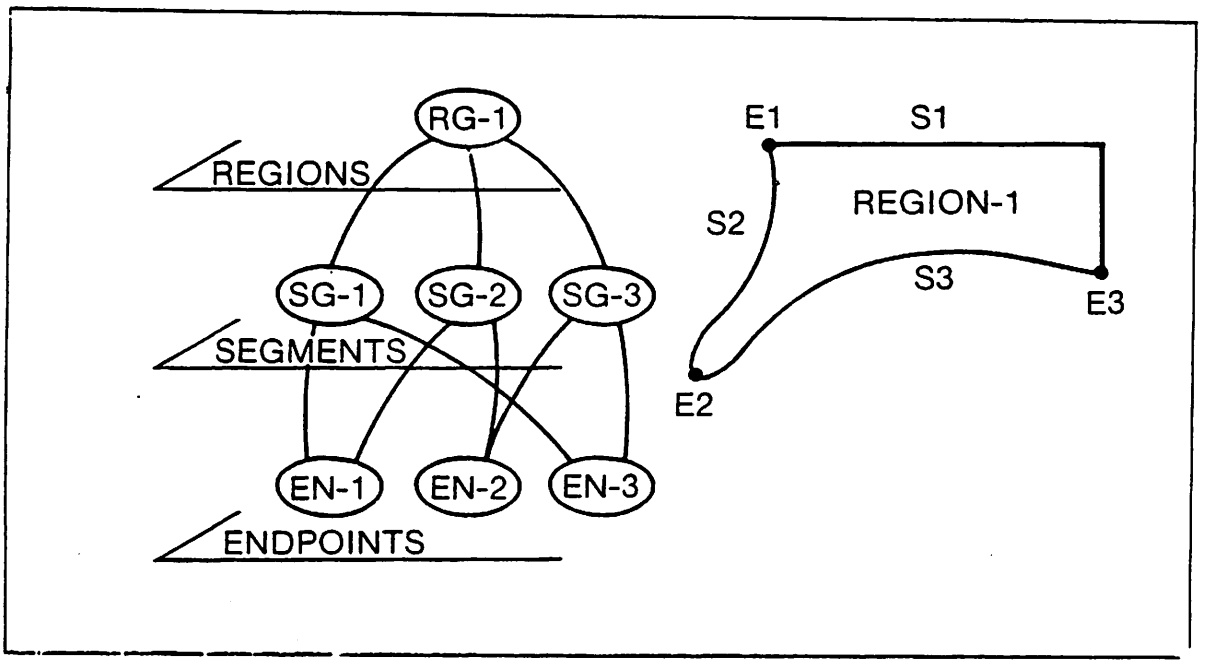

Figure 3: A portion of an RSV graph structure

The two basic techniques for segmenting a digital image are region analysis and edge analysis. Region analysis collects pixels into groups on the basis of common primitive features and partitions the image based upon these groupings and spatial proximity rules. Once the image is partitioned into regions, the boundaries between the regions as well as their segments and endpoints are derived in a straightforward manner.

Edge analysis determines the existence of edges in the digital image on the basis of contrast between features of adjacent pixels. Once the edges have been found, tracking algorithms are used to link adjoining edges into digital curves. Endpoints are specified where digital curves meet or where a single digital curve ends. Regions can be determined by extracting a succession of digital curves which form a closed boundary.

Clearly, both region and edge analysis are capable of producing the output graph described above and the low-level system makes use of both approaches to produce segmenta-

tions. Since this dissertation is concerned only with the output of the low-level system, the interested reader is referred to (Nagin 1979), (Prager 1979), (Hanson and Riseman 1978a), (Hanson, Riseman, and Glazer 1980) for more detail on segmentation.

The output graph of the low-level system is the input source for the high-level system. The output of the high-level system is an interpretation, i.e. a model of the 3D scene depicted in the image represented by RSV. The 3D model consists of a list of objects, their locations and orientations in a 3D world coordinate system, and associations between those objects (their 3D surfaces, edges, and vertices) and the regions, segments, and vertices of RSV.

In order to produce a 3D model the high-level system must possess a great deal of knowledge about the physical world. For example, it must make use of the physics and geometry of the image formation process, Euclidean geometry, and the physics of solid matter. In addition the system must have an effective means of representing and applying that knowledge in the image interpretation process. In the VISIONS system, knowledge is separated into two distinct types, short-term (or image-specific knowledge) and long-term (or general knowledge).

Let us consider long-term knowledge first. The long term memory (LTM) of VISIONS is decomposed into a number of levels of abstraction (see Figure 4). Certain common entitities are stored at each level and relationships between those classes of entities (objects, volumes, surfaces,...) may be intra-level or inter-level. The choice of levels shows a bias toward shape and spatial interpretation, and this bias is natural since shape is the only invariant feature of every rigid object.

Figure 4 is an example sketch of a partial interpretation and the decomposition of declarative knowledge. The knowledge is divided into: 1) a hierarchy of levels of representa-

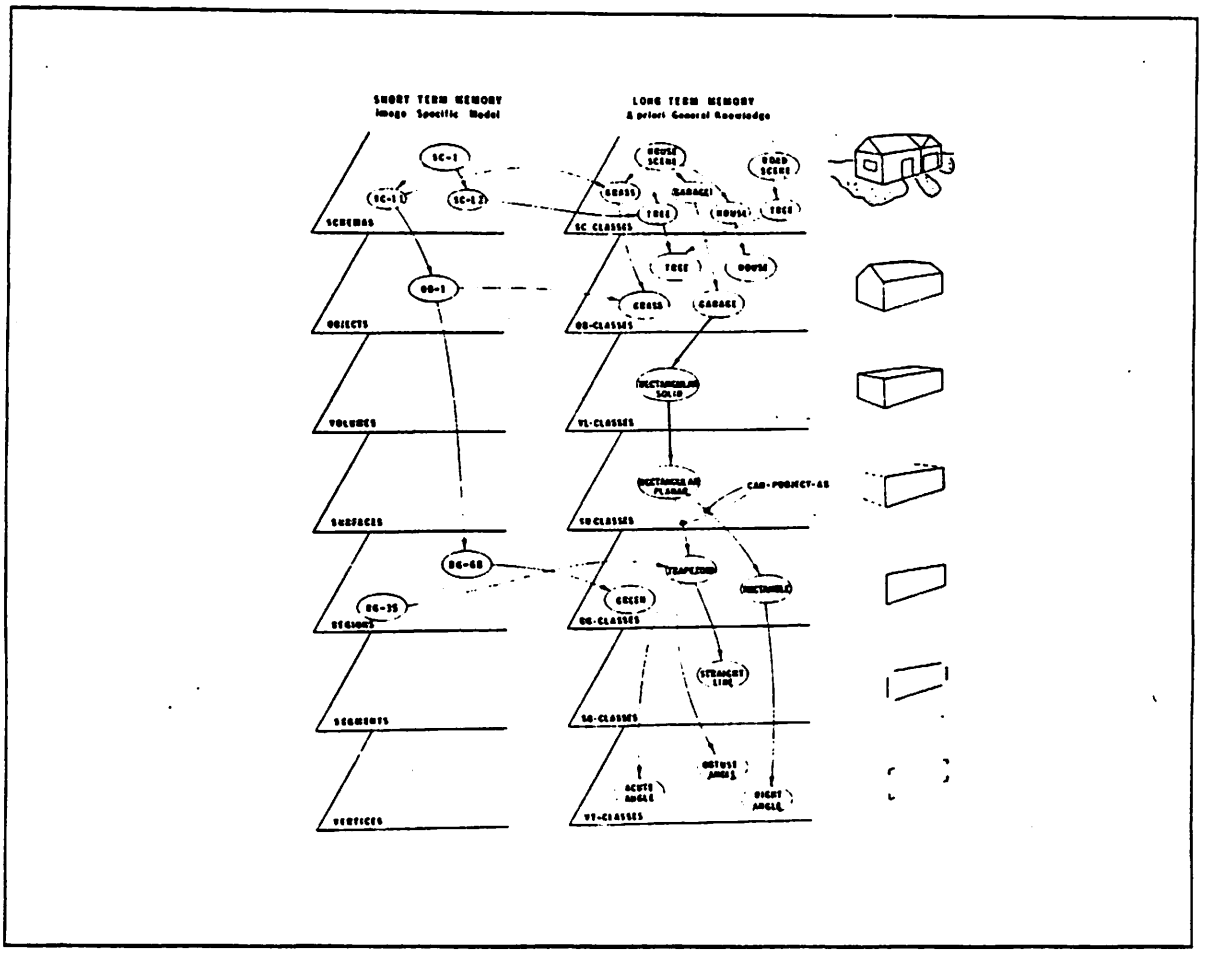

Figure 4: Hierarchical decomposition of knowledge

tion defining the key levels of abstraction which are necessary for a general system of visual perception; and 2) short-term memory (STM) representing an interpretation of the specific image and long-term memory (LTM) representing general visual knowledge of the world.

The short-term memory consists of the same levels of abstraction as the long-term memory; however, the lower three levels contain the input data (RSV) and the upper four levels organize the world coordinate system in which the 3D model is constructed. When an interpretation is complete, the STM contains the interpretation. The upper four levels contain the 3D model and the edges down into RSV contain the associations of objects to regions,

segments, and endpoints. STM and LTM are represented naturally by graph structures; associated with each node in the graph is a list of attribute-value pairs representing the features of the entity which the node represents.

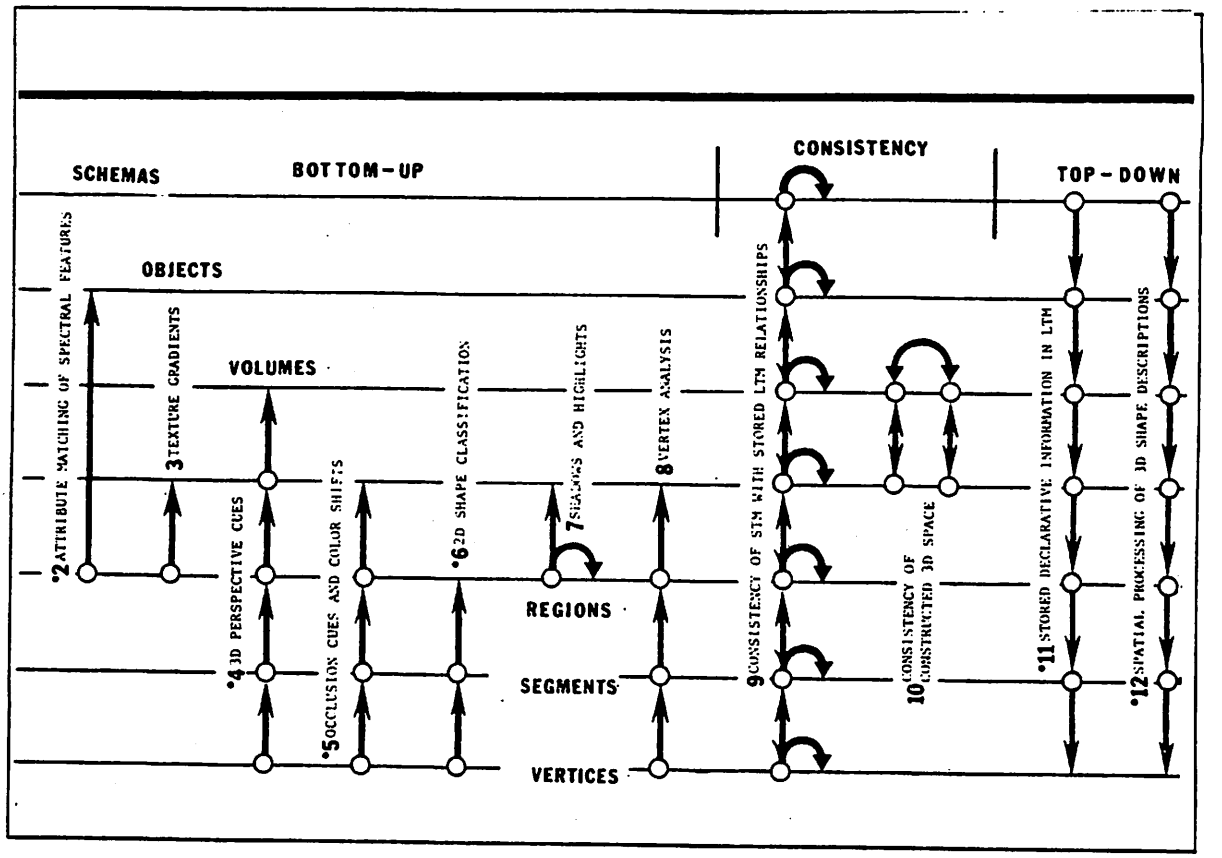

Figure 5: Knowledge sources in VISIONS[1]

The construction of a 3D model results from the application of knowledge which is embodied in procedures called knowledge sources(KS's). Figure 5 outlines the KS's in the VISIONS system and depicts the levels of abstraction upon which they work. There are both bottom-up and top-down paths for hypotheses. Knowledge sources, using both declarative and procedural forms of knowledge, can generate and verify hypotheses along many paths through the multi-level representation. These are some of the prime cues in static, monocular, 2D color images of 3D scenes. Additional paths are available from motion and stereo data.

Paths 6 and 12 are of particular interest in this thesis. Path 6 is directly concerned with the analysis and classification of shapes of regions which appear in 2D images. This process involves curve fitting the digital curves which comprise a region's boundary and classifying the shapes of the the fitted curves. Path 12 has to do with the top-down matching of a 3D model of an object to the shape descriptions derived from a 2D image. For a more detailed description of the high-level system, see (Hanson and Riseman 1978b, Parma, Hanson, and Riseman 1980). In this dissertation our primary concerns are the detailed representation of 3D objects in LTM and the KS's which are useful in the recognition of shape.

1.4 Literature Review

The review of related work is divided into four basic areas. Each area has a theme which becomes apparent as the reader delves more deeply into the dissertation. The first section is concerned with theories of form. Its theme is that, despite the absence of a theory of form perception, existing experimental results in the psychophysics of form perception can be effectively utilized to guide research in the computer perception of form. The second section presents the range of currently available 3D shape representations in computer vision. The outstanding characteristics of each representation are briefly outlined. The third section discusses 2D shape representations and feature extraction techniques, while the final section presents certain relevant topics from the fields of computer graphics and computer-aided geometric design.

I.4.1 Theories of form.

In the early 1950's Attneave (Attneave 1954) began to develop the notion of perception as an information processing task. He conjectured that much of the sensory information received by humans is redundant and that such redundancy is exploited during perceptual tasks. He devised a simple experiment to validate his conjecture and based upon the results of that experiment he drew the following conclusion:

> "It is evident that redundant visual stimulation results from either (a) an area of homogeneous color (color is used in its broad sense here, and includes brightness), or (b) a contour of homogeneous direction or slope. In other words information is concentrated along contours (i.e. regions where color changes abruptly) and is further concentrated at those points on a contour at which its direction changes most rapidly (i.e. at angles or peaks of curvature)."[2]

As visual evidence of this assertion, he offered a drawing (see Figure 6) in which he connected 38 points of maximum curvature from the contour of a sleeping cat, and he noted how remarkably easy it is for humans to recognize the drawing as that of a sleeping cat. Attneave also asserted that symmetry is a source of redundancy and that a good gestalt is a figure with a high degree of internal redundancy. These assertions form the foundation upon which much of this dissertation rests.

Attneave saw the need for a psychophysics of form and the quantitative study of shape. This need led to a further study by Attneave and Arnoult (Attneave and Arnoult 1956). They carefully pointed out that "shape is a multi-dimensional variable and that the number of dimensions necessary to describe a shape is not fixed or constant but increases with the complexity of the shape."[4] They developed a procedure for generating arbitrary closed planar figures, both polygonal and curved. Sample shapes were generated, the contours were

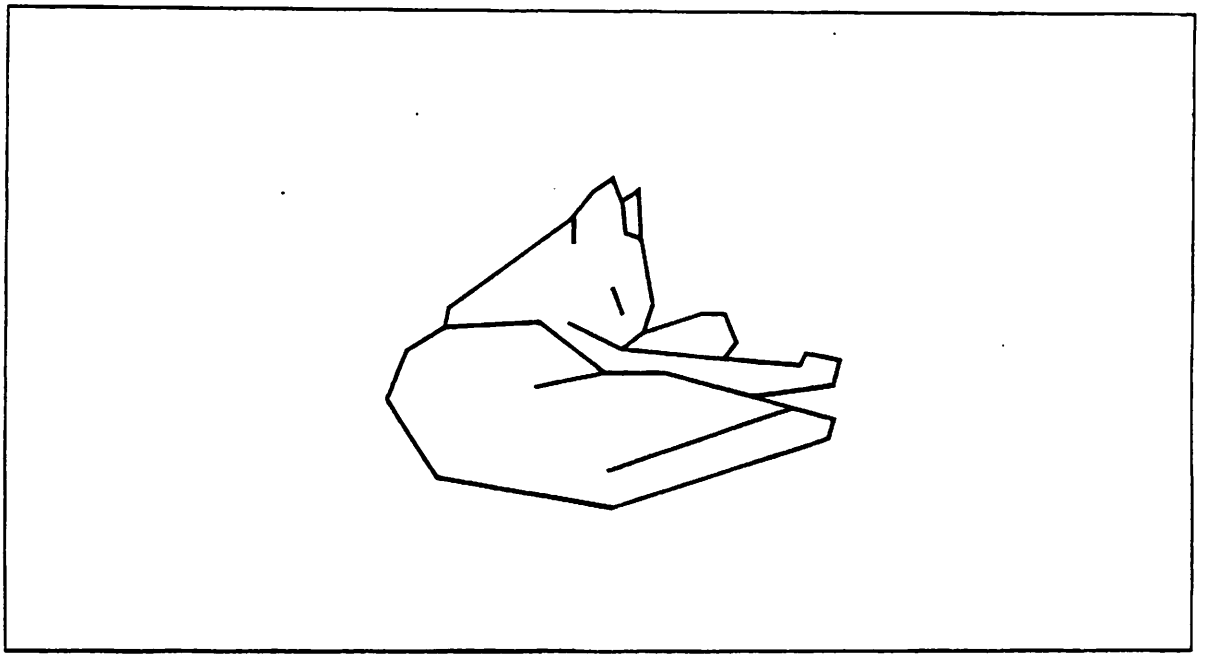

Figure 6: Attneave's cat[3]

analyzed, and certain gestalt variables were measured. Attneave and Arnoult drew the following conclusions:

(1) The number of tricoordinates required to describe a shape constitutes a first order approximation of its psychological complexity. (Tricoordinates describe the shapes of successive sides of a planar polygon, taken in pairs. There are many systems of tricoordinates (see (Attneave and Arnoult 1956, p.465)).

(2) Perimeter squared divided by area (P^2/A) is an important gestalt measure of shape. P^2/A is a measure of compactness and a related measure $D = 1 - ((2\sqrt{\pi A})/P)$ is a measure of dispersion (D).

(3) A deeply convoluted or jagged figure will have a high dispersion value, as will a thin rectangle or a thin ellipse.

(4) The circle is the most compact figure possible and has a value of D=0.

These preliminary results created an appetite for more quantitative studies of shape. Questions arose as to the roles of these features in human perception for judging the complexity of a shape and in determining the similarity of two shapes.

Attneave (Attneave 1957) studied the impact of the five features on the judged complexity of certain closed planar shapes. The features were curvedness, symmetry, number of turns, P^2/A, and angular variability. The results were obtained for 4- and 12- sided planar polygons randomly generated and they varied only slightly with the number of sides. Number of turns was by far the most significant variable in the determination of judged complexity, accounting for nearly 80% of the variance. Surprisingly, symmetric shapes were judged more complex than asymmetric shapes and curvedness did not significantly affect judged complexity. P^2/A and angular variability were partially correlated and together they accounted for approximately 7% of the variance. For image resolutions of 8x8 up through 64x64 there was no significant variance. Based upon the results of these experiments Attneave developed a statistical predictor of judged complexity; however, he did not report on the performance of his predictor on other data samples.

Attneave's results spurred new research into quantitative measures of shape and in the use of such measures for form discrimination. Brown and Owen studied the psychological literature on quantitative measures of shape and produced a thorough analysis (Brown and Owen 1967) of the types of measures, their correlations and uses. They were able to find 130 separate measures in the literature. This number was reduced to 80 by a screening process which eliminated similar measures which were highly correlated. Most of the measures were based upon sizes of angles, side lengths, area units, perimeter units, radial lengths, or coordinates. Also, the first four moments were computed for interior angles, side lengths, radial lengths, perimeter and areal distribution. These measures were statistically analyzed for

correlations in an attempt to find the number of independent dimensions of shape. A number of key results were obtained.

Compactness (P^2/A) correlated very highly with several of the 80 measures. The measures which correlated most highly with compactness included (1) the largest complementary radial pair, (2) the area of the enclosing rectangle, (3) the first four moments of radial lengths, (4) the second moment of side lengths, (5) the second areal moment of x, (6) the second perimeter moment of x, (7) the maximum second perimeter moment of x, and (8) the maximum second areal moment of x.

Jaggedness, a measure based upon the relative magnitudes of interior angles, was highly correlated with all measures based upon angles. Skewness of the area and perimeter with respect to the y axis was found to correlate highly with the mean of the y coordinate, the y coordinate of the first moment of area and the third areal and perimeter moments of y. Similar results were obtained for skewness with respect to the x axis. Finally, directionality (direction of the dominant axis) was found to be highly correlated with the second areal and perimeter moments of x, the second moment of area and perimeter of y, and the variance of the x and y coordinates.

Important guides for perceptual research are implied by these results. For instance, Brown and Owen suggest that, because of the high degree of correlation among shape measures one can expect to effectively use a limited number of them to obtain reasonable predictions of form; however, they make no claims as to the efficacy of these measures in form discrimination tasks.

In this thesis we explore ways in which to exploit these results in computer systems for the perception of form by developing shape measures for 3D objects. The objects must have a suitable geometric representation to permit the computation of such measures.

I.4.2 Three-dimensional shape representations.

One of the earliest successful attempts at the machine perception of form was achieved by Roberts (Roberts 1965). He developed a program for recognizing certain polyhedra (cubes, wedges, and hexagonal prisms). Basically his system worked in a bottom-up fashion on scenes consisting of one or more blocks on a flat surface. The grey scale image of the scene was segmented with a simple edge-detection operator (referred to as the Robert's cross operator). Feature points were selected and probable orientations for lines through them were hypothesized. Certain reduction techniques were used to remove spurious points and finally a sequential least mean-square-error procedure was used to fit straight lines to the remaining points. The output of this whole process was a perfect line drawing.

The line drawing was analyzed for polygons and polygon descriptions were built up in terms of exterior angles for each possible polygon in the perfect line drawing. A selection criteria based upon the number of sides and whether or not the polygon was closed was used to select an image polygon to be matched against the polygonal components of 3D polyhedral models in the data base. The look-up procedure found all models containing polygons of the required type and a refinement process reduced the number of acceptable models by performing some simple adjacency tests. Finally a mean-square-error technique was used to determine if the image polygon was an allowable transformation of any of the acceptable models. The model with the smallest mean-square-error was declared to be the matching object.

Roberts' work broke new ground on several fronts. He was one of the first researcher to store internal 3D models of objects -- in this case the polyhedra were described as collections of points in homogeneous coordinates (x y z w) with specific connection topologies. Secondly, he developed an edge operator and a line finder which worked reasonably well on a restricted class of images. Finally, and most significantly, he developed a procedure for testing the similarity of a 2D polygon or collection of polygons with a 3D polyhedron or collection of polyhedrons by exploiting knowledge of perspective projection and the geometry of trihedral polyhedra.

Despite this enormous contribution, his work is sometimes criticized because it avoids certain serious problems. For example, only a small group of polyhedral objects (cube, wedge and hexagonal prism) are represented in his system. Polyhedra have a straightforward representation and do not pose the problems presented by curved objects. In addition, he was careful to control the color of the blocks, the color of the support surface, and the lighting conditions to ensure high contrast edges in his images. In the general scene analysis problem none of these variables are under the control of the experimenter. In spite of these criticisms Roberts' work inspired many researchers to believe that the general scene analysis problem might be tractable.

Guzman (Guzman 1968) attacked the problem of determining the number of polyhedral blocks appearing in an image. His system was based upon the analysis of trihedral vertices and how they project as picture junctions. Although he did not explicity represent shapes or attempt to recognizes shapes, he re-focused attention on the importance of vertices and picture junctions as loci of concentrated information. Guzman's approach was heuristic in nature and it was based on his intuitive observations about polyhedra and their projections. By 1971 Huffman (Huffman 1971) and Clowes (Clowes 1971) each had developed systems

whereby the constraints imposed by the types of edges (convex, concave), which connect vertices, were used to reduce the number of possible interpretations for an image. Neither of these systems explicitly represented shapes, but both were used to eliminate certain nonsense interpretations of an image as a collection of trihedral polyhedra, thus contributing to the solution of the figure-ground separation problem. Waltz (Waltz 1972) expanded upon this work by analyzing the further constriaints imposed by shadows in such scenes. Mackworth (Mackworth 1973) included non-trihedral vertices and Turner (Turner 1974) considered vertices formed by curved edges. In all of these cases the number of possible interpretations grew exponentially with the number of vertex and edge labels.

During the same period progress was being made on the 3D representational front. Baumgart (Baumgart 1972a,b) developed the winged-edge polyhedron representation. Objects were represented in terms of primitive polyhedra with specific attachment relations. The representation was a bit cumbersome, but it adequately captured all objects which are easily decomposed into polyhedra. Baumgart embedded the representation in a system for geometric modelling, GEOMED (Baumgart 1972a), which allows the user to construct and manipulate objects in the winged-edge representation. The major drawback of this representation is that it does not provide for the representation of curved objects.

In 1971 Binford (Binford 1971) introduced a representation for curved objects which he called the "generalized cylinder" representation. An object in the generalized cylinder representation is described in terms of a skeleton together with descriptions of cross sections along the axes of the skeleton (Figure 7). In its most primitive form simple cylinders are used -- i.e. each object or part of an object is represented by a straight-line axis in space and a radius for a circular cross section which is swept down the length of the axis. The cross section remains orthogonal to the axis at all points along the axis. In its more general form the

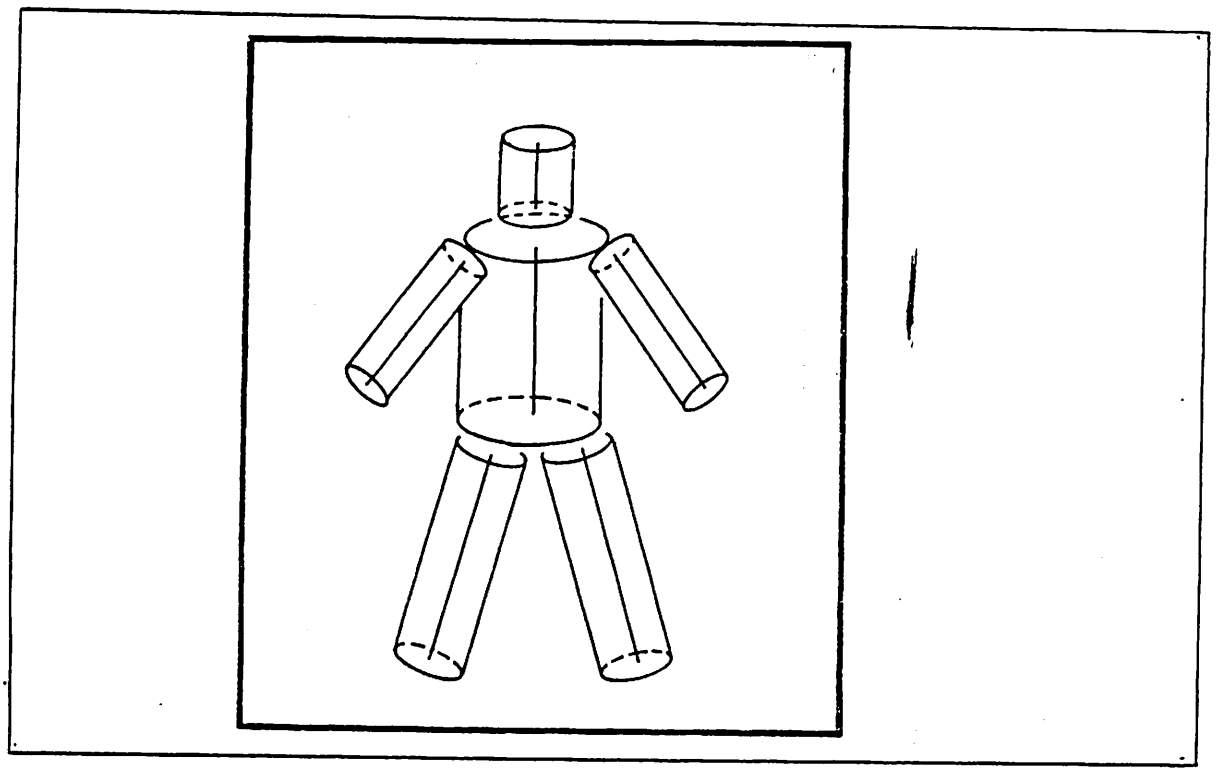

Figure 7: A cylinder representation of a human body[5]

axis may be an arbitrary space curve and the cross section may be any closed planar figure (Figure 8).

Agin implemented a curved object description system based upon generalized cylinders (Agin 1972). The system was able to segment TV image data which had been augmented by a depth map. The depth map was generated by a laser range finder and Agin's program derived generalized cylinder descriptions from such data. The program consisted of a cylinder-finder which would iteratively converge on a best estimate of a cylinder in the data, given an initial estimate of the position, orientation, and extent of the axis. The cylinder-finder used a curve fitter to fit conics to sequences of points in the 3D data. The centers of these conics (only circles and ellipses) were used to revise the current estimate of the axis. When a reasonable

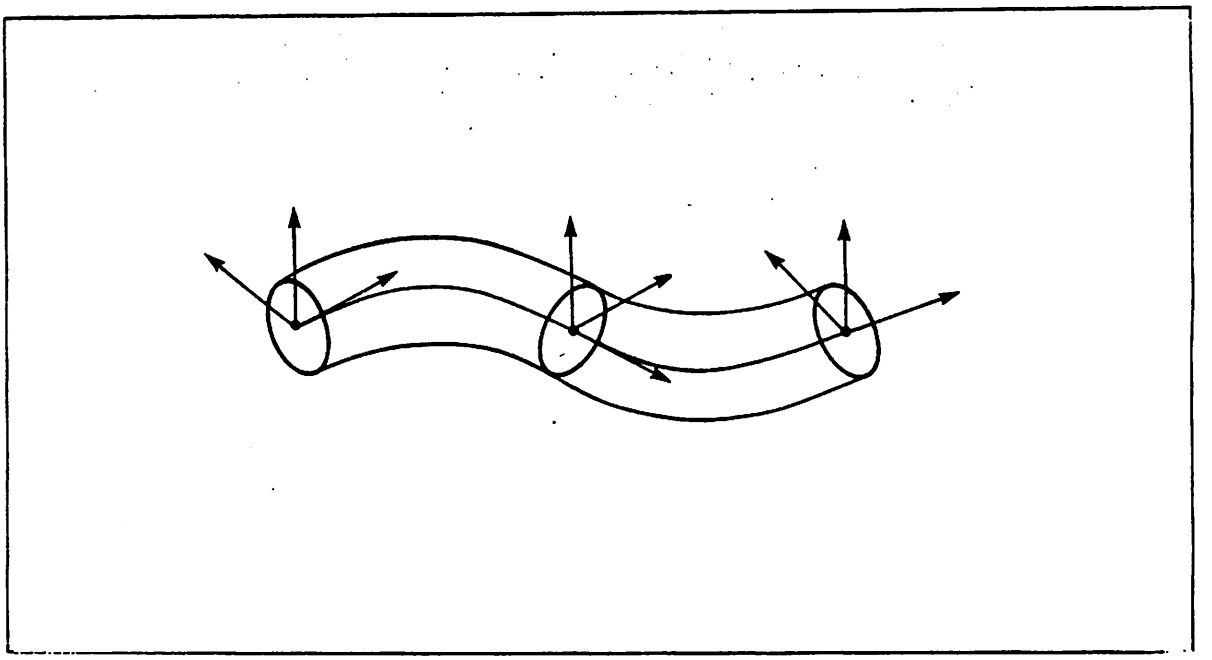

Figure 8: A generalized cylinder

estimate was obtained, the system attempted to extend the cylinder in both directions along its axis. It continued to extend the cylinder until an obvious break point was reached.

Agin achieved reasonable results with his program. It worked well for simple cylindrical objects such as the torus and the hammer; however, it did not perform well for complex objects. In his thesis Agin pointed out certain drawbacks to using his program in an object recognition system: (1) the procedure does not always converge (2) it becomes confused at T-joints (3) a reasonable initial guess for the principal axis must be provided. Agin does provide an automated procedure for producing an initial guess; however, its effectiveness is limited.

Nevatia (Nevatia 1974) took a similar approach to that of Agin, but he differed in three significant ways. First, he used a "generalized cone representation" which allowed the radius of the circular cross section to vary along the axis. Secondly, he generated structured symbolic

descriptions of objects. These descriptions encoded the joints (places where axes came together) and the pieces (generalized cones) in a graph structure (see Figure 9). Each piece was described by certain features

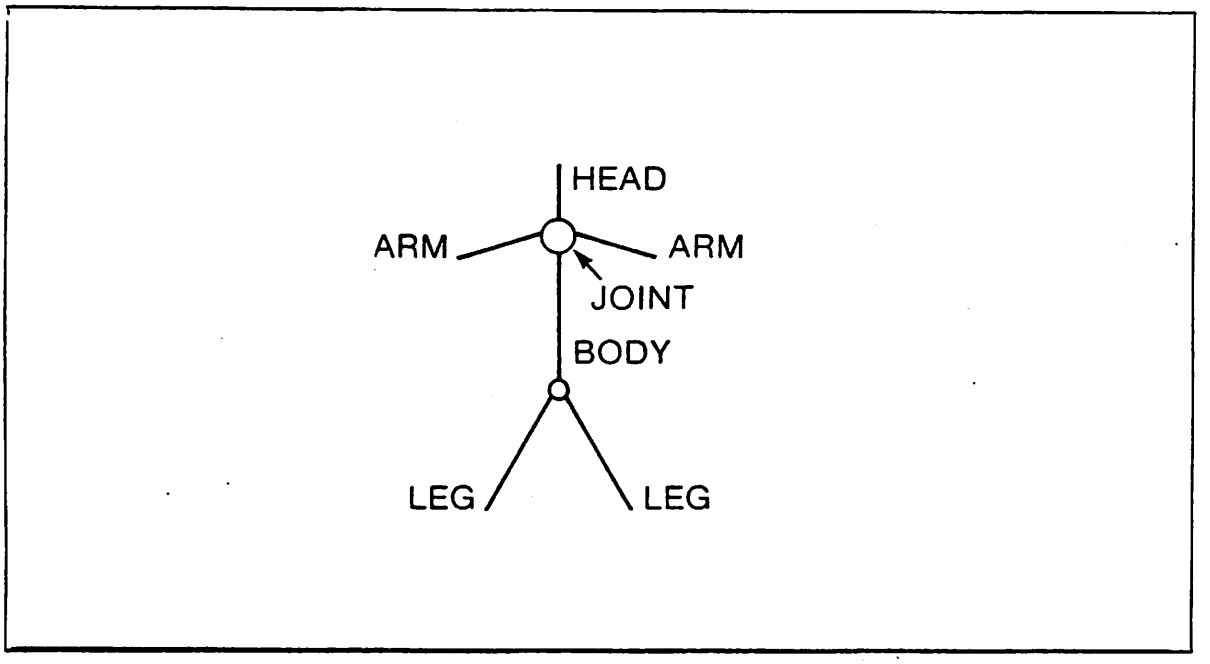

Figure 9: Nevatia's skeletal description of a human[6]

such as axis length, widths of cross-sections, and the ratio of axis length to average cross section width. Each joint was described by the number of axes entering the joint, the angles formed, and the relative sizes of the axes. Nevatia created graph matching routines for generating partial matches on these symbolic descriptions. In addition, his system contained a memory of object models which could be indexed easily on these features. He was able to obtain fairly good results extracting and matching object descriptions on a data base of 6-10 object models. Most of the objects were elongated and thus were suitable for representation by generalized cones.

The difficulty with Nevatia's work is that he was limited to the class of objects representable with generalized cones. He does hint at the possible use of splines for axes which are curved in space. Another problem is that he would sometimes store multiple descriptions for a single object.

Nevatia made the following positive contributions. He demonstrated the importance and the effectiveness of indexing into a class of models to reduce the search space. He showed that semantics and partial matches could be effectively utilized in a recognition system. And finally he demonstrated the efficacy of hierarchical symbolic shape descriptors.

By 1976 Marr and Nishihara at MIT had begun experimenting with the generalized cylinder representation. The major difference between their task and those of Agin and Nevatia was that they were working with 2D data. They were attempting to infer a generalized cylinder representation of a single object from the digital contours of a 2D image of the object (Marr and Nishihara 1976, 1977). They developed a hierarchical object representation using simple cylinders and an adjunct relation for relating the auxiliary axes of an object to its principal axis. The skeletal description in terms of this adjunct relation was more flexible than those proposed by Agin and Nevatia.

The derivation of a generalized cylinder from 2D data involved finding points of "strong segmentation" along the contour and hypothesizing the positions in the image of the principal and auxiliary axes (Figure 10). Again relative lengths of principal and auxiliary axes were used to index into a data base of stored generalized cylinder descriptions. The best candidates were obtained and they were matched against the contour in the image by a using a relaxation technique to test various views of the 3D model. The convergence of the procedure could not

be established; however, reasonable results were obtained for a small data base of objects which could be easily recognized by their stick figure representations.

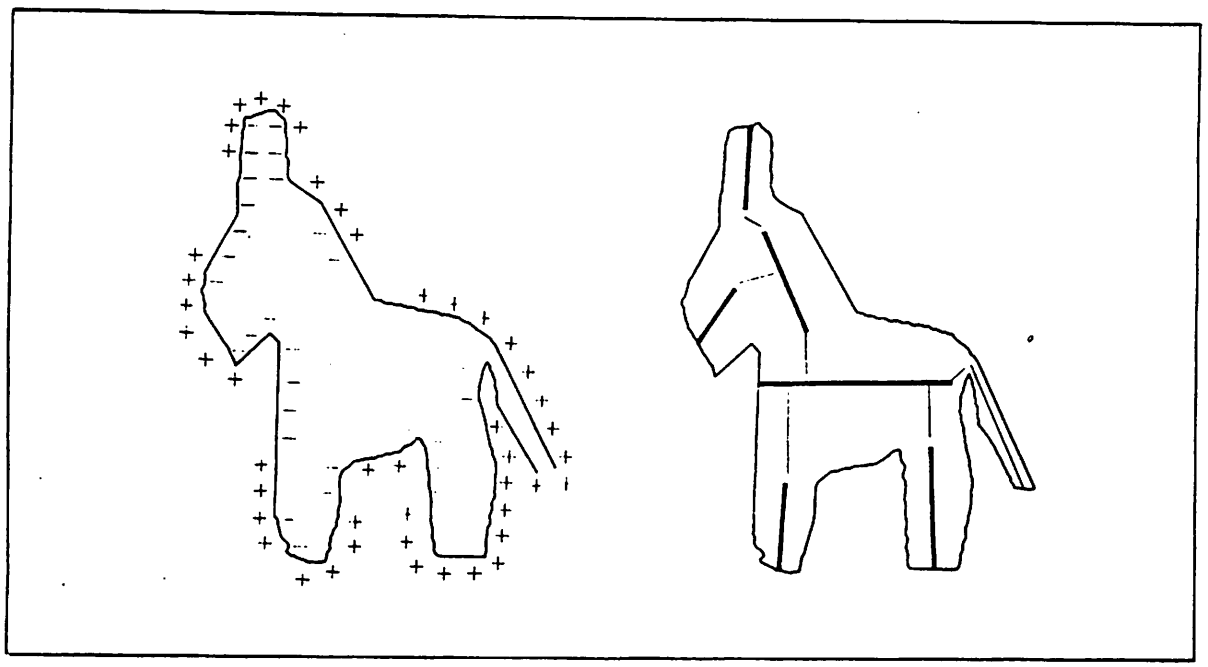

Figure 10: Hypothesized axes in a digital figure[7]

The more significant contribution of this work was the explicit statement of criteria for designing a shape representation. These criteria are discussed in detail in Chapter II.

Hollerbach (Hollerbach 1975) developed programs for describing ancient Greek vases in terms of generalized cylinders and for recognizing a vase from a 2D outline of its contour. Moreover, he extended the notion of generalized cylinder to include polyhedra - i.e. the cross section is a planar polygon and the axis is a straight line.

Soroka (Soroka 1979) implemented a system which extracts "elliptical cylinder" (EC) descriptions of 3D objects from stacks of 2D slices through a 3D data space. An elliptical cylinder is a generalized cylinder restricted to elliptical cross sections (and of course includes circular cross sections).

Soroka's program performs four major functions -- slice selection, slice description, hypothesis formation, and hypothesis update. He presents the results of running his program on four objects, a cylindrical Y, a cylindrical helix, a phantom heart, and a digitized heart. The phantom heart was a synthetic object consisting of three cylinders with the approximate structure of a heart. The digitized heart was an actual digitization of a canine heart.

The program was able to develop EC descriptions for the Y and the helix despite the junction of the Y and the curved axis of the helix (the curved axis was approximated in a piecewise linear fashion). It encountered some difficulty with the phantom heart, but it achieved surprising success with the more complex digitized heart. Slice selection is critical to the process and Soroka developed a notion of "curiosity" for cleverly investigating unexplored portions of the data space during slice selection. Soroka did not define measures for evaluating the goodness of the EC description nor did he present routines for matching extracted descriptions to stored prototypes.

Finally Shirai (Shirai 1978) has developed a system for recognizing various real world objects such as a desk lamp or a telephone in an office scene. His system consists of four basic processes -- edge finding, straight line and elliptic curve fitting, object recognition by a hierarchy of features, and a process monitor which controls the other three processes. These processes embody a common paradigm in computer vision and Shirai uses it effectively. The major drawbacks of this system are that objects have only 2D descriptions in terms of elliptical curves and straight lines and that a separate recognition program is constructed for each object (a procedural appraoch to object recognition). The significant contribution of Shirai's work is the total system approach to object recognition. He integrates the strategy for object recognition with the low-level processes of edge extraction and curve fitting. Such integration accounts for a great deal of the success of his system.

The research described in this section is representative of the bulk of 3D shape research over the past several years. It is clear that most shape research has been focused on volumetric primitives. The representation presented in this thesis draws upon research in the areas of computer graphics and numerical approximation. We will propose a surface primitive and show that it can be stored economically and manipulated easily.

I.4.3 Two-dimensional shape representations.

There is a wide variety of 2D shape representations and Pavlidis (Pavlidis 1976) presents an excellent taxonomy of the techniques for extracting descriptions in those representations from digital data. Only a few of the better known representations and techniques are discussed in this section.

Two of the older representations describe a 2D object as a sequence of scalar values and for this reason they are referred to as "scalar transform" representations. The first represents an object as a series of moments and the second represents it as a series of Fourier coefficients. Both of these representations require the characteristic function of the object for the computation of a description and consequently they are called "internal" scalar transform techniques. These representations are information-preserving; however, they are computationally very expensive.

The most commonly used external scalar transform technique is the Fourier transform of the object's boundary (Zahn and Roskies 1972). It is also computationally expensive and not sensitive to local information in the boundary. Recently Schudy (Schudy 1979) employed spherical harmonics to obtain 3D Fourier descriptors of objects. He was successful in

characterizing shapes of the human heart at various phases of expansion and contraction with as few as six to eight coefficients in many cases.

Another of the older representations was invented by Blum in 1964 and described in (Blum 1973). In this representation an object is described by its skeleton which is computed via the medial axis transformation (MAT). The problem with this representation is that the transformation is extremely sensitive to noise in the data. The two objects in Figure 11 have very different skeletons despite their similar shapes. Pavlidis (Pavlidis 1976) suggests that the sensitivity to noise and the computational overload may be relieved somewhat by first computing the polygonal approximation of the boundary and then computing the MAT of the approximation. Montanari (Montanari 1970) actually implemented this idea and the problem with this approach is that the MAT adds little useful shape information on top of the polygonal approximation.

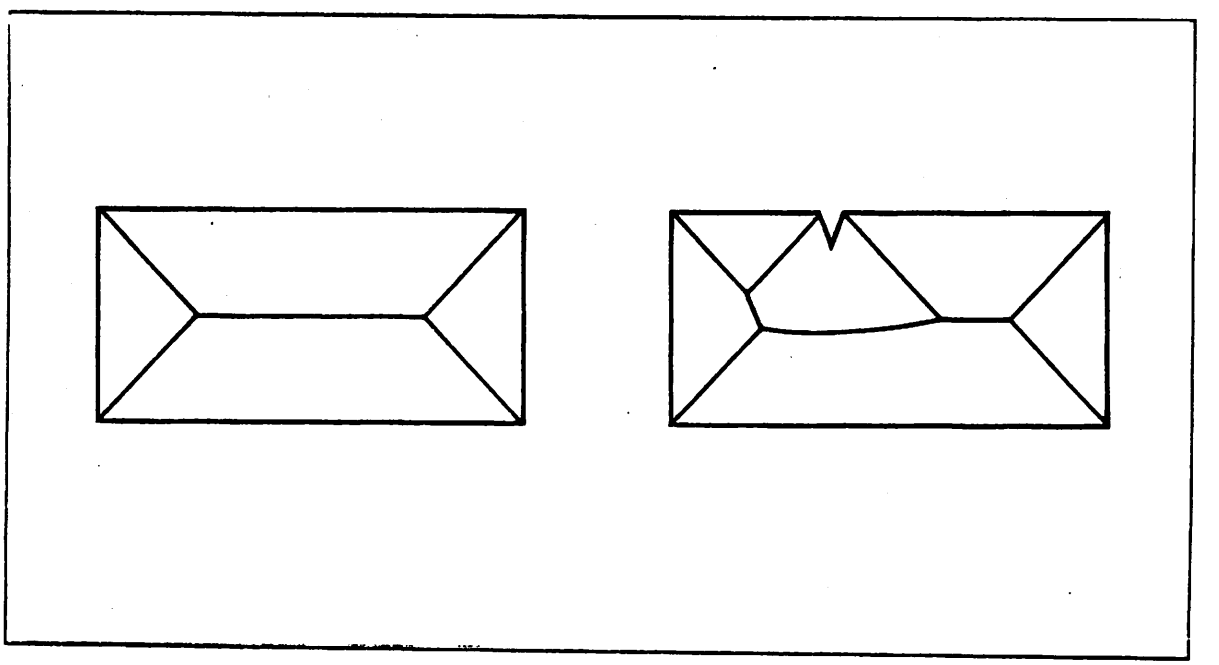

Figure 11: Skeletons resulting from the medial axis transformation[8]

Pavlidis (Pavlidis 1972a,b) popularized techniques for the decomposition of polygons into primary convex sets. This structure is represented by a labelled graph with the juxtaposition relation. The restriction to convex sets was too strict a requirement so Feng and Pavlidis (Feng and Pavlidis 1975) relaxed it to obtain decompositions into convex sets and spirals, where spirals are polygons which have all of their concave vertices adjacent to one another. This representation is rotation and translation invariant and it is to a great degree size invariant. The major drawbacks are that the algorithms are very complex and computationally expensive.

Davis (Davis 1976a) developed a relaxation technique for segmenting digital curves into angle-side descriptions. With a relaxation labelling - graph matching algorithm (Davis 1976b) he was able to match very successfully the descriptions of coastlines extracted from images with prototypes stored in this representation. His matching algorithm is similar to spring-loaded template matching (Fischler and Eschlager 1973).

Finally, there are the syntactic techniques. Shapes are represented by sentences in a formal language. Each sentence is a string of symbols where each symbol represents some local shape feature of the object represented by the string. The recognizable shapes in the system are precisely those shapes whose feature strings are sentences in the language. Gips (Gips 1974) devised several shape grammars for recognition of configurations of certain polygons. The problem with syntactic techniques is that when the grammar is simple only a small specific class of objects can be recognized. As the class of objects grows and global properties are desired the grammars become very complex.

Clearly there are difficulties with each of these approaches, but there is hope that a combination of approaches will be effective. Pavlidis (Pavlidis 1976) suggests that a combina-

tion of local approximation and syntactic techniques may prove fruitful. It is our belief that this is a practical course to follow. In Chapter IV we discuss a particular local approximation scheme.

I.4.4 Computer graphics and computer-aided geometric design.

It would be inappropriate to close this chapter without acknowledging the impact of related work in the fields of computer graphics and computer-aided geometric design (CAGD), although the contributions are readily apparent when one reads the remainder of this thesis. These two disciplines are concerned with representation of objects and the tools for designing such objects. Computer vision is concerned with the representation of objects for the purpose of recognition. Design and recognition are related tasks and perhaps the representations for one are useful for the other.

This thesis is based on the observation that 3D shape representation research has not progressed very far in the last five years. Our goal is to change the focus from volumetric representations to surface-based representations. An investigation of surface representations, currently used in CAGD, led us to cubic splines and Coons patches.

For a theoretical treatment of splines see (Ahlberg et al. 1967). (deBoor 1979) provides a more practical coverage complete with algorithms in FORTRAN. (Baudelaire 1977) presents fast algorithms for cubic spline interpolation and for plotting the resulting curves on a raster display. (Bezier 1972) describes curve techniques and the use of splines in numerically controlled machines at Renault. (deBoor and Rice 1968a,b) contain algorithms for least-

squares cubic spline curve fitting, while (Schweikert 1966) shows algorithms for computing splines in tension.

In the area of surface representation there are several good references. Coons introduced his symmetric surface patch representation in (Coons 1967). (Rogers and Adams 1976) describe several surface patch representations from the simple bilinear patch to the generalized Coons patch. Each of these representations has limitations which are pointed out in Appendix II. (Forrest 1968) and (Gordon and Riesenfeld 1974) are good treatments of surface interpolation and (deBoor 1979) presents techniques for tensor product bicubic spline approximation.

CHAPTER II

REPRESENTATION OF THREE-DIMENSIONAL SHAPE

II.1 Introduction to 3D Representation

The problem of shape representation in computer vision is intrinsically difficult because the concepts of "shape" and "representation" are ill-defined. Before one can design a shape representation, practical definitions must be given to both of these terms. These definitions must be sufficiently precise to allow the direct development of data structures within the intended system implementation.

A practical definition of shape must admit both quantitative and qualitative descriptions of objects for it to be useful in a computer vision system. The quantitative description permits the geometric reconstruction of the object, while the qualitative description provides symbolic features which permit various levels of similarity and difference between objects to be detected without recourse to the detailed geometric description of the object. We use the following definition. The quantitative shape description of an object is a collection of geometric entities (points, lines, surfaces, ...) and relationships between those entities such that the relationships remain invariant under the spatial transformations of rotation, translation, and uniform scaling in all dimensions. This constraint merely states that points are transformed into points, lines into lines, and surfaces into surfaces; the relationships between these entities such as relative distances and connectedness are retained.

The qualitative shape description of an object is a collection of features and their corresponding values. These features may be scalar or structured, and the values for these features must be computable from the quantitative description. For example, the position of a point on an object may be part of the quantitative description of an object's shape; however, it cannot be a qualitative shape descriptor since position is not invariant under the transformations mentioned above.

Throughout the remainder of this thesis, the term "shape feature" will often be used to encompass both quantitative and qualitative shape description. Distinction will be made only when necessary to avoid confusion. Let us define a "shape representation" to be a uniform structuring of shape features which facilitates concise object description and rapid object recognition. This is a general definition which encompasses many possible structures. Thus, the shape designer's art consists of choosing the proper quantitative and qualitative shape descriptors so as to meet certain criteria. We elaborate on those criteria later in this chapter.

This chapter describes a practical exercise in the design of a shape representation for three dimensional objects. The target class consists of all rigid 3D objects.

II.2 Overview of Chapter

The purpose of this chapter is to give the reader a complete description of our shape representation. We begin with a few criteria for measuring the goodness of a shape representation in order to provide a context for the evaluation of the representation which is proposed herein. It is followed by descriptions of the representation primitives, their properties, the structure of complex objects, an analysis of shape features, and shape classification. Several

example objects are described in detail and the chapter concludes with a comparison to the generalized cylinder representation and a discussion of the limitations of our representation.

II.3 Criteria for a Good Shape Representation

The following design criteria proposed by Marr and Nishihara (Marr 1976) were used as guidelines in our design. They provide a framework within which to critically appraise the proposed shape representation. Briefly stated the three criteria are:

(1) Scope and Uniqueness

(2) Stability and Sensitivity

(3) Accessibility

These criteria are usually augmented by two additional practical criteria:

(4) Cost effectiveness

(5) Matchability

Scope refers to the class(es) of shapes which may be described in the given representation. Obviously a broad scope is desirable, and in the ideal situation all objects would be described within a single shape representation. However, the ideal is not easily attained in practice and in terms of computational overhead, it is probably not desirable for domains which contain a small number of objects with limited shape complexity. For example, in the domain consisting of a small number of regular polyhedra, a representation which makes specific use of the special geometric characteristics of regular polyhedra would be sufficient. On the other hand, such a representation would certainly be insufficient for the description of complex, curved objects.

Uniqueness refers to the situation in which distinct shapes have distinct descriptions within the shape representation. A description is a (possibly structured) collection of values associated with the collection of shape features in the representation. Uniqueness is a matter of degree, since qualitative features partition the space of objects into equivalence classes and the fineness of the partitioning is dependent upon the choice of features.

Stability and *sensitivity* are measures of the continuity of the representation. Objects which have similar shapes should, in some sense, have similar descriptions. For example, it is desirable for an apple and a ball to have similar descriptions -- a stability at some level in the representation. It is also desirable to be able to discriminate between apples and balls on certain occasions - a sensitivity at some level of the representation. Clearly stability and sensitivity are closely related to uniqueness and the three criteria must be carefully balanced during the design process.

Accessibility is concerned with the ability to gain access to the representation of an object. What information is required? What is the source of that information and how does that information relate to the qualitative and quantitative shape descriptions of the object? In the case of 2D scene analysis, the accessibility issue raises the following questions. What features from the 2D image supply access information about the 3D object depicted in the image? How do we capture this information and use it to gain access to the representation of the 3D object?

Cost effectiveness is an important practical consideration. A representation which is very expensive in terms of storage or computation is of very little use in a computer vision system.

Matchability refers to the suitability of the representation for matching operations. After access has been gained to a set of representations, a matching routine must determine which

member of the set best matches the data. The effectiveness of such a matching process is directly related to the form of the representation and the nature of the data. For instance, 3D data allows the direct computation of shape features which appear in the 3D representation, whereas 2D data requires the hypothesis of 3D shape features from 2D shape features. Clearly the matching process is heavily dependent upon the nature of the data for the kinds of shape features which can be computed, upon the indexing mechanism for obtaining initial candidates, and upon the nature of the internal representation for the discrimination ability of the match.

The reader should keep these criteria in mind during the remainder of the thesis.

II.4 Representation of 3D Objects

II.4.1 Introduction

An object in our representation is a hierarchically structured collection of parts. Parts, themselves, are objects and thus hierarchically decomposed into parts. At some stage of the decomposition there are primitive parts. Primitive parts are represented as collections of surface patches. These surface patches are geometrically represented in the Coon's surface patch form (Coons 1967).

II.4.2 The Coons Surface Patch

The Coons surface patch is an interpolative patch form developed by S.A. Coons in 1964 and later extended in 1967 (Coons 1967). It was originally designed for use in the large

computer-aided geometric design (CAGD) systems of the aircraft, shipbuilding, and automobile industries; however, it did not achieve a high degree of popularity. The most common criticism by CAGD practitioners was that the manipulation of the parameters of the patch caused non-intuitive changes in the shape of the surface, and for this reason it was difficult to use. Our purpose was not to judge the validity of the criticism, but to choose the best surface patch representation for computer vision on the basis of the aforementioned criteria. That is, our goal was not to build a computer-aided design system, but to construct a representational system. We felt that the Coons patch representation best met our criteria.

The Coons patch equation is an interpolation formula in which the position of every point on the surface patch is represented by a sum of interpolations between opposing pairs of boundary curves and opposing pairs of across-boundary derivative curves. Coons requires only that the boundary and across-boundary derivative curves be expressed as vector-valued functions of a single parametric variable and that boundary curves meet at the corners of the patch. In our representation we chose to represent these curves by cubic B-splines in three dimensions, due to a suggestion made by Coons(Coons 1974).

Only brief mathematical descriptions of the Coons patch and the cubic B-spline will be presented here; more complete descriptions appear in Appendices I and II. Our major objective is to describe the properties of these two representation primitives and to illustrate how they are used to represent complex 3D objects. At this point we strongly recommend that the reader who is unfamiliar with Coons surface patches and cubic splines review the appendixes and become familiar with the notation.

II.4.2.1 Notation

Let u and w represent two independent parameters, each restricted to the interval [0,1]. Consider the unit parametric square in the u-w plane. The surface patch P is a vector-valued function of these two independent parameters. Thus the head of the vector P(u,w) designates a point on the surface P associated with the pair of parameters u and w. We will make use of the following notation developed by Coons.

uw denotes P(u,w), an arbitrary point on surface P.

0w, 1w, u0, u1 denote the four boundary curves P(0,w), P(1,w), P(u,0) and P(u,1) of surface P.

$u0_w$ denotes the partial derivative of the surface P with respect to parameter w, evaluated at w=0.

$$u0_w = \frac{\partial(uw)}{\partial w}\Big|_{w=0} \qquad u1_w = \frac{\partial(uw)}{\partial w}\Big|_{w=1}$$

$$0w_u = \frac{\partial(uw)}{\partial u}\Big|_{u=0} \qquad 1w_u = \frac{\partial(uw)}{\partial u}\Big|_{u=1}$$

00_{uw} denotes the second partial derivative with respect to u and w, evaluated at u=0 and w=0. This derivative is referred to as a *corner twist vector.*

$$00_{uw} = \frac{\partial^2(uw)}{\partial u \partial w}\Big|_{\substack{u=0 \\ w=0}}$$

and similarly for 01_{uw}, 10_{uw}, and 11_{uw}

The univariate real-valued blending function $F_0(u)$ will be written F_0u and similarly for parameter w and the blending functions F_1, G_0, and G_1.

With this notation the generalized Coons patch equation may be written in matrix form as follows:

$$uw = [F_0u \quad F_1u \quad G_0u \quad G_1u] \begin{vmatrix} 0w \\ 1w \\ 0w_u \\ 1w_u \end{vmatrix} + [u0 \quad u1 \quad u0_w \quad u1_w] \begin{vmatrix} F_0w \\ F_1w \\ G_0w \\ G_1w \end{vmatrix}$$

$$-[F_0u \quad F_1u \quad G_0 \quad G_1] \begin{vmatrix} 00 & 01 & 00_w & 01_w \\ 10 & 11 & 10_w & 11_w \\ 00_u & 01_u & 00_{uw} & 01_{uw} \\ 10_u & 11_u & 10_{uw} & 11_{uw} \end{vmatrix} \begin{vmatrix} F_0w \\ F_1w \\ G_0w \\ G_1w \end{vmatrix}$$

A generalized Coons surface patch may be thought of as the sum of an intrinsic surface and a slope correction surface. Figure 12 gives a graphical illustration of the meaning of the intrinsic component of the above equation. Later we will show examples of surfaces which may be obtained using only the intrinsic component of this equation.

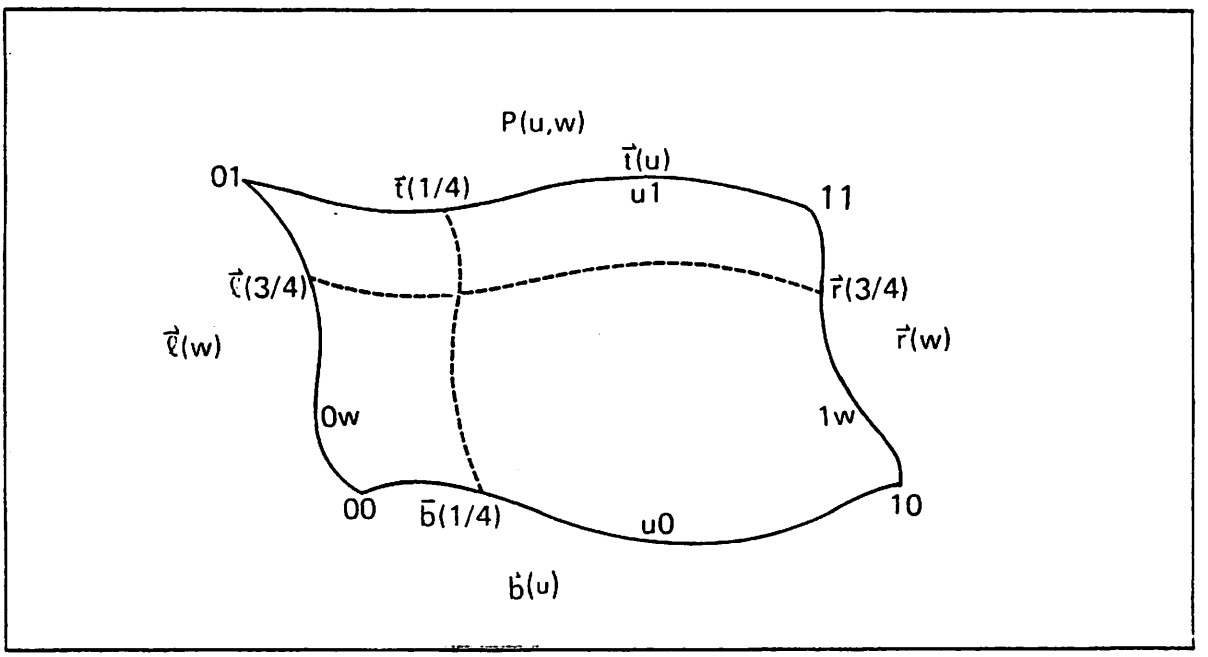

Figure 12: A Coons surface patch-- graphical view of interpolation

The Coons patch form has the following nice properties:

(1) The patch equation is *symmetric* in u and w.

(2) Two patches may be *smoothly* joined (curvature continuous) along a common boundary if the blending functions satisfy a certain criterion called the δ-condition(see Appendix II). The δ-condition is a restriction on the derivatives of the blending functions of the two patches being joined.

(3) Surface normals, surface area and surface curvature may be directly computed.

(4) The Coons patch form subsumes other commonly used forms as special cases.

(5) The complexity of the shape of the patch is a function of the complexity of the boundary curves and blending functions only -- i.e. the patch is completely defined by boundary conditions.

For a short development of interpolative patch forms and the Coons equations see Appendix II. In section II.4.4 we demonstrate the usefulness of these properties in the design of our shape representation, but first we must briefly describe cubic B-splines and their properties.

II.4.3 Cubic Splines

Basically a spline is a piecewise polynomial function which meets certain continuity conditions at the points where the pieces come together. These points are called *knots* and a cubic spline function is curvature continuous everywhere in the interval over which it is defined. A B-spline function is a spline function represented in a special form as a linear combination of special basis functions. The term B-spline was introduced by Schoenberg and originally referred only to those basis functions; however, the term has come to be used with reference to any spline function represented as a linear combination of these basis functions. Corresponding to the knot vector (the ordered list of knots) for the spline representation is the defining polygon (the ordered list of vertices) in the B-spline representation. Both representations are useful and we shall point out their uses later in this chapter.

The interpolation formula for evaluating a uniform cubic B-spline function which is defined by the ordered list of vertices $\{V_i\}$ is represented in matrix form as follows:

$$P(u) = [s^3 \quad s^2 \quad s \quad 1]\ 1/6 \begin{vmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{vmatrix} \begin{vmatrix} V_i \\ V_{i+1} \\ V_{i+2} \\ V_{i+3} \end{vmatrix}$$

where $V_i = (x_i, y_i, z_i)$
n = the number of vertices -3
i = the integer part of $n-u$
s = the fractional part of $n \times u$

Cubic B-spline curves are obtained from cubic B-spline functions by parameterization. If we wish to define a cubic B-spline curve in 3D space, we merely select the knots (x_i, y_i, z_i) and consider the curve to be the triple of cubic B-spline functions $(S_X(t), S_Y(t), S_Z(t))$ where t is a parameter. Throughout this work we use *uniform* parameterization; the effects of nonuniform parameterizations are demonstrated in (Baudelaire 1977) and (York 1979).

Cubic B-splines have the following nice properties:

(1) Local approximation - The cubic B-spline approximation of an underlying curve is a local approximation scheme. A change in the position of a vertex of the defining polygon only affects the shape of the curve local to the point of change (see Figure 13).

(2) Variation diminishing - The cubic B-spline approximation to an underlying curve will always be smoother than the underlying curve. This property has the curious but useful side effect that cubic B-splines approximate straight lines exactly.

(3) Discontinuities - Derivative discontinuities can be achieved at a knot or vertex location through the use of multiple knots or vertices. A multiple knot occurs when the same triple of x-y-z coordinates appears more than once in succession in the knot vector. A multiple vertex is similarly defined (see Figure 14).

(4) Strong Convex Hull - This property states that the cubic B-spline curve will always remain within the local convex hull defined by each set of four successive vertices (see Figure 15). This property is the geometric manifestation of properties (1) and (2).

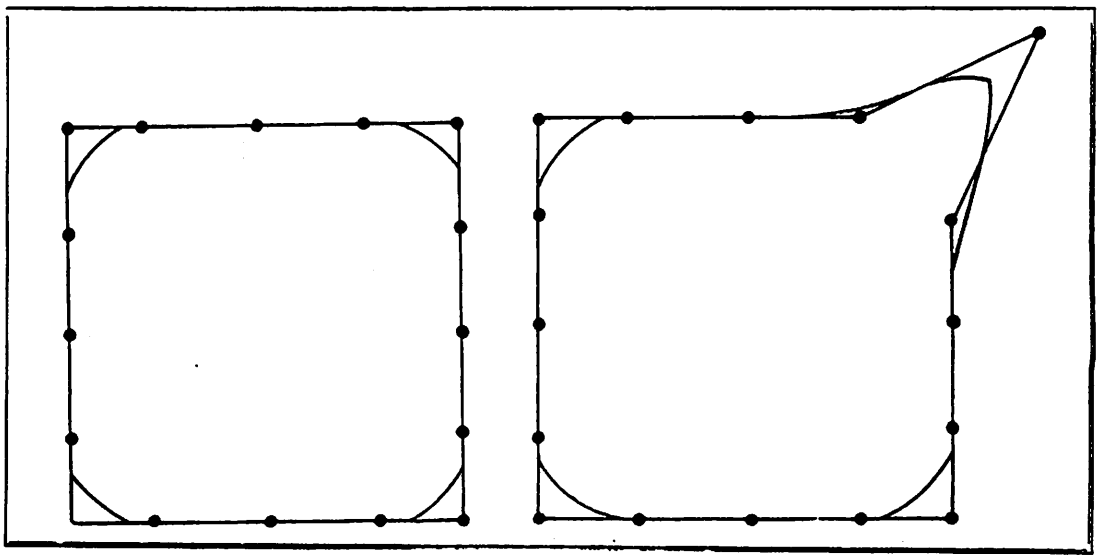

Figure 13: Local nature of Bspline approximation

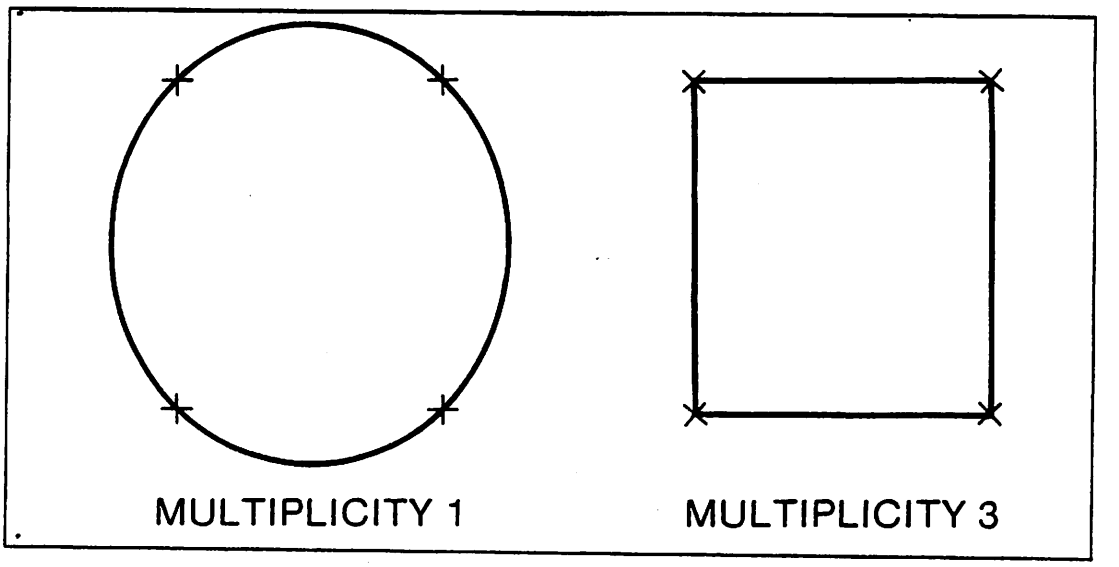

Figure 14: Discontinuities due to multiple knots

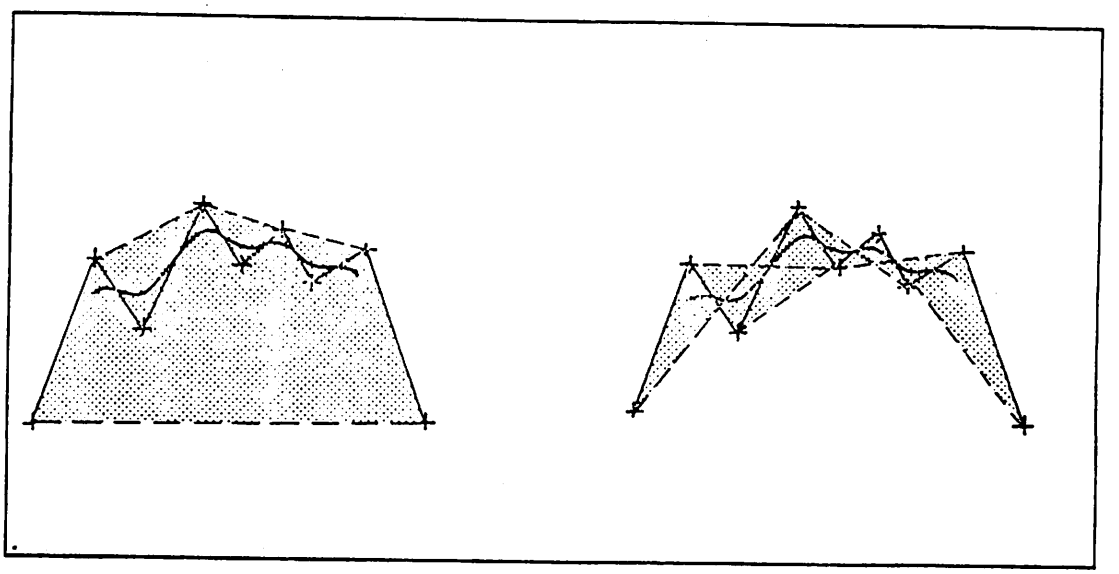

Figure 15: Strong convex hull property of cubic B-splines

These properties of cubic B-splines make the B-spline curve form an ideal candidate for representing the boundary curves, the across-boundary derivative curves, and even the blending functions of the Coons surface patch form. For a derivation of the defining equations for cubic splines and certain relevant theorems, see Appendix I.

Before proceeding to a description of how objects are constructed from these primitives, we wish to remind the reader that this thesis is not concerned with producing better algorithms for spline and surface approximation, or graphical display. Those problems fall in the provinces of numerical approximation and computer graphics. We are concerned here mainly with the use of these constructs and the properties they bring to the overall shape representation.

II.4.4 Building Objects out of Surface Patches

As mentioned earlier, each object in the data base has a hierarchical description in terms of its component parts. The parts themselves are objects and they may be further decomposed until, at some level, the parts are represented as surface patches. The description in this

section will proceed in a bottom-up fashion. First, we describe how complex surface patches are constructed from simple surface patches. Next, we describe how patches are put together to form volumes and finally, how volumes are composed to form objects. Each of these steps involves a structuring mechanism, or attachment relation, which must be understood by other components of the shape subsystem which use the representation.

II.4.4.1 Attachment Relations

The first attachment relation allows two Coons surface patches to be brought together continuously along a common edge. In order to use this structuring mechanism the two component surfaces must share a common continuous boundary curve and must satisfy the δ-condition on blending functions (see Appendix II). In our representation every surface patch is represented as a quad tree of surface patches (see Figure 16) in which the four component surfaces are joined via this attachment relation. Simple surfaces are obviously quad trees with three null leaves.

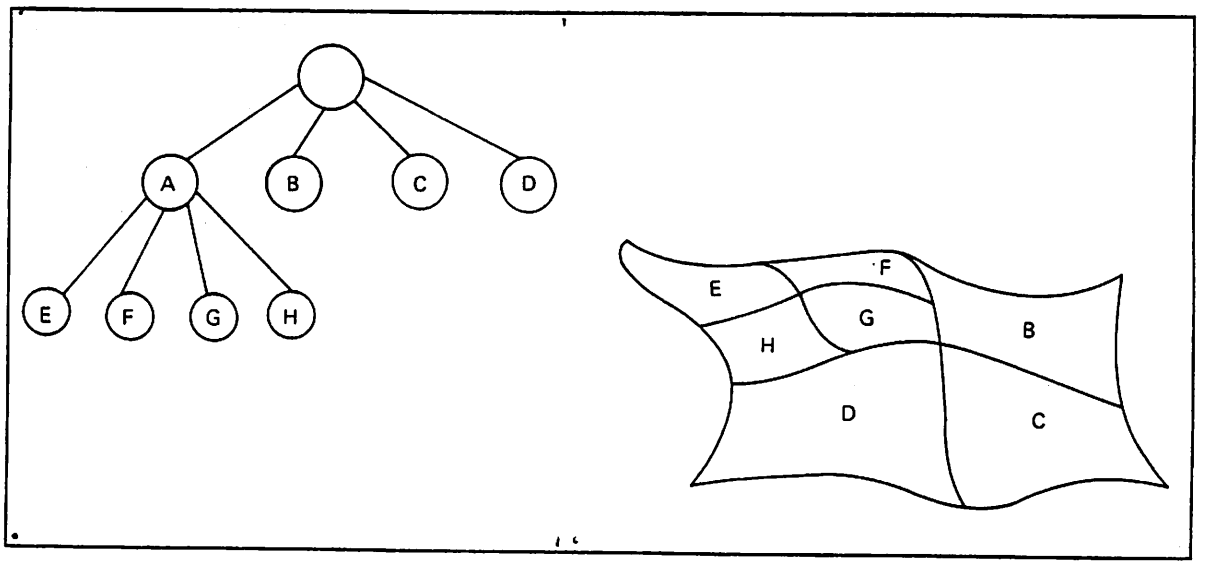

Figure 16: Complex surfaces as quad trees

The second attachment relation allows two surface patches to be joined discontinuously along a common edge(see Figure 17). This structuring mechanism permits the representation

of volumes where two curved surfaces come together in a discontinous fashion also along a common boundary.

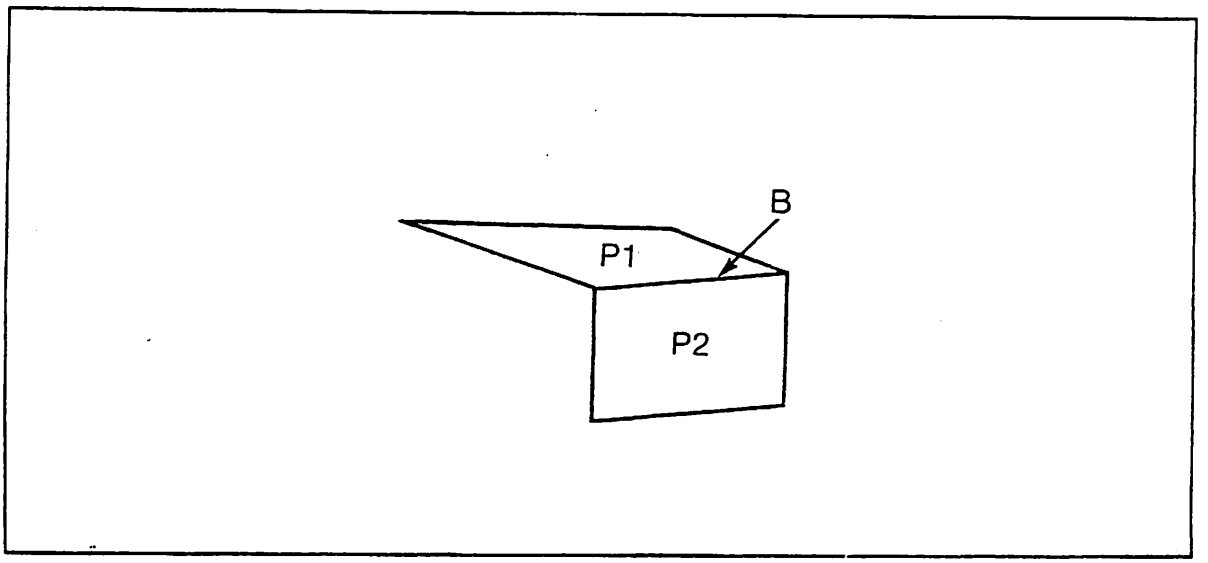

Figure 17: Discontinous join of two surfaces

The final attachment relation may be characterized as surface-to-surface attachment. This structuring mechanism allows the representation of volumes composed of two volumes which are connected at a common surface(see Figure 18).

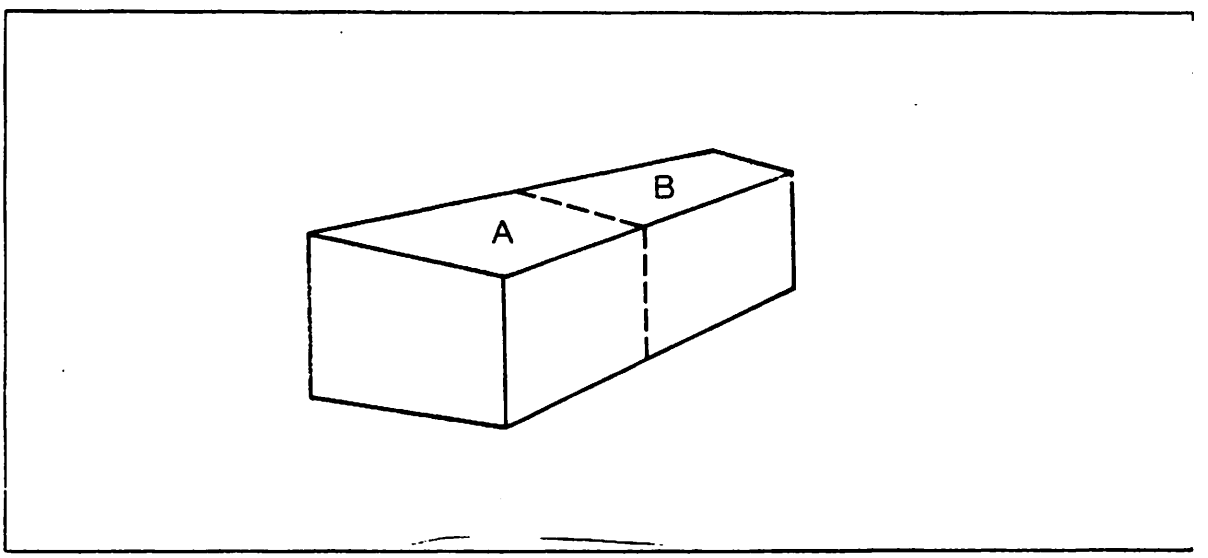

Figure 18: Two cubes attached at a common surface

The constraint on this attachment relation is that the two surfaces patches must be identical in order to form such an attachment. The quad tree decomposition of surfaces allows one surface to be attached to a portion of another surface, if that portion is a node of the quad tree and is identical to the other surface.

These attachment relations require a certain amount of geometry for their specification and enforcement. However, for a general understanding it is sufficient to know that each geometric representation primitive in the system (boundary curves, blending functions, corner definitions, etc.) is defined in its own local Cartesian coordinate system. An object is constructed by putting these primitives together to form surface patches and volumes according to the Coons patch equation and the three structuring mechanisms outlined above. Basically, each structural description of an object has a corresponding geometric description in which the attachment relations are represented by 4x4 matrices. Each matrix represents the transformation in homogeneous coordinates which must be performed to properly position the local coordinate system of a part with respect to the local coordinate system of the object to which it is being attached. Thus, the hierarchical decomposition of an object into its component parts corresponds to the hierarchical embedding of local coordinate systems shown in Figure 19. This technique is widely used in the field of computer graphics and it has been employed by several researchers in computer vision.

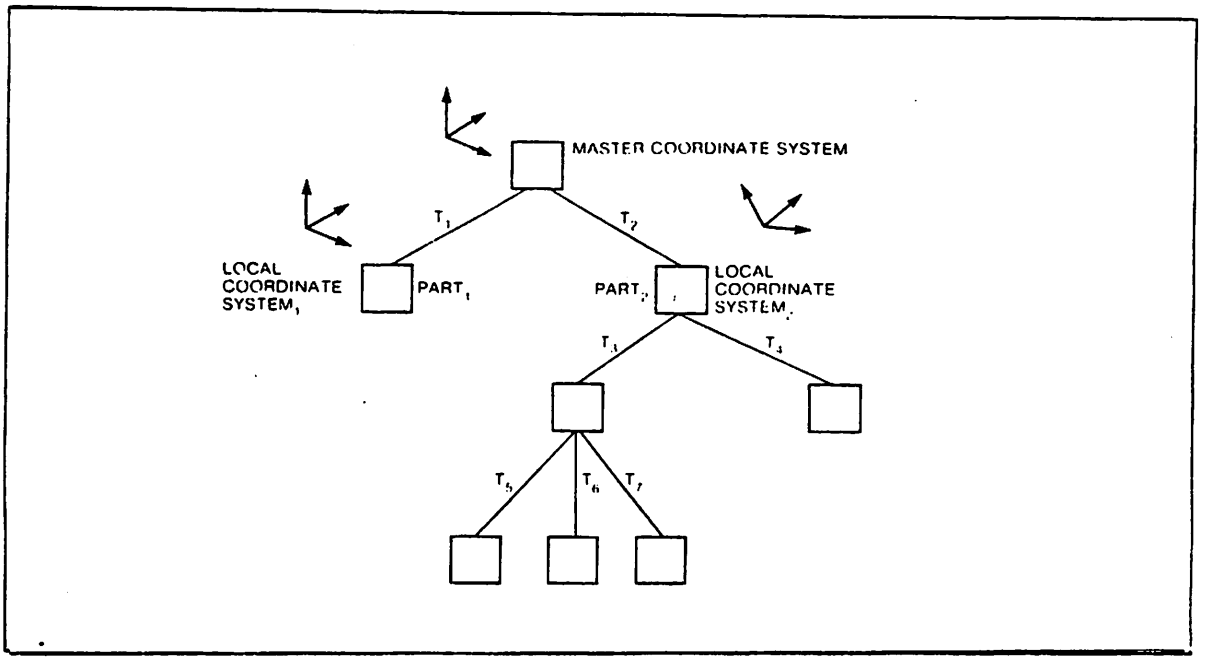

Figure 19: Hierarchical decomposition via local coordinate systems

II.4.4.2 Properties

Before we present several examples of objects described in our representation, let us first summarize the properties of the overall representation and indicate the contributions of the underlying primitives. The important properties of the overall representation are:

(1) High Sensitivity - The smooth join property of the Coons surface patch and the quad tree representation allow arbitrarily fine detail in the representation of an object.

(2) Broad Scope - Through the multiple knot/vertex mechanism, cubic B-splines may represent polygons as well as smooth curves. This feature together with the Coons patch interpolation equation permits the representation of both curved and polyhedral objects within the same representation scheme.

(3) Cost - Since the cost of representing a single Coons patch is constant, the cost of storage does not increase dramatically with increase in complexity of the surface shape. The local coordinate systems allow a single primitive or part to be shared among many objects.

(4) High accessibility - The sharing of object part descriptions provides access to all objects which contain a given part as a component.

(5) Good Qualitative Descriptors - The surface patch representation admits the computation of many features which are not easily obtainable from other geometric representations. These features are useful both in gaining access to object descriptions and in matching.

(6) Flexible Generation of Structural Descriptions - Our representation provides a more flexible geometric form from which to generate structural descriptions. Representations based on volumetric primitives do not possess the variety of structural attachment relations available in our representation.

In the next section several examples of object descriptions are provided. They are offered mainly to illustrate the broad scope engendered by the Coons patch and cubic B-spline as representation primitives.

II.5 Examples

The first few examples in this section consist of objects which are represented by a restricted form of Coons surface patch, which we shall refer to as a type 0 patch. In the corner definition matrix of a type 0 patch the corner tangent vectors (00_u, 01_u, 10_u, 11_u, 00_w, 01_w, 10_w, and 11_w) and the corner twist vectors (00_{uw}, 01_{uw}, 10_{uw}, and 11_{uw}) are all zero. Also no across boundary derivative curves are associated with a type 0 patch. A type 1 patch would have at least one non-zero corner tangent or corner twist vector. It may or may not have across boundary derivative curves associated with it.

As these examples demonstrate, an interesting and broad class of object shapes may be represented with type 0 patches. The notion of *identification mapping* is introduced as a didactic aid for describing certain shapes, and its role in the classification of object shapes is explained.

II.5.1 The Right Circular Cylinder

The right circular cylinder is, in some sense, the simplest curved surface which can be formed from a unit square and it clearly illustrates the idea of an *identification mapping*. Bringing together two opposite edges of a square sheet of paper in the obvious way forms a cylinder as in Figure 20. This action is described mathematically as a mapping from the domain set or unit square (both boundary and interior) to the range set or unit cylinder. It is called an *identification mapping* because the edges a and c have been *brought together* or *identified* with one another. When the identification mapping is continuous and onto and the domain set is a topological space, a topology is induced on the range set. This induced topology is referred to as the *identification topology*. In these examples we illustrate how an identification map is achieved in our representation.

For the right circular cylinder, we achieve the identification in the following way:

(1) choose the boundary curves:

u0 - a unit circle in the plane z=0 centered at the origin (0,0,0)

u1 - a unit circle in the plane z=1 centered at (0,0,1)

0w - a straight line connecting the point (1,0,0) on the curve u0 to the point (1,0,1) on the curve u1

1w - the same straight line as 0w

(2) set the corners of the patch:

00 = (1,0,0)
01 = (1,0,1)
10 = (1,0,0)
11 = (1,0,1)

(3) set the tangents and twist vectors in the corner definition to zero

(4) use simple linear blending functions

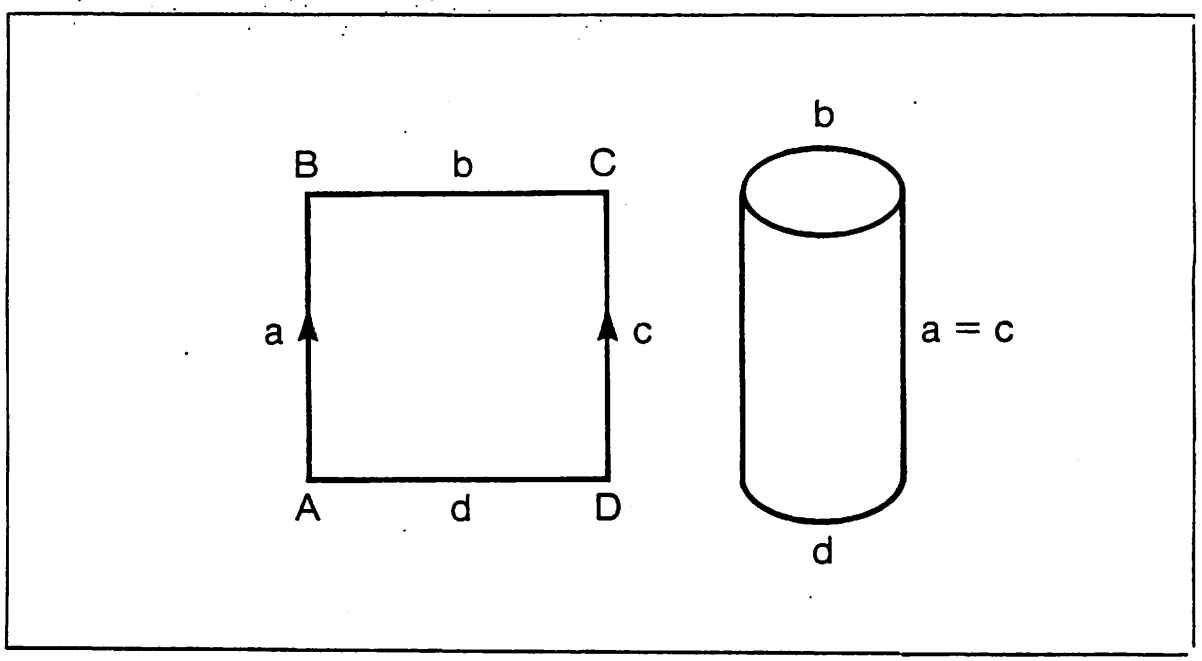

Figure 20: Identification map of unit square to cylinder

The reader should observe the following aspects of this particular representation. First, it is clear that the Coons patch equation is symmetric in u and w; thus, swapping the roles of the u boundary curves for the w boundary curves and making the corresponding changes to the corner points does not change the shape of the surface. Secondly, a type 0 Coons patch was used. In this form, the tangents and twist vectors are set to zero, thus eliminating the "slope correction" component of the surface equation. Finally, the identification was achieved by using a closed curve (in this case a circle) to identify two pairs of points (A is identified with D and B is identified with C in figure 20). A single curve (a straight line in 3-space) is used to identify the other edges (a and c) of the unit square.

Note that, in some sense, we have decribed a volume with a single, type 0 Coons surface patch by appropriate choice of boundary curves and corner points. Nothing has been implied about the physical characteristics of the surfaces of the cylinder; we have merely described a method of obtaining points on one of its surfaces. The existence of the top and bottom surfaces is implied by the planarity of the the curves u0 and u1. In addition, all of the

surfaces of the cylinder could have additional characteristics such as virtual/real, transparent/opaque, etc. associated with them.

Nothing in this particular representation required cubic B-splines; however, their usefulness will become clear as we extend the representation to more complex shapes. In the above representation the unit circle was approximated by a cubic B-spline with ten vertices and the straight line was approximated exactly by a cubic B-spline with two vertices, each of multiplicity 3. It is clear that a cylinder of any scale may be constructed in this manner, since the unit square may be continuously deformed into a rectangle of any size before the identification is performed.

II.5.2 The Right Circular Cone

The right circular cone is easily achieved (see Figure 21) by an identification which is almost identical to the mapping for the right circular cylinder. (A figure whose title is preceded by an * was machine-generated and plotted on a CRT without hidden-line removal.) The difference is that the curve u1 becomes a single point--i.e. for all values of u the value of the curve u1 is the point (0,0,1). The curves 0w and 1w become the straight line from (1,0,0) to (0,0,1). The corner definition is:

$$00 = (1,0,0)$$
$$01 = (0,0,1)$$
$$10 = (1,0,0)$$
$$11 = (0,0,1)$$

Note that this corner definition has precisely the same pattern as the corner definition for the right circular cylinder -- namely that 00 = 10 and 01 = 11. A critical difference in the description of the two shapes is contained in the relative shapes of the two boundary curves u0 and u1. In the case of the cylinder the two curves are planar, lie in parallel planes, have identical shape and scale; for the cone one of the curves is a point and the other is a closed planar figure.

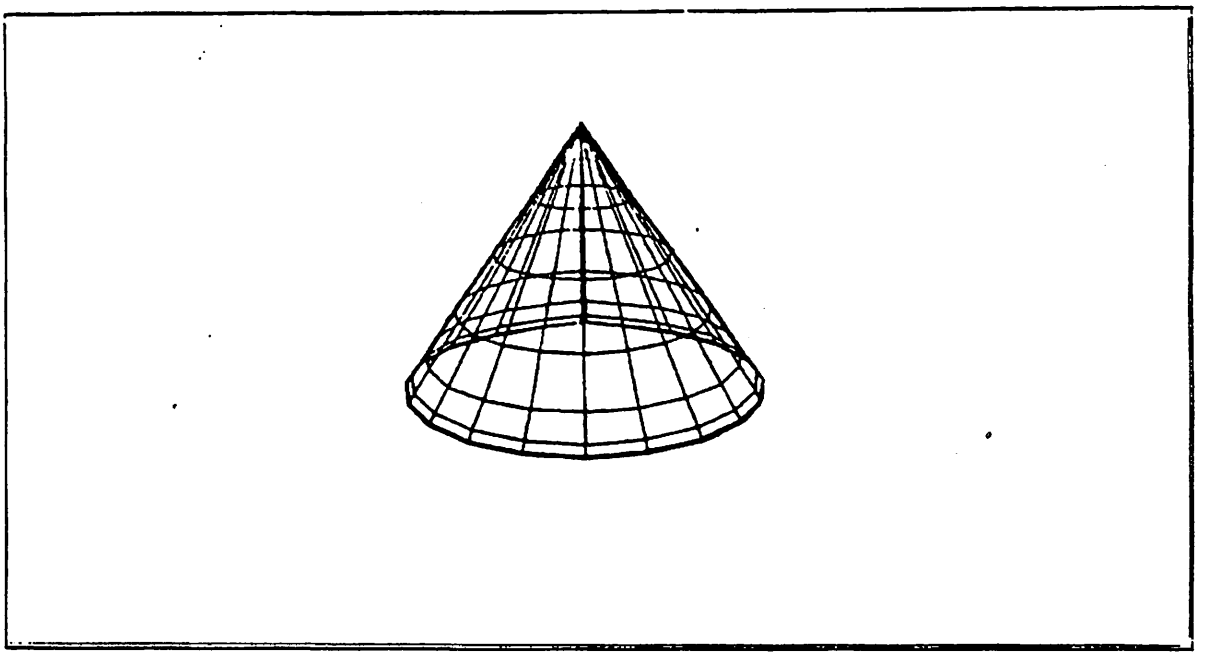

Figure 21: *Patch representation of the right circular cone

II.5.3 The Wedge

By making use of the multiple knot feature of splines, one can obtain piecewise planar surface patches and thereby represent polyhedral volumes/objects. A simple polyhedral object which falls in the same general class as the right circular cylinder (in that it has same identification map) is the wedge. It consists of two triangular curves lying in parallel planes and connected by straight lines. Each triangle is a closed cubic B-spline curve generated by the following defining polygons:

u0				u1			
x	y	z	multiplicity	x	y	z	multiplicity
1	0	0	3	1	0	5	3
0	1	0	3	0	1	5	3
-1	0	0	3	-1	0	5	3
1	0	0	3	1	0	5	3

The connecting curves 0w and 1w are generated by the defining polygon for the straight line

from (1,0,0) to (1,0,5):

0w				1w			
x	y	z	multiplicity	x	y	z	multiplicity
1	0	0	3	1	0	0	3
1	0	5	3	1	0	5	3

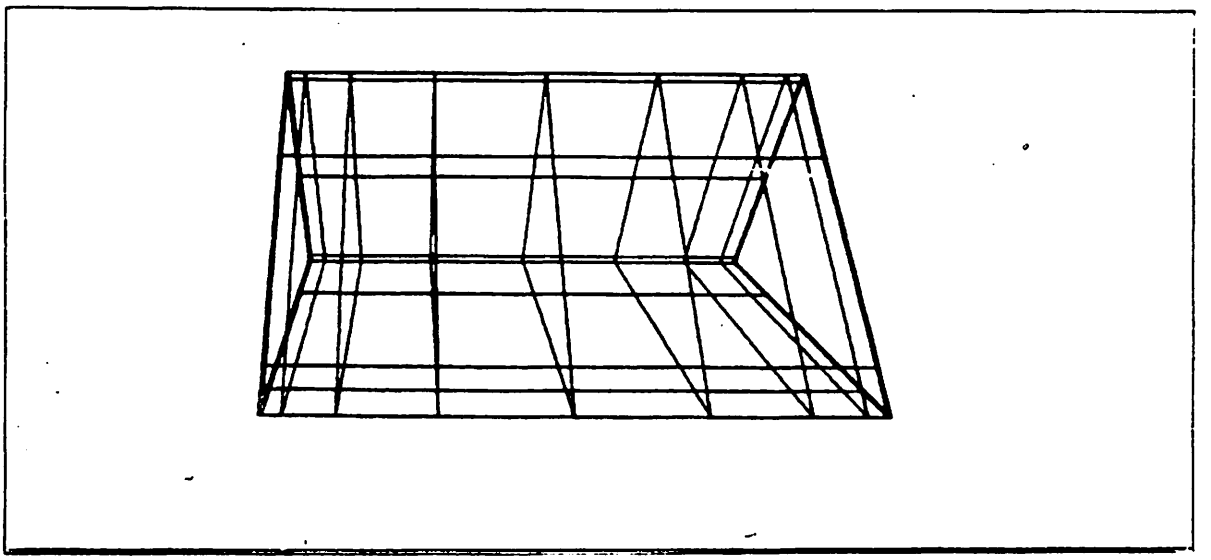

Figure 22: *Patch representation of the Wedge

II.5.4 The Tetrahedron

The tetrahedron is obtained from the wedge in a fashion analogous to the way in which the cone is obtained from the cylinder. The identification map remains the same and one of the triangular curves degenerates to a point (see Figure 23)

II.5.5 The Cylinder and Cone Shape Classes

It is clear from the previous examples that there is a certain commonality between the right circular cylinder and the wedge and between the right circular cone and the tetrahedron. The similarities are expressed in the identification mappings for the boundary curves of the

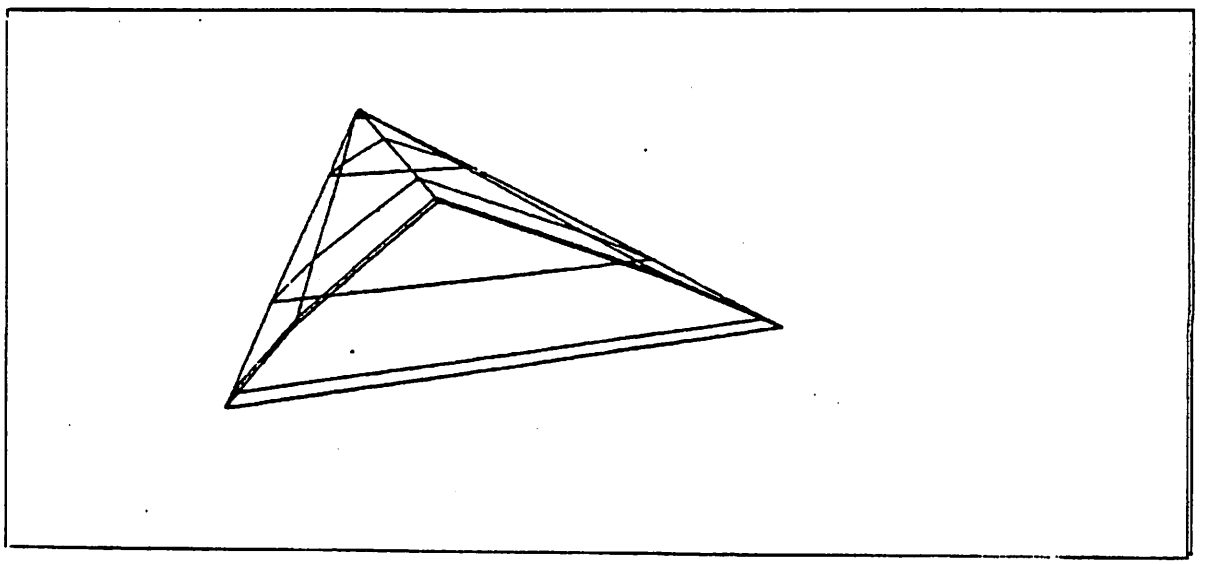

Figure 23: *The Tetrahedron

patch, the nature of the curves (closed, planar), and relationships between the curves (they lie in parallel planes). These are precisely the kinds of invariant characteristics which are useful for classifying shapes.

Thus, any single type 0 patch of the cylinder class must satisfy the identification mapping of the right circular cylinder and the two curves u0 and u1 must be closed planar curves lying in parallel planes. The curve 0w must be identical to the curve 1w and it must connect the starting point of u0 to the starting point of u1. Figures 24 and 25 show two objects of the cylinder class; one object has a non-circular curved base and the other has a non-convex polygonal base.

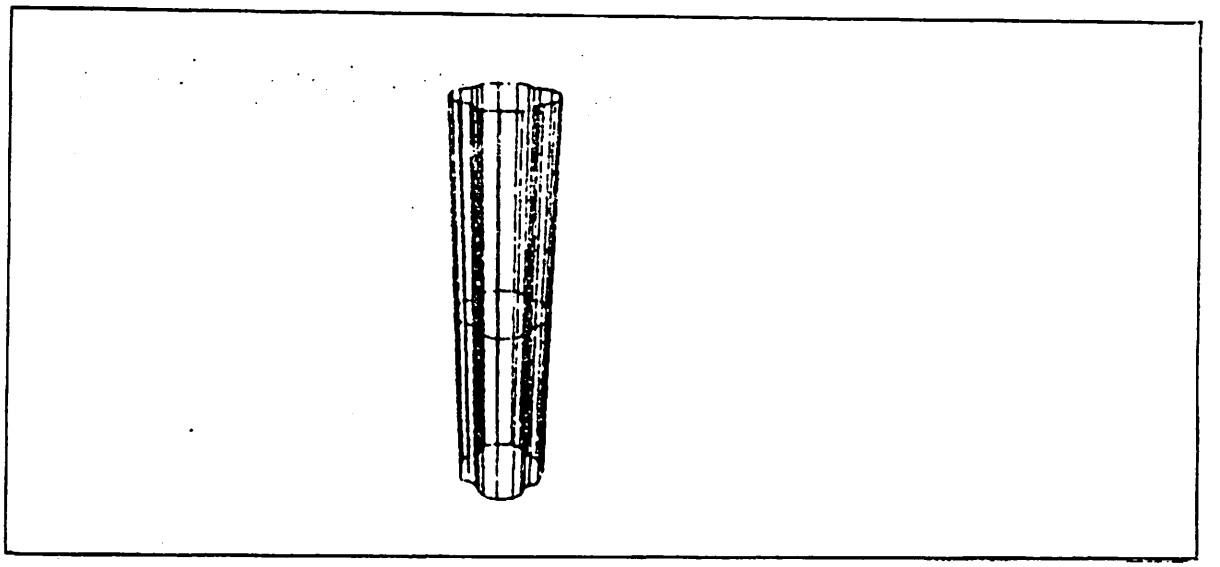

Figure 24: *A Knurled Screwdriver Handle

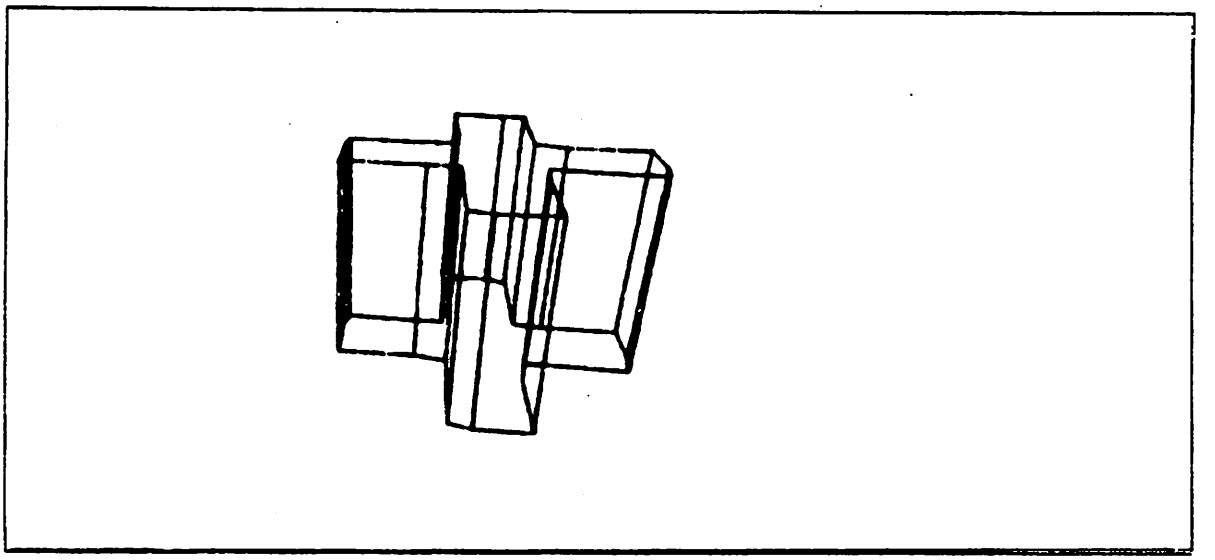

Figure 25: *A Generalized cylinder with non-convex polygonal base

II.5.6 The Running Shoe

The running shoe (see Figure 26) is a practical example of a real world object which is easily represented by a type 0 patch of the cylinder class. The upper "elliptically-shaped" curve is connected to the lower "sole-shaped" curve by a straight line at the rear of the shoe.

Only 7 vertices are required in the defining polygon of the upper curve and 15 vertices for the lower curve. Of course, two vertices of multiplicity 3 are required for the straight line down the rear of the shoe. The line represents the position at which the two halves of the physical shoe would normally be stitched together. In this particular case, a geometric component of the representation has been made to coincide with a meaningful part of the object. Because of the flexibility of designing object descriptions in our representation, object semantics may be more easily associated with geometric components of the description. Representations with volumetric primitives are severely restrictive in this regard.

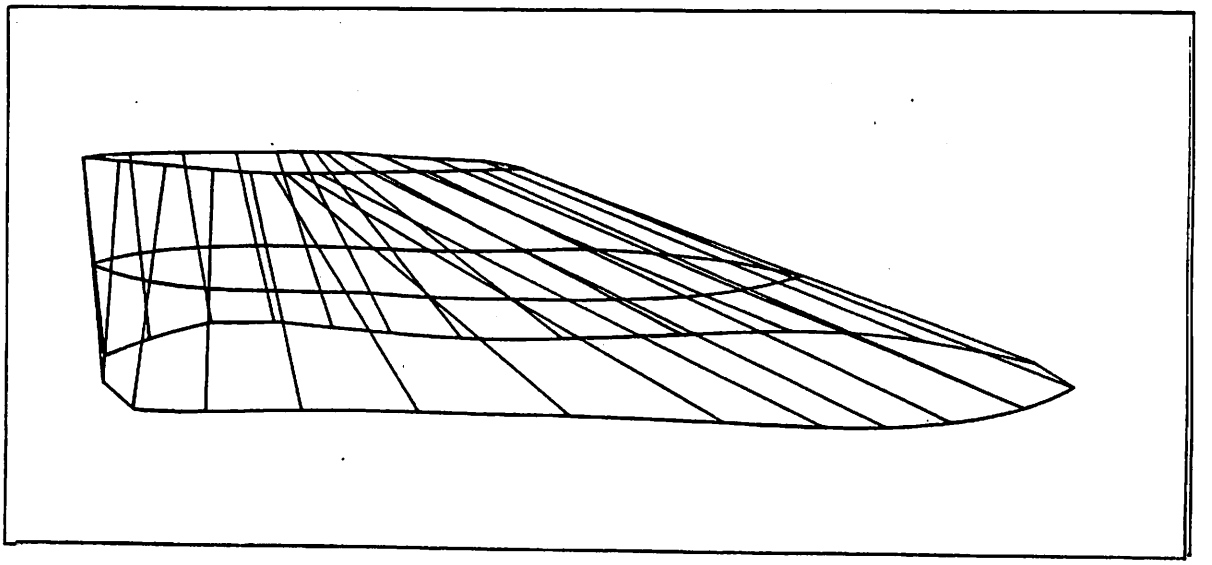

Figure 26: *A Running Shoe

II.5.7 The Telephone Receiver

The telephone receiver is also a single type 0 patch. The unusual aspects of this representation are that the two circular curves lie in the same plane and they are not connected by a straight line. The connecting curve (0w=1w) starts at the starting point of circle u0, rises up out of the plane, goes across to just above the starting point for u1, and falls down into the plane at the starting point of u1 (see Figure 27)

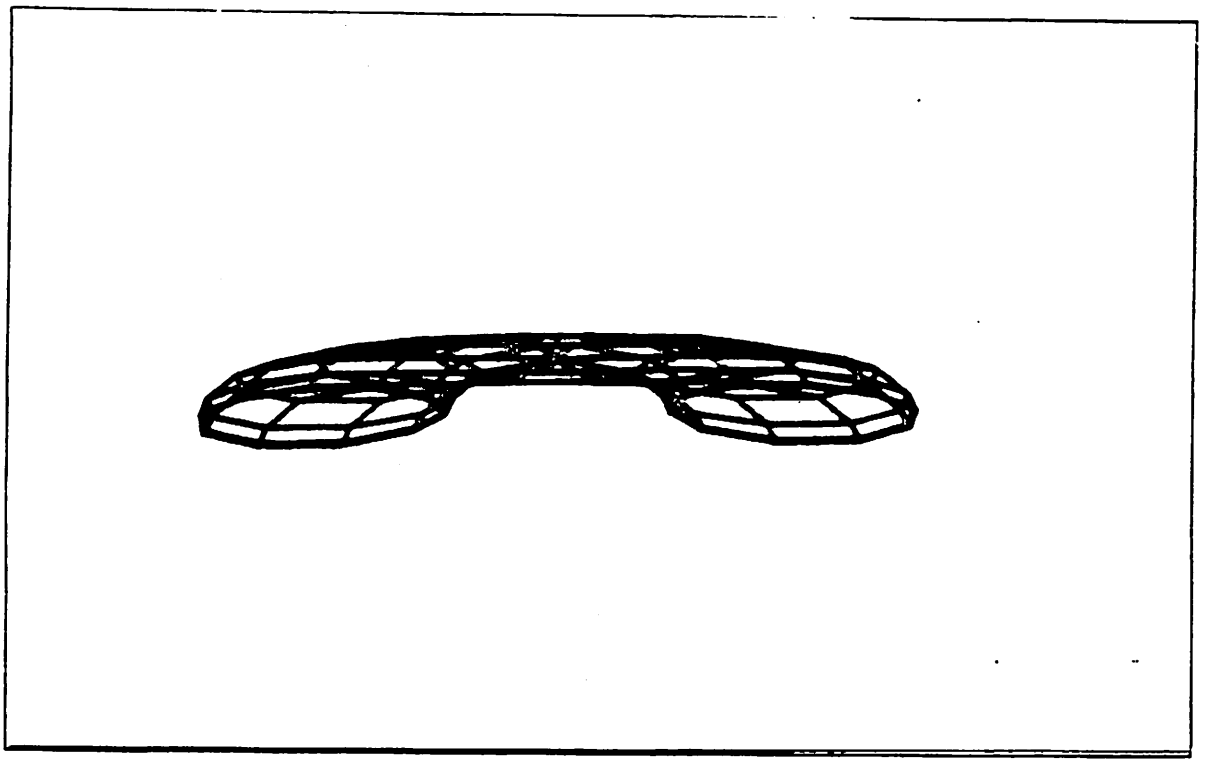

Figure 27: *A Telephone Receiver

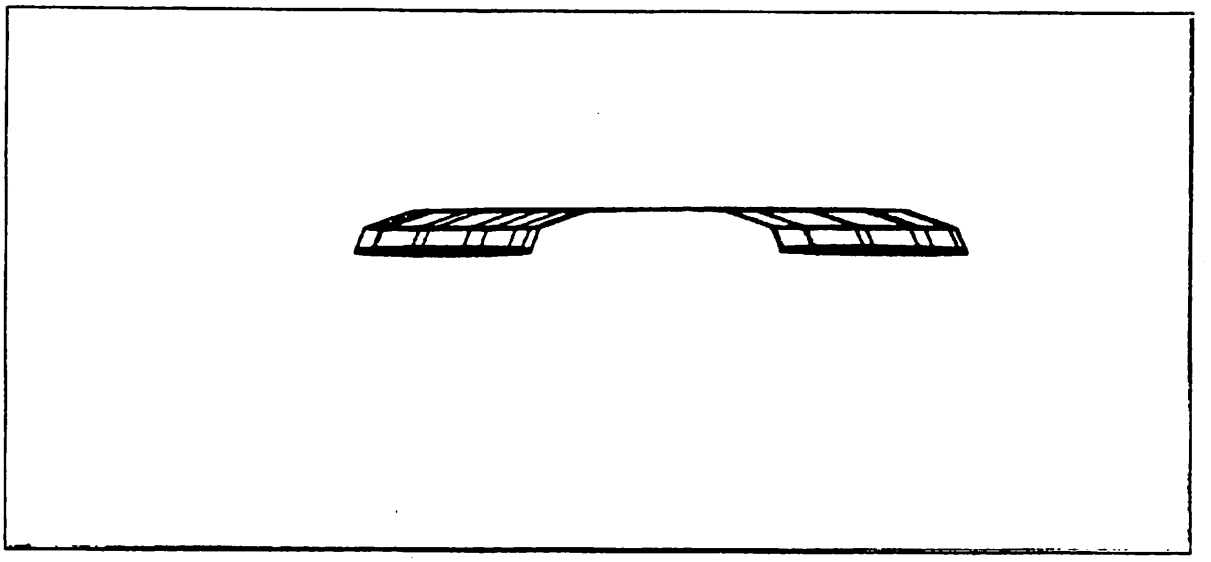

Figure 28: *A Cross-sectional View of a Telephone Receiver

Notice in the cross-sectional view that the receiver has infinitesimal thickness along the central portion of the connecting curve. This results from making the curves 0w and 1w coincide. A complete telephone receiver would require a second patch to give it thickness.

II.5.8 The Screwdriver

The screwdriver may be formed from three type 0 patches. The blade is represented by a single type 0 patch formed from a circle, a rectangle, and a straight line connecting them. The circle at the base of the blade is smoothly blended into a small rectangle at the tip of the blade. The handle is represented by two patches. The first patch is planar; it is an annulus formed from two concentric circles with the radius of the inner circle the same as that of the base of the screwdriver blade. The second patch is a right circular cylinder with bases of the same radius as the outer circle of the annulus patch. Only edge-to-edge attachment is required to compose this object. The base of the blade is attached to the inner circle of the annulus and the outer circle of the annulus is attached to one end of the cylindrical handle.

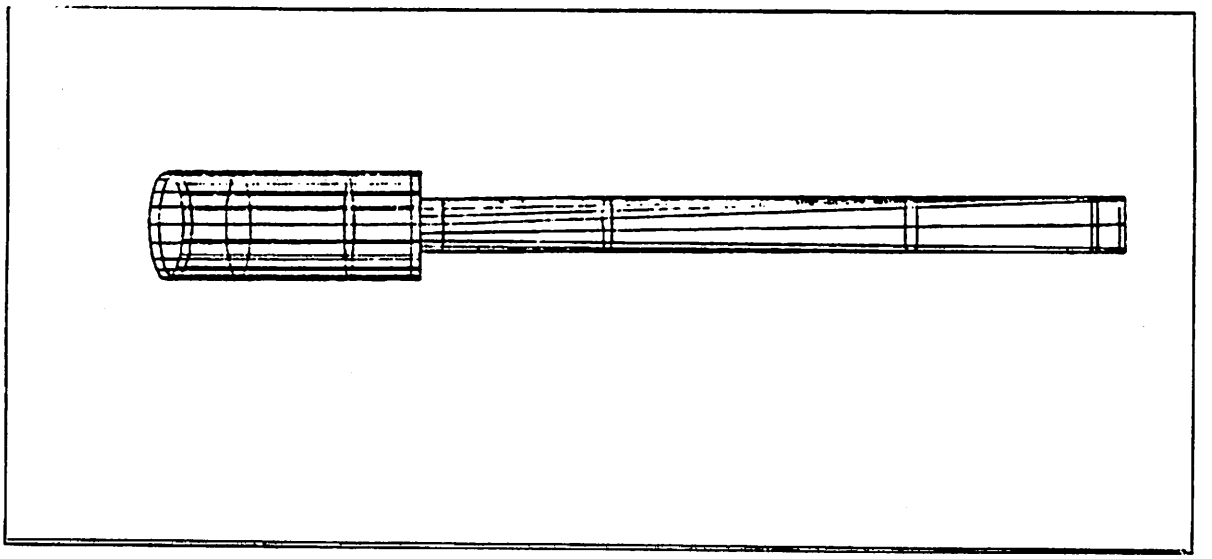

Figure 29: *A Screwdriver

II.6 3D Shape Features

The purpose of this section is to describe a number of features which may be directly computed from our representation and which are useful in the shape classification of objects described in our representation. Sub-sections II.6.1 and II.6.3 describe the features computed for cubic B-spline curves and Coons patches respectively. II.6.2 shows a tabular summary of the values of these features for the example objects in section II.5 and II.6.4 describes the impact on shape of varying components of the surface patch.

II.6.1 Features of Curves

All curves whether they are boundary curves, across-boundary derivative curves, or blending functions are represented by their defining polygons in cubic B-spline representation (see Appendix I). The defining polygon is just an ordered list of points called vertices and these vertices are assumed to be connected by straight lines. A particular point may occur more than once in succession in the defining polygon. In such a case the point is referred to as a multiple vertex. The uniform cubic B-spline curve associated with a given defining polygon having N vertices is generated by the following formula applied to each of the coordinates of the defining polygon independently.

$$P_X(u) = [s^3 \quad s^2 \quad s \quad 1]\ 1/6 \begin{vmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{vmatrix} \begin{vmatrix} x_i \\ x_{i+1} \\ x_{i+2} \\ x_{i+3} \end{vmatrix}$$

where x_i = x-coordinate of vertex V_i
n = the number of vertices -3 (i.e. $n = N-3$)
i = the integer part of $n-u$
s = the fractional part of $n \times u$

Similarly for Y and Z,

$$P_Y(u) = [s^3 \quad s^2 \quad s \quad 1]\ 1/6 \begin{vmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{vmatrix} \begin{vmatrix} y_i \\ y_{i+1} \\ y_{i+2} \\ y_{i+3} \end{vmatrix}$$

$$P_Z(u) = [s^3 \quad s^2 \quad s \quad 1]\ 1/6 \begin{vmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{vmatrix} \begin{vmatrix} z_i \\ z_{i+1} \\ z_{i+2} \\ z_{i+3} \end{vmatrix}$$

Boundary curves and across-boundary derivative curves are generally 3D space curves, whereas blending functions are actually univariate functions. Thus, the defining polygon of a blending function may be represented by an nx2 array of points.

The following low-level features characterize the shapes of all cubic B-spline curves in the system and they are used to aid in the organization of the data base for indexing purposes.

Open/Closed (OC)

This is a simple binary attribute which indicates whether the curve is open or closed. For all closed cubic B-splines the first and last vertex of the defining polygon are the same point.

$$OC = \begin{cases} 0 & \textit{if open} \\ 1 & \textit{if closed} \end{cases}$$

Planar/Non-planar (PL)

Planarity is achieved when a given coordinate has the same value for every vertex in the defining polygon.

$$PL = \begin{cases} 0 & \textit{if planar} \\ 1 & \textit{if non} - \textit{planar} \end{cases}$$

Smoothness (SM)

Smoothness is destroyed whenever at least one vertex in the defining polygon occurs with multyplicity greater than one.

$$SM = \begin{cases} 0 & \text{if } smooth \\ 1 & \text{if } non-smooth \end{cases}$$

Number of Vertices (NV)

The absolute number of vertices in the defining polygon is a gross measure of the complexity of the shape of the curve. A very smooth curve will generally have fewer vertices than a very wiggly curve.

Number of Multiple Vertices (MV)

This is also a measure of the complexity of the curve. Vertices of multiplicity 3 indicate a curvature discontinuity at the vertex. The portions of the spline entering either side of the vertex must have zero curvature in the neighborhood of the vertex. Generally multiple vertices indicate straight line components of the spline and, therefore, simpler shapes.

Number of Angular Turns(NT)

The number of angular turns differs from the number of vertices because consecutive segments of the vertex polygon are sometimes collinear.

Ratio of Multiple Vertices to Number of Vertices (RV=MV/NV)

This is a significant measure of complexity of the curve. As the ratio tends toward unity, the shape tends toward polygonal.

Ratio of Angular Turns to Vertices(RT=NT/NV)

This is also a measure of the complexity of the curve.

Angular Variability (AV)

Angular variability is a measure of smoothness of the curve represented by the defining polygon. Attneave (Attneave 1957, p.223) defined this measure for closed 2D polygons as "...the arithmetic mean (sign ignored) of the algebraic differences in degrees of the slope-change (sign observed) between all successive or adjacent angles taken in overlapping pairs about the contour, convex angles being considered positive and concave ones negative". This definition is sufficient for closed 2D polygons; however, our representation requires corresponding definitions for open 2D and 3D polygons as well as closed 3D polygons. The term "polygon" is used loosely to refer to the geometrical object consisting of the straight lines connecting the points which define the vertices of the cubic B-spline.

2D Angular Variability

Closed Polygon

$$AV = \frac{1}{NT}\left[(\theta_1 - \theta_n) + \sum_{1}^{n-1} (\theta_{i+1} - \theta_i)\right]$$

where θ_i = the interior angle at the ith vertex V_i

Open Polygon

$$AV = \frac{1}{NT}\sum_{2}^{n-2}(\theta_{i+1}-\theta_i)$$

where θ_i is the interior angle of the triangle formed by connecting vertex V_{i-1} to vertex V_{i+1} (see Figure 30).

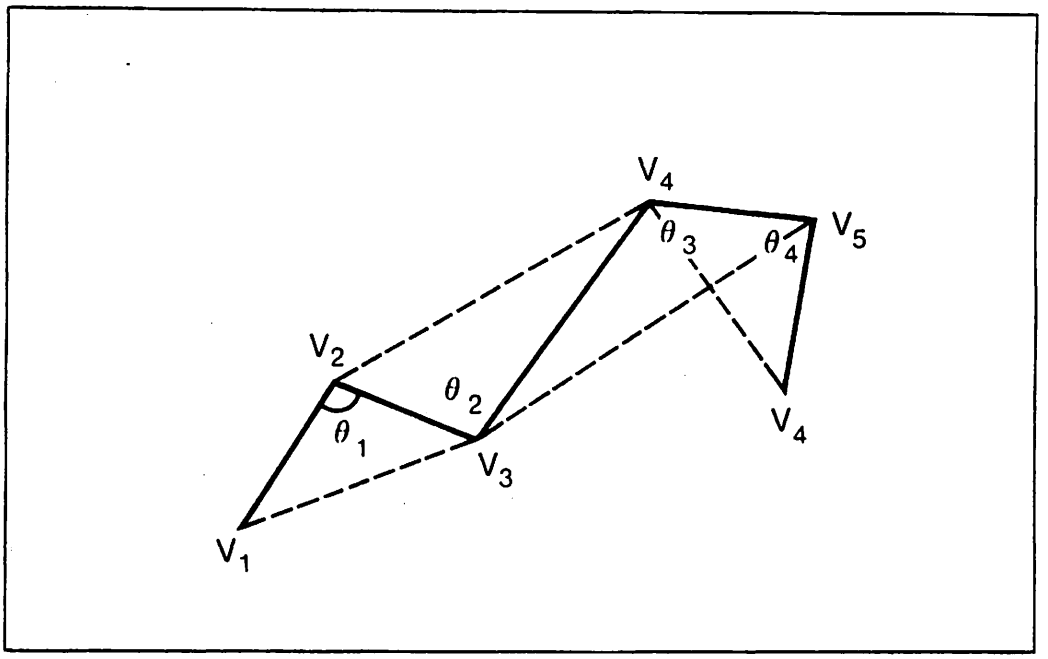

Figure 30: Angular Variability for Open 2D Polygon

3D Angular Variability

The angular variability for a 3D defining polygon is a 3-vector consisting of the angular variability in each of the planes, X-Y, Y-Z, and X-Z.

$$AV_{3D} = (AV_{XY}, AV_{YZ}, AV_{XZ})$$

This measure is used for both open and closed 3D defining polygons.

Polygon Length/Baseline(PGL/BL)

This feature is a rough measure of how much the vertex polygon deviates from a straight line. It is computed only for open polygons.

$$PGL/BL = \frac{1}{d(V_1,V_n)} \sum_{i=1}^{n-1} d(V_i,V_{i+1})$$

where $d(V_i,V_{i+1})$ is the Euclidean distance from vertex V_i to vertex V_{i+1}.

Compactness(C)

This is the standard 2D compactness measure of perimeter squared divided by area. When a boundary curve is closed and planar, the compactness measure is computed in lieu of PGL/BL

Maximum Angular Turn(MAXAT)

This measure is an approximation to the maximum curvature along the curve.

Maximum Leg Length/Minimum Leg Length(MAXL/MINL)

This is a measure of the range in lengths of legs along the vertex polygon. If its value is 1, all of the legs are of equal length.

Before proceeding to a description of the shape features for Coons patches, we first show an analysis of the boundary curves for the example objects in section II.5. No analysis of blending functions is shown. In all cases we used the standard blending functions described in Appendix II.

II.6.2 Boundary Curve Feature Tables

Table 1 The right circular cylinder

	u0	u1	0w	1w
OC	1	1	0	0
PL	0	0	1	1
SM	0	0	0	0
NV	10	10	2	2
MV	0	0	2	2
NT	10	10	0	0
MV/NV	0	0	1	1
NT/NV	1	1	0	0
PGL/BL	0	0	1	1
C	12.57	12.57	0	0
MAXAT(DEG)	144	144	0	0
MAXL/MINL	1	1	1	1
AV_{3D}	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)

Table 2 The right circular cone

	u0	u1	0w	1w
OC	1	1	0	0
PL	0	1	1	1
SM	0	1	0	0
NV	10	1	2	2
MV	0	1	2	2
NT	10	1	0	0
MV/NV	0	1	1	1
NT/NV	1	0	0	0
PGL/BL	0	0	1	1
C	12.57	0	0	0
MAXAT(DEG)	144	0	0	0
MAXL/MINL	1	0	1	1
AV_{3D}	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)

Table 3 The wedge

	u0	u1	0w	1w
OC	1	1	0	0
PL	0	0	1	1
SM	1	1	0	0
NV	3	3	2	2
MV	3	3	2	2
NT	3	3	0	0
MV/NV	1	1	1	1
NT/NV	1	1	0	0
PGL/BL	0	0	1	1
C	23.31	23.31	0	0
MAXAT(DEG)	60	60	0	0
MAXL/MINL	1	1	1	1
AV_{3D}	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)

Table 4 The tetrahedron

	u0	u1	0w	1w
OC	1	1	0	0
PL	0	1	1	1
SM	1	1	0	0
NV	3	1	2	2
MV	3	1	2	2
NT	3	0	0	0
MV/NV	1	1	1	1
NT/NV	1	0	0	0
PGL/BL	0	0	1	1
C	23.31	0	0	0
MAXAT(DEG)	60	0	0	0
MAXL/MINL	1	0	1	1
AV_{3D}	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)

Table 5 The running shoe

	u0	u1	0w	1w
OC	1	1	0	0
PL	0	0	1	1
SM	0	0	0	0
NV	8	16	2	2
MV	0	0	2	2
NT	6	16	0	0
MV/NV	0	0	1	1
NT/NV	.75	1	0	0
PGL/BL	0	0	1	1
C	16.6	7.6	0	0
MAXAT(DEG)	152	177	0	0
MAXL/MINL	22.83	9.82	1	1
AV_{3D}	(16.7,0,0)	(11.7,0,0)	(0,0,0)	(0,0,0)

Table 6 The telephone receiver

	u0	u1	0w	1w
OC	1	1	0	0
PL	0	0	0	0
SM	0	0	1	1
NV	10	10	7	7
MV	0	0	2	2
NT	10	10	2	2
MV/NV	0	0	.29	.29
NT/NV	1	1	.29	.29
PGL/BL	0	0	1.33	1.33
C	12.57	12.57	0	0
MAXAT(DEG)	144	144	90	90
MAXL/MINL	1	0	2	2
AV_{3D}	(0,0,0)	(0,0,0)	(0,0,90)	(0,0,90)

The reader should note that the 2D angular variability measure for straight lines in the plane is zero as you would expect. Consequently, the 3D angular variability for a straight line in space is (0,0,0). Straight lines are classified as smooth despite the fact that they are terminated by vertices of multiplicity 3. In addition, straight lines in space are not considered to be planar; planar curves are those curves which lie in a *single* plane. A curve which consists of a single point of multiplicity 3 (such as the apex of the right circular cone) is classified as both *non-planar* and *non-smooth*.

There is some overlap in the information provided by the set of features presented above and this redundancy should be exploited in an implementation which uses these features.

II.6.3 Features of surfaces patches

In this section we describe shape features which may be computed directly from the geometric description of the Coons surface patch. Some of these features have direct 2D analogues and the relationships between these 2D and 3D features should be exploited during indexing. For the present we shall merely describe the features and how they are computed.

Compactness(C)

The compactness measure, $C = P^2/A$,has a direct 2D analogue (Attneave 1956). It is computed by dividing the square of the perimeter of the surface patch by its surface area. The perimeter is approximated by summing the lengths of the defining polygons for the four boundary curves (u0,u1,0w,1w). The surface area of a Coons surface patch is computed by the following formula:

$$A = \int_0^1 \int_0^1 \sqrt{J_X^2 + J_Y^2 + J_Z^2}\, du dw$$

where the jacobians, J_X, J_Y, and J_Z are defined by the determinants:

$$J_X = \begin{vmatrix} y_u & z_u \\ y_w & z_w \end{vmatrix} \quad J_Y = \begin{vmatrix} x_u & z_u \\ x_w & z_w \end{vmatrix} \quad J_Z = \begin{vmatrix} x_u & y_u \\ x_w & y_w \end{vmatrix}$$

The perimeter,P , is computed by:

$P = dlength(u0) + dlength(u1) + dlength(0w) + dlength(1w)$.

where *dlength* is defined to be $\sum_{i=1}^{n-1} d(V_i, V_{i+1})$

and

$$d(V_i, V_{i+1}) = \sqrt{(x_i - x_{i+1})^2 + (y_i - y_{i+1})^2 + (z_i - z_{i+1})^2}$$

Volume/Surface Area x maximum Edge(V/SAE)

By considering the vectors from 00 to 01, 00 to 10, and 00 to 11 of a Coons surface patch, we may easily compute the volume of the parallelpiped defined by those

vectors using the scalar triple product(see Figure 31): Recall that 00 is a shorthand notation for P(0,0) which is the vector value in Cartesian 3-space of the bivariate function P(u,w) evaluated at u=0, w=0. If the reader is still uncomfortable with this notation, he should review Appendix II.

$$c.(a \wedge b) = \begin{bmatrix} a_x & a_y & a_z \\ b_x & b_y & b_z \\ c_x & c_y & c_z \end{bmatrix}$$

where a is the vector from 00 to 01, b is the vector from 00 to 10, and c is the vector from 00 to 11 ($\wedge$ denotes the vector product and the brackets indicate the determinant in this instance).

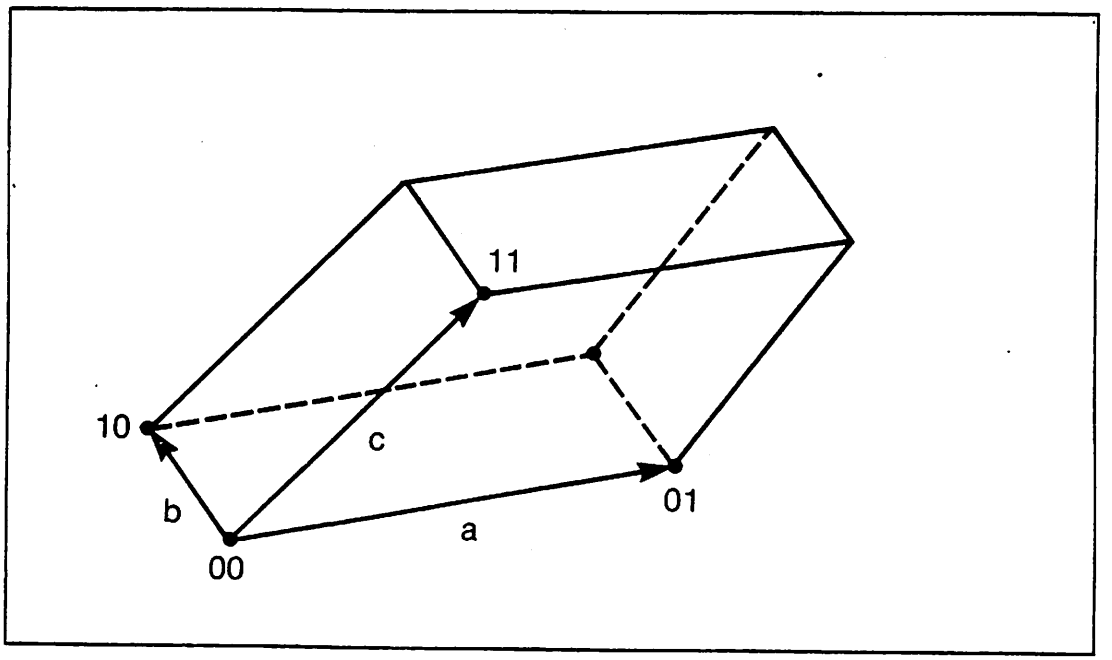

Figure 31: Volume of parallelpiped for corner definition

Since any two vectors lie in a common plane, we choose a and b (two edges of the surface patch corner definition) and compute the area of the parallelogram defined by these two vectors using the vector product equation

$$a \wedge b = An$$

where A is the area of the parallelogram and n is the unit normal vector to the plane of the parallelogram. The projection of the vector c onto the normal n is given by the dot product $c.n$. This projection is the signed distance, d, of the upper parallelogram above/below the plane defined by the vectors, a and b. Thus, the signed volume is the signed distance times the area

$$\pm V = \pm d \times A$$

and

$$\pm d \times A = c.(a \wedge b)$$

V/SAE is a *rough* measure of how the area of the surface patch relates to the volume defined by the corner position matrix. For example, a right circular cylinder with radius 1/2 and height 1 has a V/SAE value of 1/4 and a unit cube has a V/SAE value of 1/6. Of course, a planar patch has a zero volume and this feature is meaningless in that case. Many of the patches with which we deal have a corner definition matrix for which all four corners lie in a plane, but the surface it defines is not planar (e.g. the corvette hood). In such cases, the volume is computed by multiplying the area of the enclosing rectangle of the corner points times the maximum height that the surface goes above the plane of the corner points. The surface may go above and below the plane, in which case the sum of these distances is used in the volume calculation. A similar situation arises with cylindrical objects such as the running shoe. In this case the area of the enclosing rectangle for each of the two planar closed curves is computed. The larger area is selected and it is used in the volume calculation. The height is easily determined from the corner definition matrix.

Identification Mapping(IM)

Identification mappings are used to classify surface patches. Because of the symmetry of the Coons patch representation, only three distinct identification mappings are observed. There are four IM codes defined as follows:

0 = no identification of corners

1 = cylindrical identification -- i.e.

$00 \Leftrightarrow 10$

$01 \Leftrightarrow 11$

or

$00 \Leftrightarrow 01$

$10 \Leftrightarrow 11$

2 = half-cylindrical -- i.e.

Any of the following:

(1) $00 \Leftrightarrow 01$

(2) $00 \Leftrightarrow 10$

(3) $01 \Leftrightarrow 11$

(4) $10 \Leftrightarrow 11$

3 = four-point identification -- i.e.

$00 \Leftrightarrow 01 \Leftrightarrow 10 \Leftrightarrow 11$

Figure 32 demonstrates each of these identification mappings.

The following surface patch shape features are all easily computed from the basic Coons patch representation. They are basically ratios of magnitudes of position, tangent, and twist vectors. Some are angles between such vectors.

Position Vector Magnitude Ratios(PVMR's)

The ratios of the lengths of the sides of the quadrilateral formed by connecting the corners of the patch contain a certain amount of shape information. If two patches have corner definitions which are similar, then there is a chance that they will have similar shapes depending on the shapes of the corresponding boundary and tangent curves. However, if the corner definitions are dissimilar, then the two patches cannot possibly have similar shapes.

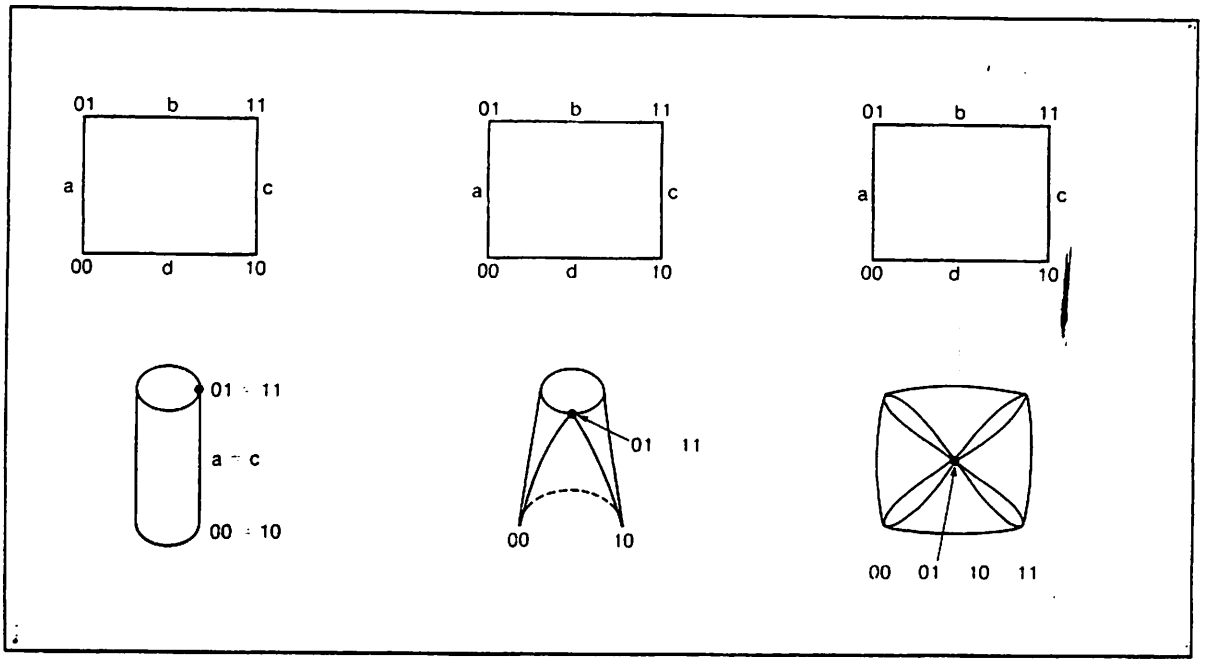

Figure 32: Identification mappings

The position vector magnitude ratios are easier to compute than the side length ratios and together with the position vector angles the position vector magnitude ratios carry the same information. The position vector magnitude ratios are:

(1) | 00 | / | 01 |

(2) | 01 | / | 11 |

(3) | 00 | / | 10 |

(4) | 10 | / | 11 |

where | 00 | denotes the length of the vector from the origin to the of x-y-z space to the corner point in x-y-z space represented by 00 in u-w parameter space.

Position Vector Angles(PVA's)

The position vector angles are easily determined from the dot product formula:

$$a.b = |a|\,|b|\cos(\theta)$$

The four position vector angles which are computed are:

(1) Angle between the vectors 00 and 01

(2) Angle between the vectors 01 and 11

(3) Angle between the vectors 00 and 10

(4) Angle between the vectors 10 and 11

Tangent Vector Magnitude Ratios(TVMR's)

Eight tangent vector magnitude ratios are computed. Each tangent vector is compared with the vector corresponding to the side of the quadrilateral on which it is defined. The eight ratios are:

(1) $| 00_u | / | 00\text{-}10 |$

(2) $| 10_u | / | 00\text{-}10 |$

(3) $| 00_w | / | 00\text{-}01 |$

(4) $| 01_w | / | 00\text{-}01 |$

(5) $| 01_u | / | 01\text{-}11 |$

(6) $| 11_u | / | 01\text{-}11 |$

(7) $| 10_w | / | 10\text{-}11 |$

(8) $| 11_w | / | 10\text{-}11 |$

Of course, some of these ratios are meaningless under the identification mappings which cause the denominator to be a zero vector.

Tangent Vector Angles(TVA's)

For each corner of the patch definition, we compute the angle between the two tangent vectors meeting at the corner. The four angles computed are:

(1) The angle between the vectors 00_u and 00_w

(2) The angle between the vectors 01_u and 01_w

(3) The angle between the vectors 10_u and 10_w

(4) The angle between the vectors 11_u and 11_w

Twist Vector Magnitude Ratios(TWVMR's)

The twist vectors at each corner are compared to the normal vector for the plane defined by the tangent vectors at each corner. The four ratios are:

(1) $| 00_{uw} | / | 00_u \wedge 00_w |$

(2) $| 01_{uw} | / | 01_u \wedge 01_w |$

(3) $| 10_{uw} | / | 10_u \wedge 10_w |$

(4) $| 11_{uw} | / | 11_u \wedge 11_w |$

Twist Vector Angles(TWVA's)

The twist vector angle is the angle between the twist vector and the normal to the plane defined by the tangent vectors at each corner. They are:

(1) $\arccos 00_{uw} \cdot (00_u \wedge 00_w) / | 00_{uw} | \, | 00_u \wedge 00_w |$

(1) $\arccos 01_{uw} \cdot (01_u \wedge 01_w) / | 01_{uw} | \, | 01_u \wedge 01_w |$

(1) $\arccos 10_{uw} \cdot (10_u \wedge 10_w) / | 10_{uw} | \, | 10_u \wedge 10_w |$

(1) $\arccos 11_{uw} \cdot (11_u \wedge 11_w) / | 11_{uw} | \, | 11_u \wedge 11_w |$

II.6.4 Variation of Surface Patch Shape

In this section we briefly summarize some of the effects on surface patch shape which result from the variation of components of the patch -- the corner position vectors, the corner tangent vectors, and the corner twist vectors. Ideally, we would like to classify a given surface patch on the basis of its values for the features described in the preceding section; however, in order to perform the classification, *similarity* classes must be determined by a *ceteris paribus* sensitivity analysis. A complete sensitivity analysis for a broad class of objects is beyond the

scope of this dissertation. We shall give brief qualitative descriptions of the effects of variation of these vector components.

II.6.4.1 Position Vector Variation

The position vectors of a surface patch may be varied in such a way as to change the position vector ratios. If all of the position vectors are scaled uniformly so as to keep the position vector ratios unchanged, the resulting change in surface patch shape is due to the change in tangent vector magnitude ratios(this change will be discussed in the section II.6.4.2).

Given a planar patch as in Figure 33 the side lengths and thus the corner position vector ratios are changed when the 11 corner is moved slightly in the plane of the quadrilateral. An obvious change in the shape is perceived because of our familiarity with regular polygonal figures.

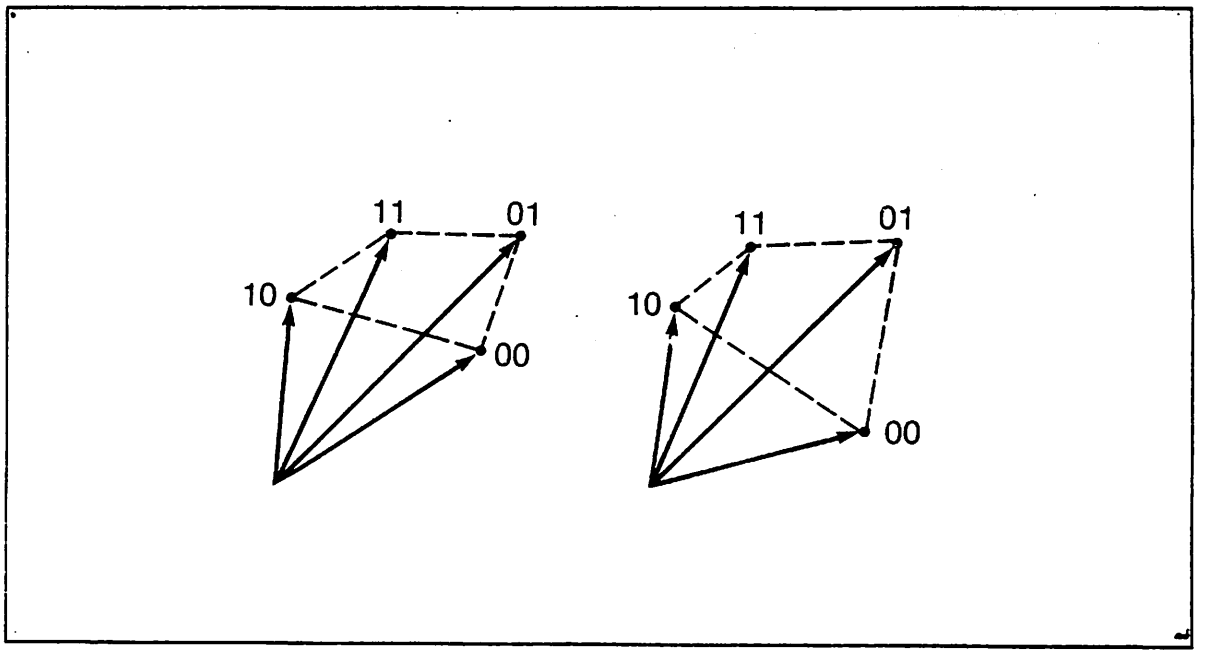

Figure 33: Position Vector Variation

One of the open questions raised by this thesis relates to changes changes in the shape of surface patches with curved boundaries and planar, rectangular, corner position quadrilaterals.

In our example both the position vector magnitude and direction were varied simultaneously. A proper sensitivity analysis would vary each separately, as well.

II.6.4.2 Tangent Vector Variation

Variation in the tangent vectors at the corners of the patch are reflected in the slope at every point of the patch. There are two types of effects which may be achieved. One results from varying the magnitude and the other from varying the direction of the tangent vector. When the magnitude is increased and the direction held constant, the surface becomes steeper -- i.e. the rate of change of the position vector is greater (see Figure 34(a)). When the direction of the tangent vector is changed and the magnitude held constant, the direction from which the associated boundary curve enters the corner is changed. Such a change alters the shape of the surface in the manner shown in Figure 34(b).

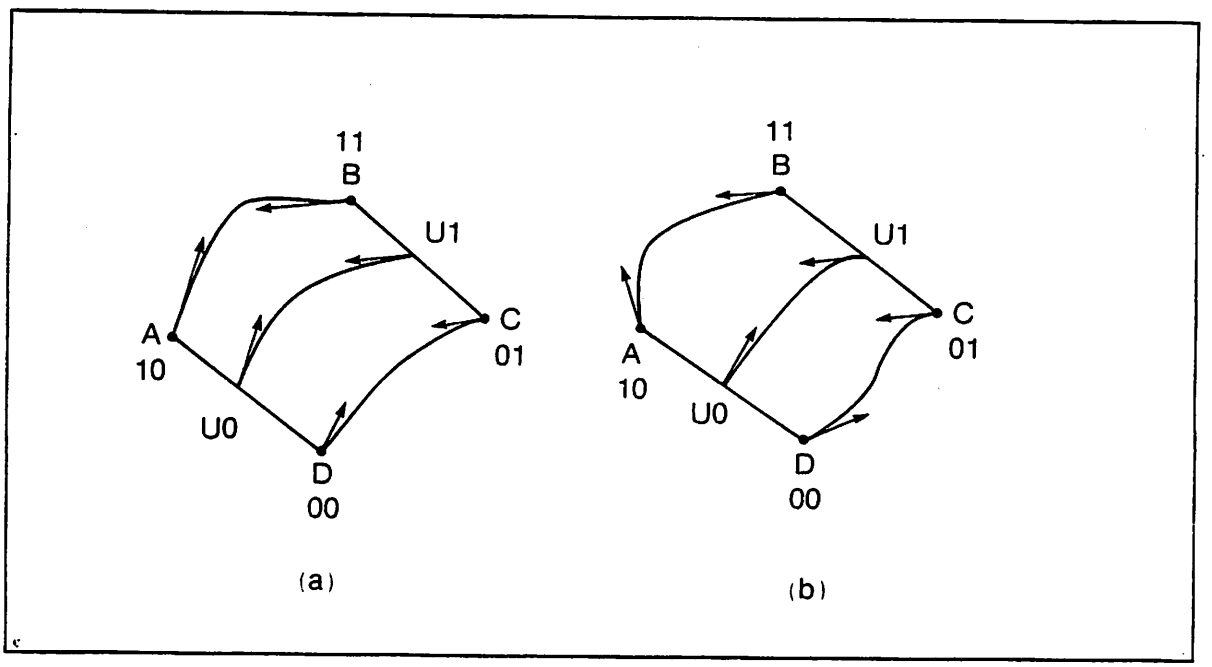

Figure 34: Variation of Tangent Vector[9]

II.6.4.3 Twist Vector Variation

The corner twist vectors determine the amount of twist at each point of the surface. Twist is the rate of change in the w direction of the slope in the u direction. One must be careful when giving geometric interpretations to the twist vector; since the value of the twist at a point on the surface depends upon the parameterization. Faux and Pratt (Faux and Pratt 1979, p.203) show two different parameterizations of a planar surface. Under one parameterization the twist is zero and under the other the twist is non-zero. The non-zero twist for a flat surface is counter to our geometric intuition and it is totally the artifact of the unusual parameterization. Figure 35 shows the effect of changing from zero twist at the 01 corner to a non-zero twist.

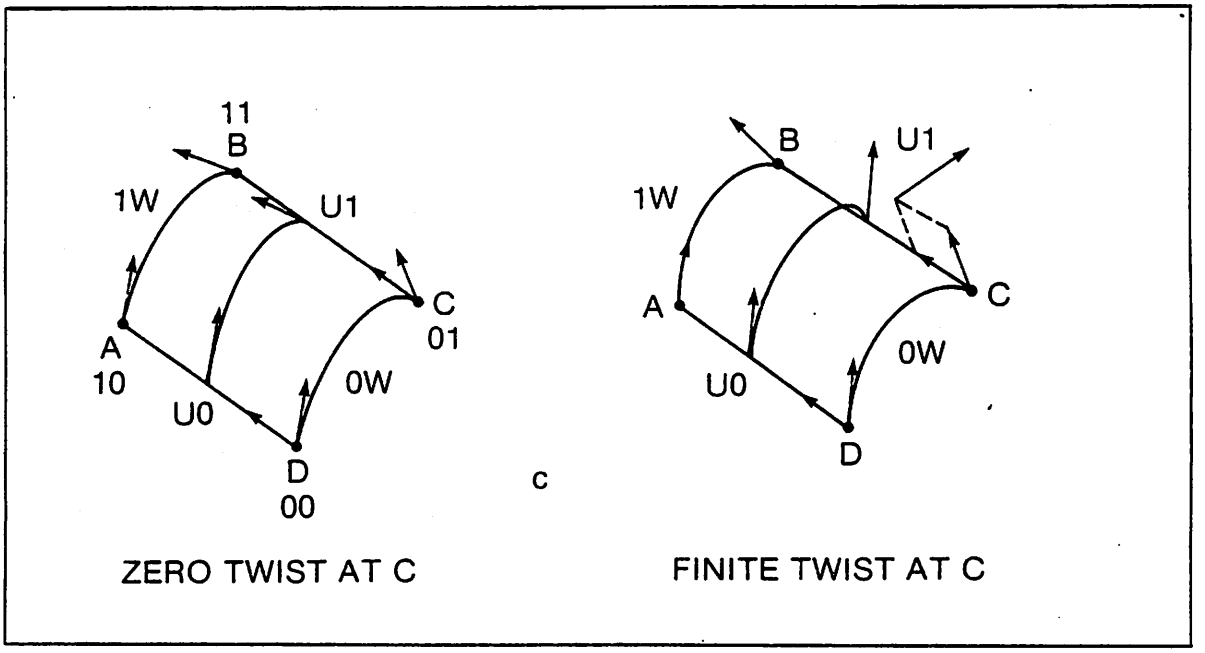

Figure 35: Variation of Corner Twist Vector[10]

For a more graphic example of the effect of changes in the magnitude and the direction of the corner twist vector see Figure 36. The surface patch depicted in the Figure 36(a) is an approximation to the hood of a Corvette automobile. It is described by a single type 0 Coons patch. The lower left corner(00) of the patch has an x-y-z axis superimposed for assisting the viewer with orientation (the x direction runs from left to right in the figure). When a corner

twist vector of (50,0,0) is provided at the 00 corner(Figure 36(d)), the shape of the patch does not change dramatically; however, the first few surface curves appear stretched out in the x-direction. This effect is due to the fact that the patch has zero slope in the x-direction. Thus, increasing the twist in this direction has no effect on the slope of the surface. Notice that the effect is diminished as the other three corners of the patch are approached.

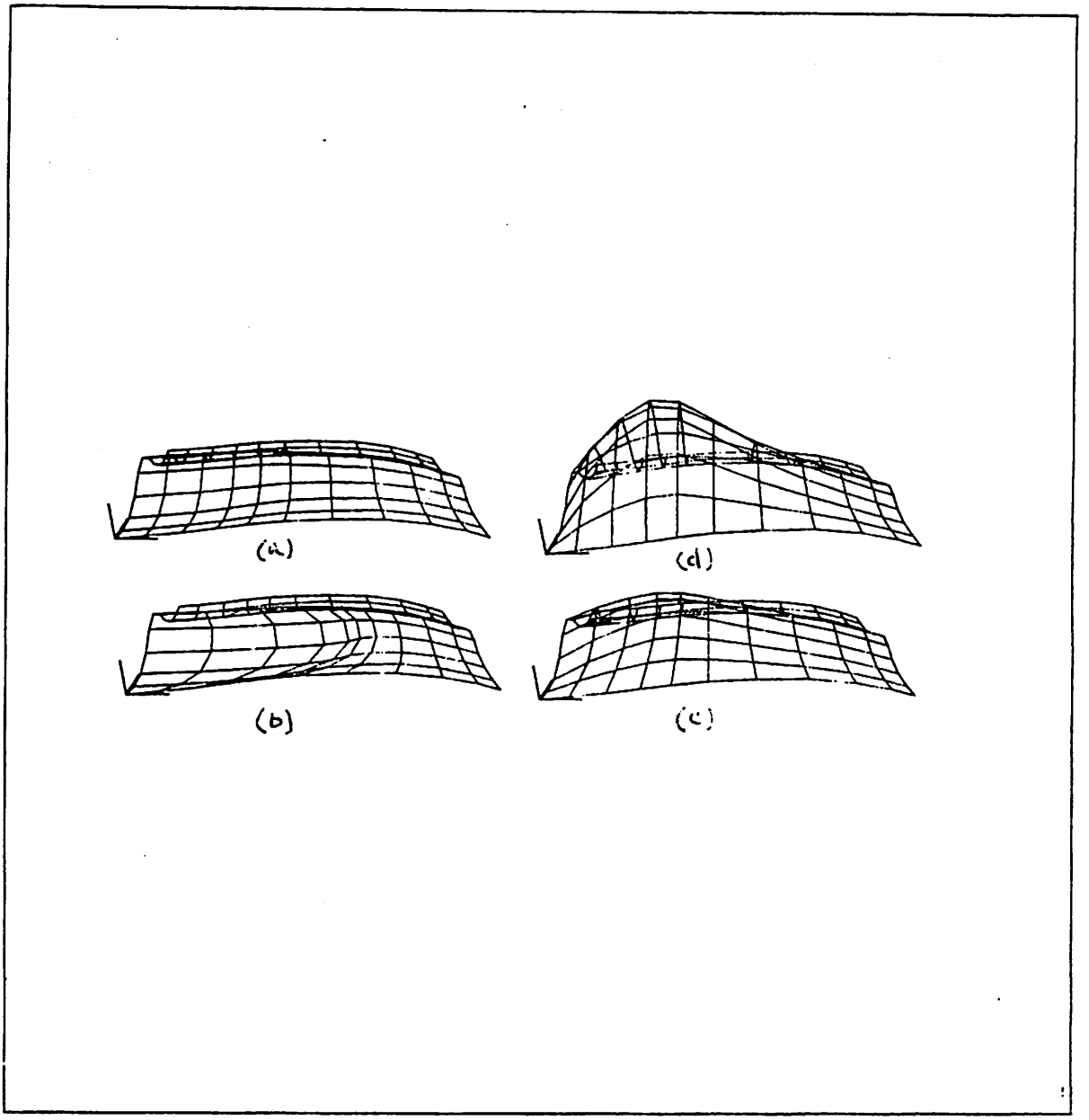

Figure 36: *Effect of twist variation on Corvette hood

The patch in Figure 36(c) displays the effect of adding a twist vector of (0,50,0) at the 00 corner of the original patch. Since the slope of the surface in the y-direction is non-zero as is clear from examining the slope of the boundary curve, the twist has the effect of increasing the slope in the y-direction. In the final patch, Figure 36(b), a twist of (0,0,50) has been added at the 00 point of the original patch. This produces a large hump in the patch.

The four patches of Figure 36 have shape similarities and differences which are captured in the features described in the preceding sections. The shape classification problem requires the analysis of these features to provide discrete ranges of these features over which two patches will be considered similar in shape.

II.7 A complete Representation of the Running Shoe

The purpose of this section is to provide a complete representation of a single object -- the running shoe described in the previous section. It is a simple curved object which serves well as an example. The description will proceed in a top-down fashion from the high-level description of the object to successively lower-level descriptions of its components and their representations.

As described in Chapter I there are many levels of representation in the long term memory (LTM) of the VISIONS system (see Figure 4). These levels contain prototypes for entities. For example, the object class level contains prototypical objects and the schema class level contains prototypical scenes. These prototypes are actually represented by paramaterized routines which, when invoked with actual arguments, produce an instance of the prototype. Thus, the running shoe prototype may be invoked to produce a 3D representation for a particular instance. These instances are also stored in LTM in a set of planes which parallel the class planes (see Figure 37).

We shall describe the representation of a single instance of the running shoe prototype. In this example, the running shoe instance is represented by a single Coons surface patch, although other decompositions are possible. The patch has several components associated

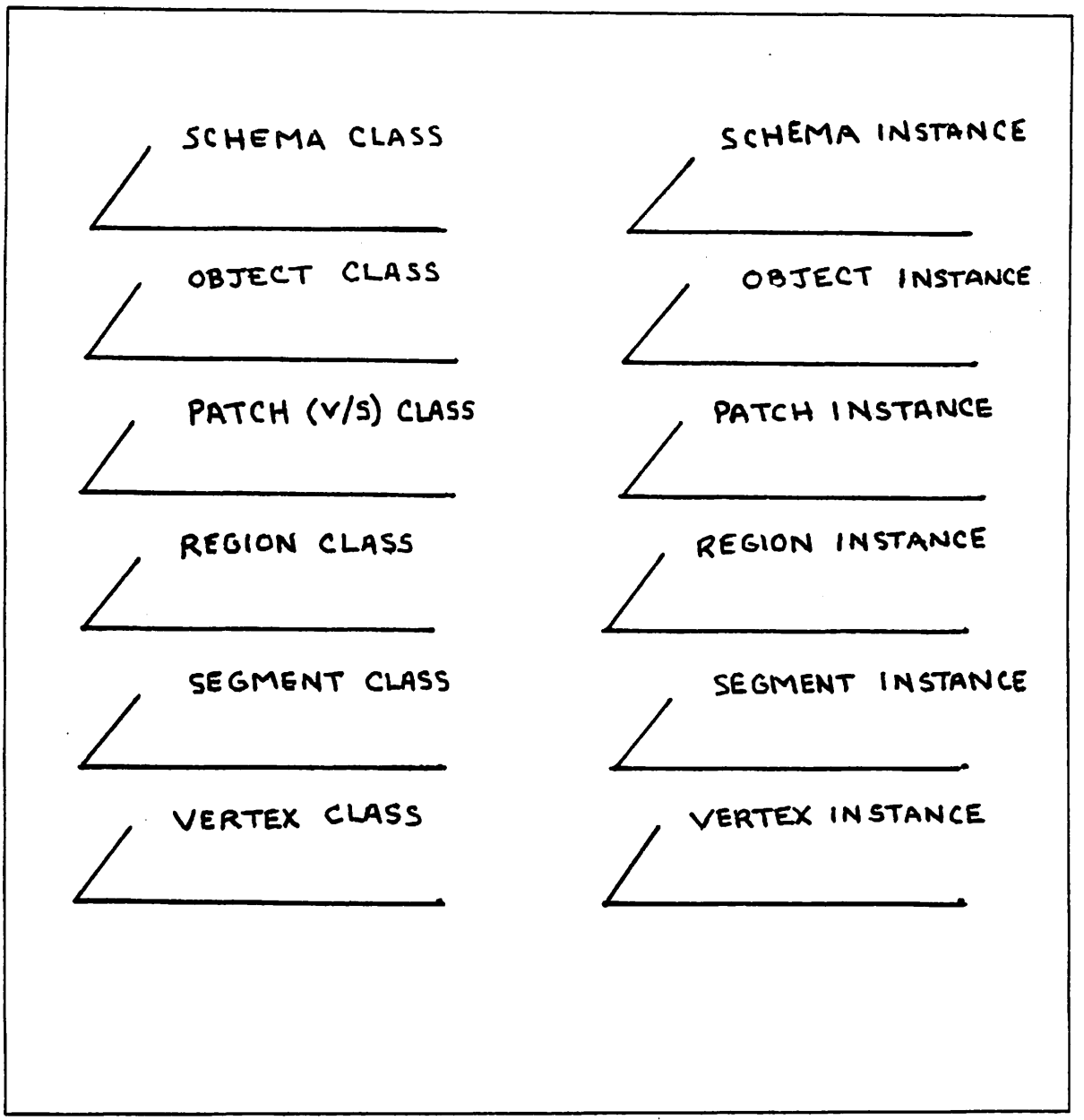

Figure 37: LTM Class and Instance Planes

with it -- a corner definition matrix, boundary curves, across boundary derivative curves, and blending functions. Figure 38 shows the structure of LTM and the components of the running show representation.

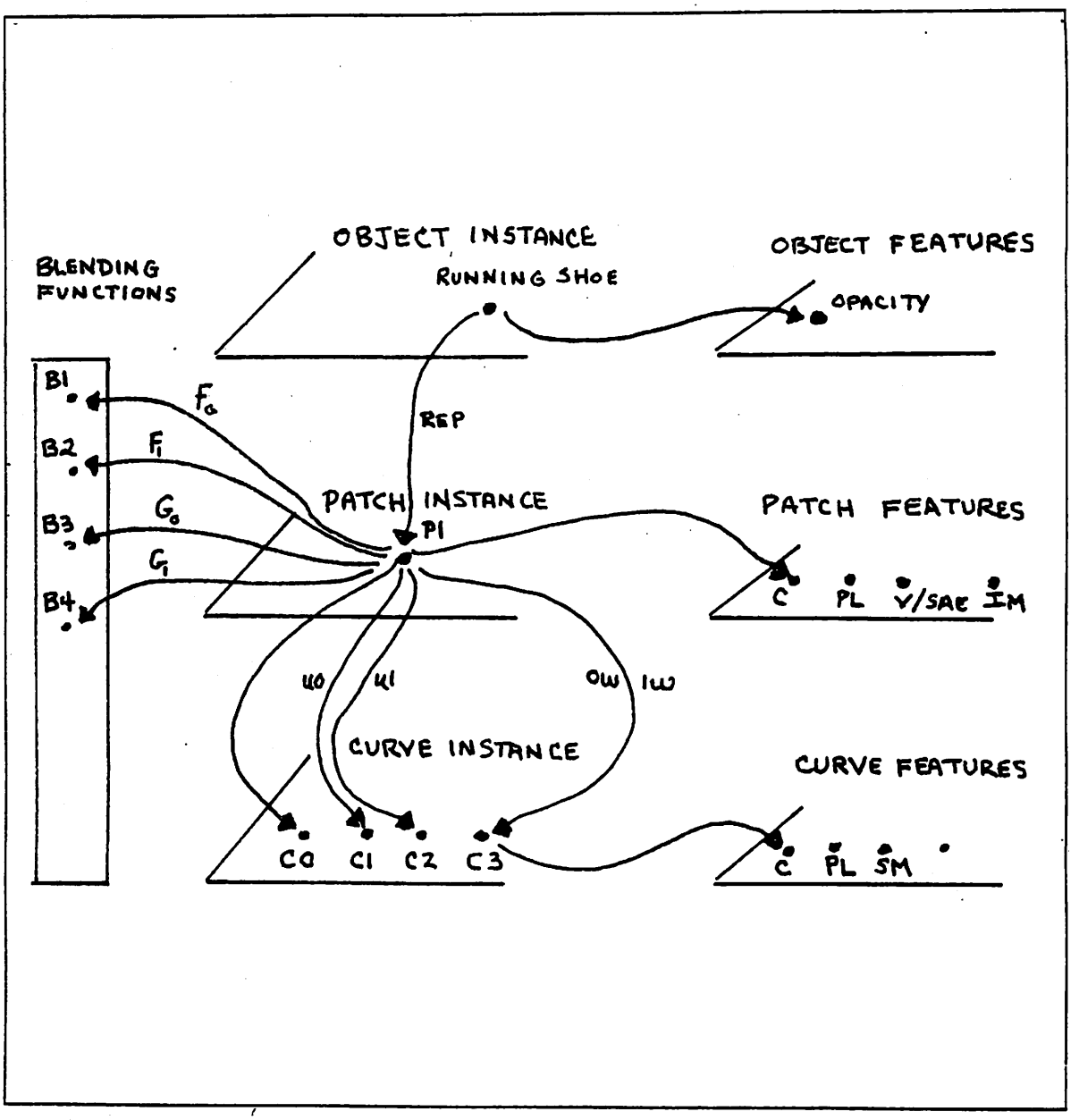

Figure 38: Running Shoe Sub-network in LTM

II.7.1 Corner Definition Matrix

The corner definition matrix for the running shoe patch instance is the value stored at the node P1 and it contains:

00 = 6 0 3
01 = 6 0 0
10 = 6 0 3
11 = 6 0 0

All other components of the corner definition matrix are set to zero. The across boundary derivative curves are set to the zero space curve and the blending functions are set to the standard blending functions (see Appendix II).

II.7.2 Blending Functions

Each of the blending functions is represented by the following 2D cubic B-spline vertex polygons:

F0		F1		G0		G1	
x	y	x	y	x	y	x	y
-.333	1	-.333	0	-.333	-.333	-.333	0
0	1	0	0	0	0	0	0
.333	1	.333	0	.333	.333	.333	0
.667	0	.667	1	1	0	.667	-.333
1	0	1	1	1	0	1	0
1.333	0	1.333	1	1.333	0	1.333	.333

These vertex polygons are stored as the values of the nodes B1,B2,B3,B4 in the GRASPER space for blending functions(see Figure 38). Note that the purpose of this representation is to be able to perform shape analysis; in the actual implementation of the display component of the VISIONS system certain optimizations have been performed. For example, blending function values are precomputed and saved for repetitive computation of Coons patches.

II.7.3 Boundary Curves

The boundary curves of the running shoe instance have the following vertex polygons stored as values at the corresponding nodes in the curve instance space. From Figure 38 it is clear that curve instance C1 participates in patch P1 as boundary curve u0 and curve instance C2 participates as boundary curve u1. Curve instance C3 participates as both boundary curves 0w and 1w. The vertex polygons for those curve instances are:

C1				C2				C3			
x	y	z	mult	x	y	z	mult	x	y	z	mult
				-6.25	0	0	1				
-6.75	0	3	1	-6.125	.5	0	1	6	0	3	3
				-5.125	-2.5	0	1				
-6.6875	-.25	3	1	-3.125	-2	0	1	6	0	0	3
				-3.125	-1.95	0	1				
-2.1875	-1.25	3	1	4.625	-2.225	0	1				
				5.875	-2.475	0	1				
-3	-1.25	3	1	7.75	-1.975	0	1				
				11	0	0	1				
2.75	0	3	1	7.625	1.975	0	1				
				6.5625	2.65	0	1				
-3	1.25	3	1	1.6875	1.3	0	1				
-2.1875	1.25	3	1	-1.1875	1.375	0	1				
				-2.875	2.125	0	1				
-6.6875	.25	3	1	-5.1875	2.5	0	1				
				-6.125	-.5	0	1				
-6.75	0	3	1	-6.25	0	0	1				

Since C1 and C2 are closed curves, the first and last vertices in each list coincide and they are treated as one vertex. They are duplicated here only to indicate closure.

II.7.4 Across Boundary Derivative Curves

The across boundary derivative curves are all set to C0, the zero space curve:

C0

x	y	z	mult
0	0	0	4

II.7.5 Curve Features

The 3D curve features are those that appear in Table 5 and they are repeated here for convenience:

Table 7 The running shoe

	u0	u1	0w	1w
OC	1	1	0	0
PL	0	0	1	1
SM	0	0	0	0
NV	8	16	2	2
MV	0	0	2	2
NT	6	16	0	0
MV/NV	0	0	1	1
NT/NV	.75	1	0	0
PGL/BL	0	0	1	1
C	16.6	7.6	0	0
MAXAT(DEG)	152	177	0	0
MAXL/MINL	22.83	9.82	1	1
AV_{3D}	(16.7,0,0)	(11.7,0,0)	(0,0,0)	(0,0,0)

II.7.6 Patch Features

The Patch shape features are as described below:

C	0
IM	1
PL	1
V/SAE	.8296
PVA-00-01	26.6°
PVA-00-10	0°
PVA-01-11	0°
PVA-10-11	26.6°
PVMR-00-01	1.118
PVMR-00-10	1
PVMR-01-11	1
PVMR-10-11	1.118

The remaining patch features are all set to zero because the running shoe is formed from a type 0 patch -- i.e. corner tangent vectors and corner twist vectors are zero. Compactness is set to zero for all non-planar patches. For closed cylindrical patches the volume is approximated by height (length of 0w(1w)) times the average area of the enclosing rectangles for the two planar boundary curves (u0,u1).

II.7.7 Standard Views

The problem of how to determine standard views is still not resolved. We have arbitrarily chosen certain standard views and computed the 2D shape features for the boundary curves from those views. For the running shoe patch instance, we computed two standard views. The first view is where the line of sight is perpendicular to the plane of the curve u0 and to the plane of the curve u1. The viewer is at a point directly above the line going down the rear of the shoe, so that the line projects as a point from this view. The distance is close enough so that none of the legs of the vertex polygons for u0 and u1 are appreciably foreshortened. The 2D projections of the curves u0,u1,0w,and 1w are shown below (see Figure 39):

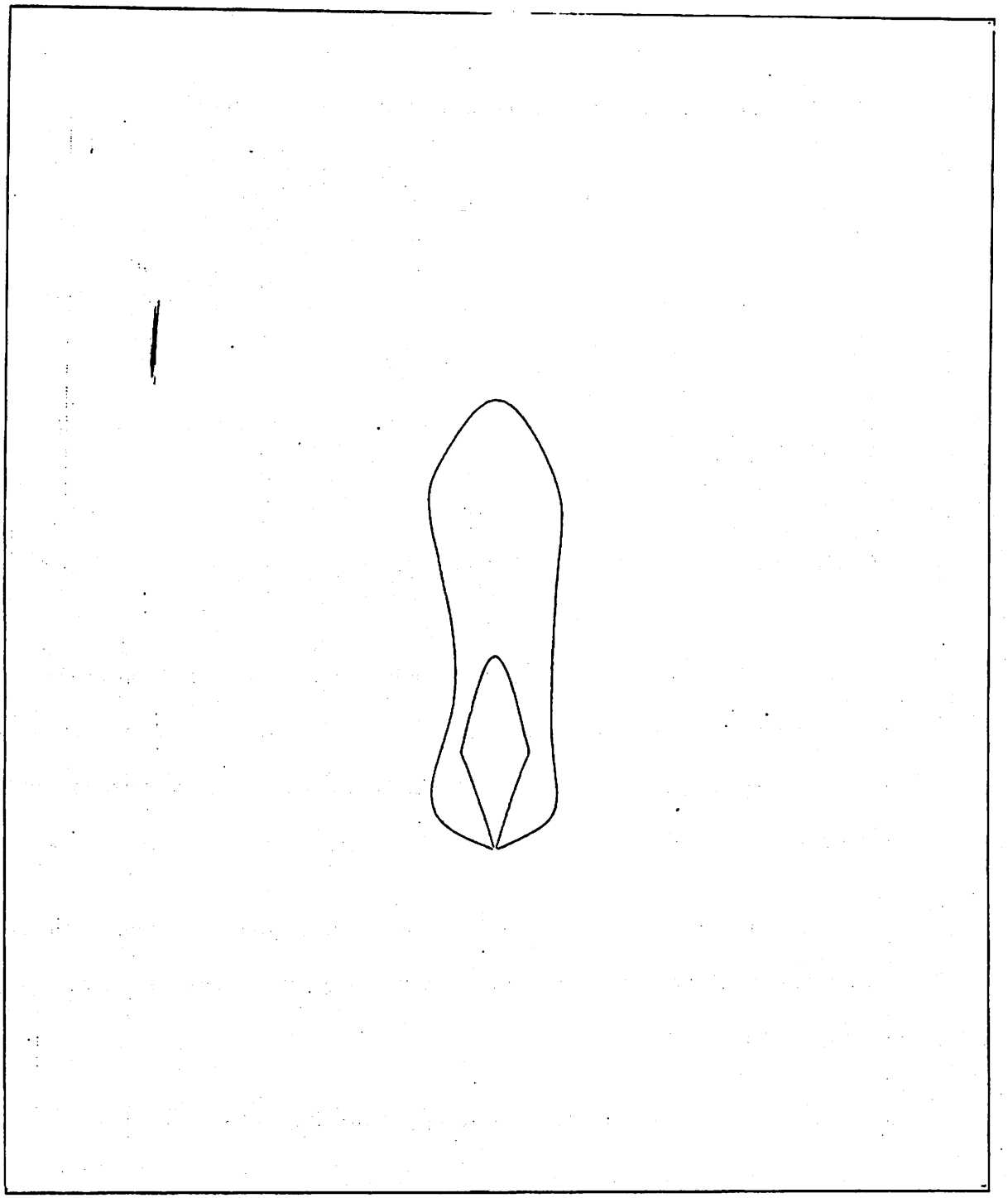

Figure 39: u0 and u1 Boundary Curves of Running Shoe -- standard view

The 2D shape features for these curves have the following values:

	u0	u1	0w(1w)
OC	1	1	0
PL	0	0	0
SM	0	0	0
NV	8	16	1
MV	0	0	1
NT	6	16	0
MV/NV	0	0	1
NT/NV	.75	1	0
PGL/BL	0	0	0
C	16.6	7.6	0
MAXAT(DEG)	152	177	0
MAXL/MINL	22.83	9.82	0
AV_{2D}	16.7	11.7	0

From this view basically all of the 3D shape properties of u0 and u1 have been preserved; however, 0w(1w) projects as a point and several of its properties have changed -- most notably NV, MV, PGL/BL, and MAXL/MINL. These changes are characteristic of degenerate views of a straight line.

The second standard view is a view from the side and above. Projections of the boundary curves appear in Figure 40. This view was chosen arbitrarily; however, it examplifies the point we are trying to make.

The 2D projections of the boundary curves onto a virtual coordinate space with extents −10 to +10 in both the x and y directions are shown below:

u0

x	y
−11.8037732	2.143349892
−11.39259372	2.244955816
−6.70032968	1.532783896
−7.962710634	1.532783896
0.8312516335	2.143349892
−5.958391958	2.600228422
−5.013768843	2.600228422
−12.06815541	2.035718909
−11.8037732	2.143349892

u l

x	y
−9.286176114	−1.093870638
−8.728500851	−0.7792573918
−10.57819894	−3.26324759
−7.204876514	−2.726240332
−3.325360345	−2.676029141
3.437031478	−2.959778479
5.454252438	−3.234790736
7.696641973	−2.701060363
9.840574687	−1.093870638
5.12620608	0.0008965216
3.9222187	0.2991597907
−0.4312632757	−0.3317406305
−3.244505817	−0.2928998863
−4.604438999	0.06995238174
−6.538448561	0.2355736128
−9.608923213	−1.440218146
−9.286176114	−1.093870638

0w(1w)

x	y
−10.80627124	2.143349892
−10.80627124	2.143349892
−10.80627124	2.143349892
−9.008976827	−1.093870638
−9.008976827	−1.093870638
−9.008976827	−1.093870638

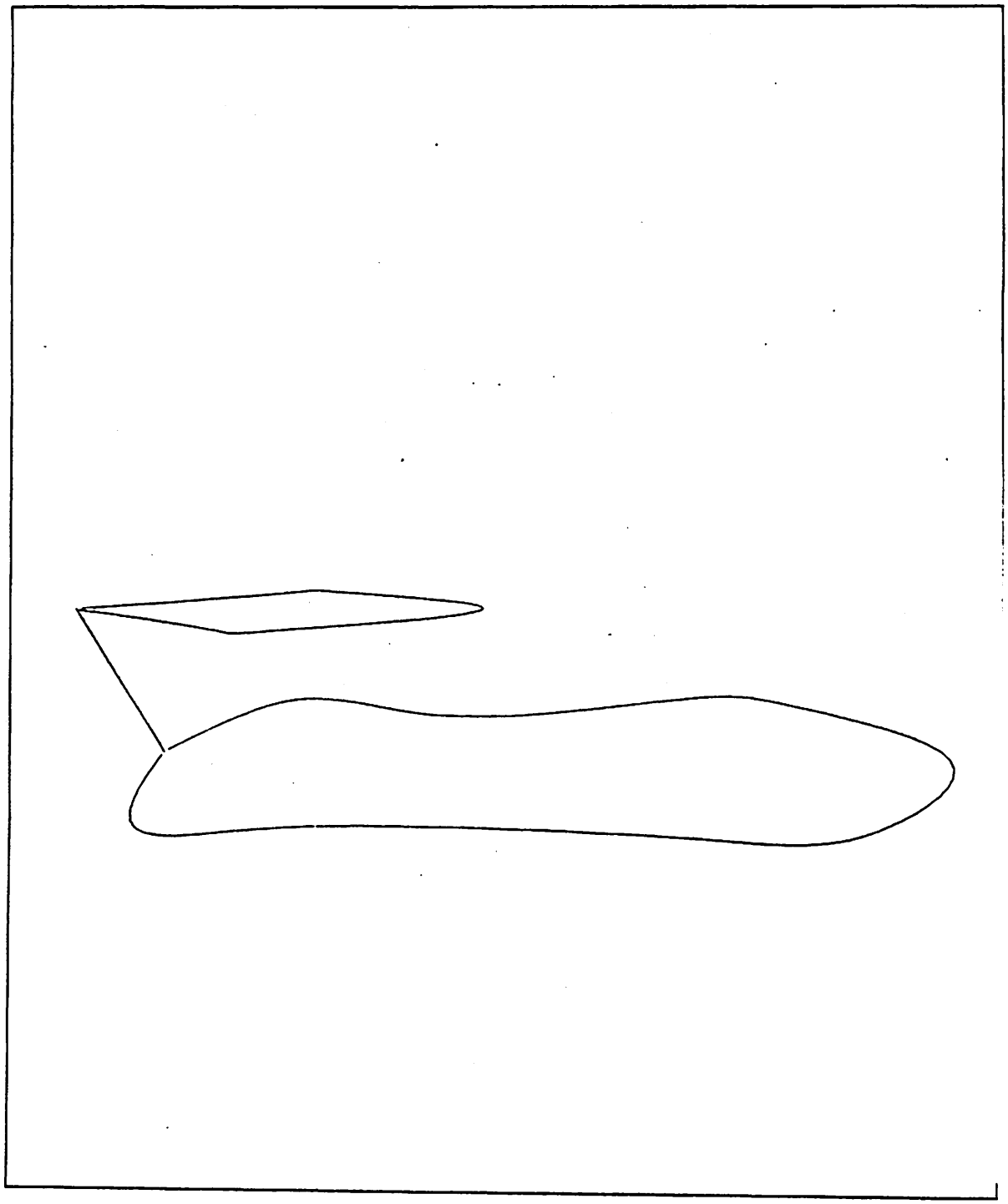

Figure 40: Alternative Standard View of Running Shoe Boundary Curves

The 2d shape feature values are shown below:

	u0	u1	0w(1w)
OC	1	1	0
PL	0	0	0
SM	0	0	0
NV	8	16	2
MV	0	0	2
NT	6	16	0
MV/NV	0	0	1
NT/NV	.75	1	0
PGL/BL	0	0	1
C	26.11	10.36	0
MAXAT(DEG)	179	172	0
MAXL/MINL	14.30	30.88	0
AV_{2D}	17.0	20.9	0

It is clear that the features which are most affected are C, MAXAT, MAXL/MINL, AV_{2D}. When one compares the first standard view with the second, the following changes are noted. The maximum angular turn for the defining polygon of u0 increased to 179 degrees from 152 degrees and it decreased to 172 degrees from 177 degrees for u1. The maximum leg length divided by the minimum leg length decreased from 22.83 to 14.30 for u0 and increased from 9.82 to 30.88 for u1. The angular variation remained about the same for u0 and almost doubled for u1 (from 11.7 to 20.9). Compactness rose in both cases, from 22.83 to 26.11 for u0 and from 9.82 to 10.36 for u1.

0w projects as a straight line segment again as reflected by NV, PGL/BL, and MAXL/MINL.

Of course, a systematic study of the variation of shape features with the angle of view must be carried out to fully exploit these feature values in a more sophisticated accessing mechanism.

It should be noted that a portion of u1 may be hidden from view if the patch P1 has the property of opaqueness. In this case additional information would be added to indicate that

an approximate portion of the boundary curve u1 between vertices 8 and 16 is not visible from this standard view. For this case, the standard view would contain a conditional assertion which states that if P1 is opaque, then u1 is hidden from vertex 8 to vertex 16 approximately.

The purpose of this section was to give the reader an understanding of how the components of this representation fit together and to give hints as to how they might be used in the indexing and matching phases of the process. We did not spend a great deal of effort on the representation of non–shape properties of objects (eg. opaqueness); however, as is clear from the example, some of these properties have significant impact on the determination of 2D shape features. In the next two chapters we shall discuss the problems of indexing and 2D shape feature computation, and in the concluding chapter we will indicate how this representation might impact the matching process. However, we must first briefly contrast our representation with the generalized cylinder representation.

II.8 Comparison with Generalized Cylinder Representation

Scope

Since a right circular cylinder may be described by a single type 0 Coons surface patch in our representation, it is clear that the class of objects representable in our representation properly contains the class of objects representable in the simple generalized cylinder representation -- i.e. one with only right circular cylinders. Secondly, our representation encompasses an even broader class of generalized cylinder representations; namely, those in which the axes are straight lines and the cross-sections have arbitrary shape (smooth or polygonal closed curves). It should be noted that no implementation of this broader class of generalized cylinders has been achieved to the best of our knowledge. Finally, cylindrical objects which are a smooth blend of one planar cross-section into another are easily represented in our representation. Examples of such objects are the running shoe and the screwdriver blade. For

a generalized cylinder representation to capture such shapes, it would have to allow the shape of the cross-section to vary as a function of the length of the axis. Again we are not aware of any generalized cylinder implementation which accomplishes this.

Accessibility

Our representation admits a variety of shape features which range from gestalt features such as compactness to very local features such as tangent vector magnitude ratios. Some of the features have 2D analogues which may be correlated under change in viewpoint. For example, one might examine how the 2D compactness measure (for the projection of a given surface patch with a single 3D compactness value) varies in value under changes of viewpoint of the 3D surface patch. These correlations together with other shape information may provide useful indexing paths from the 2D image descriptions to the 3D descriptions in the data base. For the generalized cylinder representation one must assume that elongated regions in the image represent projections of cylinders. Given this assumption an axis is hypothesized through the center of the elongated region in each of the primary orientations. The ratio of the width to length of the region is used to access cylinders with similar ratios for the radius to axis length. Because of the paucity of shape features under the generalized cylinder representation, fewer features of the image regions may be utilized in indexing and thus fewer indexing paths exist. More features provide more information about the correct alternative; however, they also tend to provide more alternatives. This may have both good and bad consequences for matching.

Stability and Sensitivity

Not only does our representation admit a greater number of features, but in addition the particular features which we have selected give our representation an advantage over others in

terms of possible stabilities and sensitivities. We have an hierarchical structuring of patch shapes based upon a shape classification scheme imposed by the identification mapping. For example, all cylindrical patches have the same identification mapping (or a symmetric reflection). Thus, at the top level of the scheme a right circular cylinder and a wedge have similar shape (a stability). If further discrimination is required, the position vector ratios are examined, then the tangent vector ratios, and so on until a difference is detected. If no differences are detected in the patch description per se, the boundary curve and tangent vector curve shape features are examined.

In addition to the stability/sensitivity of the shape features, the hierarchical structuring of complex patches as quad trees admits variation in the coarseness of the geometric representation. For example, the complex surface patch designated by A in Figure 16 has a coarse patch description which is different from the finer description defined by the four patches E,F,G,H. Patches E,F,G, and H need agree with A only where they share common boundary. In the generalized cylinder representation the paucity of features dramatically limits the stability and sensitivity of the representation. There are merely two levels of features allowed -- adjunct relations and radius/axis length ratios.

Cost

The storage cost for our representation is quite reasonable, given its tremendous scope. The cost of storing a surface patch appears to be the sum of the costs of storing the patch matrix, the 4 boundary curves, the 4 tangent curves, and the 4 blending functions; however, this sum is misleading. Since many patches use the same blending functions, the cost of storage for blending functions is an overhead which is distributed over all the patches in the data base. Also, a single boundary curve or tangent curve may participate in several different patches; thus, the cost of storage for that curve is shared among the patches in which it participates.

The incremental cost of storing a new patch which shares existing boundary curves, tangent curves, and blending functions is basically the cost of storage for the patch matrix and the pointer information which allows sharing. This cost varies depending upon the type of the patch. For a general, type 1 Coons patch a 4×4 matrix of 3-vectors is required. In addition, 4 pointers to boundary curves, 4 pointers to tangent curves, and 4 pointers to blending functions are required. For a type 0 patch, only a 2×2 matrix of corner positions, 4 pointers to boundary curves, and 4 pointers to blending functions are needed.

In addition to the pointer information, scale and orientation information is required in order to properly attach the boundary curves to the corners of the patch. Three additional values are used to specify the scale and orientation parameters for each boundary curve and its associated tangent curve. Thus, as few as 32 values may be required to describe a type 0 patch and as many as 72 values for a type 1 patch. These are worst case estimates, since additional savings result from proper organization of the data base. For example, the four blending functions for a patch must satisfy certain relationships with each other and hence they always occur in standard groups. We exploit this fact by only allowing a patch representation to point to a group, thus saving 3 pointers per patch. A further example involves classification of patches on the basis of identification mappings. All patch matrices which are basically of the cylinder class (i.e. 00 = 10 and 01 = 11) are grouped together, requiring only 2 3-vectors to describe the patch matrix. Other possibilities for the elimination of patch storage redundancies are still being explored.

The storage of a 3D cubic B-spline curve with n points in its defining polygon generally requires the storage of 3n values; however, for most curves n is between 6 and 8. A very complex curve might require 15 vertices. Note that 10 vertices were used to approximate the circles in the earlier examples in order to increase the accuracy of approximation, although fewer vertices might have been acceptable. Additional storage is needed for B-splines with multiple vertices to indicate the location and the degree of the multiplicity. Many possibilities

exist for the compression of the storage required by the representation; however, that topic is not part of this research.

It is clear that only 7 values (the two endpoints of the axis and the radius of the cross-section) are required to represent a simple cylinder in the generalized cylinder representation. This is a major advantage of the generalized cylinder; however, the savings come at a very high price -- diminished scope.

Matchability

The issue of matchability is still open with respect to our representation. In chapter IV we explain why we feel our representation will contribute to improved shape matching.

CHAPTER III

ACCESSIBILITY AND INDEXING

III.1 Introduction to Indexing

In this chapter we are concerned with the problem of accessibility of three-dimensional object representations stored in a data base. Accessibility is the problem of gaining access to a 3D model of an object via features which are obtained from sensed data. The sensed data may be either 3D or 2D. For convenience of explication we separate the problem into three major sub-problems.

(1) Obtaining 2D shape feature values from a 2D image or obtaining 3D feature values directly from the scene.

(2) Hypothesizing 3D feature values from 2D feature values.

(3) Indexing into a data base of object descriptions on the basis of the 3D feature values.

The three major sub-problems are listed as they would be encountered in a bottom-up (data-to-model) analysis of an image. Since the focus of this research is 3D representation, the sub-problems will be treated in the reverse of the order in which they are listed. Since the availability of 3D data obviates the need for step (2) above, our approach is designed to allow evaluation of indexing based upon data derived from direct 3D measurements. In this chapter we shall explain the structure of long term memory (LTM) in greater detail than in Chapter I and we shall describe the access paths available for 3D shape feature indexing. This description is followed by a sample of indexing experiments on a small data base. Finally, it is shown

how 2D shape features for certain standard views of an object can be correlated with the 3D shape features for the object. The feature correlations for two sample objects are presented. The problem of extracting 2D shape features is treated in the following chapter.

III.2 Long Term Memory and Data Base Structure

As described in Chapter I knowledge in the VISIONS system is structured in an hierarchical fashion. There are several levels of shape representation within this hierarchy and they are depicted in Figure 41. At the highest level of LTM are the schemas. They represent common scene prototypes, such as a house scene, a farm scene, or an urban ghetto scene. A scene prototype is a collection of object prototypes. Information such as the approximate spatial relationships between object prototypes is encoded in the schema for the scene prototype. These spatial relationships may be simple assertions of the nature -- object prototype A is *above* object prototype B, or they could involve relative 3D locations in a local coordinate system. Each object prototype may have a decomposition into other object prototypes via the "part-of" relationship. For example, a table prototype may be decomposed in the obvious way into a table top prototype and a table leg prototype. Stored with this particular table prototype is the information that any instance of a table must have one instance of the table top and four instances of the table leg. In addition, these parts must satisfy certain spatial relationships specified in the table prototype.

Each primitive object prototype (i.e. one which has not been decomposed into part prototypes) is geometrically represented by a patch (volume/surface) prototype. The term "volume/surface" is used, because it is clear from the examples in the previous chapter that Coons patches may be used to represent surfaces which enclose volumes. A volume/surface

prototype is actually a parameterized routine for generating Coons surface patches of a particular form. For example, the prototype, rectangular-solid, is a program which is parameterized by length, width, and height. When this program is invoked with specific parametric values, it generates a Coons patch representation of a specific rectangular-solid. Such specific instances of prototypes are also stored in LTM and we shall discuss the "instance" planes of LTM shortly.

As we descend the LTM "class" hierarchy, we find that volume/surface prototypes are represented by boundary curve prototypes, across boundary derivative curve prototypes, and corner definition matrix prototypes -- essentially the components of the Coons patch representation. Curve prototypes are merely parameterized routines for generating curve instances and each curve instance is represented by the vertex polygon for a cubic B-spline. For example, the curve prototype, circle, is a program parameterized by the radius of the desired circle and the number of vertices in the vertex polygon for the cubic B-spline which will approximate that instance of circle.

All of the curve prototypes generate 3D space curves. However, in some cases these space curves are actually planar. In these cases it is a rather easy matter to correlate these planar 3D space curves with their corresponding 2D counterparts. Thus, below the 3D space curve level in LTM are two additional abstract levels -- the region class level and the segment class level. These levels contain prototypes or programs for generating 2D region instances and 2D curve instances. With respect to shape, 2D region instances are described by their boundary curves, which are represented by vertex polygons for 2D cubic B-splines. Thus, region instances (for shape purposes) are merely pointers to boundary curve instances.

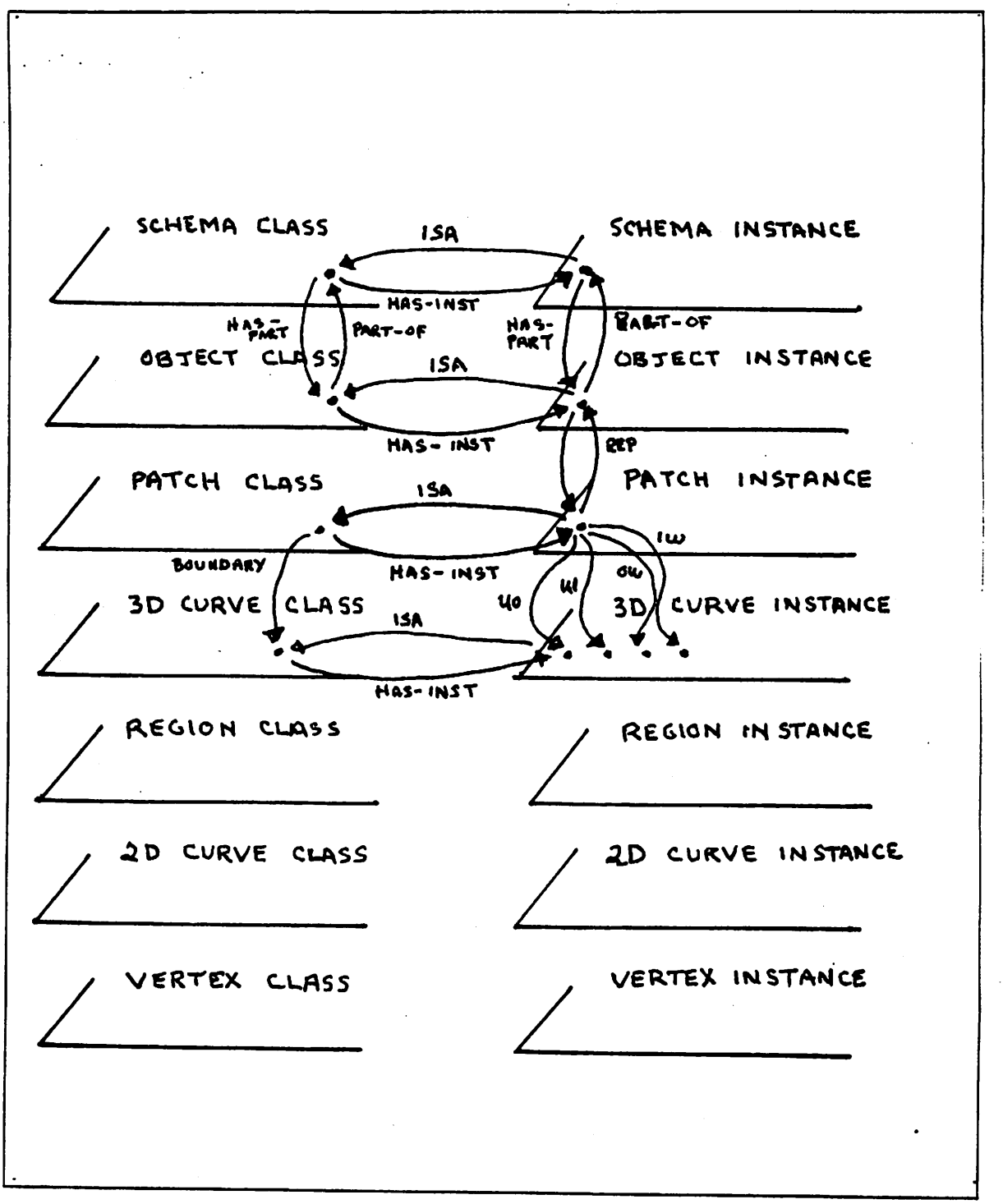

Figure 41: Network Structure of LTM

Figure 41 is a graphic depiction of the levels of LTM and the relationships between those levels. For completeness of the model there are "vertex class" and "vertex instance" planes. Use of 2D junctions and 3D vertices has not been incorporated into the research at this time; however, if the reader desires more information on vertex and junction analysis [Huff71,Clow71,Waltz72,Turner76,Parma79] are excellent references.

The vehicle for implementation of the LTM model is the GRASPER programming language extension to LISP [Low79]. GRASPER, a language for constructing arbitrary network structures, provides a grouping mechanism which supports the hierarchical LTM model. GRASPER utilizes the notion of "spaces". Spaces are collections of nodes and/or arcs. Thus, each plane of LTM maps directly onto a GRASPER space. To each node of each plane a value may be attached. In the case of "class" planes, nodes represent prototypes and the value of each such node is actually a GRASPER/LISP program for generating instances.

Figure 42 elaborates another portion of the LTM structure -- the shape feature levels. For each abstract level of the LTM hierarchy there exists a corresponding shape feature space.

In the model pictured in Figure 41, there are several types of edges. These edges represent relationships in the model. For example, between each class node and its corresponding instance node(s), there are two types of directed edges (has-instance, isa). An object prototype represented by a node in the object class space may have several different instances in the object instance space. However, each object instance may have only one "isa" edge connecting it to an object prototype. A listing of the edge types and their meanings appear in Figure 43. This set of edge types determines the set of indexing paths throughout this portion of LTM.

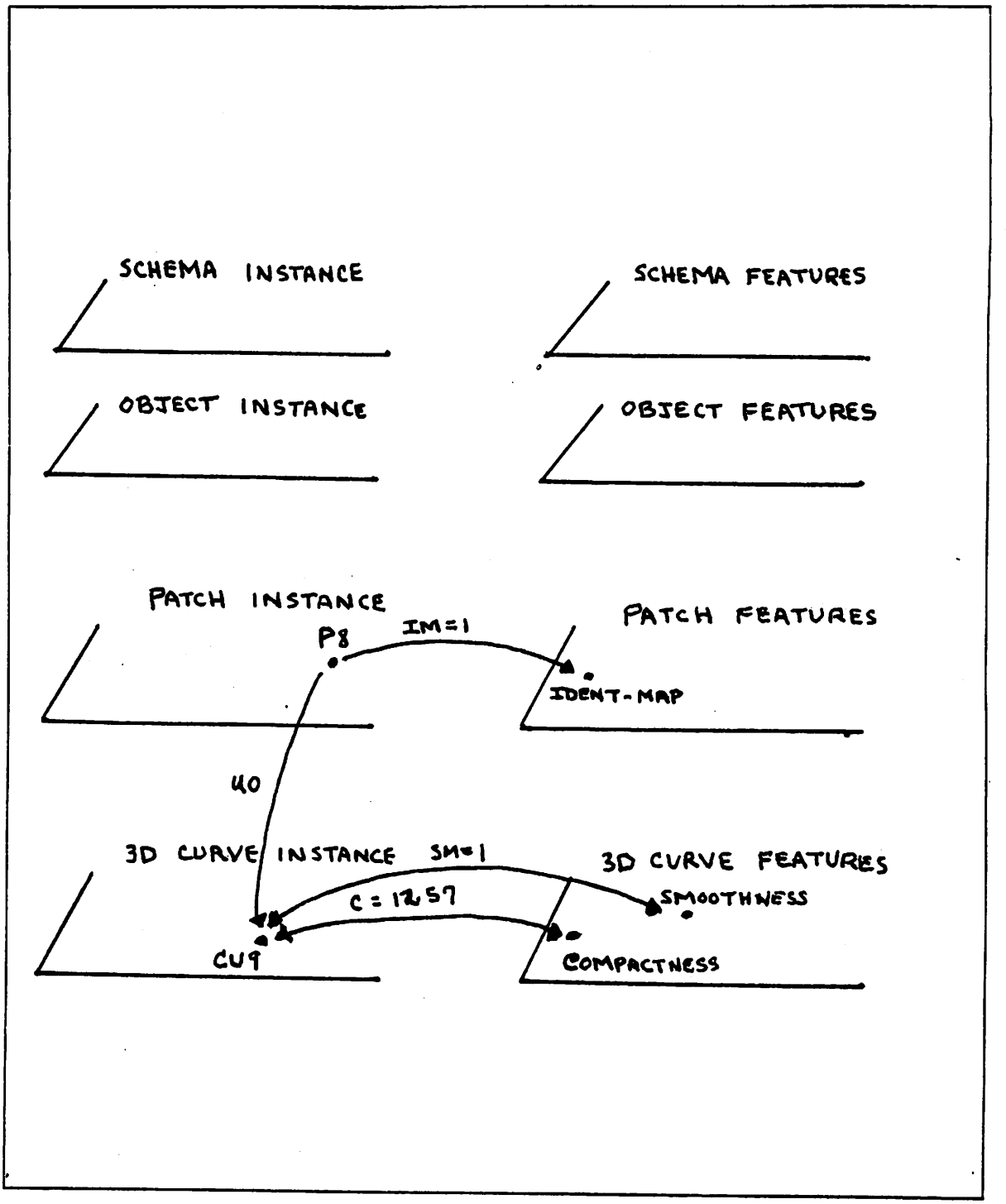

Figure 42: Shape Feature Structure in LTM

Each shape feature for each class of objects is represented by a node in the corresponding shape feature space for that class of object. Figure 42 shows the connection between a given curve instance and its associated shape features. Thus, each shape feature is represented by an edge "type" leading to a specific node in a specific feature space. Figure 42 shows only one type of feature relationship -- the one between an instance and its features. In addition, there are features which are characteristic of the class. For example, there can be features which are characteristic of the curve class circle, such as circumference = $2\pi r$ and area = πr^2, etc. However, any instance of a circle is actually an approximation to an instance of a circle. Thus these approximations have their own values for circumference, and area, and they may differ from the class values. In the experiments that follow we concerned ourselves only with features of instances. The shape features are those described in Chapter II; their node names and edge types may be found in Appendix III. Also a description of a significant portion of the LTM graph appears in Appendix III.

EDGE TYPE	TO-SPACE	FROM-SPACE
ISA	schema-class	schema-instance
	object-class	object-instance
	patch class	patch instance
	3D curve class	3D curve instance
	region class	region instance
	2D curve class	2D curve instance
HAS-INSTANCE	schema-instance	schema-class
	object-instance	object class
	V/S instance	V/S class
	3D curve instance	3D curve class
	region instance	region class
	2D curve instance	2D curve class
PART-OF	schema-class	object-class
	schema-instance	object-instance
	object-class	object-class
HAS-PART	object-class	schema-class
	object-instance	schema-instance
	object-class	object-class
REP	object-class	patch class
	object-instance	patch instance
BOUNDARY	3D curve class	patch class
	2D curve instance	region class
	2D curve instance	region instance
u0,u1,0w,1w	3D curve instance	patch instance

Figure 43: LTM Class/Instance Edge Types

III.3 3D Indexing Examples

Each indexing experiment is described in a concise form as a relational query on the network data base. The form shows the query as a feature name, a relationship, and a value. The indexing mechanism retrieves all of the desired entities satisfying that relationship and all of the entities connected to the entities which satisfy the relationship. Note that when feature values are real values, one may wish to obtain all entities which satisfy the equality relationship to a given degree of tolerance. This is achieved by specifying two separate queries and forming the intersection of the result sets. For example, to determine those closed curves

which have a compactness measure approximately equal to say 12.56, one might specify two separate queries:

(1) compactness < 13.06

(2) compactness > 12.06

The intersection of the two result sets would provide all closed curves with a compactness value within a neighborhood of radius .5 about the value 12.56.

The indexing mechanism demonstrated here is the raw relational mechanism. It merely selects all of the entities satisfying a given query. It performs each query independently. The choice of what intersections to perform and when to perform them is under the control of a more global mechanism -- the indexing strategy. The indexing strategy is part of the control mechanism for the model builder in the VISIONS system and it is not properly part of this research. For further discussion see (Hanson and Riseman 1978b, Williams et al. 1978).

Query 1

	Feature	Relation	Value
(1)	smoothness	=	0

Results

Curves (class and instances) satisfying (1)
(CIRCLE (CU9 CU14 CU17 CU18 CU21 CU22 CU24 CU25))
(STRAIGHT-LINE (CU7 CU13 CU16 CU19 CU23 CU29 CU32 CU35
CU38 CU44 CU47 CU50))
(CUBIC (CU1 CU2 CU3 CU4 CU11 CU12))

Curve-instances-to-Patch-instances
(P1 (CU1 CU2 CU3 CU4))
(P2 (CU7))
(P3 (CU9))
(P4 (CU11 CU12 CU13))
(P5 (CU14 CU16))
(P6 (CU17 CU18 CU19))
(P8 (CU21 CU22 CU23))
(P9 (CU24 CU25))

Patch-instances to object-instance and class
(CORVETTE-HOOD (O1 (P1)))
(DRINKING-CUP (O2 (P2)))
(PENCIL (O3 (P3)))
(RUNNING SHOE (O5 (P5)))
(SCREWDRIVER-BLADE (O6 (P6)))
(SCREWDRIVER-HANDLE (O6 (P6)))
(TELEPHONE-POLE (O9 (P8)))
(TELEPHONE-RECEIVER (O10 (P9)))

Comments:

This very simple query returns all of the smooth curve instances in the data base. More than half of the curves in the data base are smooth and over half of the object instances in the data base are represented by patches with smooth boundary curves.

Query 2

	Feature	Relation	Value
(1)	no-vertices	=	2

Results:

Curves
(STRAIGHT-LINE(CU7 CU13 CU16 CU19 CU23 CU29 CU32
CU35 CU38 CU41 CU44 CU47 CU50))

Curve-instances to Patch-instances
(P2 (CU7))
(P3 (CU13))
(P5 (CU16))
(P6 (CU19))
(P8 (CU23))
(P10 (CU29))
(P11 (CU32))
(P12 (CU35))
(P13 (CU38))
(P14 (CU41))
(P15 (CU44))
(P16 (CU47))
(P17 (CU50))

Patch-instance to object-instance and class

(DRINKING_CUP (O2 (P2)))
(RUNNING-SHOE (O4 (P4)))
(SCREWDRIVER-BLADE (O5 (P5)))
(SCREWDRIVER-HANDLE (O6 (P6)))
(TELEPHONE-POLE (O9 (P8)))
(WASTE-BASKET (O11 (P9)))
(TABLE-TOP (O12 (P10)))
(TABLE-LEG (O13 (P12) O14 (P13) O15 (P14) O16 (P15)))
(TETRAHEDRON (O17 (P16)))
(WEDGE (O18 (P17)))

Comments:

This query selects all curves having only two vertices. Straight lines are the only curves in this data base with just two vertices. All instances of straight line and the patches and objects

containing straight lines are retrieved. The pencil is not selected because its boundary curves consist of a circle, a point, and a piecewise linear curve with 3 vertices.

The intersection of the results of queries 1 and 2 at the object class level would yield the following 5 objects -- DRINKING-CUP, RUNNING-SHOE, SCREWDRIVER-BLADE, SCREWDRIVER-HANDLE, and TELEPHONE-POLE.

Query 3

	Feature	Relation	Value
(1)	CLOSURE	=	1
(2)	SMOOTHNESS	=	0
(3)	COMPACTNESS-C	<=	13
(4)	COMPACTNESS-C	>=	12

Results:

Curves:
(CIRCLE (CU9 CU14 CU17 CU18 CU21 CU22 CU24 CU25))

Curve-instances to Patch-intances:
(P3 (CU9))
(P5 (CU14))
(P6 (CU17 CU18))
(P8 (CU21 CU22))
(P9 (CU24 CU25))

Patch-instances to object instances and class
(PENCIL (O3 (P3)))
(SCREWDRIVER-BLADE (O5 (P5)))
(SCREWDRIVER-HANDLE (O6 (P6)))
(TELEPHONE-POLE (O9 (P8)))
(TELEPHONE-RECEIVER (O10 (P9)))

Comments:

In this query we formed the set of all curve instances which satisfy all of the feature-value relationships -i.e. the conjunction. The only curve instances which satisfy the conjunction are the instances of CIRCLE. All patches and objects containing circles are retrieved as well.

Discussion:

The queries shown above are merely examples of how the 3D curve indexing mechanism functions at the lowest level. Under the control of the indexing strategy access to 3D entites (curves, patches, and objects) may be gained by performing the proper queries on the data

base. It is the responsibility of the indexing strategy to formulate the proper queries on the basis of low-level information provided by the knowledge sources which analyze the image data and on the basis of high-level information obtained from the schema which is instantiated. How a particular schema gets instantiated is not part of this research, but how shape features are obtained is part of this research. The extraction of 2D shape features is treated in the next chapter; however, the relationship between 2D and 3D shape features is briefly explored in the next section of this chapter. Further examples of 3D indexing appear at the end of Appendix III.

Before proceeding to the next section, a few comments must be made about data base performance. GRASPER (implemented on a VAX 11/780 with 3 megabytes of real storage) is biased toward indexing on nodes. A great deal of the shape network substructure depends upon indexing upon arcs, since the values of features are stored as the values of the arcs connecting each entity instance (curve, patch, object) with each feature node. Thus, the current implementation of 3D indexing is reasonably inefficient. Average cpu time for the queries in this chapter and Appendix III ranged between 5 and 10 seconds depending upon the number of queries. There are two obvious ways in which to improve performance. First, the data base engine could be improved by using a B-tree indexing for arcs as well as nodes. Secondly, query processing could be optimized by performing intersections as early as possible.

III.4 2D Shape Features

Now that the structure of 3D shape information is clear and several access paths have been shown to exist from 3D features to curves, patches, and ultimately objects, we must describe how to get to 3D shape features from 2D shape features. It is clear that very few if

any 3D shape features are preserved under the perspective projection of a 3D entity such as a patch or space curve onto an image plane. For example, when a circle is viewed from a point orthogonal to the plane of the circle, the circle projects as a circle on the view plane. However, as the plane of the circle is rotated away from the viewer (about either the x or y axes), the circle projects as an ellipse of some sort on the view plane (see Figures 44, 45, 46).

Notice the circle and the ellipse in the view plane both share certain properties which are properties of the 3D circle. They are closed, smooth curves. However, some of the other properties of the 3D circle were changed under the perspective projection. For instance, the compactness featured defined as the square of the perimeter divided by the area is constant at 4π for a circle, but it is a function of a and b (the lengths of the semi-major and semi-minor axes) for an ellipse. The perimeter of an ellipse is defined by the approximate equation:

$$P = \pi[3(a + b) - \sqrt{(a+3b)(b+3a)}]$$

and the area of an ellipse is $A=\pi ab$. Thus, compactness for an ellipse is defined by the approximate equation:

$$\frac{P^2}{A} = \pi^2[3(a + b) - \sqrt{(a+3b)(b+3a)}]^2/\pi ab$$

Since the eccentricity for an ellipse is defined by the equation $b^2 = a^2(1-e^2)$ and $0<e<1$, the eccentricity increases as the ratio of b to a decreases. As the ratio of b to a decreases the compactness measure increases. Thus, compactness for an ellipse is an increasing function of eccentricity. Notice that for a circle the eccentricity is zero and compactness is minimal.

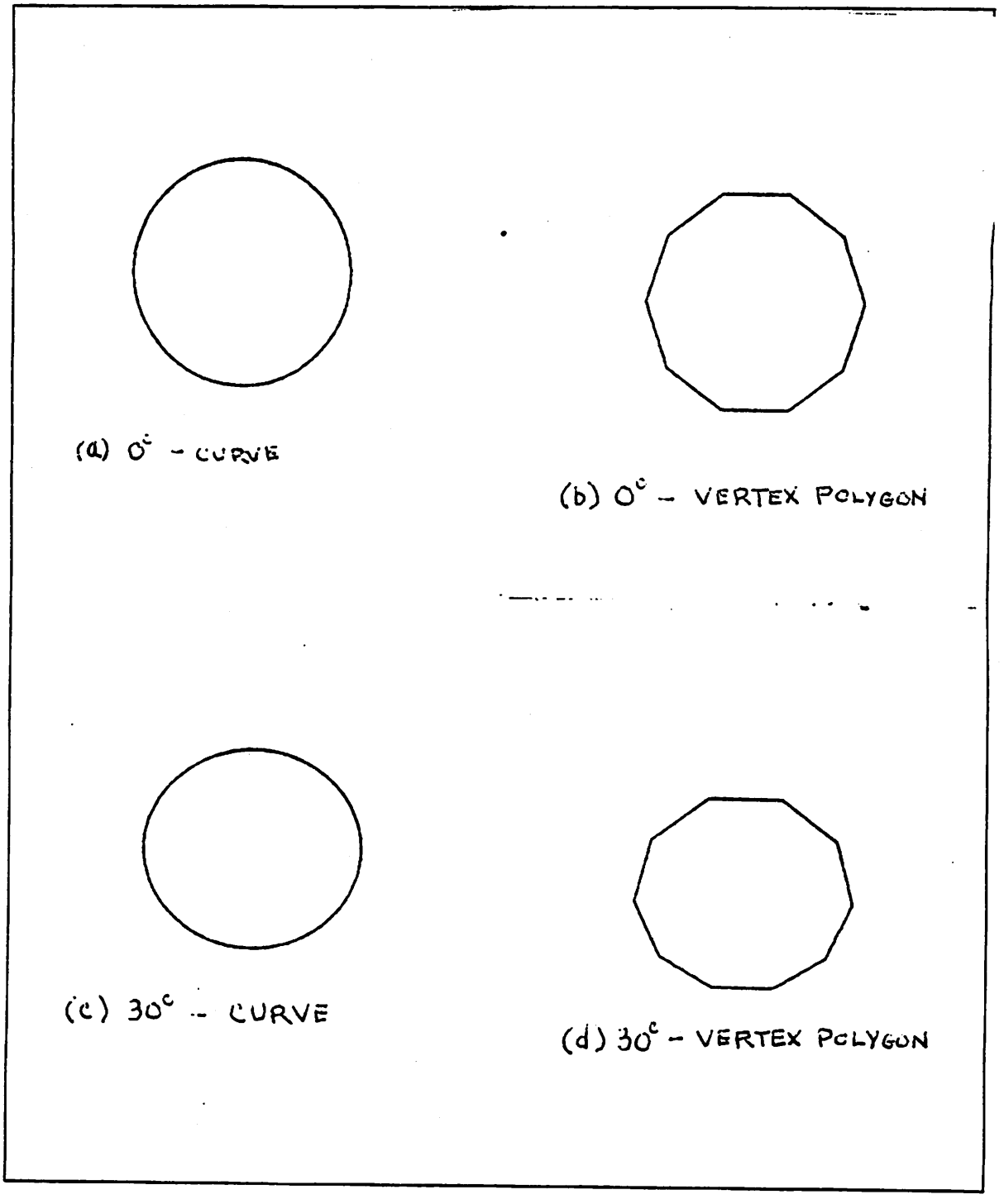

Figure 44: Projections of a circle -- 0° and 30°

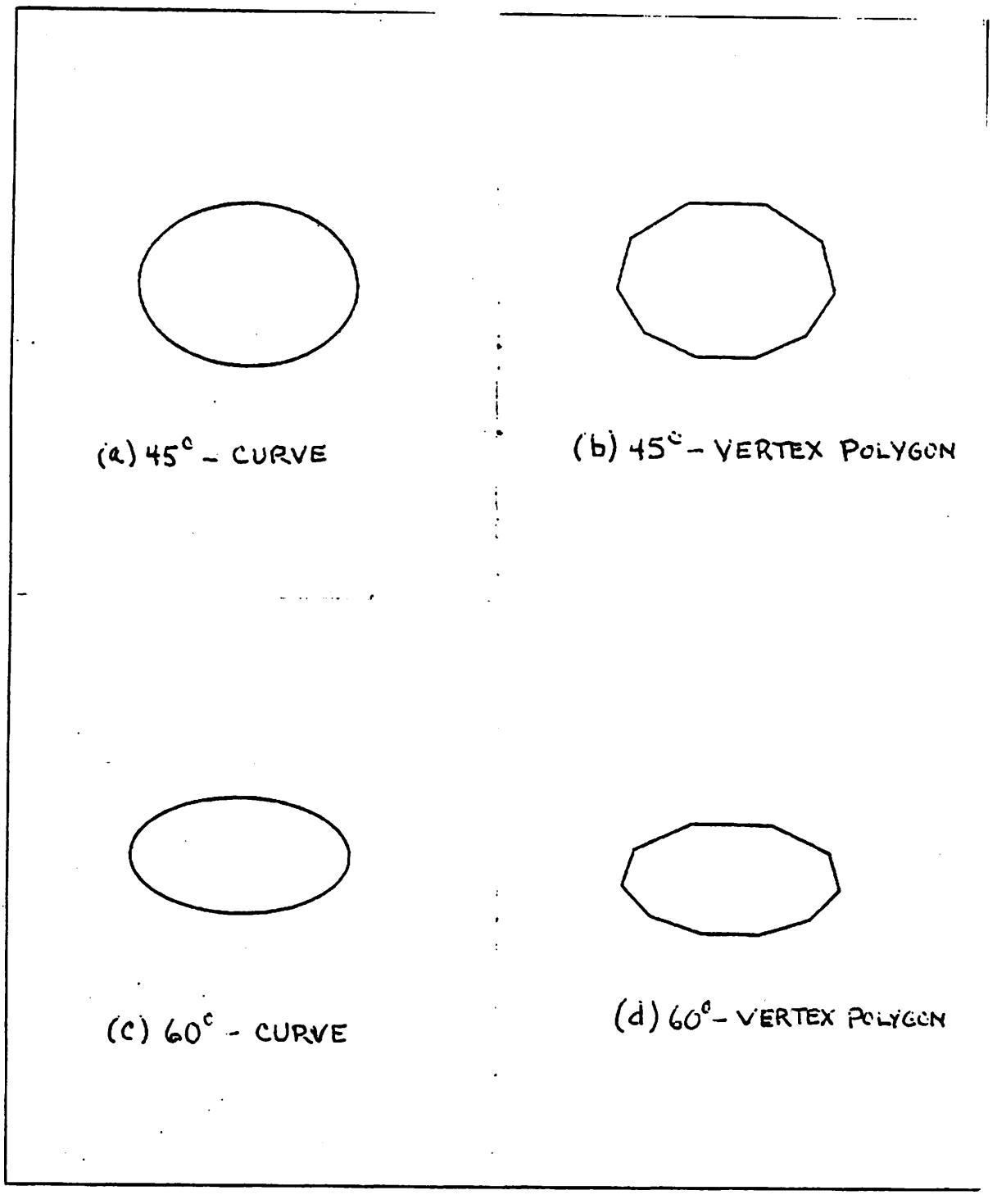

Figure 45: Projections of a circle -- 45° and 60° *

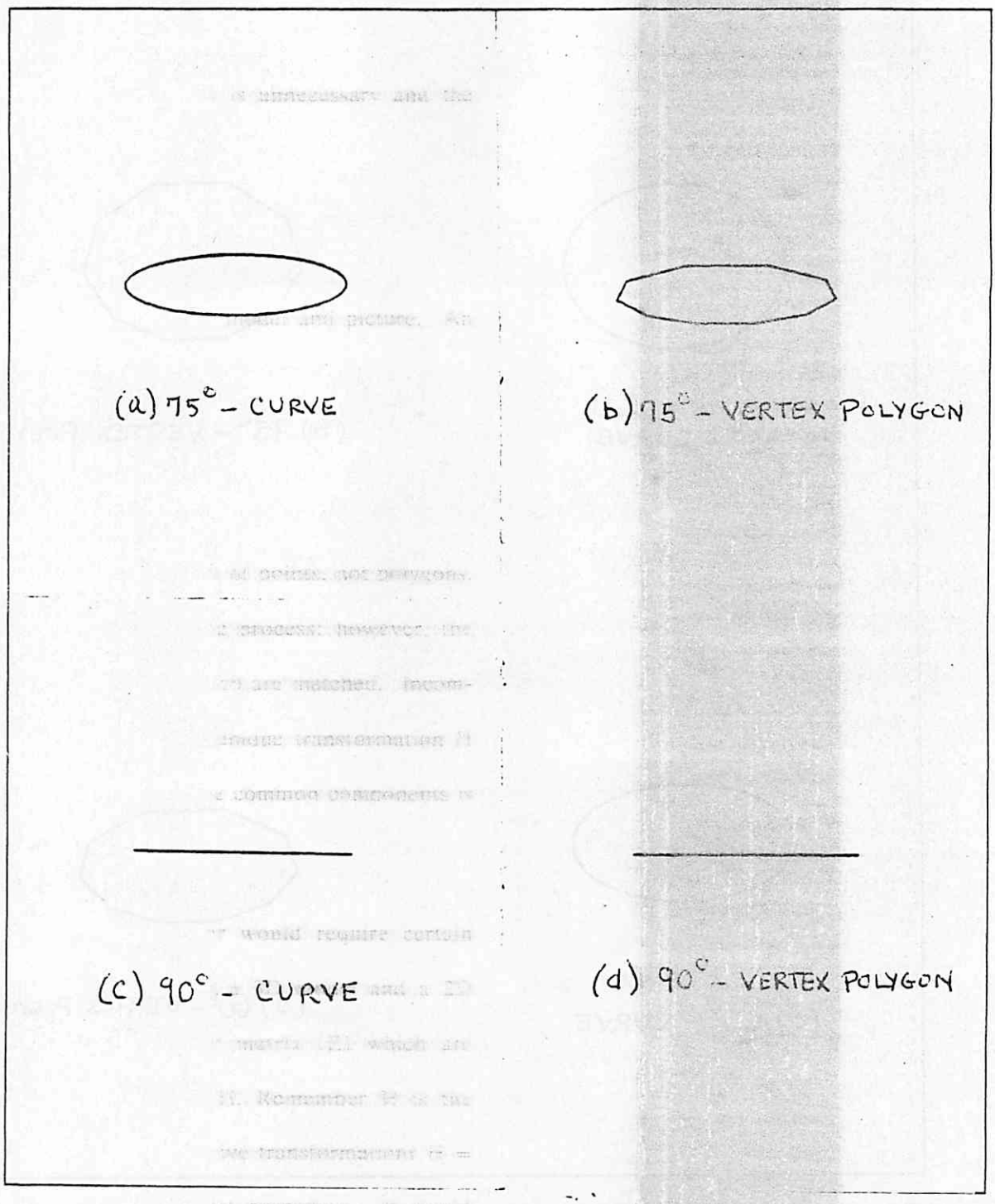

Figure 46: Projections of a circle -- 75° and 90° *

A possibly simpler example to calculate is the case of the unit square. It has area of 1 and a perimeter of 4. Thus, its compactness is 16 (P^2/A). As the square is rotated about the x-axis (i.e. away from the viewer), it will project as a narrower and narrower rectangle until it eventually projects as a straight line. When it projects as a rectangle with top and bottom edges of length 1 and left and right edges of length 1/2, the area is 1/2 and the perimeter is 3. Thus, compactness is 36(9 divided by 1/4).

The preceding discussion was concerned with the computation of the compactness feature for actual circles and ellipses. In our data base we have cubic spline approximations to these curves. Since we are concerned with the extraction of 2D splines and their shape features from images, it is important that we correlate the value of each spline approximation to a curve with the 2D projections of the spline approximation. In certain cases where the feature is constant for a whole class of curves, (such as compactness for circles) this information would be retained in the compactness feature arc for the circle *prototype*.

Obviously it is impossible for people as well as systems to correlate 3D curve features with the features of every possible 2D projection of the curve. Thus, it is necessary to select specific 2D projections which represent standard views (Minsky 1975) or important degenerate views. 2D shape features should be computed for these views and correlated with the corresponding 3D shape features. This requires a means of defining a natural coordinate system for each curve and patch and prescribing standard views with respect to that coordinate system. Table 8 shows 2D shape features for various projections of a circle (Figures 44, 45, 46) approximated by a cubic spline with 10 vertices in its defining polygon. The projections range from 0° rotation about the x-axis (plane of circle is orthogonal to line of sight) to 90° rotation about the x-axis. Similarly, Table 9 shows results for the open planar cubic spline

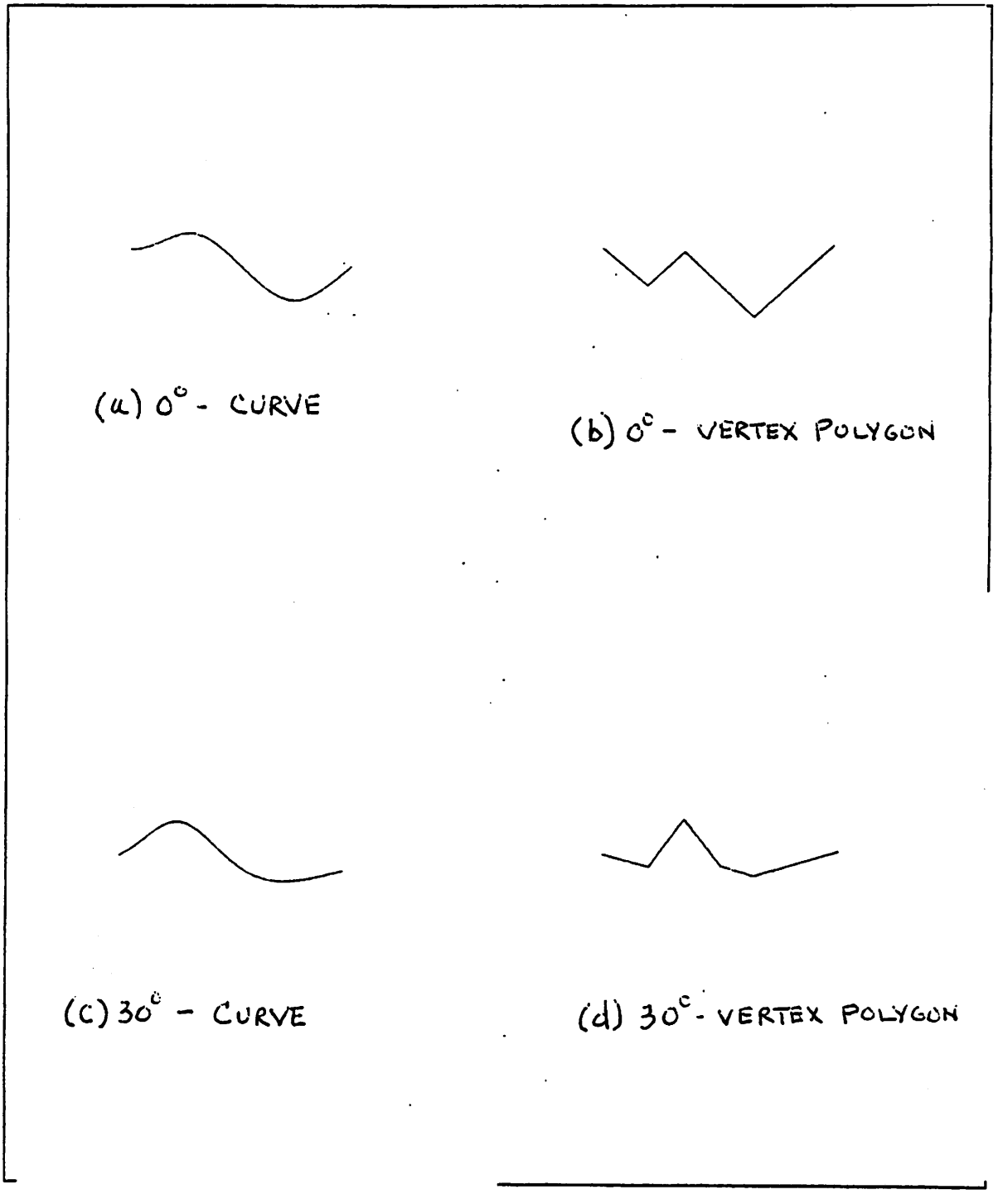

Figure 47: Projections of a cubic spline -- 0° and 30°

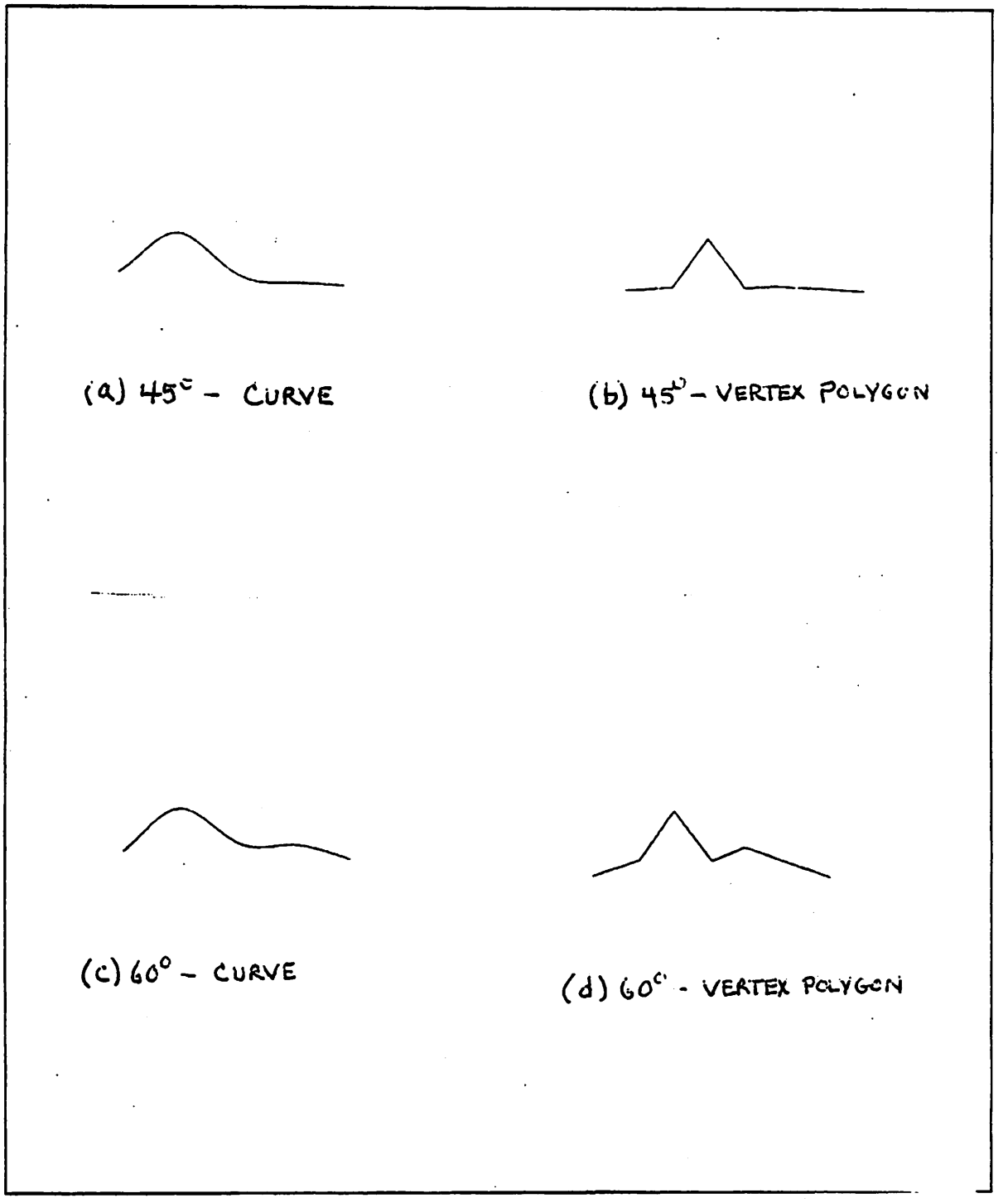

Figure 48: Projections of a cubic spline -- 45° and 60° *

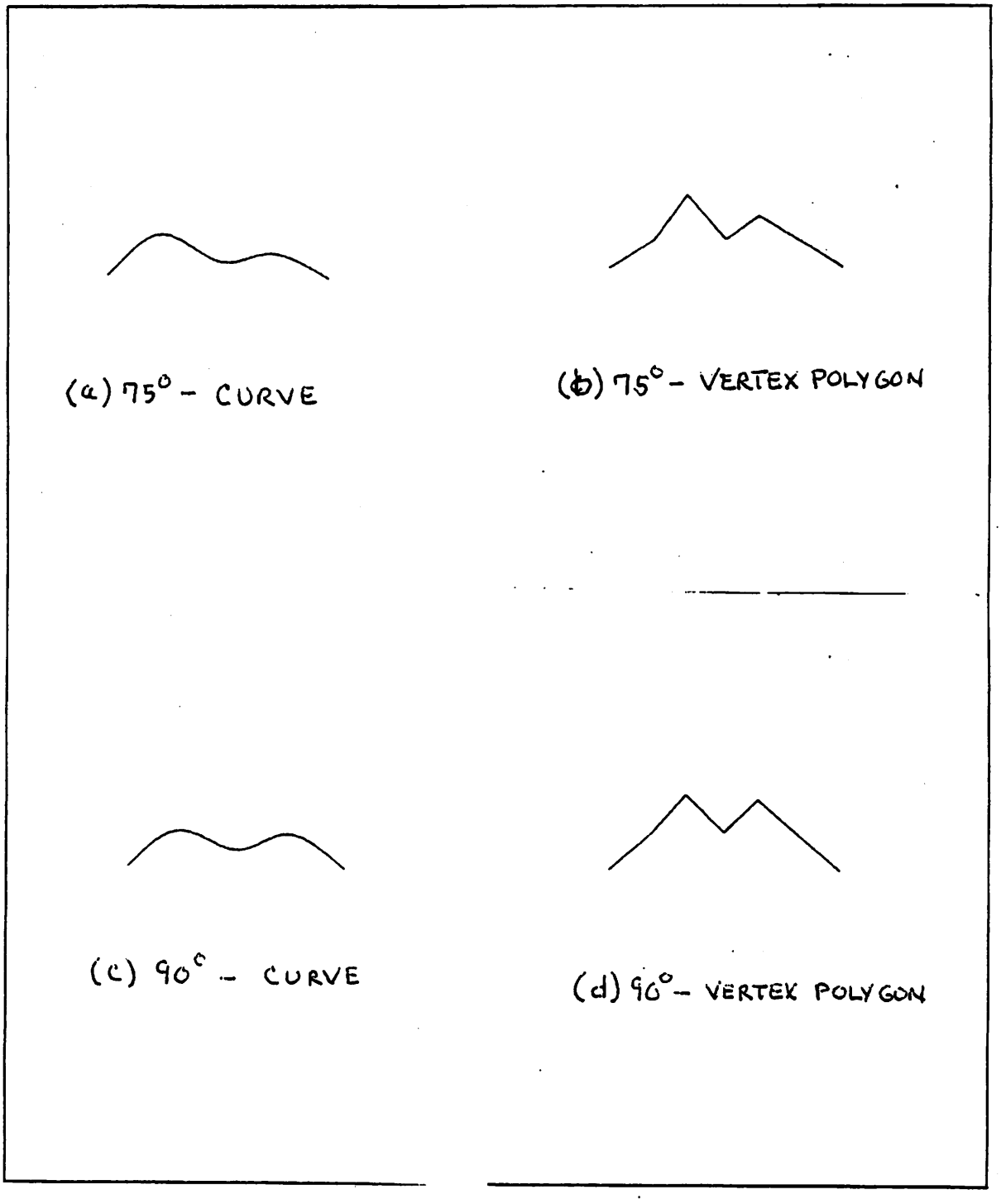

Figure 49: Projections of a cubic spline -- 75° and 90° *

space curve depicted in Figures 47, 48, 49. The 3D vertex polygon for the open cubic is the following list of points:

-1.5 0.5 0.5

-1 0 0

-0.5 0.5 -0.5

0 0 0

0.5 -0.5 -0.5

1 0 0

1.5 0.5 0.5

At present we must refrain from drawing any conclusions from these simple examples. This problem requires a rigorous analysis and it is a topic of future research.

Table 8 Projections of a circle -- 2D Shape Features

	0°	30°	45°	60°	75°	90°
OC	1	1	1	1	1	0
PL	0	0	0	0	0	0
SM	0	0	0	0	0	0
NV	10	10	10	10	10	2
MV	0	0	0	0	0	2
NT	10	10	10	10	10	0
MV/NV	0	0	0	0	.0	1
NT/NV	1	1	1	1	1	0
PGL/BL	0	0	0	0	0	1
C	12.57	12.61	12.91	14.62	21.13	0
MAXAT(DEG)	144	150.6	156.2	163.2	171.3	0
MAXL/MINL	1	1.28	1.55	2.16	3.93	1
AV_{2D}	0	.57	1.01	1.28	1.01	0

Table 9 Projections of a cubic -- 2D Shape Features

	0°	30°	45°	60°	75°	90°
OC	0	0	0	0	0	0
PL	0	0	0	0	0	0
SM	0	0	0	0	0	0
NV	7	7	7	7	7	7
MV	0	0	0	0	0	0
NT	5	5	5	5	5	5
MV/NV	0	0	0	0	.0	0
NT/NV	.71	.71	.71	.71	.71	.71
PGL/BL	1.36	1.23	1.22	1.26	1.32	1.39
C	0	0	0	0	0	0
MAXAT(DEG)	180	180	180	180	180	180
MAXL/MINL	1.23	1.76	1.89	1.74	1.44	1.22
AV_{2D}	34.6	7.25	11.71	15.18	17.31	18

III.5 Summary

In this chapter we have described a possible representation for shape information in a network structure. Within this representation certain access paths were exercised to demonstrate the indexing mechanism for 3D shape features. Obviously in a system which captures 3D data, 3D shape features may be computed from the data and no further mechanism would be required for indexing. For the 2D scene analysis problem it is necessary to correlate 2D shape features with 3D shape features in order to develop access paths from 2D to 3D. Two examples of selection of standard views and feature correlation are offered. We do not wish to imply that this complex and difficult problem is solved. We have only provided an approach that we believe is rich in possibilities and hope that the problem will receive careful attention in the near future. The significant problems are selecting a natural coordinate

system for each 3D entity, a theory for selecting standard views, the importance of degenerate views, and the prediction of feature values for non-standard views.

CHAPTER IV

2D SHAPE ANALYSIS

IV.1 Introduction to 2D Shape

In this chapter we are concerned with the problem of extracting shape descriptions from 2D images. We wish to compute 2D descriptions from the digital data which may be used to index into the data base of 3D representations as described in the previous chapter. The specific problem we address is that of fitting uniform cubic B-splines to the digital curves of a segmented image. In order to prepare the reader for the fitting algorithm, we briefly describe the form of the data, its special characteristics, and B-spline curve fitting. It is assumed that the reader is familiar with polynomial interpolation and least-squares curve fitting. An algorithm for B-spline curve fitting is described and some examples are presented.

IV.2 Input Data

We begin with a color photographic slide of a 3D scene. The slide is digitized into a 512x512 array of picture elements (pixels). For each pixel three intensity values are recorded, one for each of the primary colors (red, green, blue). The 512x512 array is referred to as a digital image (see the image in Figure 1 of Chapter I).

The image is enhanced by noise reduction techniques (Overton and Weymouth 1979) and it is segmented into regions (Riseman and Hanson 1978, Nagin 1979). The result of the segmentation is graphically depicted by the digital line drawing of Figure 50. The actual output is a network data structure (RSV) described in Chapter I (see Figure 3 of Chapter I).

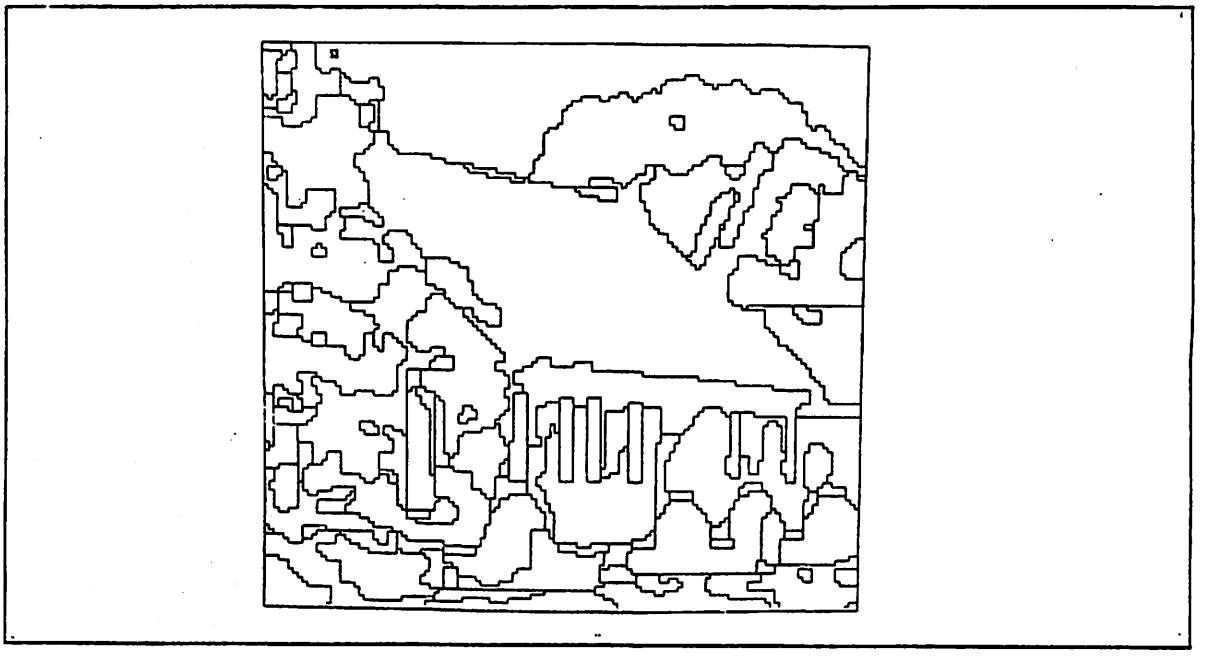

Figure 50: Segmented image of house scene

Each region node in the structure is linked to the segment nodes which represent the segments comprising its boundary. Each segment node has two endpoint nodes; in rare cases the segmentation produces a region bounded by a single closed segment; such segments have a single endpoint. Associated with each segment node is an ordered list of x-y coordinate pairs. Each list represents a 4-connected digital curve in which the x-y coordinates are integer-valued and successive pairs of points along the curve must satisfy one of four relationships. The point following any given point must be either one unit to the left, to the right, above, or below the given point. Image edges lie *between* pixels and an image curve consists of connected image edges. This is sometimes referred to as the "interpixel" representation (Prager 1979).

The problem addressed in this chapter is how to fit cubic B-spline curves to these 4-connected digital curves, taking into account their special characteristics. Once suitable cubic B-spline curves have been fitted to the data, the knot vector for each curve is converted to its

vertex polygon representation and the 2D shape features described in the previous chapter are computed. These 2D features are used to begin the indexing process and once suitable candidate 3D objects have been accessed, some form of matching must be applied. In this chapter we address only the problem of 2D cubic B-spline curve fitting to 4-connected digital curves. In the following chapter we discuss the issues related to matching in our representation.

IV.3 B-spline Curve Fitting

If the reader is unfamiliar with spline approximation, it is strongly recommended that he read Appendix I at this time. From the discussion in Appendix I it is obvious that the problem of B-spline curve fitting reduces to the problem of properly selecting the knot vector. Once the knot vector has been selected, the B-spline may be evaluated and the error of fit determined. If necessary the knot vector may be modified and a new B-spline fit generated. Since the B-spline approximation is a local approximation scheme, the cost of successive iterations is cheaper than the initial fit and error computation. We describe an algorithm for fitting uniform cubic B-splines to 4-connected digital curves.

IV.3.1 Knot Placement Algorithm

The purpose of the knot placement algorithm is to select points along the digital curve as knot locations in such a way that the associated B-spline best reflects the underlying curve - i.e. the underlying, possibly continuous curve in the scene which gave rise to the digital curve in the segmented image. The 4-connected representation allows only those lines in the scene which are parallel and perpendicular to the direction of scan to retain their smoothness in the image. All other lines whether straight or curved will have a 4-connected representation that

must be smoothed if it is to approximate the projection of the physical world in some way. The problem is that not all digital curves should be smoothed in the same way. The algorithm we present is sensitive to the data.

The first step of the algorithm is to compute a curvature approximation at each point along the digital curve. We use the following formula for normalized k-curvature (Davis 1976):

$$C^k(i) = 1 - | D^k(i) |$$

where

$$D^k(i) = a_{ik}.b_{ik} \; / \; | a_{ik} | \; | b_{ik} |$$

and where a_{ik} and b_{ik} are the vectors from point p_i to points p_{i-k} and p_{i+k} respectively and $a_{ik}.b_{ik}$ indicates the dot product of these two vectors (see Figure 51).

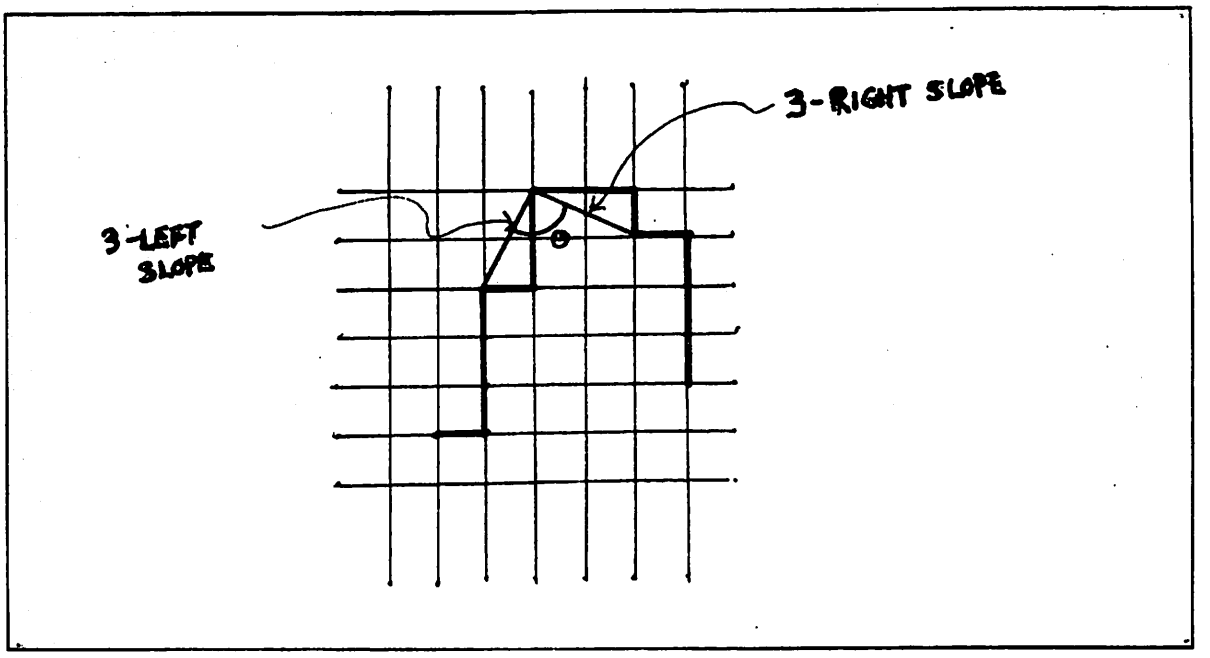

Figure 51: 3-curvature at point (x_i,y_i)

$D^k(i)$ is the cosine of the angle(θ) between the two vectors, a_{ik} and b_{ik} at the ith point along the curve. The two vectors, a_{ik} and b_{ik}, are approximations to the right and left slopes at the ith point. The measure, $1 - |D^k(i)|$, is used so that high values of $C^k(i)$ will correspond to sharp angles and low values will correspond to very broad angles -- i.e. when the vectors are nearly collinear. With this measure 1 indicates orthogonality and 0 indicates collinearity of the k-left slope and the k-right slope vectors. For 4-connected digital curves, values of k between 3 and 5 seem to produce good approximations. For $k>5$ the k-slopes do not reflect the local shape of the curve, while for $k<3$ the slopes reflect the *digital* curve (see Hanson and Riseman 1978b). Our curve fitting algorithm makes use of k-curvatures for values of k which are greater than or equal 3. Since the points near the ends of the curve may not have either a k-left slope or a k-right slope, a special approximation must be applied. The 3-curvature values at the terminal points of an open curve are set to zero. The 3-curvature values at points 2 and 3 are computed using the 1-left slope and 2-left slope, respectively. Similarly, at points n-1 and n-2 the 1-right slope and 2-right slope are used.

The k-curvature values are used to determine the placement of knots along the digital curve. The objective is to position the knots so as to get a cubic spline approximation to the digital curve which closely approximates the shape of the underlying curve. Points of high curvature are selected as initial candidates for knot positions. We include all points whose 3-curvature values are non-zero in the initial list of knot candidates. Our problem is to process this list of knot candidates so as to:

(1) eliminate spurious knots,

(2) find points of slope discontinuity and create multiple knots, and

(3) create simple knots (multiplicity 1) where required.

The three types of processing are *knot elimination*, *knot multiplication*, and *knot addition*. They are discussed in the following sections.

IV.3.2 Knot Elimination and Knot Multiplication

IV.3.2.1 Straight line segments

Straight line segments are a special case in which knot elimination is combined with knot multiplication. When a straight line portion of a curve is detected, the endpoints are included in the knot vector as knots of multiplicity 3 and the intervening knots are eliminated from the knot vector.

Determination of straight line segments is achieved via several heuristics. First, the initial 3-curvature values for all of the points of the curve are examined for runs of consecutive points of zero value. Such an analysis only detects horizontal and vertical straight lines. A more sophisticated heuristic is needed for oblique lines. For example, a line at a 45° angle with step size of 1 has consecutive 3-curvature values of .20 (actually consecutive substrings of length 1) except near the endpoints(see Table 10). The step size is the number of consecutive horizontal edges or the number of consecutive vertical edges in the digital curve. Notice that for each value of k, k knots at either end must be skipped before the consecutive run occurs.

A line at a 45° angle with a step size of 2 has alternating 3-curvature values of 0 and .20 (actually consecutive substrings of length 2) except near the endpoints and has consecutive 4-curvature values of 0 except near the endpoints (see Table 11). The 3-curvature vector for a line at a 45° angle with a step size of 3 contains consecutive substrings of length 3

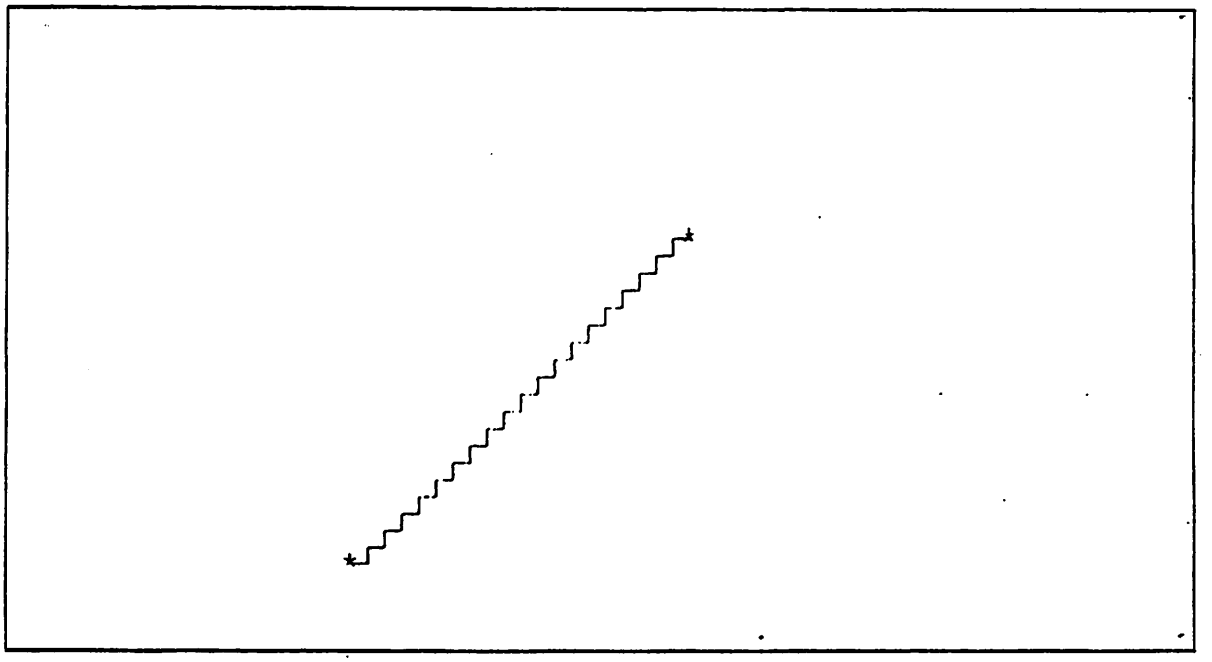

Figure 52: 4-connected digital curve for 45° line - step size 1

(1.0,.2,.2); the 4-curvature values contain consecutive substrings of length 3 (.1056,.1056,.4000); the 5-curvature values contain consecutive substrings of length 3 (.0769,.0000,.0000); and the 6-curvature values contain consecutive zeroes except near the endpoints (see Table 12). An obvious pattern has developed. The k-curvature vectors form repeating subsequences of length n where n is the step size and when k is less than 2n. Of course, whether 45° straight lines of step sizes greater than 1 may occur depends upon the level of quantization. Tables 11 and 12 have been truncated so that they would each fit on a single page; however, enough of the data appears for the cycles to be observed.

Table 10 K-curvature for 45° line - step size 1

		k-curvature					final	
X	Y	3	4	5	6	7	knot	mult
0	0	.0000	.0000	.0000	.0000	.0000	1	3
1	0	.5528	.2929	.4453	.2929	.4000		
1	1	.0513	.0000	.0194	.0000	.0101		
2	1	.2000	.0513	.1318	.0513	.1056		
2	2	.2000	.0000	.0194	.0000	.0101		
3	2	.2000	.0000	.0769	.0194	.0570		
3	3	.2000	.0000	.0769	.0000	.0101		
4	3	.2000	.0000	.0769	.0000	.0400		
4	4	.2000	.0000	.0769	.0000	.0400		
5	4	.2000	.0000	.0769	.0000	.0400		
5	5	.2000	.0000	.0769	.0000	.0400		
6	5	.2000	.0000	.0769	.0000	.0400		
6	6	.2000	.0000	.0769	.0000	.0400		
7	6	.2000	.0000	.0769	.0000	.0400		
7	7	.2000	.0000	.0769	.0000	.0400		
8	7	.2000	.0000	.0769	.0000	.0400		
8	8	.2000	.0000	.0769	.0000	.0400		
9	8	.2000	.0000	.0769	.0000	.0400		
9	9	.2000	.0000	.0769	.0000	.0400		
10	9	.2000	.0000	.0769	.0000	.0400		
10	10	.2000	.0000	.0769	.0000	.0400		
11	10	.2000	.0000	.0769	.0000	.0400		
11	11	.2000	.0000	.0769	.0000	.0400		
12	11	.2000	.0000	.0769	.0000	.0400		
12	12	.2000	.0000	.0769	.0000	.0400		
13	12	.2000	.0000	.0769	.0000	.0400		
13	13	.2000	.0000	.0769	.0000	.0400		
14	13	.2000	.0000	.0769	.0000	.0400		
14	14	.2000	.0000	.0769	.0000	.0400		
15	14	.2000	.0000	.0769	.0000	.0400		
15	15	.2000	.0000	.0769	.0000	.0400		
16	15	.2000	.0000	.0769	.0000	.0400		
16	16	.2000	.0000	.0769	.0000	.0400		
17	16	.2000	.0000	.0769	.0000	.0400		
17	17	.2000	.0000	.0769	.0000	.0101		
18	17	.2000	.0000	.0769	.0194	.0570		
18	18	.2000	.0000	.0194	.0000	.0101		
19	18	.2000	.0513	.1318	.0513	.1056		
19	19	.0513	.0000	.0194	.0000	.0101		
20	19	.5528	.2929	.4453	.2929	.4000		
20	20	.0000	.0000	.0000	.0000	.0000	1	3

Table 11 K-curvature for 45° line - step size 2

		k-curvature					final	
X	Y	3	4	5	6	7	knot	mult
0	0	.0000	.0000	.0000	.0000	.0000	1	3
1	0	.5528	.2929	.1679	.2929	.4000		
2	0	.5528	.2929	.4453	.5528	.4000		
2	1	.0000	.0513	.1318	.0513	.0161		
2	2	.2000	.0000	.0194	.0513	.0101		
3	2	.0000	.0000	.0000	.0194	.0570		
4	2	.2000	.0000	.0769	.2000	.1056		
4	3	.0000	.0000	.0000	.0000	.0000		
4	4	.2000	.0000	.0769	.2000	.0400		
5	4	.0000	.0000	.0000	.0000	.0000		
6	4	.2000	.0000	.0769	.2000	.0400		
6	5	.0000	.0000	.0000	.0000	.0000		
6	6	.2000	.0000	.0769	.2000	.0400		
7	6	.0000	.0000	.0000	.0000	.0000		
8	6	.2000	.0000	.0769	.2000	.0400		
8	7	.0000	.0000	.0000	.0000	.0000		
8	8	.2000	.0000	.0769	.2000	.0400		
9	8	.0000	.0000	.0000	.0000	.0000		
10	8	.2000	.0000	.0769	.2000	.0400		
10	9	.0000	.0000	.0000	.0000	.0000		
10	10	.2000	.0000	.0769	.2000	.0400		
11	10	.0000	.0000	.0000	.0000	.0000		
12	10	.2000	.0000	.0769	.2000	.0400		
12	11	.0000	.0000	.0000	.0000	.0000		
12	12	.2000	.0000	.0769	.2000	.0400		
13	12	.0000	.0000	.0000	.0000	.0000		
14	12	.2000	.0000	.0769	.2000	.0400		
14	13	.0000	.0000	.0000	.0000	.0000		
14	14	.2000	.0000	.0769	.2000	.0400		
15	14	.0000	.0000	.0000	.0000	.0000		
16	14	.2000	.0000	.0769	.2000	.0400		
16	15	.0000	.0000	.0000	.0000	.0000		
16	16	.2000	.0000	.0769	.2000	.0400		
17	16	.0000	.0000	.0000	.0000	.0000		
18	16	.2000	.0000	.0769	.2000	.0400		
18	17	.0000	.0000	.0000	.0000	.0000		
18	18	.2000	.0000	.0769	.2000	.0400		
19	18	.0000	.0000	.0000	.0000	.0000		

Table 12 K-curvature for 45° line - step size 3

		k-curvature					final	
X	Y	3	4	5	6	7	knot	mult
0	0	.0000	.0000	.0000	.0000	.0000	1	3
1	0	.1056	.2929	.4453	.2929	.2000		
2	0	.5528	.6838	.4453	.2929	.2000		
3	0	1.0000	.6838	.4453	.2929	.4000		
3	1	.2000	.1056	.0352	.1056	.1778		
3	2	.2000	.1056	.0000	.0194	.0570		
3	3	1.0000	.4000	.0769	.0000	.0101		
4	3	.2000	.1056	.0000	.0000	.0000		
5	3	.2000	.1056	.0000	.0000	.0000		
6	3	1.0000	.4000	.0769	.0000	.0400		
6	4	.2000	.1056	.0000	.0000	.0000		
6	5	.2000	.1056	.0000	.0000	.0000		
6	6	1.0000	.4000	.0769	.0000	.0400		
7	6	.2000	.1056	.0000	.0000	.0000		
8	6	.2000	.1056	.0000	.0000	.0000		
9	6	1.0000	.4000	.0769	.0000	.0400		
9	7	.2000	.1056	.0000	.0000	.0000		
9	8	.2000	.1056	.0000	.0000	.0000		
9	9	1.0000	.4000	.0769	.0000	.0400		
10	9	.2000	.1056	.0000	.0000	.0000		
11	9	.2000	.1056	.0000	.0000	.0000		
12	9	1.0000	.4000	.0769	.0000	.0400		
12	10	.2000	.1056	.0000	.0000	.0000		
12	11	.2000	.1056	.0000	.0000	.0000		
12	12	1.0000	.4000	.0769	.0000	.0400		
13	12	.2000	.1056	.0000	.0000	.0000		
14	12	.2000	.1056	.0000	.0000	.0000		
15	12	1.0000	.4000	.0769	.0000	.0400		
15	13	.2000	.1056	.0000	.0000	.0000		
15	14	.2000	.1056	.0000	.0000	.0000		
15	15	1.0000	.4000	.0769	.0000	.0400		
16	15	.2000	.1056	.0000	.0000	.0000		
17	15	.2000	.1056	.0000	.0000	.0000		
18	15	1.0000	.4000	.0769	.0000	.0400		
18	16	.2000	.1056	.0000	.0000	.0000		
18	17	.2000	.1056	.0000	.0000	.0000		
18	18	1.0000	.4000	.0769	.0000	.0400		
19	18	.2000	.1056	.0000	.0000	.0000		
20	18	.2000	.1056	.0000	.0000	.0000		
21	18	1.0000	.4000	.0769	.0000	.0400		

The straight line finder checks for such patterns as occur in Tables 10 and possibly those in Tables 11 and 12 depending upon parameter settings. When such a pattern is detected, the endpoints of the digital curve are retained in the initial knot vector as knots of multiplicity 3 and all intervening points are removed from the initial knot vector. The algorithm for detecting such patterns first looks for the longest horizontal and the longest vertical run. If they are of the same length, it proceeds with the check for a 45° line. The length of this run determines the maximum value of k (2 times the length of the run) for which k-curvature must be computed. If a run of length greater than some threshold (RUNLENGTH) is found, the points corresponding to the run of curvature values is made into a separate component of the digital curve by placing knots of multiplicity 3 at its endpoints. A long digital curve is broken into components in this fashion -- this is similar to the split part of a split-and-merge algorithm. The determination of long horizontal and vertical segments is relatively inexpensive compared to the computation of k-curvature. Generally, the technique of k-curvature pattern matching will be invoked only when the step size is small. Of course, this heuristic does not work for all oblique lines; however, it is justified by the high expected frequency of 45° lines in real images. In Tables 10 through 12 the points of the original curve which are retained in the final knot vector are indicated by a 1 in the *final knot* column. Figure 53 shows the cubic B-spline which results from the knot vector for the 45° digital curve of Figure 52. The algorithm which we have described works well in the ideal case; however it is sensitive to small amounts of noise in the data. We need to find ways to make the algorithm less sensitive to such noise.

The determination of oblique lines at angles other than 45° proceeds in a similar fashion. Each component of the digital curve is subjected to the following analysis if the length of its maximum horizontal run is different from the length of its maximum vertical run. The oblique

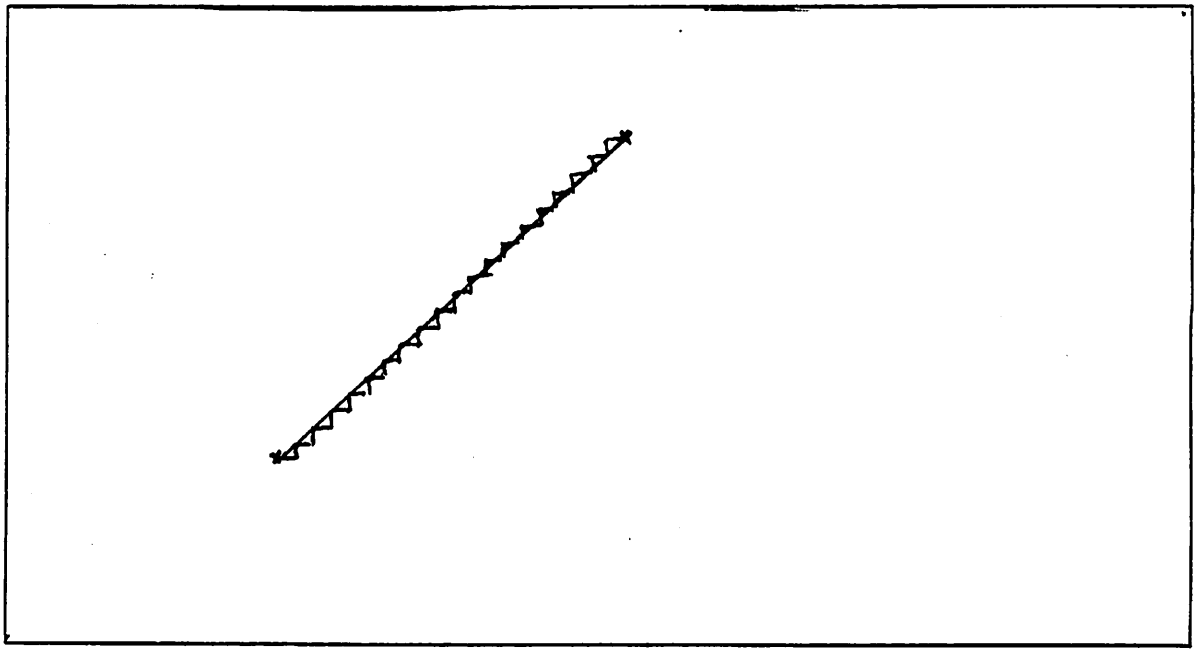

Figure 53: Digital curve for 45° line and B-spline fit

line analyzer looks for digital curves which proceed n units in the horizontal direction followed by m units in the vertical direction or n units in the vertical direction followed by m units in the horizontal direction. These patterns manifest themselves in the k-curvature sequences for successive values of k.

Tables 13 and 14 show the k-curvature sequences for two oblique curves of different slope; the two digital curves are shown in Figure 54. Table 13 contains data for an oblique digital line which increases 1 horizontal unit followed by 2 vertical units. In Table 14 the horizontal value is 3 and the vertical value is 1. It is clear that the length of the repeating subsequence is n+m and the first value of k ($k > n+m$) for which the k-curvature vector contains consecutive zeroes (except at the endpoints) is $2(n+m)$. It should be noted that this analysis is independent of the orientation of the oblique line in the image -- i.e. the lines with

step sizes (1,2), (1,-2), (-1,2), (-1,-2), (2,-1), (-2,1), and (-2,-1) all have the same pattern of k-curvature vectors.

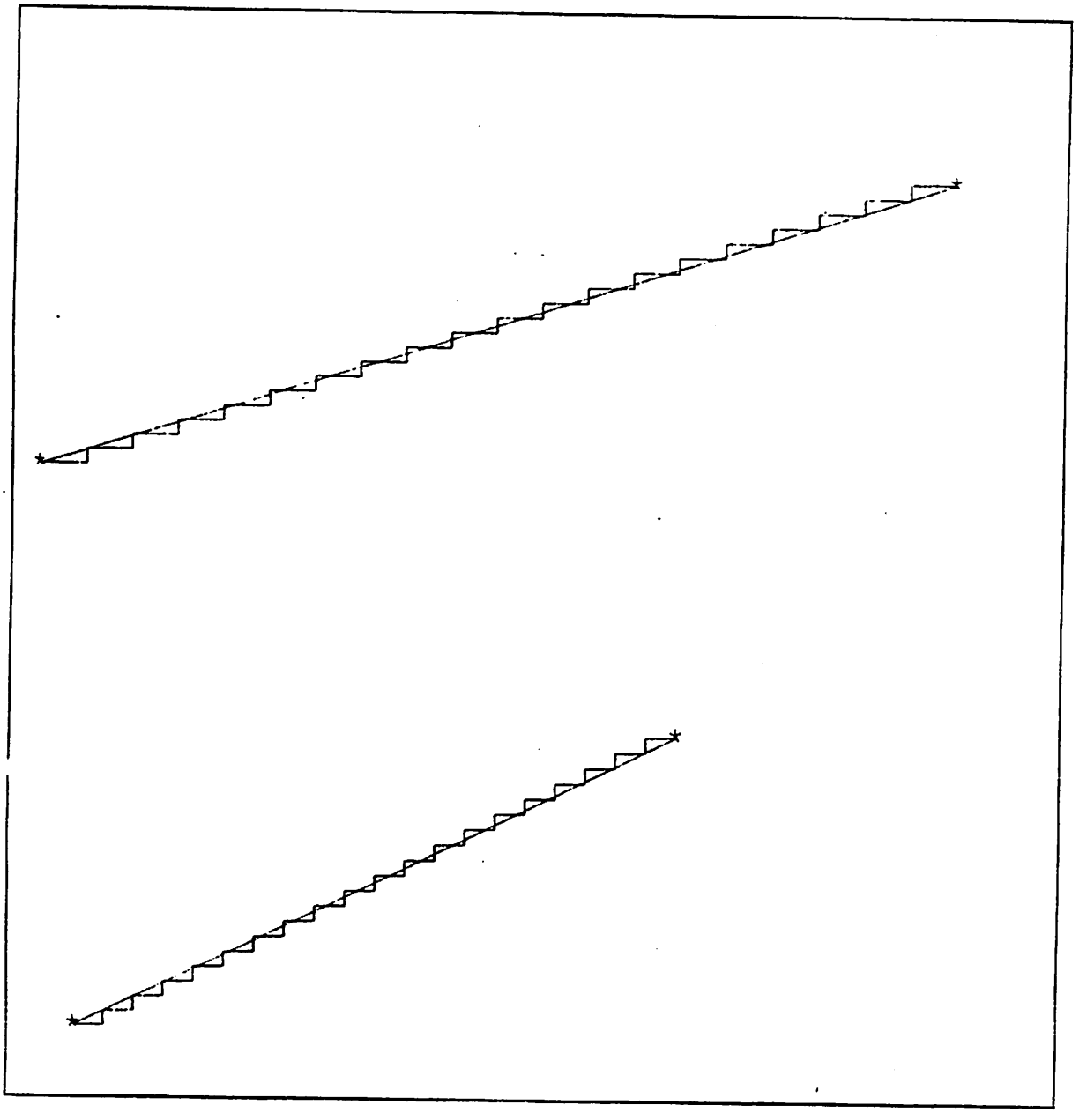

Figure 54: Two digital curves for oblique lines

Table 13 K-curvature for oblique line - step sizes (2,1)

		k-curvature					final	
X	Y	3	4	5	6	7	knot	mult
0	0	.0000	.0000	.0000	.0000	.0000	1	3
1	0	.1056	.0513	.1679	.1056	.0715		
2	0	.1056	.2929	.1679	.1056	.2000		
2	1	.0000	.0101	.0238	.0000	.0035		
3	1	.0000	.0000	.0352	.0101	.0017		
4	1	.0000	.1056	.0583	.0238	.0784		
4	2	.0000	.1056	.0583	.0000	.0035		
5	2	.0000	.0000	.0000	.0000	.0000		
6	2	.0000	.1056	.0583	.0000	.0344		
6	3	.0000	.1056	.0583	.0000	.0344		
7	3	.0000	.0000	.0000	.0000	.0000		
8	3	.0000	.1056	.0583	.0000	.0344		
8	4	.0000	.1056	.0583	.0000	.0344		
9	4	.0000	.0000	.0000	.0000	.0000		
10	4	.0000	.1056	.0583	.0000	.0344		
10	5	.0000	.1056	.0583	.0000	.0344		
11	5	.0000	.0000	.0000	.0000	.0000		
12	5	.0000	.1056	.0583	.0000	.0344		
12	6	.0000	.1056	.0583	.0000	.0344		
13	6	.0000	.0000	.0000	.0000	.0000		
14	6	.0000	.1056	.0583	.0000	.0344		
14	7	.0000	.1056	.0583	.0000	.0344		
15	7	.0000	.0000	.0000	.0000	.0000		
16	7	.0000	.1056	.0583	.0000	.0344		
16	8	.0000	.1056	.0583	.0000	.0344		
17	8	.0000	.0000	.0000	.0000	.0000		
18	8	.0000	.1056	.0583	.0000	.0344		
18	9	.0000	.1056	.0583	.0000	.0344		
19	9	.0000	.0000	.0000	.0000	.0000		
20	9	.0000	.1056	.0583	.0000	.0344		
20	10	.0000	.1056	.0583	.0000	.0344		
21	10	.0000	.0000	.0000	.0000	.0000		
22	10	.0000	.1056	.0583	.0000	.0344		
22	11	.0000	.1056	.0583	.0000	.0344		
23	11	.0000	.0000	.0000	.0000	.0000		
24	11	.0000	.1056	.0583	.0000	.0344		
24	12	.0000	.1056	.0583	.0000	.0344		
25	12	.0000	.0000	.0000	.0000	.0000		
26	12	.0000	.1056	.0583	.0000	.0344		
26	13	.0000	.1056	.0583	.0000	.0344		

Table 14 K-curvature for oblique line - step sizes (3,1)

		k-curvature					final	
X	Y	3	4	5	6	7	knot	mult
0	0	.0000	.0000	.0000	.0000	.0000	1	3
1	0	.1056	.0513	.0299	.0194	.0715		
2	0	.1056	.0513	.0299	.1056	.0715		
3	0	.1056	.0513	.1679	.1056	.0715		
3	1	.1056	.0000	.0029	.0077	.0122		
4	1	.0000	.0000	.0000	.0011	.0092		
5	1	.0000	.0000	.0000	.0352	.0167		
6	1	.1056	.0000	.0583	.0352	.0231		
6	2	.1056	.0000	.0583	.0352	.0231		
7	2	.0000	.0000	.0000	.0352	.0000		
8	2	.0000	.0000	.0000	.0352	.0000		
9	2	.1056	.0000	.0583	.0352	.0231		
9	3	.1056	.0000	.0583	.0352	.0231		
10	3	.0000	.0000	.0000	.0352	.0000		
11	3	.0000	.0000	.0000	.0352	.0000		
12	3	.1056	.0000	.0583	.0352	.0231		
12	4	.1056	.0000	.0583	.0352	.0231		
13	4	.0000	.0000	.0000	.0352	.0000		
14	4	.0000	.0000	.0000	.0352	.0000		
15	4	.1056	.0000	.0583	.0352	.0231		
15	5	.1056	.0000	.0583	.0352	.0231		
16	5	.0000	.0000	.0000	.0352	.0000		
17	5	.0000	.0000	.0000	.0352	.0000		
18	5	.1056	.0000	.0583	.0352	.0231		
18	6	.1056	.0000	.0583	.0352	.0231		
19	6	.0000	.0000	.0000	.0352	.0000		
20	6	.0000	.0000	.0000	.0352	.0000		
21	6	.1056	.0000	.0583	.0352	.0231		
21	7	.1056	.0000	.0583	.0352	.0231		
22	7	.0000	.0000	.0000	.0352	.0000		
23	7	.0000	.0000	.0000	.0352	.0000		
24	7	.1056	.0000	.0583	.0352	.0231		
24	8	.1056	.0000	.0583	.0352	.0231		
25	8	.0000	.0000	.0000	.0352	.0000		
26	8	.0000	.0000	.0000	.0352	.0000		
27	8	.1056	.0000	.0583	.0352	.0231		
27	9	.1056	.0000	.0583	.0352	.0231		

Digital curves which fail the straight line analysis are assumed to represent smooth curves with non-zero curvature at almost all points. Before we deal with the treatment of smooth curves, a few paragraphs must be devoted to curves with peaks or corners. Such curves appear in many objects and they play an important role in shape analysis.

IV.3.2.2 Peaks and Corners

The determination of peaks or corners is not as straightforward as the determination of straight lines. Whether or not a peak or corner exists is a somewhat subjective matter. It depends upon parameters which determine how long the edges which enter a peak must be and how great an angle will be tolerated between the two edges. In addition, because of the quantization level we must determine the width of a peak -- it may be one, two, three or more pixels wide depending upon the surrounding context. The problem lies in deciding whether the digital curve reflects a slope discontinuity in the underlying curve or whether the underlying curve is actually slope continuous. We provide techniques for constructing the knot vector when each of the cases is suspected; however, the determination of which case to look for depends upon global, contextual information which might be available to the control mechanism of a vision system, but which is unavailable to a local, shape analysis routine. Throughout this discussion we assume that a global control mechanism sets the parameters which determine the kinds of B-spline fits which can be achieved. Figure 56 shows the final knot placement (an * appears at each final knot location) for various parameter settings in the peak-finder algorithm. The corresponding k-curvature vectors are shown Tables 15 and 16. Figure 56 shows the cubic B-spline curves which the knot vectors in Figure 55 represent.

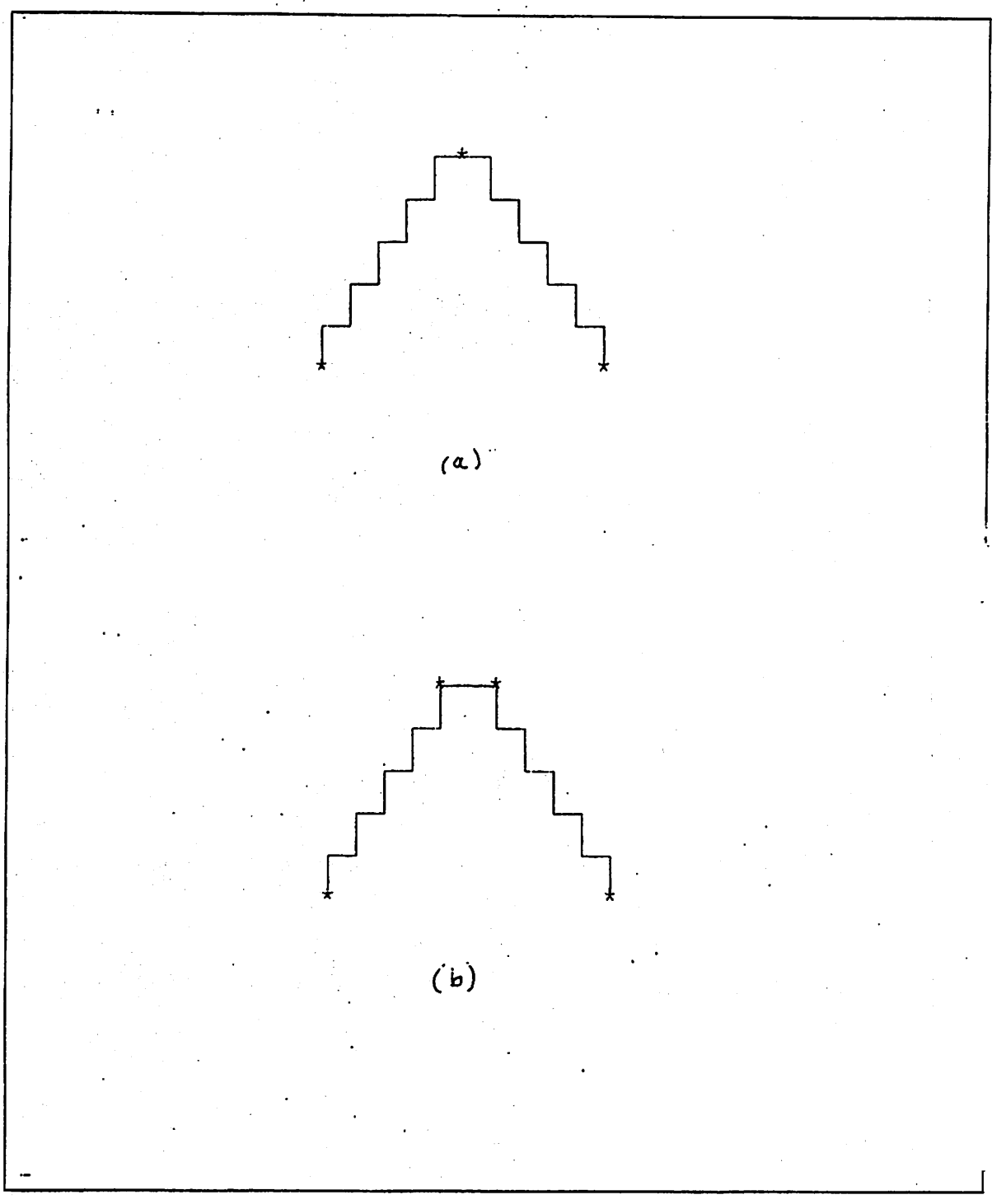

Figure 55: Examples of Peaks

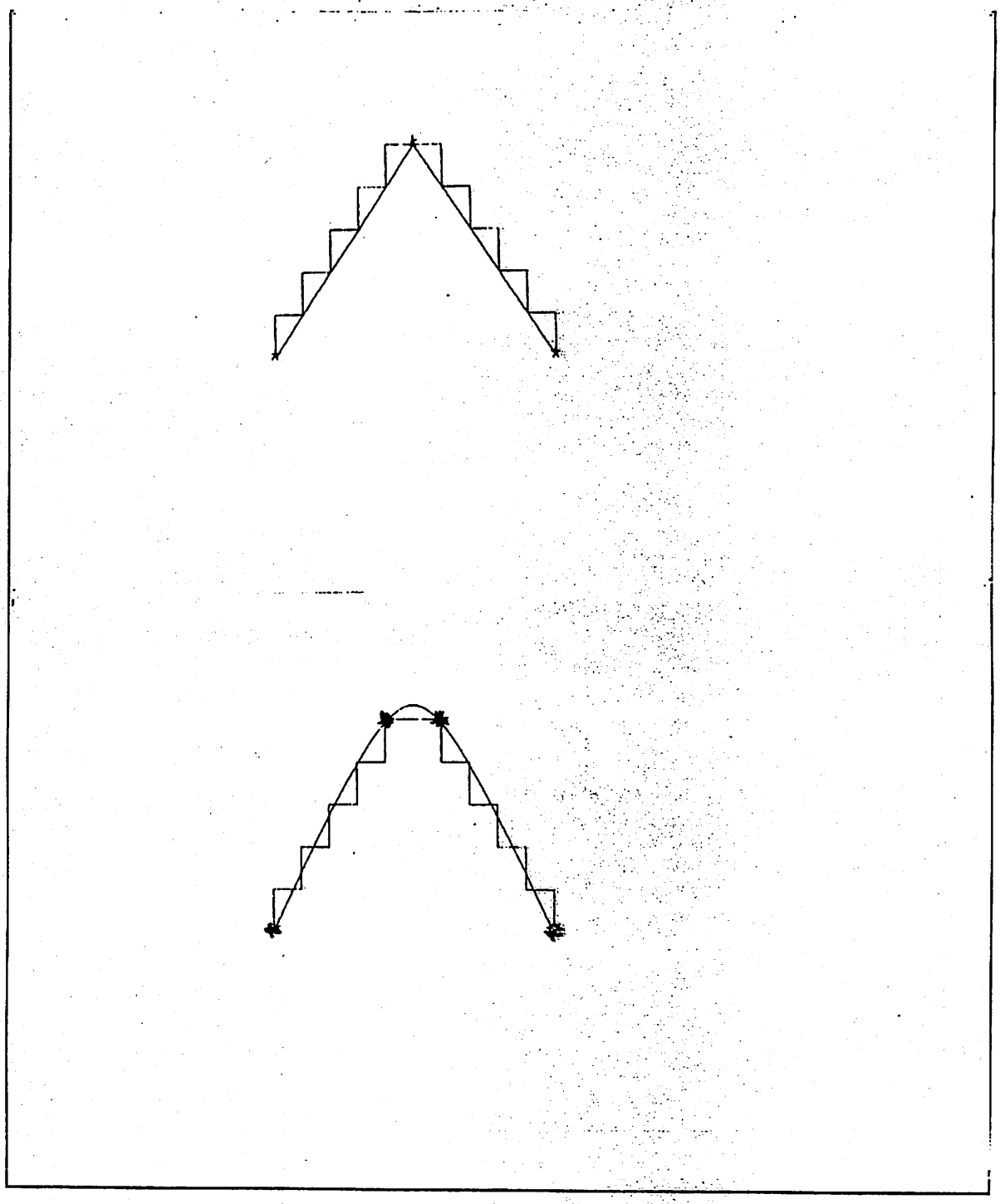

Figure 56: Cubic B-splines for Peaks in Figure 55

Table 15 K-curvature for curve in Figure 55 (a)

		k-curvature					final	
X	Y	3	4	5	6	7	knot	mult
-3	-12	.0000	.0000	.0000	.0000	.0000	1	3
-3	-11	.0000	.0513	.0299	.0194	.0136		
-3	-10	.1056	.0513	.0299	.0194	.0136		
-3	-9	.1056	.0513	.0299	.0194	.0715		
-3	-8	.1056	.0513	.0299	.1056	.0715		
-2	-8	.1056	.0513	.0000	.0011	.0032		
-2	-7	.1056	.0000	.0000	.0000	.0005		
-2	-6	.0000	.0000	.0000	.0000	.0000		
-2	-5	.1056	.0000	.0000	.0000	.0231		
-2	-4	.1056	.0513	.0000	.0352	.1122		
-1	-4	.1056	.0513	.0000	.0000	.0215		
-1	-3	.1056	.0000	.0583	.1679	.2526		
-1	-2	.0000	.1056	.3492	.8039	.7059		
-1	-1	.5528	.6838	.7831	.4453	.2705		
-1	0	.5528	.2929	.3273	.2106	.3095		
0	0	.4000	.2000	.1176	.4000	.2759	1	3
1	0	.5528	.2929	.3273	.2106	.3095		
1	-1	.5528	.6838	.7831	.4453	.2705		
1	-2	.0000	.1056	.3492	.8039	.7059		
1	-3	.1056	.0000	.0583	.1679	.2526		
1	-4	.1056	.0513	.0000	.0000	.0215		
2	-4	.1056	.0513	.0000	.0352	.1122		
2	-5	.1056	.0000	.0000	.0000	.0231		
2	-6	.0000	.0000	.0000	.0000	.0000		
2	-7	.1056	.0000	.0000	.0000	.0005		
2	-8	.1056	.0513	.0000	.0011	.0032		
3	-8	.1056	.0513	.0299	.1056	.0715		
3	-9	.1056	.0513	.0299	.0194	.0715		
3	-10	.1056	.0513	.0299	.0194	.0136		
3	-11	.0000	.0513	.0299	.0194	.0136		
3	-12	.0000	.0000	.0000	.0000	.0000	1	3

Table 16 K-curvature for curve in Figure 55 (b)

		k-curvature					final	
X	Y	3	4	5	6	7	knot	mult
-3	-12	.0000	.0000	.0000	.0000	.0000	1	3
-3	-11	.0000	.0513	.0299	.0194	.0136		
-3	-10	.1056	.0513	.0299	.0194	.0136		
-3	-9	.1056	.0513	.0299	.0194	.0715		
-3	-8	.1056	.0513	.0299	.1056	.0715		
-2	-8	.1056	.0513	.0000	.0011	.0032		
-2	-7	.1056	.0000	.0000	.0000	.0005		
-2	-6	.0000	.0000	.0000	.0000	.0000		
-2	-5	.1056	.0000	.0000	.0000	.0231		
-2	-4	.1056	.0513	.0000	.0352	.1122		
-1	-4	.1056	.0513	.0000	.0000	.0215		
-1	-3	.1056	.0000	.0583	.1679	.2526		
-1	-2	.0000	.1056	.3492	.8039	.7059		
-1	-1	.5528	.6838	.7831	.4453	.2705		
-1	0	.5528	.2929	.3273	.2106	.3095	1	1
0	0	.4000	.2000	.1176	.4000	.2759		
1	0	.5528	.2929	.3273	.2106	.3095	1	1
1	-1	.5528	.6838	.7831	.4453	.2705		
1	-2	.0000	.1056	.3492	.8039	.7059		
1	-3	.1056	.0000	.0583	.1679	.2526		
1	-4	.1056	.0513	.0000	.0000	.0215		
2	-4	.1056	.0513	.0000	.0352	.1122		
2	-5	.1056	.0000	.0000	.0000	.0231		
2	-6	.0000	.0000	.0000	.0000	.0000		
2	-7	.1056	.0000	.0000	.0000	.0005		
2	-8	.1056	.0513	.0000	.0011	.0032		
3	-8	.1056	.0513	.0299	.1056	.0715		
3	-9	.1056	.0513	.0299	.0194	.0715		
3	-10	.1056	.0513	.0299	.0194	.0136		
3	-11	.0000	.0513	.0299	.0194	.0136		
3	-12	.0000	.0000	.0000	.0000	.0000	1	3

For peaks and corners there are six basic parameters which control the action of the knot selection mechanism. They are the value used in computing k-curvature (k), the maximum allowable width for a peak (MAXWIDTH), the minimum allowable leg length for the two segments which form the angle at the potential peak or corner (MINLEG), the maximum allowable angle between the legs (MAXANGLE), the maximum allowable error for the straight line fits to the legs (MAXERROR), and the curvature threshold for selecting possible candidates (THR).

The peak-finder algorithm searches for slope discontinuities in the following way. After the search for straight lines (as described in the previous sections of this chapter) has concluded, we are left with one of three results:

(1) The digital curve represents a single straight line segment and the resulting knot vector consists of knots of multiplicity 3 at the endpoints of the digital curve.

(2) The digital curve has been broken into straight line components with knots of multiplicity 3 at the breakpoints.

(3) The digital curve has been broken into components, some of which may not be straight line segments.

In the first two cases it is clear where the points of slope discontinuity occur; however, there may be points of slope discontinuity which occur in the apparently curved components in case (3). These slope discontinuities are the ones which have become smoothed by quantization. It is just these peaks which we attempt to find with the following peak-finder algorithm:

(1) Search each apparently curved component of the digital curve for points with k-curvature greater than threshold (THR).

(2) Let each such point be the central point in an interval of MAXWIDTH and check that there are no other such points within MINLEG to either side. If there are points in this neighborhood, they are eliminated from consideration as possible peaks by the rules (3) through (5).

(3) Begin with the point with maximum k-curvature value. Eliminate all points in its MINLEG-neighborhood which have lower k-curvature value.

(4) If there are an odd number of points with k-curvature equal to the maximum in that neighborhood, then choose the central point in that point set.

(5) If there are an even number, choose the midpoint between the two central points of that set. Of course, this may result in a point which is not on the original digital curve; however, we are only attempting to approximate the underlying curve.

(6) Once the possible peaks have been chosen, an approximate angle subtended at the peak is computed. This angle is defined by the vectors from the peak to the points which are MINLEG units away on either side of the peak. Note that, when MINLEG equals k, the k-curvature at the peak is the cosine of the desired angle.

(7) If the subtended angle exceeds the maximum allowable angle (MAXANGLE), the corresponding point is retained in the knot vector as a knot of multiplicity 1.

(8) If the subtended angle is less than MAXANGLE, the point is considered a peak and it is retained in the knot vector as a knot of multiplicity 3.

In all the cases which we have considered so far, the straight-line finding process seems to be sufficient for finding corners of digital polygons, especially rectangles.; however, the peak-finder could also be used. Figure 57 shows a digital rectangle and a digital diamond shape. Both were adequately fit using the straight line heuristics without recourse to the peak-finder. The k-curvature data for Figure 57 appears in Tables 17 and 18.

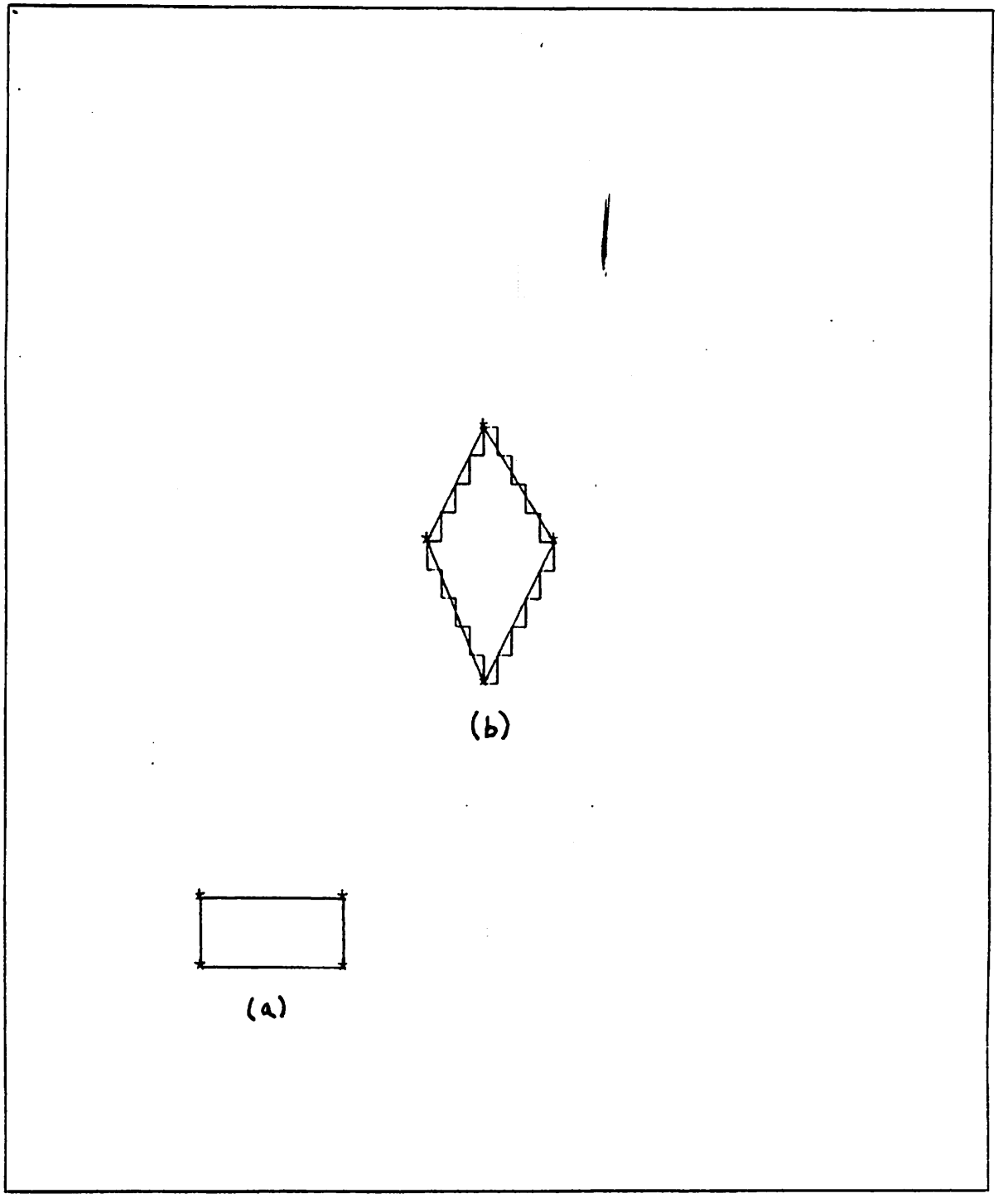

Figure 57: Digital Polygons with Cubic B-spline Fits

Table 17 K-curvature for Rectangle

		k-curvature					final	
X	Y	3	4	5	6	7	knot	mult
0	0	1.0000	1.0000	.6838	.2929	.0809	1	3
1	0	.2929	.5528	.9233	.5528	.1778		
2	0	.0000	.2929	.8039	.8039	.3861		
3	0	.1056	.2929	.7369	.7369	.2929		
4	0	.5528	.6838	1.0000	.4631	.1318		
5	0	1.0000	.6838	.4453	.2929	.0979	1	3
5	1	1.0000	.5528	.3273	.2106	.1624		
5	2	1.0000	.5528	.3273	.2106	.0715		
5	3	1.0000	.6838	.4453	.1679	.0344	1	3
4	3	.5528	.6838	.7575	.2929	.0570		
3	3	.1056	.5528	1.0000	.5528	.2929		
2	3	.1056	.5528	1.0000	.8039	.3861		
1	3	.5528	.6838	.9233	.5528	.1778		
0	3	1.0000	1.0000	.6838	.2929	.0809	1	3
0	2	.5528	.8586	.4855	.2929	.1056		
0	1	.6838	.6838	.3861	.2407	.1624		

Table 18 K-curvature for Diamond Shape

		k-curvature					final	
X	Y	3	4	5	6	7	knot	mult
0	0	.4000	.5528	.3273	.4000	.4801	1	3
1	0	.4000	.2000	.6154	.4000	.2759		
2	0	.4000	.5528	.3273	.4000	.4801		
2	1	.0000	.0513	.0299	.1056	.3273		
3	1	.0000	.0000	.1679	.2929	.5528		
4	1	.0000	.1056	.3492	1.0000	1.0000		
4	2	.0000	.0000	.4453	.5528	.7831		
5	2	.5528	.5528	.8760	.6838	.4335		
6	2	.4000	.5528	.3273	.4000	.4801		
6	3	.4000	.5528	.3273	.4000	.4801	1	3
5	3	.5528	.5528	.8760	.6838	.4335		
4	3	.0000	.0000	.4453	.5528	.7831		
4	4	.0000	.1056	.3492	1.0000	1.0000		
3	4	.0000	.0000	.1679	.2929	.5528		
2	4	.0000	.0513	.0299	.1056	.3273		
2	5	.4000	.5528	.3273	.4000	.4801		
1	5	.4000	.2000	.6154	.4000	.2759		
0	5	.4000	.5528	.3273	.4000	.4801	1	3
0	4	.0000	.0513	.0299	.1056	.3273		
-1	4	.0000	.0000	.1679	.2929	.5528		
-2	4	.0000	.1056	.3492	1.0000	1.0000		
-2	3	.0000	.0000	.4453	.5528	.7831		
-3	3	.5528	.5528	.8760	.6838	.4335		
-4	3	.4000	.5528	.3273	.4000	.4801		
-4	2	.4000	.5528	.3273	.4000	.4801	1	3
-3	2	.5528	.5528	.8760	.6838	.4335		
-2	2	.0000	.0000	.4453	.5528	.7831		
-2	1	.0000	.1056	.3492	1.0000	1.0000		
-1	1	.0000	.0000	.1679	.2929	.5528		
0	1	.0000	.0513	.0299	.1056	.3273		

IV.3.2.3 Smooth Curves

A knot vector containing only knots of multiplicity 1 (except maybe at its endpoints) defines a smooth (curvature continuous) curve. The heuristics described above try very hard to find apparent slope discontinuities and place multiple knots at such locations. Wherever multiple knots are not appropriate, knots of multiplicity 1 have been left at the points of high curvature. Thus, a smooth curve results whenever all of the special effort to find discontinuities fails. Observe the two cubic B-spline fits to the digital curve in Figure 58. By setting the discontinuity parameters so broad, a smooth curve results (Figure 58 (a)). When they are set more tightly, the discontinuos curve with peaks (Figure 58 (b)) results.

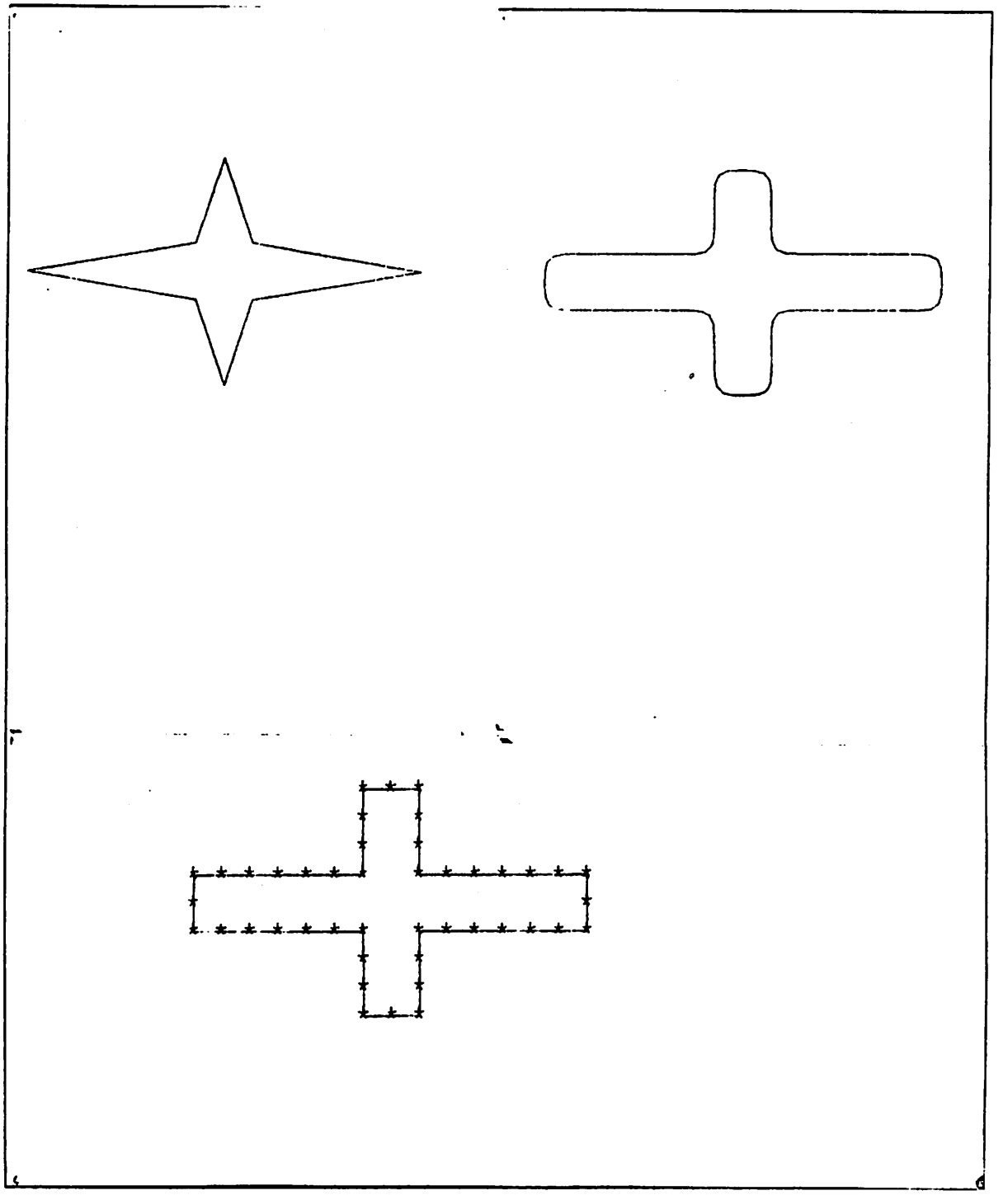

Figure 58: Smooth and Discontinuous fit to same digital curve

In certain situations we may be confident that the right type of cubic B-spline curve has been fitted to the digital curve, but dissatisfied with the closeness of the fit. In these cases we may wish to adjust some of the knots to reduce the error of fit. We propose the following knot adjustment procedure and error calculation. Currently the knots may be adjusted interactively at a graphics terminal.

IV.3.3 Knot Adjustment

When the total error (the error computation is described in the next section) exceeds some threshold, the individual errors are examined. The point with the maximum error is selected. In case there is more than one point with maximum error, each point is analyzed in succession. If the absolute value of the error at this point exceeds twice the average error, then knot adjustment is attempted in the neighborhood of the selected point. Several situations pertain:

(1) if the knots on both sides of the selected point are more than k points away along the digital curve (where k is the value used to compute k-curvature), and both have multiplicity 1, then a new knot of multiplicity 1 is added to the knot vector at the selected position. The spline is then recomputed using the new knot vector and a new error is calculated.

(2) If the knots on both sides of the selected point are k points away and each has multiplicity 3, then a new knot of multiplicity 1 is added at the selected position. The spline is recomputed only locally since the two knots of multiplicity 3 isolate that portion of the spline from changes which occur in the knot vector outside of the interval defined by the two knots of multiplicity 3. The algorithm is then applied recursively to this portion of the spline as if it were the whole spline.

(3) If a knot on one side of the selected point has multiplicity 3 and both knots are not within k points of the selected postion, a knot of multiplicity 1 is added. However, only the proper portion of the spline is recomputed. This is a one-sided version of step (2).

(4) If either knot is within k points of the selected point, we move the closest knot one position along the digital curve towards the selected point and recompute

the spline. If both knots are equidistant from the selected point, both are moved one position closer.

IV.3.4 Error Calculation

The error measure is computed separately for each subinterval of the knot vector. If the error on any subinterval exceeds a threshold, then the knots on that portion of the spline are adjusted in a manner which we shall describe briefly. First the error measure and the threshold must be discussed. The error measure is the distance from each point on the digital curve to the spline curve. The distance need not be computed at the knots, since the spline is guaranteed to go through the knots and they are selected from points on the digital curve.

The distance calculation is performed as follows:

(1) If both ends of the spline subinterval are bounded by knots of multiplicity 3, we use the formula for the distance from a point to a line:

$$d = (ax_0 + by_0 + c)/\sqrt{a^2 + b^2}$$

where the equation of the line between the two multi-knots is $ax + by + c = 0$ and the point under consideration on the digital curve is (x_0,y_0).

(2) If either end is bounded by a simple knot, we compute the distance between the interior point of the digital curve and the corresponding point on the spline under uniform parameterization. For example, if a subinterval contains n-1 points, the spline is computed at parametric values 1/n, 2/n,..., n-1/n. Thus, the distance from a point to the spline curve is defined:

$$d_i = \sqrt{(x_i - S_X(i/n))^2 + (y_i - S_Y(i/n))^2}$$

For smooth curves all distances less than 1 are ignored in the error calculation, because they indicate that the smooth curve and the digital curve are less than a pixel apart and thus the error is below the noise level. For horizontal and vertical straight lines the error must be zero at every point since cubic B-splines approximate straight lines exactly. For other curves we use the square root of the mean of the squared distances as a measure of total error:

$$E^2 = (1/n - 1)\sum_{i=1}^{n-1} d_i^2$$

It is clear that rule (2) subumes rule (1); however, the calculation of rule (1) is much less expensive when it is known that a straight line connects the knots.

IV.4 Results

House Roof

Finally, we have an example of these techniques applied to a region boundary from the image of the house scene shown earlier in the chapter. Figure 59 shows the points around the region in the image which represents the roof of the house. The segmentation process was unable to find a boundary between the edge of the roof and the tree on the right hand side of the image, accounting for the unusual shape of the region. At the resolution employed the two regions were almost visually indistinguishable and higher level processes are required to fill in the subjective contour (Parma, Hanson, and Riseman 1980). Figure 60 shows a piecewise linear fit to the endpoints of the components of the digital curve after thresholding on 3-curvature at the value .30 and applying the straight line heuristics with a RUNLENGTH threshold of 10. Figure 61 shows the piecewise linear fit to the components resulting from

thresholding 3-curvature at .60. Figure 62 shows the cubic B-spline fit to the knot vector resulting from thresholding 3-curvature at .60. The knot placement algorithm has done a good job of finding the linear components and smoothing the apparently curved components.

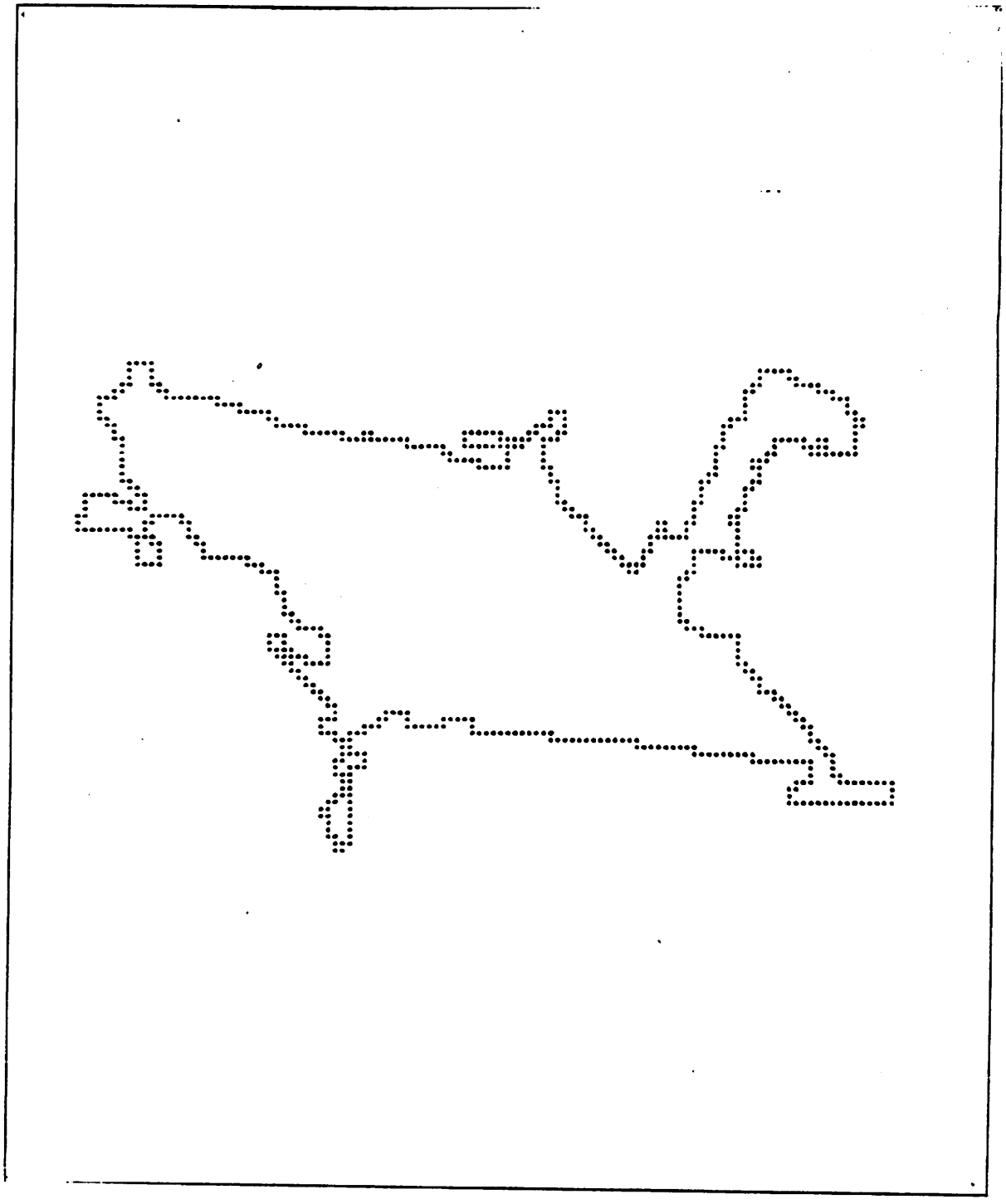

Figure 59: House Roof -- Points of digital boundary

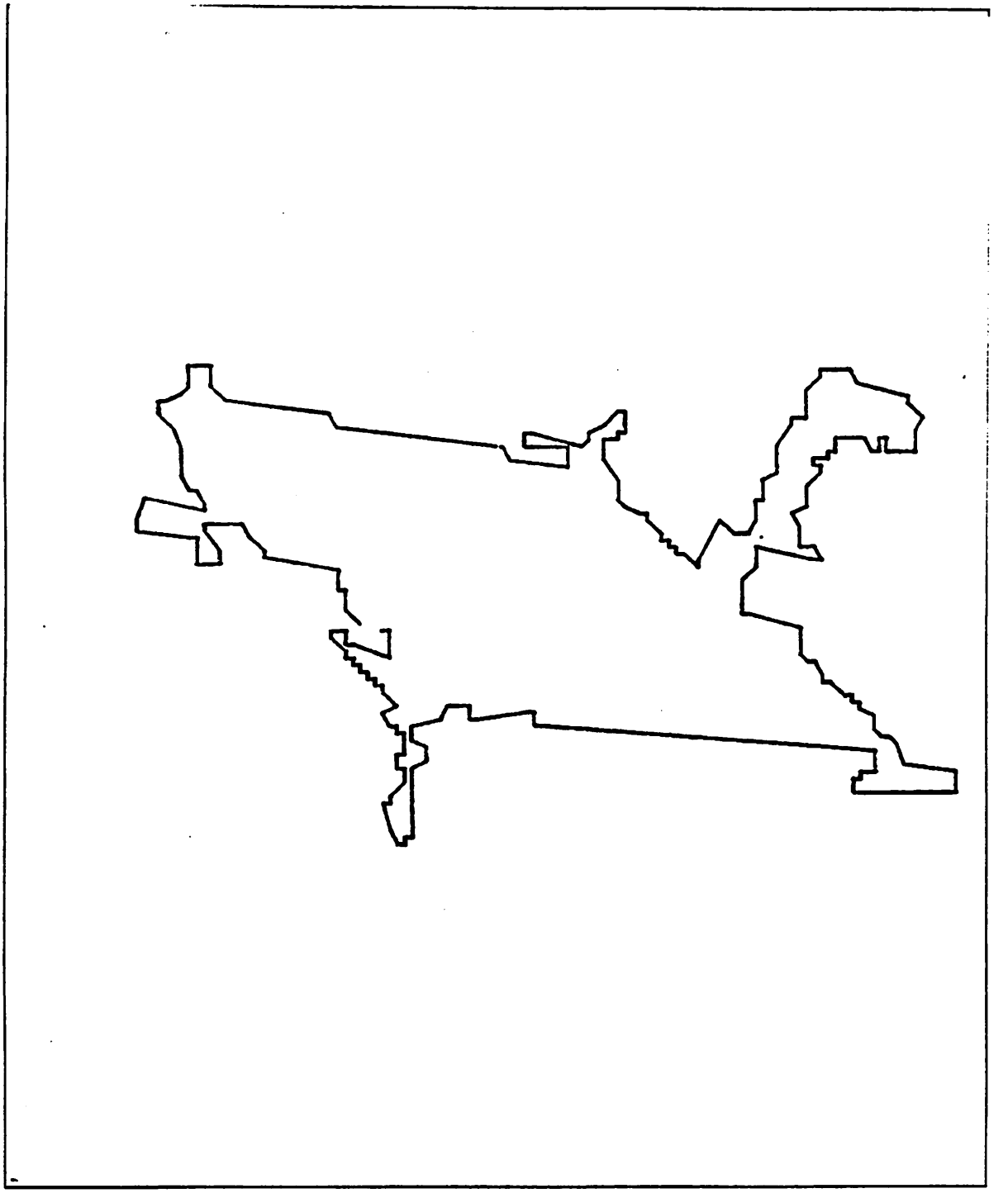

Figure 60: House roof -- Piecewise Linear Fit after 3-curvature threshold at .30

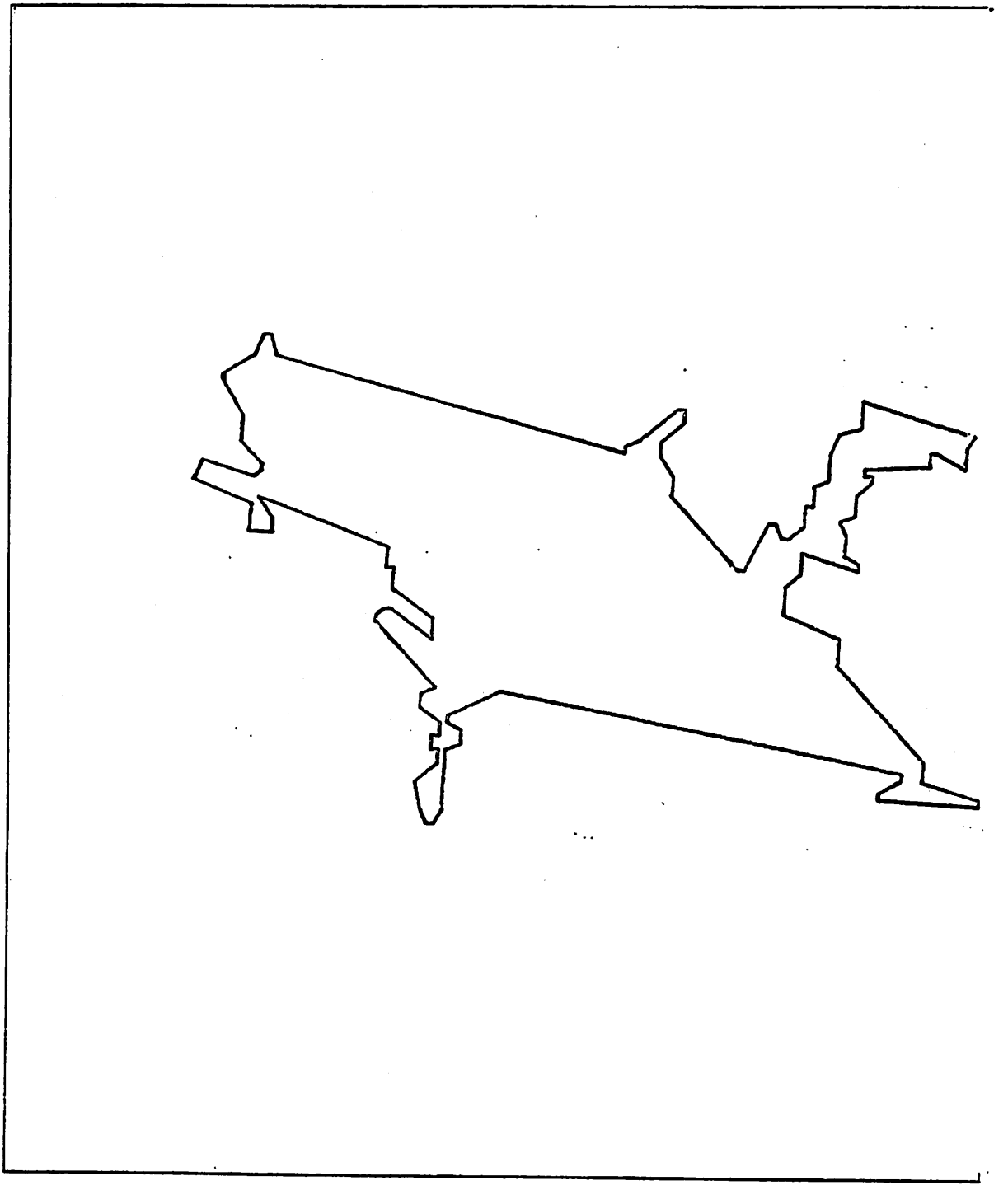

Figure 61: House roof -- Piecewise Linear Fit after 3-curvature threshold at .60

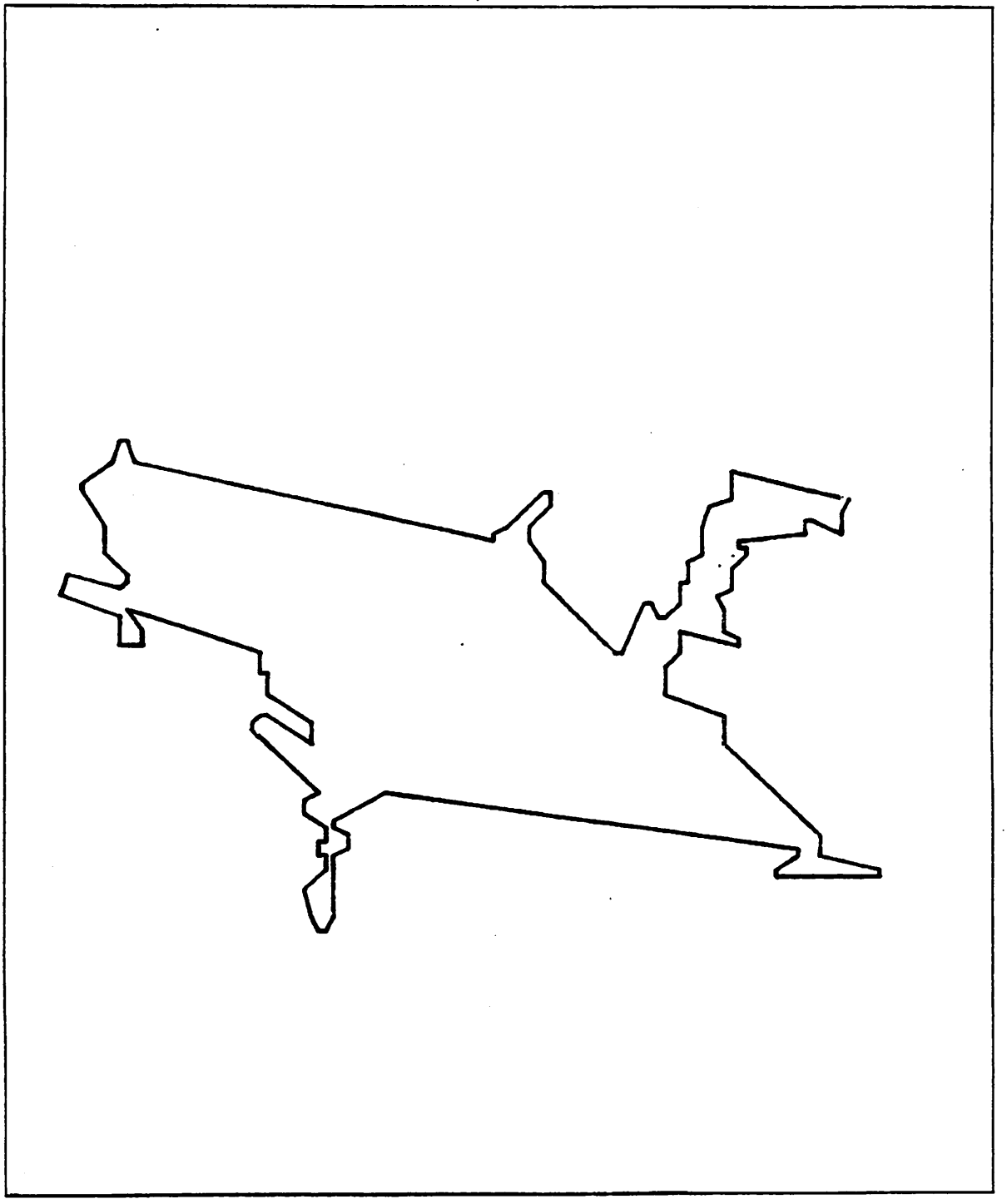

Figure 62: House roof -- Cubic B-spline Fit after 3-curvature threshold at .60

IV.5 Conclusions

In this chapter we have given techniques for fitting cubic B-spline curves to 4-connected digital curves extracted from a 2D segmented images. Specific heuristics were given for analyzing a variety of curve types to obtain a knot vector for a cubic B-spline which reflects the shape of the underlying curve. Both smooth and slope discontinuous curves may result from this fitting process. The search for points of discontinuities is controlled by a few parameters. These parameters must be set by a global control mechanism which has access to context information not available to the curve fitting procedures. An interesting research problem would be to determine how to automatically set these parameters.

Now that we have presented 3D shape representation, 2D curve fitting, and indexing, we shall discuss how this work might be used to achieve shape matching for object recognition in computer vision systems.

CHAPTER V

AN APPROACH TO 3D SHAPE MATCHING

V.1 Introduction

The purpose of this chapter is to describe how shape matching might work in a computer vision system based upon our representation. We are fully aware that some of our ideas about matching may not prove feasible; however, it is important to describe how we expect the matching process to be integrated with the other components of the system.

Consider the following scenario for shape matching in 2D scene analysis. Suppose a computer vision system encounters a scene containing familiar objects - i.e. object instances which it has processed at some previous time. This is a situation in which specific 3D object models have been created and exist in the object-instance, surface-instance, and curve-instance planes of LTM. Also, suppose the system is given expectations of a scene containing those object instances. This is analogous to the situation when one wakes up in the morning and expects to see specific object instances (a bed, desk, bureau,...) which have specific well-understood shapes. In VISIONS terminology there is a specific schema-instance which is instantiated upon awakening. The schema-instance provides the objects, their descriptions, and their approximate locations.

There are three kinds of problem which the system might attempt to solve. The first is to verify that each of the objects in the room is still in the room at the location it had been when this schema-instance had last been instantiated. A second and more difficult problem would be to verify that all of the objects are still in the room, given that the room was rearranged

while one slept or since the last instantiation of that schema-instance. Finally, a third problem would be to determine whether an object was missing from the rearranged room.

These tasks involve difficult shape matching problems; however, they are simpler than the general problem of 2D scene analysis. First, the scene has been seen before. The shape and location of every object in the scene are known. Secondly, there are specific object model instances to match against, instead of object class prototypes which are general object models. Because the location of each object is known, the segmentation problem is somewhat alleviated. The question then arises: How might a computer vision system employing our shape representation go about matching a specific 3D object model instance against the regions, segments, and vertices of the projection of the object in a 2D image?

The next few sections of this chapter describe how these issues might be addressed in the future.

V.2 Object Recognition and Shape Matching

In general, the object recognition process will involve many steps some of which are not directly related to shape matching. For example, color and texture are useful cues for object identification, but are unrelated to shape. In this section we shall outline the relevant image processing issues which directly impact the shape matching process.

Let us assume we have a computer vision system with our internal representation for objects and that when presented with an image of a scene containing 3D objects, it performs a suitable segmentation of the image as described in Chapter I. These are 2D entities which may or may not correspond to 3D entities. For example, due to the partial occlusion of one object by another, a 2D edge which is not the projection of a 3D edge will appear in the

image. In addition, it is often the case that a surface which is partially in shadows will appear as two or more regions in the image. The point is that 2D segmentation alone is insufficient to provide the kind of information needed for object recognition Further image analysis must be performed to obtain hypotheses for the 3D entities which give rise to the regions and boundaries of the image. Some of the required processing includes:

(1) Region merging/splitting based upon shadow, occlusion, and junction analyses.

(2) Boundary merging/splitting based upon shadow, occlusion, and junction analyses.

(3) Depth hypotheses based upon perspective analysis.

Once these kinds of processing have been accomplished, 2D shape descriptions must be obtained from this augmented segmentation. They are obtained by cubic B-spline curve fitting followed by computation of the 2D shape features described in Chapter II. From these 2D shape descriptions and the depth information in the augmented segmentation, surface and volume hypotheses might be generated for the regions of the image. These hypotheses together with the aforementioned shape features for curves provide indexing paths into the data base of stored 3D object descriptions. Only at this point would matching be invoked. The matching process would attempt to decide whether a region or collection of regions in the image is the projection (from some unknown point of view) of one or more of the candidates (3D objects, volumes, or surfaces) obtained by indexing. As we present how we would attempt to solve this problem, the reader should bear in mind that this approach has not yet been tested. It is the subject of future research.

Central to the matching process is the determination of whether or not two collections of geometric entities represent the same shape. Several algorithms (Aki and Toussaint 1978, Bykat 1979, and Manacher 1976) have been developed for determining the similarity of two

2D polygons under the affine transformations of translation, rotation, and uniform dilation or contraction. Bykat's algorithm is computationally efficient (O(n)) and he is currently extending it to three dimensions. These algorithms are useful for matching polygons of the same dimensionality; however, they are inadequate for matching 2D polygons to 3D polygons.

Roberts (Roberts 1965) developed a technique which derives the best match between a set of 3D points and a set of 2D points. His technique derives the best transformation which takes the set of 3D points into the set of 2D points -- i.e. the best transformation in terms of minimum squared error. Both Bykat's and Roberts' algortihms will be described in more detail in the next few sections and we shall describe how these algorithms might be used in a matching process based upon our representation.

V.3 Polygon Matching

Bykat's algorithm is typical of those currently used for 2D polygon matching. It determines whether two polygons are similar under some isogonal affine transformation. An isogonal affine transformation is a reflection, rotation, translation, or uniform dilation/contraction. The algorithm makes use of some of the following properties of 2D polygons which remain invariant under isogonal affine transformations:

(1) straight lines

(2) ratio of areas

(3) angles

(4) number of vertices

(5) ratio of segment lengths

The algortihm proceeds as follows:

" Let P be the polygon defined by the vertices $(a_1,a_2,...,a_n)$ and Q be the polygon defined by the vertices $(b_1,b_2,...,b_k)$.

Step 1: If $n \neq k$, then P is <u>not</u> similar to Q.

Step 2: Compute γ, the centroid of P, and γ', the centroid of Q.

Step 3: Compute the values $s_i = d(a_i,\gamma)$, $l_i = d(a_i,a_{i+1})$, $m_i = d(b_i,\gamma')$, and $z_i = d(b_i,b_{i+1})$ for all i.

Step 4: Compute the magnification factor $\frac{1}{\alpha} = \sum_{i=1}^{n} l_i / \sum_{i=1}^{n} z_i$

Step 5: Compute $M_i = \frac{m_i}{\alpha}$, $Z_i = \frac{z_i}{\alpha}$

Step 6: Define the triangulation sequences $p = s_1 l_1 s_2 l_2 \ldots s_n l_n$ and $q = M_1 Z_1 M_2 Z_2 \ldots M_n Z_n$. If p is a cyclic permutation of q, then the polygons are similar.

Step 7: Define p' to be the reflection of p -- i.e. $p' = l_n s_n l_{n-1} s_{n-1} \ldots l_1 s_1$ If p' is a cyclic permutation of q then the polygons are similar, otherwise they are not. " 11

Let us explore the ways in which Bykat's algorithm might be incorporated into the shape matching process of a computer vision system. Since the algorithm requires both polygons to be of the same dimensionality, it may be used directly in applications in which the data and the internal representation are of the same dimensionality. Both the internal representation and the descriptions extracted from the data must be based on polygons. In our domain of applications the internal representation is three dimensional and the external descriptions are two dimensional. Thus, one or the other must be converted in dimensionality before the algorithm may be applied. The conversion in dimensionality may go in either direction -- (1) 3D to 2D via perspective projection or (2) 2D to 3D via depth hypotheses for image entities.

The conversion from 3D to 2D requires that a center of projection and a view reference point be determined, whereas the 2D to 3D conversion requires that depth hypotheses be made for image points prior to application of the algorithm. Roberts chose to develop a technique which did not explicitly convert to the same dimensionality *before* matching. His algorithm derives the best compound transformation (isogonal affine transformation in 3D and a perspective transformation) which could transform the given 3D point set into the given 2D point set.

Before going on to describe the Roberts technique, we should mention two important ways in which the Bykat algorithm must be modified in order to be adapted for shape matching. First, it is often not sufficient to merely determine if two polygons match or not. In many cases useful information can be derived from the manner in which they mismatch. Thus, Bykat's algorithm should be modified to return the features(angles, lengths, ratios,...) upon which the polygons mismatch and the amount of mismatch. The second modification would be to permit a certain amount of tolerance in the matching of the components of each triangulation. In the algorithm as presented the components of the triangulation, p, must precisely match the components of the triangulation, q. This is too stringent a requirement for applications involving noisy data and approximate internal representations.

Fischler and Eschlager have achieved some success with spring-loaded template matching (Fischer and Eschlager 1973), a technique which matches the graph structures associated with object structures. Each node in the graph has the features of the corresponding object. The nodes of one object graph are matched with each of the nodes of a second object graph and the error in the local match on the features is represented as tension in a spring stretched between the nodes. The spring-loaded template matcher minimizes global tension in the bipartite graph formed by the initial matching of nodes. During the minimization process some

of the springs are broken and the resultant graph contains an optimal match under the tension criteria. Fischler and Eschlager formulated the spring-loaded template matching as a form of dynamic programming algorithm which drastically reduces their computational requirements. A similar approach could be taken with respect to modifying the Bykat algorithm.

V.4 Roberts Similarity Test

The Roberts similarity test may be succinctly described,

> "We are given a matrix A of n points (x,y,z,w) from a model and want to find a transform H that will most nearly fit n points (y,z,w) in a matrix B. Thus we hope for
>
> $$AH \cong B$$
>
> However, we cannot write an equality sign above without introducing a diagonal scale matrix D which will allow the w_i values of AH to differ from the w_i values of B,
>
> $$AH = DB \qquad \text{" 12}$$

The matrices A and B represent lists of points in three and two dimensions respectively. Both A and B are expressed in homogeneous coordinates. Roberts derives the solution to this equation and the constraints on the solution.

The first constraint required for a complete solution to exist is that n must be greater than or equal to 6. By using a minimum squared technique, he solves the equation for H,

$$H = (A^TA)^{-1}A^TDB$$

where A^T denotes the transpose of A.

However, this solution depends upon the unknown diagonal scale matrix D. Roberts solves for the diagonal elements of D by the following means:

(1) The minimum squared error equations imply that the diagonal elements of AHB^T must equal those of DBB^T.

(2) Substituting the solution for H from above allows him to define $G = A(A^TA)^{-1}A^T - I$. Thus, the matrix $GDBB^T$ has zero diagonal terms.

(3) Define the matrix $Q = BB^T$.

(4) Form the nxn matrix S by multiplying the terms of G by those of Q^T, term by term.

$$S_{ij} = g_{ij}q_{ji}.$$

(5) Solve the system of equations $Sd = 0$ where $d = d_1 \ldots d_n$ the diagonal elements of the matrix D.

The equation $Sd = 0$ requires S to be a singular matrix with degeneracy at least one (i.e. the rank of S is at most n-1). If the degeneracy of S is exactly one, then D and H have the same scale and the scale factor may be ignored in the solution. If the degenracy is greater than one, then either too few equations were given ($n<6$) or there was no perspective in the picture. In the latter case 1's may be substituted for each of the undefined values d_i of the matrix D, thus giving an incomplete solution for H.

Roberts includes the following three notes in his analysis:

"It should be noted that if n=4, A^{-1} will probably exist and the best solution obtainable is the one with no perspective,

$$H = A^{-1}B$$

If solutions without perspective are expected, the matrix D is unnecessary and the ordinary minimum squared error solution holds,

$$H = (A^TA)^{-1}A^TB$$

An error criterion can be found to indicate the mismatch of model and picture. An error matrix E is found,

$$E = AH - DB \text{ or } E = GDB \quad \text{" } [13]$$

The reader should notice that the Roberts technique matches sets of points, not polygons. The connectedness of the points is not considered in the matching process; however, the Roberts algorithm does compute the error for each pair of points which are matched. Incomplete solutions are actually classes of solutions -- i.e. instead of a unique transformation H being the result of the algorithm, a class of transformations with some common components is the result.

Incorporation of the Roberts algorithm into a shape matcher would require certain modifications. The matcher would have to apply the algorithm to a 3D model and a 2D description, analyze the transformation matrix (H) and the error matrix (E) which are returned, and modify the perspective matrix (P) associated with H. Remember H is the product of two matrices, R an affine transformation and P a perspective transformation(H = RP). Such an algorithm would be an iterative hypothesize-and-test procedure. It would depend heavily on the error analysis and perspective information to determine the manner in which to move the center of projection and the view reference point in order to improve the

match. It is basically a large search problemand it is virtually impossible to move through such a large search space and converge upon a solution without a certain amount of heuristic guidance.

Both the Roberts and Bykat techniques require that there be precisely the same number of points/vertices in the two elements being compared. This is a severe restriction and any effective shape matcher would have to be able to match an entity to substructures of the other entity. Roberts deals with this issue by using features of the image object to restrict his system to portions of the model containing the proper number of points. His system deals only with trihedral polyhedra, so a few feature constraints suffice. For a broader class of objects a more general technique similar to spring-loaded template matching might be required.

Now let us consider how our representation might facilitate shape matching.

V.5 An Approach to Improved Shape Matching

This section contains a list of suggestions for improving the shape matching process. They should be construed only as suggestions, since experiments to prove their usefulness have not yet been completed.

Based on our representation an approach that we feel would lead to improved shape matching would include the following major steps:

(1) Curve fitting --

Cubic B-spline curves should be fitted to the digital curves which form region boundaries in the image. Fitting cubic B-splines amounts to finding the knot vector or vertex polygon which produces the best approximation to the digital curve.

(2) Region and Boundary merging/splitting --

Regions must be merged/split to eliminate superfluous regions caused by shadows or surface markings.

(3) 2D Shape descriptions --
Shape features (NT,AV2D,...) should be computed for the vertex polygons which were determined in step (1).
Shape features such as area, centroid, compactness should be computed for each region in the image. Also, if the region has some regular 2D shape such as a circle, rectangle, etc. that shape should be recorded.

(4) 2D Spatial Relationships --
The relative position of the centroids of adjacent regions should be computed.

(5) 2D Structural Descriptions --
Collections of adjacent regions which appear to be parts of the same object, determined on the basis of information other than shape such as color or texture; a structural description consisting of a graph of these region nodes and their relative positions should be prepared.

(6) Index and select candidate object models --
Index on boundary curve features (number of vertices, angular variability,...)
Index on region features (compactness, geometrical shape,...)
Index on structural relationship of adjacent regions (connectedness, relative position)

(7) Match the vertex polygons for the boundary curves of the surface patches of the object models against the boundary curves of the 2D regions using a modified Roberts approach

(8) Match the corner definitions of the patches against hypothesized corners points for 2D regions.

(9) Determine the object model with the best fit (minimum - error transformation) and attempt to improve the fit by a hypothesis-and-test algorithm. The hypotheses will consist of relative depth estimates for points along the region's boundary (dimensionality conversion) and the test will be a modified version of Bykat's algorithm for 3D polygons.

Obviously, two very critical phases of such a matching process are (1) the initial indexing on features for object model candidates, and (2) the basis for modifying relative depth estimates in step (9). The performance of the indexing mechanism depends upon the features which may be computed from the image, the features which are captured in the internal 3D representation, and the relationships between those features. Our representation allows many

more features than volumetric representations do and, in addition, many of the 3D shape features have natural 2D analogues which have been shown to be indicators of shape complexity. However, some work remains to be done on the correlation of 3D and 2D features. The basis for modifying relative depth estimates requires studying the effects of depth changes upon our collection of features so as to be able to define an objective function which is optimized as part of the algorithm. The optimization algorithm will have to achieve a balance between local and global optimization and this balance will vary depending upon the situation. For example, in certain cases a strong localized match may overide a moderate match over a larger area. As an alternative one might devise relaxation processes for accepting distributed, parallel, local contributions to a control process which varies the view parameters in order to improve the match.

We conclude this chapter with mention of two other significant issues relating representation and matching. The work of Hinton (Hinton 1979) demonstrates that a single object may have several structural descriptions, where a specific structural description is associated with a set of orientations in which the object may be recognized easily. For example, a cube might have one structural description consisting of six square planar surface patches with the appropriate attachment relations. Alternatively, it might consist of two square planar surface patches and a single closed four-sided piecewise planar surface patch.

We feel that an effective representation system will have to support multiple structural descriptions and a means of associating each structural description with the class of views for which it is most appropriate. To avoid consuming large quantities of storage, the system would need a means of transforming a detailed geometric representation in one structural description into a different detailed geometric representation in another structural description. Thus, the representation of a 3D object might consist of a distinguished structural description

(patches and attachment relations), a detailed geometric description of the patches, a set of standard views, and a set of ancillary structural descriptions.

The second issue concerns the amount of generality provided by the object recognition system. We wish to recognize classes of objects as well as specific objects. In order to recognize classes of objects, the representation must be parameterized in such a way as to capture the specific shapes of all the objects in the general class. Parameterization of an object consisting of a single surface patch is easily accomplished by specifying ranges for the shape featuees provided in the representation. The problems with parameterization are introduced when an object is composed of two or more parameterized parts and one part constrains the set of values for the parameters of the second part. Thus, parameterization requires a mechanism for specifying parameter constraints.

It is clear from this short exposition that there is still a great deal of work to be done in the area of shape matching. Some of the specific problems for future research are outlined in the next chapter.

CHAPTER VI

CONCLUSION

VI.1 Conclusions

The purpose of this chapter is to summarize the contributions of this thesis and to describe future research based upon this work.

The major contribution of this dissertation is the demonstration of the desirability and feasibility of surface-based 3D object representations for computer vision systems. As part of the demonstration we synthesized a particular 3D surface-based representation from existing technology in the fields of computer graphics, computer-aided geometric design, numerical analysis, functional approximation, and artificial intelligence. The most significant advantages of our representation are its broad scope and potential for matching.

Our representation marks a significant advance in scope for representation of 3D objects for computer vision systems. Both curved and polyhedral objects are captured in a single uniform representation. We take advantage of the multiple knot feature of cubic B-splines and the identification mapping to achieve piecewise planar surfaces and closed polyhedra as well as cylindrical and conical closed surfaces. An object whose thickness is neglible in determining its shape may be naturally represented as a surface instead of awkwardly represented as a volume or collection of volumes with infinitesimal thickness.

The scope of the representation is enhanced by the possible types of attachment relations. Two surfaces may be joined smoothly (curvature continuous) along a common edge or

discontinuously along a common edge. Two complex surface patches may be joined at a common sub-patch component in a surface-to-surface fashion. The flexibility of the attachment relations and the variety of shapes describable by a simple patch open up great possilities for multiple structural descriptions of the same object. We feel multiple structural descriptions will be essential to the success of a general object recognition system.

The storage costs of our representation are highly economical in light of the broad scope and our representation admits the direct computation of surface area and surface normals. These features may be useful for incorporating illumination information into the matching process.

Our representation permits the hierarchical structuring of objects from parts. These parts are constructed from surface patches which are defined in terms of corner definition matrices boundary curves, and tangent vector curves. Various shape features are computed for each of these primitive components. Many objects may share a part or primiive component; thus, indexing on primitive features leads to all parts and objects possessing components with those features. All of the shape features are invariant under isogonal affine transformation and many have 2D analogues proven to be significant in the determination of shape similarity and complexity by humans.

Finally, the representation has tremendous potential for matching. All primitive components of the representation have underlying polygonal descriptions (vertex polygons, space quadrilateral of corner definition matrix) and there are existing algorithm for determining the similarity of polygons.

Another, but less significant, contribution of this work is the development of an algorithm for curve fitting 2D cubic B-splines to 4-connected digital curves. The algorithm detects

discontiniuties and produces knot vectors possessing multi-knots at the discontinuities. The knot vectors are inverted to produce vertex polygons which are analyzed for shape characteristics.

The final major contribution of this thesis is the description of a general approach to the problem of matching 3D object descriptions against 2D descriptions extracted from image data. Some of the more important problems and possible approaches to their solution are described in the next section.

V.2 Future Work

There are five significant problems whose solutions will have major impact on the matching problem for computer vision systems employing our representation. The first involves adapting the Bykat and Roberts algorithms to the hypothesize-and-test matching approach described in chapter IV. Error analysis and matching tolerance would have to be added to the Bykat algorithm; however, the more important problems are (1) understanding how to adjust the viewpoint in the Roberts algorithm based upon error analysis, (2) what to do with incomplete solutions in the Roberts algorithm, (3) how to generate depth hypotheses for the Bykat algorithm given the result of one run of the Roberts algorithm, and finally (4) how to match polygons to subsets of other polygons.

The second problem involves the uniqueness of representation for cubic B-splines. For a given cubic B-spline there may be many knot vectors which will generate the desired curve. In general one can add a knot between two existing knots without changing the shape of the curve by insuring that the new knot lies on the spline defined by the original knots. Current algorithms for inverting the knot vector to obtain the vertex polygon make no attempt to

achieve minimality in the . resulting number of vertices. Thus, it is possible to have two or more different vertex polygons for the same cubic B-spline curve. Obviously it is desirable to have a unique representation or at least a means of forming a canonical representation for various vertex polygons for the same curve.

One of the more well-defined problems is to perform a complete sensitivity analysis of changes in the values of 2D features (angular variation, compactness,...) associated with changes in view of a given patch possessing a specified set of 3D features. Correlations between the view, the 3D features-values, and the 2D feature values will provide the information for indexing into the 3D data base.

Next is the problem of parameterizing object description to achieve more generality in the data base. The mechanism for specifying the constraints on the parameter ranges must be well integrated with the hierarchical structure, so that constraint specification at any level in the object description must not be overly burdensome. In addition, the mechanism must be very efficient.

Finally, there is the problem of how to create and maintain multiple structural descriptions of a single object. Multiple structural descriptions are desirable because certain descriptions facilitate recognition from certain views. What is needed is a means to move quickly between different structural descriptions and their corresponding geometric representations

The solutions of these five problems will greatly contribute to the solution of the overall problem of object recognition in computer vision.

FOOTNOTES

[1] Hanson, A. and Riseman, E. VISIONS: a computer system for interpreting scenes, p.306. In Computer Vision Systems. A. Hanson and E. Riseman (eds.). p.303-334. Academic Press. New York. 1978.

[2] Attneave, F. Some informational aspects of visual perception. Psychological Review. 61:183-193. 1954.

[3]Ibid.,

[4] Attneave, F. and Arnoult, M. 1956. The quantitative study of shape and pattern perception.

[5] Marr, D. and Nishihara, H. Representation and recognition of the spatial organization of three dimensional shapes, p.15. MIT Artificial Intelligence Laboratory. Memo 377. August 1976.

[6] Nevatia, R. Structured descriptions of complex curved objects for recognition and visual memory, p. 21. October 1974. Stanford AI Project. AIM-250. Stanford University. Stanford, Ca.

[7] Marr, D. and Nishihara, H. Representation and recognition of the spatial organization of three dimensional shapes, p. 11. MIT Artificial Intelligence Laboratory. Memo 377. August 1976.

[8] Agin, G. J. 1972. Representation and description of curved objects, p. 22. Ph.D. Dissertation, Computer Science Department, Stanford University, October 1972.

Stanford, California.

[9] Rogers, D. F. and Adams, J. A. Mathematical Elements of Computer Graphics. p. 175. McGraw-Hill, New York. 1976.

[10]Ibid., p.175

[11] Bykat, A. On polygon similarity. p.24-25. Information Processing Letters. 9(1):23-25. July 1979.

[12] Roberts, L. G. Machine perception of three-dimensional solids. p. 195-196. In Optical and Electro-optical Information Processing. p. 159-197. J.T. Tippett (ed.). MIT Press. Cambridge, Massachusetts.

[13]Ibid., p. 196.

Bibliography

Agin, G. J. 1972. Representation and description of curved objects. Ph.D. Dissertation, Computer Science Department, Stanford University, October 1972. Stanford, California.

Ahlberg, Nilsson, and Walsh. 1967. Theory of Splines and Their Application. Academic Press. New York. 1967.

Aki, S. G. and Toussaint, G. T. 1978. An improved algorithm to check for polygon similarity. Information Processing Letters 7(3):127-8.

Arbib, M. 1975. Artificial intelligence and brain theory: unities and diversities. Annals of Biomed. Eng. 3:238-274.

Attneave, F. 1954. Some informational aspects of visual perception. Psychological Review. 61:183-193.

_____. 1957. Physical determinants of the judged complexity of shape. J. Exp. Psychol. 53:221-227.

Attneave, F. and Arnoult, M. 1956. The quantitative study of shape and pattern perception. Psychol. Bull. 53:452-457.

Baker, H. 1976. Building models of three dimensional objects. M. Phil. Department of Machine Intelligence, University of Edinburgh. Edinburgh, U.K.

Barrow, H. G. and Burstall, R. M. 1976. Subgraph isomorphism, matching relational structures and maximal cliques. Information Processing Letters 4:83-84.

Barrow, H. G. and Tenenbaum J. M. 1978. Recovering intrinsic scene characteristics from images. SRI International. Menlo Park, Ca.

Barrow, H. G., Tenenbaum, J. M., Bolles, R., and Wolf, H. 1977. Parametric correspondence and chamfer matching: two new techniques for image matching. Proc. of Fifth Intl. Joint Conf. on Artificial Intelligence. 659-664. MIT, Cambridge, Massachusetts.

Baudelaire, P., Flegal R. M., and Sproull R. F. 1977. Spline curve techniques. Technical report. Xerox Palo Alto Research Center. Palo Alto, California. May 1977.

Baumgart, B. G. 1972a. Winged edge polyhedron representation. Stanford AIM-179. Stanford Artificial Intelligence Laboratory. Stanford University.

_____. 1972b. GEOMED - A geometric editor. Stanford AIM-232. Stanford Artificial Intelligence Laboratory. Stanford University.

Bezier, P. 1972. Numerical-Control Mathematics and Applications. John Wiley and Sons. London.

Binford, T. O. 1971. Visual perception by computer. Invited paper at IEEE Systems, Science, and Cybernetics Conference. Miami, Florida. December 1971.

Blum, H. 1973. Biological shape and visual science(Part I). J. Theor. Biol. 38:205-287.

Braid, I. C. 1974. Designing with Volumes Cantab Press. Cambridge, England. 1974.

Brown, D. R. and Owen, D. H. 1967. The metrics of visual form. Psychol. Bull. 68:243-249.

Bykat, A. 1979. On polygon similarity. Information Processing Letters. 9(1):23-25.

Clowes, M. B. 1971a. On seeing things. Artificial Intelligence. 2(1):79-116.

Coons, S. A. 1967. Surfaces for computer-aided design of space forms. Department of Computer Science, M.I.T., Cambridge, Massachusetts. NTIS AD-663 504.

_____. 1974. Surface patches and B-spline curves. In Computer Aided Geometric Design. R. Barnhill and R. Riesenfeld (eds.). Academic Press. New York. 1974.

Davis, L. S. 1976a. Understanding shape, 2: symmetry. Technical report 441. Computer Science Center. University of Maryland. College Park, Maryland 20742.

_____. 1976b. Shape matching using relaxation techniques. Technical report 480. Computer Science Center. University of Maryland. Colege Park, Maryland 20742.

_____. 1977. Understanding shape: angles and sides. IEEE Trans. on Computers. C-26(3):236-243. March 1977.

Davis, L. S. and Rosenfeld, A. 1976a. Applications of relaxation labelling,2: spring-loaded template matching. Technical report 440. Computer Science Center. University of Maryland. College Park, Maryland 20742.

deBoor, C. 1979. A Practical Guide to Splines. Springer-Verlag. New York. 1979.

deBoor, C. and Rice, J. R. 1968a. Least squares cubic spline approximation I - fixed knots. Technical report 20. Computer Science Department. Purdue University. West Lafayette, Indiana.

_____. 1968b. Least squares cubic spline approximation II - variable knots. Technical report 21. Computer Science Department. Purdue University. West Lafayette, Indiana.

Duda, R. O. and Hart, P. E. 1973. Pattern Classification and Scene Analysis. John Wiley and Sons. New York. 1973.

Faux, I. D. and Pratt, M. J. 1979. Computational Geometry for Design and Manufacture. Ellis-Horwood limited. Chicester, West Sussex, England.

Feng, H. and Pavlidis, T. 1975. Decomposition of polygons into simpler components: feature generation for syntactic pattern recognition. IEEE Trans. on Computers. C-24(6):636-651. June 1975.

Fischler, M. A. and Elschlager, R. A. 1973. The representation and matching of pictorial structures. IEEE Trans. on Computers C-22(1): 67-93. January 1973.

Forrest, A. R. 1968. Curves and surfaces for computer-aided design. Ph.D. Dissertation. University of Cambridge. Cambridge, U. K.

Gibson, J. J. 1950. The Perception of the Visual World Houghton-Mifflin. Boston.

Gordon, W. and Riesenfeld, R. 1974. B-spline curves and surfaces. In Computer Aided Geometric Design. R. Barnhill and R. Riesenfeld (eds.). Academic Press. New York. 1974.

Grossman, D. D. 1976. Procedural representation of three dimensional objects. IBM Journal of Research and Development 20:582-589.

Guzman, A. 1968. Computer recognition of three-dimensional in a visual scene. MAC-TR-59. MIT Cambridge, Massachusetts.

Hanson, A. and Riseman, E. 1978a. Segmentation of natural scenes. In Computer Vision Systems. A. Hanson and E. Riseman (eds.). p. 129-164. Academic Press. New York. 1978.

Hanson, A. and Riseman, E. 1978b. VISIONS: a computer system for interpreting scenes. In Computer Vision Systems. A. Hanson and E. Riseman (eds.). p.303-334. Academic Press. New York. 1978.

Hanson, A., Riseman, E. and Glazer, F. 1980. Edge relaxation and boundary continuity. In Consistent Labelling Problems in Pattern Recognition. R. M. Haralick (ed.) Plenum Press.

Hinton, G. 1979. Some demonstrations of the effects of structural descriptions in mental imagery. Cognitive Science. 3:231-250.

Hollerbach, J. M. 1975. Hierarchical shape description of objects by selection and modification of prototypes. AI-TR-346. Artificial Intelligence Laboratory. M.I.T. Cambridge, Massachusetts

Horn, B. K. P. 1970. Shape from shading: a method for obtaining the shape of a smooth opaque object from one view. MAC-TR-79. MIT Cambridge, Massachusetts.

Huffman, D. 1971. Impossible objects as nonsense sentances. Machine Intelligence. 6:295-324.

Konolige, K., York, B., Hanson, A. and Riseman, E. 1977. Between regions and objects--surfaces and volumes. Proc. of Fifth Intl. Joint Conf. on Artificial Intelligence. 646-648. MIT, Cambridge, Massachusetts.

Lieberman, L., Grossman, D., Lavin, M., Lozano-Perez, T., Wesley, M., 1979. Three-dimensional modelling for automated mechanical assembly. Proc. of Workshop on representation of three-dimensional objects. Univ. of Pennsylvania. Philadelphia, Pa. May 1979.

Lowrance, J. D. 1978. Grasper 1.0 Reference Manual. COINS technical report 78-20. Dept. of Computer and Information Sciences. University of Massachusetts. Amherst, Ma. 01003.

Mackworth, A. K. 1973. Interpreting pictures of polyhedral scenes. Artificial Intelligence 4:121-137.

_____. 1976. Model-driven interpretation in intelligent vision systems. Technical report 76-2. Department of Computer Science. University of British Columbia. Vancouver, B.C. Canada.

Manacher, G. 1976. An application of pattern matching to a problem in geometrical complexity. Information Processing Letters. 5(1):6-7.

Marr, D. 1976. Analysis of occluding contour. MIT Artificial Intelligence Laboratory. Memo 372. October 1976.

_____. 1978. Representing visual information. In Computer Vision Systems. A. R. Hanson and E. M. Riseman (eds.) p.61-80. Academic Press. New York. 1978.

Marr, D. and Nishihara, H. 1975. Spatial disposition of axes in a generalized cylinder representation of objects that do not encompass the viewer. MIT Artificial Intelligence Laboratory. Memo 341. December 1975.

Marr, D. and Nishihara, H. 1976. Representation and recognition of the spatial organization of three dimensional shapes. MIT Artificial Intelligence Laboratory. Memo 377. August 1976.

Marr, D. and Nishihara, H. K. 1978. Representation and recognition of three dimensional shapes. Proc. R. Soc. Lond. B. 200,269-294.

Michels, K. M. and Zusne, L. 1965. Metrics of visual form. Psychol. Bull. 63:74-86.

Minsky, M. 1975. A framework for representing knowledge. In The Psychology of Computer Vision. Ed. P. H. Winston. McGraw-Hill. New York. 1975.

Montanari, U. 1970. A note on minimal length polygonal approximation of a digitized contour. CACM. 13:41-

Nevatia, R. 1974. Structured descriptions of complex curved objects for recognition and visual memory. Stanford AI Project. AIM-250. Stanford University. Stanford, Ca.

Newman, W. and Sproull, R. 1973. Principles of Interactive Computer Graphics. McGraw-Hill. New York.

O'Callaghan, J. 1972. A review of some computer methods for the recovery of shape features. In Pictorial Organization and Shape. p. 35-46. J. F. O'Callaghan (ed.). CSIRO Division of Computing Research. Canberra.

O'Rourke, J. and Badler, N. 1979. Decomposition of three-dimensional objects into spheres. IEEE Trans. on PAMI.

Parma, C., Hanson, A., and Riseman, E. 1980. Experiments in schema-driven interpretation of a natural scene. Proc. of NATO advanced study institute on digital image processing and analysis. Bonas, France. June 1980.

Pavlidis, T. 1978. A review of algorithms for shape analysis. Computer Graphics and Image Processing. 7(2):243-259. April 1978.

Pavlidis, T. and Ali, F. 1976. A general syntactic shape analyzer. Technical report 221. Department of Electrical Engineering and Computer Science. Princeton University. Princeton, New Jersey.

_____. 1979.A hierarchical syntactic shape analyzer. IEEE Trans. on PAMI. PAMI-1(1):2-10. January 1979.

Prager, J. M. 1979. Segmentation of static and dynamic scenes. Ph.D. dissertation. Dept. of Computer and Information Sciences. University of Massachusetts. Amherst, Ma. 01003.

Prager, J., Nagin, P., Kohler, R., Hanson, A. and Riseman, E. 1977. Segmentation processes in the VISIONS system. Proc. of Fifth Intl. Joint Conf. on Artificial Intelligence. 642-643. MIT, Cambridge, Massachusetts.

Reddy, D. R. and Rubin, S. 1978. Representation of three-dimensional objects. Technical report 113. Department of Computer Science. Carnegie-Mellon University. Pittsburgh, Pennsylvania.

Riseman, E. and Arbib, M. 1977. Computational Techniques in the visual segmentation of static scenes. Computer Graphics and Image Processing. 6:

Roberts, L. G. 1965. Machine perception of three-dimensional solids. In Optical and Electro-optical Information Processing. p. 159-197. J.T. Tippett (ed.). MIT Press. Cambridge, Massachusetts.

Rock, I. 1979. Form and orientation. Proc. of Workshop on representation of three-dimensional objects. Univ. of Pennsylvania. Philadelphia, Pa. May 1979.

Rogers, D. F. and Adams, J. A. 1976. Mathematical Elements of Computer Graphics. McGraw-Hill, New York. 1976.

Rosenfeld, A. 1969. Picture Processing by Computer. Academic Press. New York. 1969.

Schmidt, D, C, and Druffel, L. E. 1976. A fast backtracking algorithm to test directed graphs for isomorphism using distance matrices. JACM. 23(3):433-445.

Schudy, R. B. 1979. Spherical harmonic surface models. TR-46. Dept. of Computer Science. Univ. of Rochester. Rochester, N.Y.

Shapiro, L. and Haralick, R. 1979. Decomposition of two-dimensional shapes by graph-theoretic clustering. IEEE Trans. on PAMI. PAMI-1(1):10-20. January 1979.

Shneier, M. 1979. A compact relational structure representation. Proc. of Workshop on representation of three-dimensional objects. Univ. of Pennsylvania. Philadelphia, Pa. May 1979.

Shirai, Y. 1972. A heterarchical program for recognition of polyhedra. AIM 263. Artficial Intelligence Laboratory. M.I.T. Cambridge, Massachusetts.

_____. 1978. Recent advances in 3-D scene analysis, Proc. of Fourth Intl. Joint Conf. on Pattern Recognition. 1978, 86-94

Soroka, B. I. 1979. Understanding objects from slices: Extracting generalized cylinder descriptions from serial sections. TR-79-1. Department of Computer Science. University of Kansas. Lawrence, Kansas 66045.

Stevens, K. A. 1976. Occlusion cues and subjective contours. AIM 363. Artificial Intelligence Laboratory. M.I.T. Cambridge, Massachusetts.

_____. 1979. Surface perception from local analysis of texture and contour. Ph.D. Thesis. Dept. of Computer Science. MIT Cambridge, Massachusetts.

Tsuji, S. and Matsumoto, F. 1977. Detection of elliptic and linear edges by searching two parameter spaces. Proc. of Fifth Intl. Joint Conf. on Artificial Intelligence. 700-706. MIT, Cambridge, Massachusetts.

Turner, K. J. 1974. Computer perception of curved objects using a television camera. Ph.D. Dissertation. University of Edinburgh. Edinburgh, U. K.

Ullman, J. R. 1976. An algorithm for subgraph isomorphism. JACM. 23(1):31-42.

Waltz, D. L. 1975. Understanding line-drawings of scenes with shadows. In The Psychology of Computer Vision. Ed. P. H. Winston. McGraw-Hill. New York. 1975.

Williams, T., Lowrance, J., Hanson, A. and Riseman, R. 1977. Model-building in the VISIONS system. Proc. of Fifth Intl. Joint Conf. on Artificial Intelligence. 644-646. MIT, Cambridge, Massachusetts.

Winston, P. H. 1975. Learning structural descriptions from examples. In The Psychology of Computer Vision. Ed. P. H. Winston. McGraw-Hill. New York. 1975.

Woodham, R. 1977. A cooperative algorithm for determining surface orientation from a single view. Proc. of Fifth Intl. Joint Conf. on Artificial Intelligence. 635-642. MIT, Cambridge, Massachusetts.

Zahn, C. 1971. Graph-theoretical methods for detecting and describing Gestalt clusters. IEEE Trans. on Computers. C-20(1):68-86.

Zahn, C, and Roskies, R. 1972. Fourier descriptors for plane closed curves. IEEE Trans. on Computers. C-21(3):269-282. March 1972.

Zusne, L. 1965. Moments of area and of the perimeter of visual form as predictors of discrimination performance. J. Exp. Psychol. 69:213-220.

Zusne, L. 1971. Visual Perception of Form. Academic Press. New York. 1971.

APPENDIX I: SPLINES

The purpose of this appendix is to provide the reader with an overview of splines which is sufficient for understanding the body of the dissertation. Intuitive definitions and explanations are interspersed with their formal counterparts and appropriate pointers into the literature are provided. Spline applications and extensions to the theory are also mentioned.

Spline functions

A spline function (in one dimension) is a piecewise polynomial function defined over an interval $\{a,b\}$ which has been partitioned into non-overlapping subintervals (Figure &afig.). The polynomials which define the spline on each subinterval are all of the same degree and they satisfy certain continuity conditions at the boundaries of the subintervals. The partitioned interval, $a=x_0<x_1<...x_N=b$, is often called a mesh and the reason for such terminology is obvious when splines in two dimensions (Figure 64) are considered.

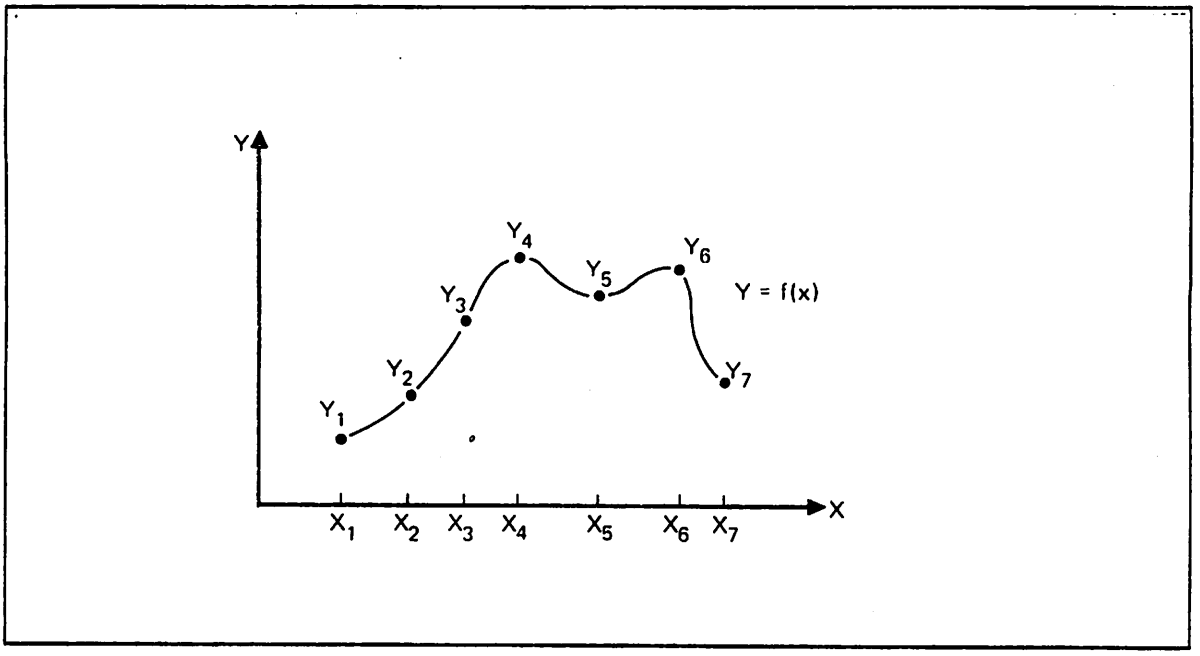

Figure 63: A cubic spline with seven knots

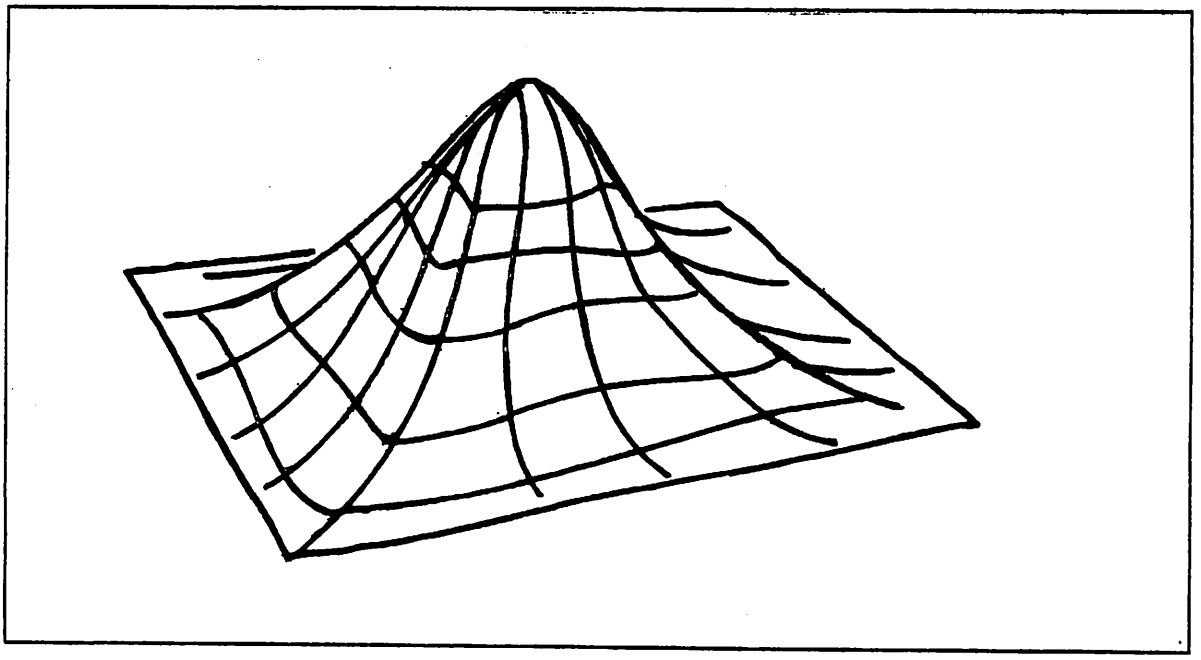

Figure 64: A cubic spline in two dimensions (a bicubic surface patch)

There are infinitely many spline functions defined over a given mesh and the spline interpolation problem may be succinctly stated: Given a set of values (y_i) $i=0,1,....,N$ which are values of some unknown function $f(x)$ at the mesh points (x_i), find the spline function $S(x)$ of a given degree such that $S(x_i)=y_i$ *for* $i=0,1,....,N$. $S(x)$ is, in a restricted sense, an approximation of $f(x)$.

Definition: Let Δ designate the mesh which partitions the interval $[a,b]$, such that $a=x_0<x_1<...<x_N=b$. A function S is a simple spline function of degree M-1 (order M) if it satisfies the following two conditions:

1. S is a polynomial of degree M-1 on every subinterval (x_i,x_i+1) $i=0,1,....,N-1$ of Δ.

2. S and its derivatives of orders 1,2,...,M-2 are everywhere continuous on Δ.

Thus, a simple spline function is a piecewise polynomial function with maximum smoothness -- i.e. there is at most a jump discontinuity in the (M-1)st derivative at a mesh point. Not all splines have maximum smoothness; those having less than maximum smoothness are sometimes referred to as "deficient splines."

When a jump discontinuity occurs in a derivative greater than the (M-k-1)st derivative at mesh point x_i *(where* $k>0$*)*, then the spline is said to have deficiency k at mesh point x_i. If the spline has deficiency k at all interior mesh points, then the spline is said to have deficiency k. Thus, simple splines have deficiency 1 and the degree N polynomial which interpolates the data has deficiency 0.

The ordered pair formed by combining a mesh point, x_i, with its associated value, y_i, is often called a knot and the ordered collection of these pairs, (x_i,y_i) $i=0,1,...,N$, is referred to as the knot vector. When a given knot appears more than once in the knot vector, a deficiency occurs in the spline of interpolation at that mesh point. The order of deficiency is equal to the number of occurrences of the knot.

Some properties of splines

Splines have many interesting and useful properties. In this section only a few of the more significant properties are presented.

Minimum curvature property

The minimum curvature property for cubic splines was established by Holladay(Holladay 1957). Holladay's theorem basically states that, given a knot vector, the simple cubic spline with natural end conditions is the smoothest function through those knots. Natural end conditions require the second derivative and hence the curvature to be zero at the endpoints of the interval. The formal statement of the theorem is as follows:

Holladay's theorem

Let the mesh Δ: $a = x_0<x_1<...<x_N = b$ and the set of real numbers (y_i) $(i=0,1,...,N)$ be given. Then of all functions $f(x)$ having a continuous second derivative on $[a,b]$ and such that $f(x_i)=y_i$ $(i=0,1,...,N)$, the spline function $S(\Delta;(y_i);x)$ with junction points at the x_i and with $S''(\Delta;(y_i);a) = S''(\Delta;(y_i);b) = 0$ minimizes the integral

$$\int_a^b |f''(x)|^2 dx$$

The proof of this theorem actually established an important result in spline theory which is referred to as the *first integral relation:*

$$\int_a^b |f''(x)|^2 = \int_a^b |S''(\Delta;f;x)|^2 dx + \int_a^b |f''(x)-S''(\Delta;f;x)|^2 dx$$

where $S(\Delta;f;x)$ denotes the spline interpolation to $f(x)$.

Because the second derivative is an approximation to curvature, this property has come to be known as the minimum curvature property of splines. The notation used above is a slight modification of the notation employed by Ahlberg (Ahlberg 1967). The minimum curvature property is a special case of the minimum norm property for more general splines -- i.e. for a given mesh and a given odd degree (M-1), the set of spline functions of degree M-1 forms a Hilbert space under the appropriate choice of norm.

It was not until the early 1960's (Ahlberg 1964) that the Hilbert space approach to spline theory began to develop. This approach allowed the easy formulation of the notion of "best approximation" and also a number of results on the convergence of spline approximations.

The class of functions with continuous second derivatives defined on the interval $[a,b]$ forms a Hilbert space $H^2[a,b]$ with inner product defined by :

$$\langle f,g \rangle = \int_a^b f''(x)g''(x)dx$$

The norm for this Hilbert space is $\|f\| = \langle f,f \rangle^{\frac{1}{2}}$ Ahlberg et al. (Ahlberg 1964) were able to find an orthonormal basis for this Hilbert space which consisted entirely of cubic spline functions. Thus, every function, $f(x)$, in $H^2[a,b]$ has an expansion in terms of such a basis. Let $B_i(x)$ denote the basis functions and let $f(x)$ denote an arbitrary function in $H^2[a,b]$. For

any positive integer N, a *best approximation* , g, formed from a linear combination of the first N basis functions is one which minimizes the quantity $\|f-g\|$ where $g(x) = \sum_{i=1}^{4} a_i B_i(x)$. The function , g, which is the best approximation, is the function whose coefficients are precisely the first N coefficients of the expansion of $f(x)$ in the complete basis.

There is an important result due to Walsh et al. (Walsh 1962) with respect to best approximation with periodic cubic splines. It provides an alternative characterization of the notion of best approximation for smooth periodic functions.

Theorem: Given a mesh Δ: $a=x_0<x_1<\ldots<x_N=b$, then of all the simple periodic cubic splines on Δ, the spline that interpolates to a periodic function $f(x)$ at the mesh points furnishes the best approximation in the preceding sense.

There are a number of results dealing with convergence of spline approximations under certain conditions. The convergence problem is simply stated:
Do the spline approximations $S^{(\alpha)}(\Delta;f;x)$ converge (and if so, how rapidly) as the mesh norm $\|\Delta\| = max\ |x_j+1 - x_j|$ approaches zero (where f is some underlying function defined on Δ).

Results

(1) For $f(x)$ *in* $C^2[a,b]$, it was shown that for cubic splines of interpolation to $f(x)$ at the mesh points, $S^{(\alpha)}(\Delta;f;x)$ converges uniformly to $f^{(\alpha)}(x)$ for $\alpha=0,1$.

(2) For $f(x)$ *in* $C^2[a,b]$. $S''(\Delta;f;x)$ converges uniformly to $f''(x)$ as $\|\Delta\|$ approaches zero.

(3) For polynomial splines of degree 2n-1, convergence of derivatives through order n-1 can be proven with the aid of the *second integral relation:*

$$\int_a^b \{f^{(n)}(x) - S^{(n)}(\Delta;f;x)\}^2 dx = \int_a^b \{f(x) - S(\Delta;f;x)\} f^{(2n)}(x) dx.$$

Convergence results have been established for $f(x)$ in $C[a,b]$ and $C^1[a,b]$ as well.

In the next section the defining equations for cubic splines are presented in order to provide the reader with an understanding of the computational complexity of spline interpolation.

Defining equations

Given a mesh Δ: $a=x_0<x_1<\ldots<x_N=b$ and an associated set of values (y_i) $i=0,1,\ldots,N$, how is the simple cubic spline which interpolates the points (x_i,y_i) $i=0,1,\ldots,N$ found? Since the spline is required to be simple, its second derivative must be continuous everywhere in Δ. Thus, at each interior subinterval boundary the one-sided limits of the second derivatives of the cubic polynomials, which define the spline on adjacent subintervals, must be equal -- i.e. for every x_j $j=1,2,\ldots,N-1$ it must be true that

$$S(\Delta;f;x_j^+) = S(\Delta;f;x_j^-).$$

When the expressions for the second derivatives are equated, a set of simultaneous equations is obtained. These equations are called the defining equations and they may be represented in matrix form:

$$\begin{vmatrix} 2 & \lambda_0 & 0 & \ldots & 0 & 0 & 0 \\ \mu_1 & 2 & \lambda_1 & \ldots & 0 & 0 & 0 \\ 0 & \mu_2 & 2 & \ldots & 0 & 0 & 0 \\ \cdot & \cdot & \cdot & \ldots & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \ldots & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \ldots & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \ldots & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & \ldots & 2 & \lambda_{N-2} & 0 \\ 0 & 0 & 0 & \ldots & \mu_{N-1} & 2 & \lambda_{N-1} \\ 0 & 0 & 0 & \ldots & 0 & \mu_N & 2 \end{vmatrix} = \begin{vmatrix} M_0 \\ M_1 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ M_{N-2} \\ M_{N-1} \\ M_N \end{vmatrix} \quad \begin{vmatrix} d_0 \\ d_1 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ d_{N-2} \\ d_{N-1} \\ d_n \end{vmatrix}$$

where

$$\mu_j = 1 - \lambda_j, \qquad \lambda_j = h_j/(h_j+h_{j+1})$$
$$\text{and } h_j = x_{j+1} - x_j$$

and

$$d_j = 6([(y_{j+1}-y_j)/h_{j+1}]- [(y_j-y_{j-1})/h_j])/(h_j+h_{j+1}).$$

are all known quantities. The defining equations are solved by matrix inversion techniques. The best inversion techniques are of $O(N^{2.81})$ complexity.

For the nonperiodic spline there are N-1 equations in N+1 unknowns. For a solution two of the unknown values must be specified (M_0 and M_N). They are called the end conditions and the case where $M_0 = M_N = 0$ is referred to as "natural end conditions." For the periodic cubic spline there are N equations in N unknowns (Ahlberg 1967,p. 12), so no additional values need be specified. In both the periodic and nonperiodic cases the defining equations may be recast in terms of the first derivatives (slopes) of the spline at the mesh points. The two formulations are equivalent in the sense that each may be easily derived from the other.

B-splines

"B-spline" is a term coined by Schoenberg (Schoenberg 1967) to describe a concept first introduced by Curry and Schoenberg (Curry 1947). The B-splines of order k are special spline functions of order k for a given mesh, such that every spline function of order k defined on that mesh may be written as a linear combination of these *basis splines* or B-splines. B-splines provide an alternative means of representing the spline functions defined earlier and there are certain computational advantages to computing with B-splines (deBoor 1979,p. 129-52).

B-spline representation is an alternative form in which all piecewise polynomial functions may be represented. This includes splines with simple knots and splines with multiple knots. In B-spline form a piecewise polynomial function $S(x)$ is represented as a linear combination of basis functions.

$$S(x) = \sum_{i=1}^{n} a_j B_j(x)$$

where a_j are real numbers and $B_j(x)$ is the jth B-spline basis function. Now let us be a bit more precise.

Given a knot vector $\{(x_0,y_0),(x_1,y_1),\ldots,(x_n,y_n)\}$ such that $x_0<x_1<\ldots<x_n$, we will denote by X the vector of x-values $X = \{x_0,x_1,\ldots,x_n\}$ and we call X the *mesh*. The set of all spline functions of order M (for cubic splines, M=4) defined over a mesh X forms a linear space of functions which we shall denote by S(M,X). This space has dimension n (the number of intervals in the mesh) and there are n basis functions of degree M-1 for this space. These basis functions are referred to as the "B-spline basis" of S(M,X). $B_{i,M}(x)$ will be used to denote the order M B-spline basis function associated with mesh point x_i. Before giving formulas for computing the B-spline basis function, a few definitions are required to aid the presentation.

Definition 1: The restriction of a spline function S(x) to the subinterval (x_i,x_{i+1}) is called the ith span.

Definition 2: The support of a function is the interval over which it has non-zero values.

Every B-spline basis function has a finite support of M spans -- i.e. the interval $(x_i,x_{i+M \bmod n})$. B-spline basis functions may be defined in various way; however, the most computationally useful definition is due to Cox and deBoor(in deBoor 1979, p. 121).

Recursive Definition of B-spline Basis functions

$$B_{i,1}(x) = \begin{cases} 1 & if x_i \le x \le x_{i+1} \\ 0 & otherwise \end{cases}$$

$$B_{i,M} = \frac{x-x_i}{x_{i+M-1}-x_i} B_{i,M-1}(x) + \frac{x_{i+M}-x}{x_{i+M}-x_{i+1}} B_{i+1,M-1}(x)$$

for M>1

Figure 65 shows the graph of the canonical uniform cubic B-spline basis function for the mesh $X = \{0,1,2,3,4\}$. For the periodic case, successive B-spline basis functions are merely cyclic translates of the canonical basis function. For the non-periodic case, the end conditions create problems. End conditions manifest themselves as multiple mesh values at the ends of the interval. Thus the basis functions at the ends of the interval differ in shape from the interior basis functions. Figure 66 contains a graph of the non-periodic uniform cubic B-spline basis functions for the mesh $X = \{0,0,0,1,2,3,4,5,6,6,6,\}$.

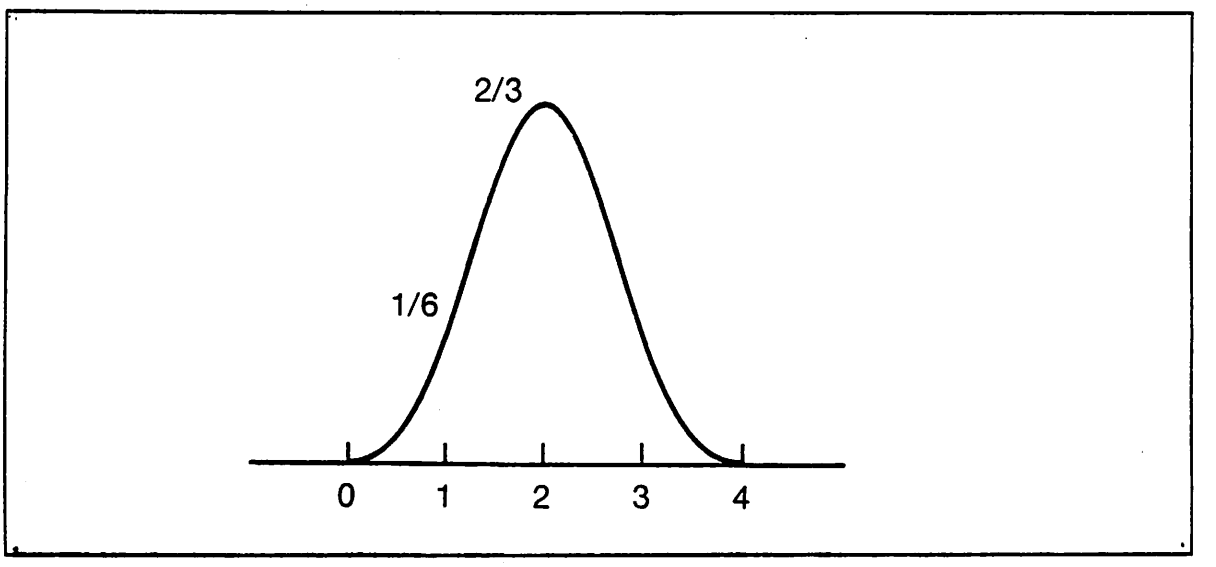

Figure 65: Canonical uniform cubic B-spline basis function

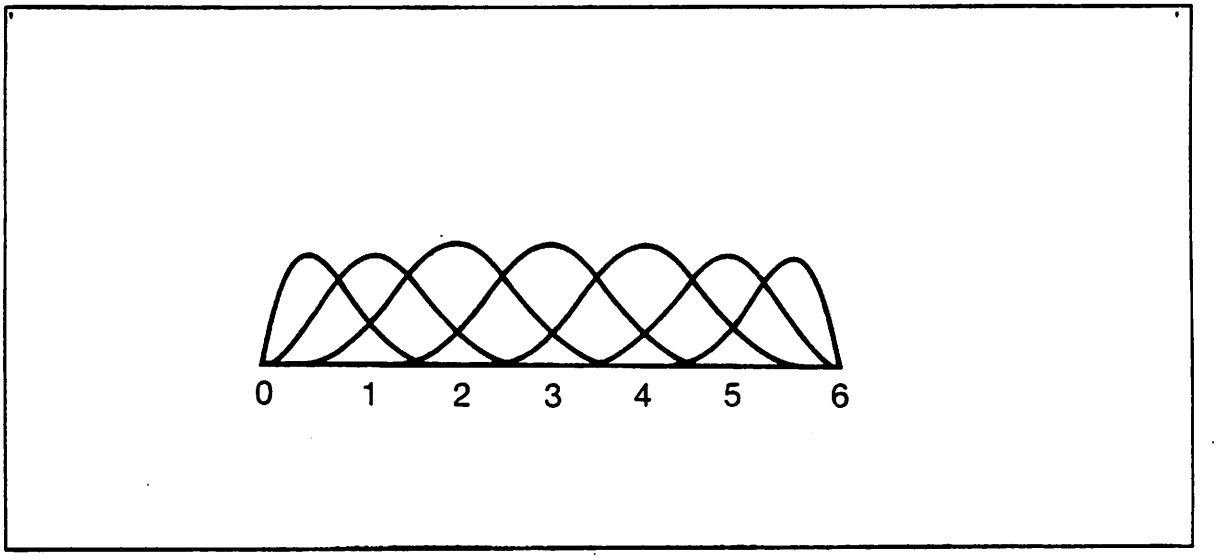

Figure 66: Non-periodic cubic B-spline basis functions

Cox and deBoor developed a stable algorithm for evaluating B-splines and their derivatives. It is based on the recursive definition presented above.

The representation just presented allows an arbitrary piecewise-polynomial function, $f(x)$, of order M to be written as a linear combination of B-spline basis functions:

$$f(x) = \sum_{i=1}^{n} a_i B_{i,M}(x)$$

The reader may wonder how this representation relates to the matrix formula in Chapter II and to the defining polygon. Well, any uniform cubic spline may be represented as a weighted sum of the following four polynomial functions:

$$\begin{aligned}
E_0(t) &= -1/6t^3 + 1/2t^2 - 1/2t + 1/6 \\
E_1(t) &= -1/2t^3 - t^2 + 2/3 \\
E_2(t) &= -1/2t^3 + 1/2t^2 + 1/2t + 1/6 \\
E_3(t) &= 1/6t^3
\end{aligned}$$

Thus, the value of an arbitrary cubic B-spline function $S(t)$ for some parameter $0 \le t \le 1$ is:

$$S(t) = E_0(t)V_i + E_1(t)V_{i+1} + E_2V_{i+2} + E_3V_{i+3}$$

where V_i, V_{i+1}, V_{i+2}, and V_{i+3} are consecutive vertices of the defining polygon for the cubic B-spline $S(t)$. Viewed in this way, the coeficients of the polynomial functions E_0, E_1, E_2, and E_3 are just the columns of the matrix in th following formula:

$$P(u) = [s^3 \quad s^2 \quad s \quad 1]\ 1/6 \begin{vmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{vmatrix} \begin{vmatrix} V_i \\ V_{i+1} \\ V_{i+2} \\ V_{i+3} \end{vmatrix}$$

where $V_i = (x_i, y_i, z_i)$
n = *the number of vertices* -3
i = *the integer part of* $n-u$
s = *the fractional part of* $n \times u$

There exist algorithms for deriving the vertices of the defining polygon from the knot vector. This process is called inversion and it is not always possible to invert a knot vector. deBoor(1979) elaborates on the difficulties.

B-spline curve interpolation derives from B-spline function interpolation in the obvious way -- parameterization. Obvious choices for the parameterization scheme are uniform and cumulative chord length. Cumulative chord length is preferrable to uniform in terms of approximation accuracy; however, it requires a good deal more computation. We have used uniform parameterization throughout our work, because the approximations achieved are adequate for the data under investigation.

Extensions to the theory

The matrix representation of the defining equations may be compactly written in terms of the linear operator D^{2n} where $D = d/dx$. Now a polynomial spline of degree $2n-1$ (M=2n) must satisfy the equation $D^{2n}S = 0$ in each mesh interval (the fourth derivative of a cubic polynomial must be zero). There are two important extensions to the theory of splines which are easily expressed by linear operators. The first is the notion of *generalized splines*. Let the operator L be defined as:

$$L = a_n(x)D^n + a_{n-1}(x)D^{n-1} + \ldots + a_0(x)$$

and let L^* denote the formal adjoint of L. The spline S which satisfies the equation $L^*LS=0$ is called a generalized spline. An example of a class of generalized splines is the set of trigonometric splines studied by Schoenberg (Schoenberg 1964).

The second extension is called "splines in tension." Under this scheme the operator L becomes $L = D(D-\sigma)$ where σ is some real number. Splines in tension concentrate the curvature near the junction points and suppress points of inflection which are not indicated by the data (Ahlberg 1967).

Applications

Despite their fairly recent introduction, mathematical splines have already been usefully applied in many areas of science and engineering. They are extensively used in the computer-aided geometric design of airplanes, ships, and automobiles (Bezier 1972). In addition spline

techniques are employed by numerical analysts to solve problems in curve fitting, interpolation, numerical integration and differentiation, and numerical solution of integral and differential equations.

Ahlberg (Ahlberg 1967) provides a good theoretical treatment of splines while deBoor (deBoor 1979) offers a very practical treatment. A short introduction for non-mathematicians is provided by York (York 1979).

Appendix II: Coons Surfaces

This appendix provides a basic understanding of the notions involved in surface patch interpolation and it presents some of the more commonly used patch forms.

There are many different ways in which one may mathematically represent a surface patch. In this appendix a variety of patch forms are presented which are generally classified in the category of "interpolation over rectangles." The term derives from the use of four points (corners of a rectangle) in a plane to delimit the ranges of values for the arguments of the vector-valued function, P(u,w), which describes the surface (Figure 67). Since all rectangles are homeomorphic to the unit square, the unit square has been chosen as the canonical rectangle over which all surface patch interpolations are defined.

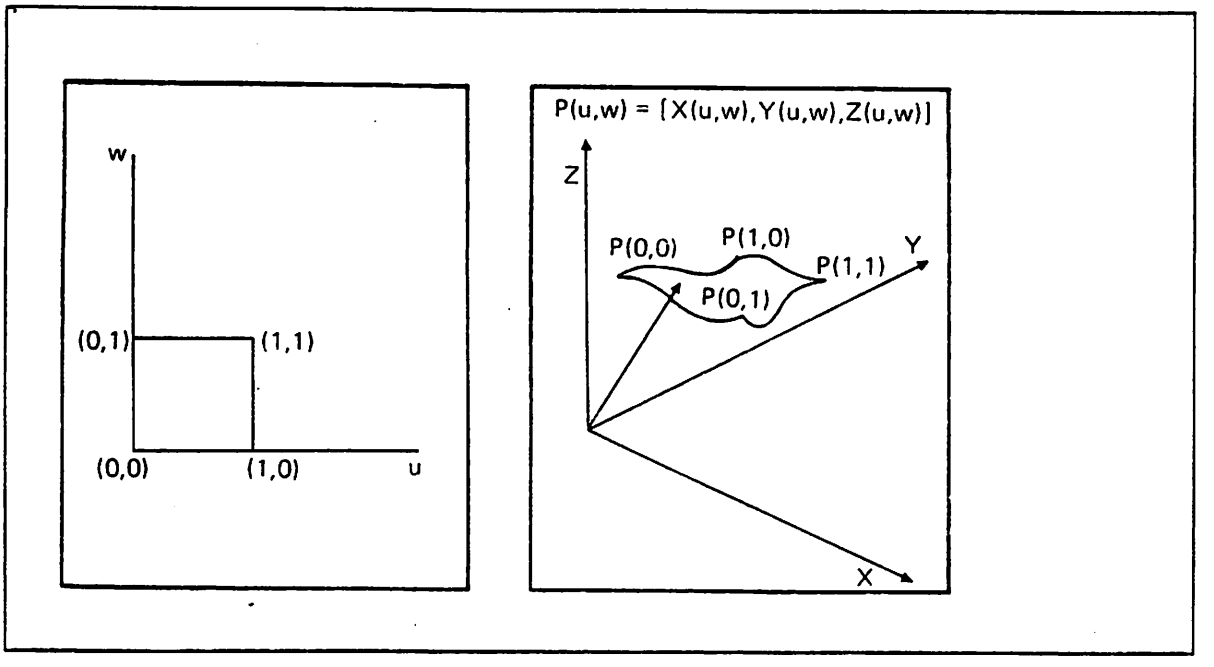

Figure 67. Surface patch over unit square.

In the descriptions to follow, each surface patch is described as a vector-valued function, P(u,w), of two independent parametric variables, u and w. Each variable ranges over the real

interval [0,1]; thus the surface patch is defined over the unit square [0,1] x [0,1] in the u-w plane. P(u,w) is a point vector, (X(u,w),Y(u,w),Z(u,w)), where X,Y, and Z are bivariate functions over the unit square. Parametric representation has the advantage of axis-independence, thus avoiding the problems of infinite slope values that may occur when evaluating a function with respect to a particular Cartesian coordinate system.

A. Bilinear surfaces

One of the least complex forms is the bilinear patch. In this form the x-y-z coordinates of the four corner points of the patch (P(0,0), P(0,1), P(1,0), P(1,1)) are specified and all other points on the surface patch are obtained via the bilinear interpolation formula,

$$P(u,w) = P(0,0)(1-u)(1-w) + P(0,1)(1-u)w + P(1,0)u(1-w) + P(1,1)uw. \quad (1)$$

This equation is of the general form

$$P(u,w) = P_1F_0(u)F_0(w) + P_2F_0(u)F_1(w) + P_3F_1(u)F_0(w) + P_4F_1(u)F_1(w), \quad (2)$$

where F_0 and F_1 are real-valued functions defined on the interval [0,1] and satisfying the relationship $F_0 + F_1 = 1$ and where P_1, P_2, P_3, P_4 are points in 3-space.

The interpolation equation (1) is merely a weighted sum of the four point vectors which determine the corners of the patch. When the four corner points are coplanar, a planar patch is defined; however, when they are not coplanar, they form a tetrahedron which bounds the volume in which the patch must lie.

In order to better understand the significance of the bilinear interpolation equation, consider the tetrahedron defined by the four corner points in Figure 68(a). Let us translate the tetrahedron so that the point P_1 is at the origin, and then label the edges as in Figure 68(b). The interpolation equation can be rewritten as a vector equation which describes every point on the bilinear surface patch as a linear combination of the three vectors, A, B, and C-(A+B). Notice that the sum of the coefficients of the corner points in the bilinear interpolation equation is unity.

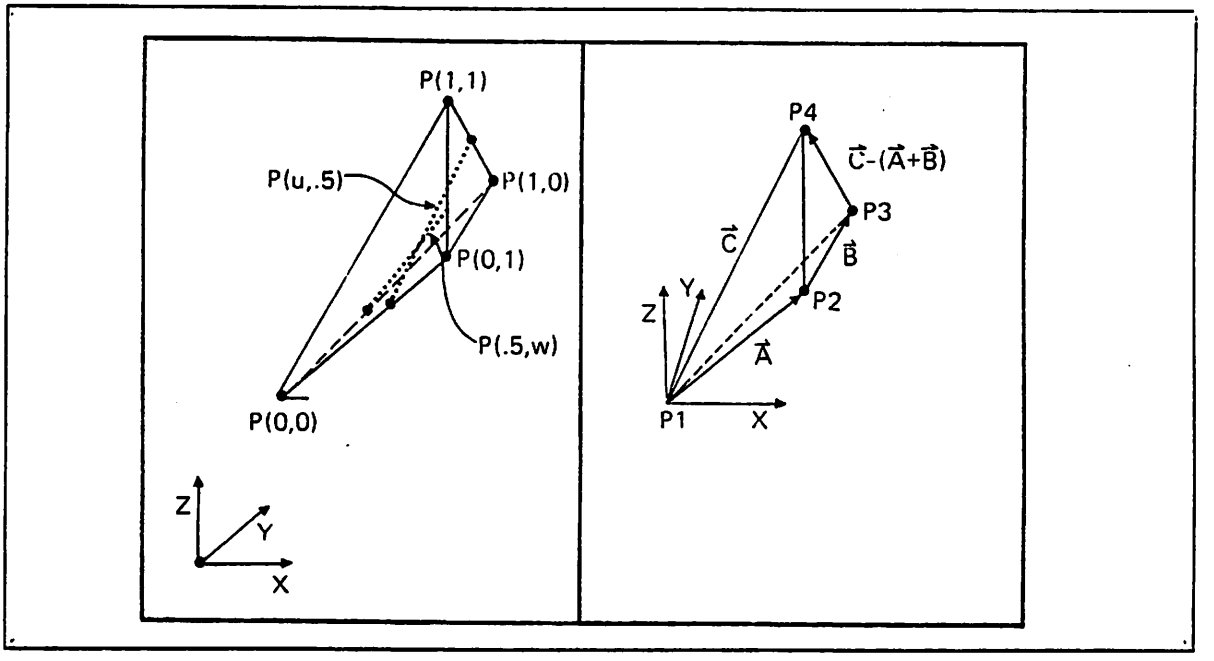

Figure 68. (a) Bilinear patch with enclosing tetrahedron

(b) Bilinear patch as a linear combination of vectors for defining tetrahedron

Thus, the translation to the origin can be effected by subtracting P_1 from both sides of equation (2):

$$P(u,w) = P_1(1-u)(1-w) + P_2(1-u)w + P_3u(1-w) + P_4uw$$

$$-P_1 = -(P_1(1-u)(1-w) + P_1(1-u)w + P_1u(1-u) + P_1uw)$$

gives

$$P(u,w)-P_1 = 0 + (P_2-P_1)(1-u)w + (P_3-P_1)u(1-w) + (P_4-P_1)uw.$$

which can be rewritten

$$P(u,w) - P_1 = (1-u)wA + u(1-w)B + uwC \qquad (3)$$

Expanding this vector equation gives,

$$P(u,w) - P_1 = wA + uB + uw(C - (A+B)) \qquad (4)$$

It is apparent from this equation that every point on a bilinear surface patch is a linear combination of the three vectors A,B and C-(A+B) and, since all the coefficients are less than unity, the surface patch must be contained within the tetrahedron formed by the corner points. The sum of the first two terms, wA+uB, provides a vector in the plane defined by A and B. The third term provides an elevation above the plane defined by A and B in the direction of C-(A+B). Alternatively, one may consider the vectors A, B, and C-(A+B) to be a non-orthonormal basis for the tetrahedron.

The class of surface patch shapes which may be represented by a bilinear patch form has the characteristic that every constant-parameter curve in the surface patch must be a straight line. Thus, when u is fixed at some constant value, say a, and w varies, then P(a,w) traces out a straight line in the surface. The only surfaces which may be represented in this form are those surfaces for which the slope is constant along any constant-parameter curve in either direction. Thus, it is not possible to represent a surface containing a hill or valley in bilinear patch form.

The bilinear patch equation may be rewritten in matrix form:

$$P(u,w) = \begin{bmatrix}(1-u) & u\end{bmatrix} \begin{vmatrix} P(0,0) & P(0,1) \\ P(1,0) & P(1,1) \end{vmatrix} \begin{vmatrix} 1-w \\ w \end{vmatrix} \qquad (5)$$

Matrix form is useful for compactly expressing a summation of products. The elements of the matrix are vectors in 3-space, thus the matrix is actually a tensor.

B. Lofted surfaces

Lofted surfaces are the simplest surfaces which are interpolants of opposing boundary curves. The general equation for a lofted surface is:

$$P(u,w) = P(u,0)(1-w) + P(u,1)w \qquad (6)$$

In equation (6) P(u,0) represents the boundary curve of the surface patch where w=0 and u is allowed to vary from 0 to 1. P(u,1) is the opposite boundary curve.

The lofted surface may be thought of as a generalization of the bilinear surface, in which two of the boundary curves are no longer constrained to be straight lines. The relaxation of this constraint allows a greater variety of surface shapes to be represented. The equation requires the slope of the surface to be constant in only one direction - i.e. all constant-parameter curves in the w-direction are straight lines, while constant-parameter curves in the u-direction may have slopes which vary non-linearly depending upon the shapes of the boundary curves P(u,0) and P(u,1). Since P(u,0) and P(u,1) are not required to be linear, the lofted patch is not constrained to lie within the tetrahedron of corner points (see Figure 69).

C. Linear Coons surfaces

A straightforward extension of the lofted surface is obtained by providing both pairs of opposing boundary curves, computing the lofted surfaces associated with each pair, and adding the two lofted surfaces together. When the two lofted surfaces are added, each corner point is counted twice in the sum. Thus, the sum surface does not go through the corners of the patch -- a violation of the interpolation constraint. The corner points must be subtracted from the

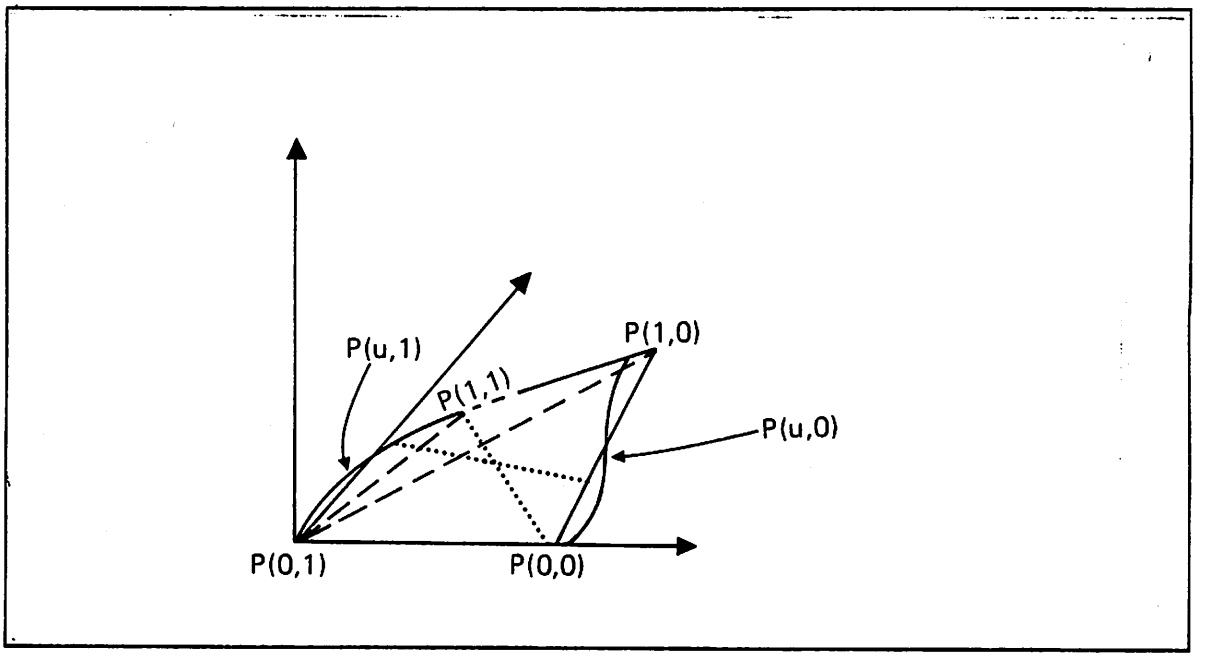

Figure 69. Lofted patch with associated tetrahedron.

sum to satisfy this condition and the patch form which results is sometimes referred to as the linear Coons surface patch. The name is appropriate because this form is merely a special case of the generalized Coons form as are the previous forms. The linear Coons surface patch equation in matrix form is:

$$P(u,w) = [(1-u) \quad u \quad 1] \begin{vmatrix} -P(0,0) & -P(0,1) & P(0,w) \\ -P(1,0) & -P(1,1) & P(1,w) \\ P(u,0) & P(u,1) & 0 \end{vmatrix} \begin{vmatrix} 1-w \\ w \\ 1 \end{vmatrix} \qquad (7)$$

The slope at an interior point P(u,w) of the surface in the u-direction is a linear combination of the slopes of the boundary curves P(u,0) and P(u,1), evaluated at the proper value of u. Simlarly, the slope in the w-direction is a linear combination of the slopes of the boundary curves P(0,w) and P(1,w) evaluated at the proper value of w. Since all four boundary curves may be of arbitrary shape, the slopes in the u and w directions may vary non-linearly.

Initially, the linear Coons patch appears to be quite a flexible patch form for surface design; however, it contains certain deficiencies. The first and most noticeable deficiency is that only linear interpolation is allowed. In the Coons generalized equation the interpolation weight functions are allowed to be functions of arbitrary complexity and they are called

blending functions. Arbitrary blending functions provide the designer with increased control over the shape of the surface, since they determine the weighting of the boundary points which are used in the calculation of every interior point P(u,w).

The second deficiency is the inability to independently control the slope of the surface. In all the preceding forms, once the boundary curves and blending functions are specified, the interior shape of the surface is completely determined; however, it is often the case that a designer is fairly content with the general shape of a given surface, but would like to change its slope slightly.

In order to solve this problem Coons developed the notion of a slope correction surface. He considered a surface interpolation to be a sum of two interpolants. The first component or intrinsic surface looks like a linear Coons patch in which the blending functions may be non-linear. The second component or slope correction surface also looks like a linear Coons surface with non-linear blending functions; however, the boundary curves are actually surface tangent vector curves. These curves describe the slope of the surface at the patch boundaries. They are also described as vector-valued functions of a single parametric variable. In his dissertation Coons (Coons 1967) extended the notion to higher order correction surfaces; however, we have not considered them in this dissertation. For our purposes the generalized Coons Patch form with slope correction surface will be sufficient.

E. Generalized Coons surface patch

Before the generalized Coons surface patch equation is described, certain notational conventions due to Coons (Coons 1967) must be presented. Just as matrix notation was introduced to reduce the complexity of multiple summation notation, so too Coons introduced a shorthand for representing the points, curves, blending functions and derivatives involved in the Coons patch equation. The notation P(u,w) for representing the vector-valued function of two variables is shortened to uw. As a result the boundary curves may be abbreviated in a similar manner (see Figure 70).

P(0,w) is represented by 0w

P(1,w) is represented by 1w

P(u,0) is represented by u0

P(u,1) is represented by u1

As far as the blending functions are concerned, the parentheses are removed to get the following notation:

$$F_0(u) = F_0u$$
$$F_1(u) = F_1u$$

and similarly for G_0 and G_1.

Now, for each of the boundary curves, the tangent to the surface in the direction of the parameter which crosses the boundary curve is the partial derivative of the surface in that direction. For example, u0 is a boundary curve in the u direction, or alternatively it may be thought of as the constant parameter curve where the parameter w is equal to the constant, zero. The slope of the surface (tangent vector) at this boundary is a vector-valued function of a single parametric variable, u. It is the partial derivative of uw with respect to w, evaluated at w=0. Thus, we get the notation:

$$u0_w = \frac{\partial(uw)}{\partial w}\Big|_{w=0} \qquad u1_w = \frac{\partial(uw)}{\partial w}\Big|_{w=1}$$

$$0w_u = \frac{\partial(uw)}{\partial u}\Big|_{u=0} \qquad 1w_u = \frac{\partial(uw)}{\partial u}\Big|_{u=1}$$

Finally, when the rate of change in the w-direction of the tangent vector curve in the u-direction is evaluated at the corners, the corner twist vectors are obtained. The corner twist vector at 00 is:

$$00_{uw} = \frac{\partial^2(uw)}{\partial u \partial w}\Big|_{\substack{u=0 \\ w=0}}$$

and similarly for 01_{uw}, 10_{uw}, and 11_{uw} With this shorthand it is now possible to express the

generalized Coons surface patch equation fairly succinctly. Figure 70 gives a pictorial description of the vector quantities involved in equation (8).

$$uw = [F_0u \quad F_1u \quad G_0u \quad G_1u] \begin{vmatrix} 0w \\ 1w \\ 0w_u \\ 1w_u \end{vmatrix} + [u0 \quad u1 \quad u0_w \quad u1_w] \begin{vmatrix} F_0w \\ F_1w \\ G_0w \\ G_1w \end{vmatrix} \qquad (8)$$

$$-[F_0u \quad F_1u \quad G_0u \quad G_1u] \begin{vmatrix} 00 & 01 & 00_w & 01_w \\ 10 & 11 & 10_w & 11_w \\ 00_u & 01_u & 00_{uw} & 01_{uw} \\ 10_u & 11_u & 10_{uw} & 11_{uw} \end{vmatrix} \begin{vmatrix} F_0w \\ F_1w \\ G_0w \\ G_1w \end{vmatrix}$$

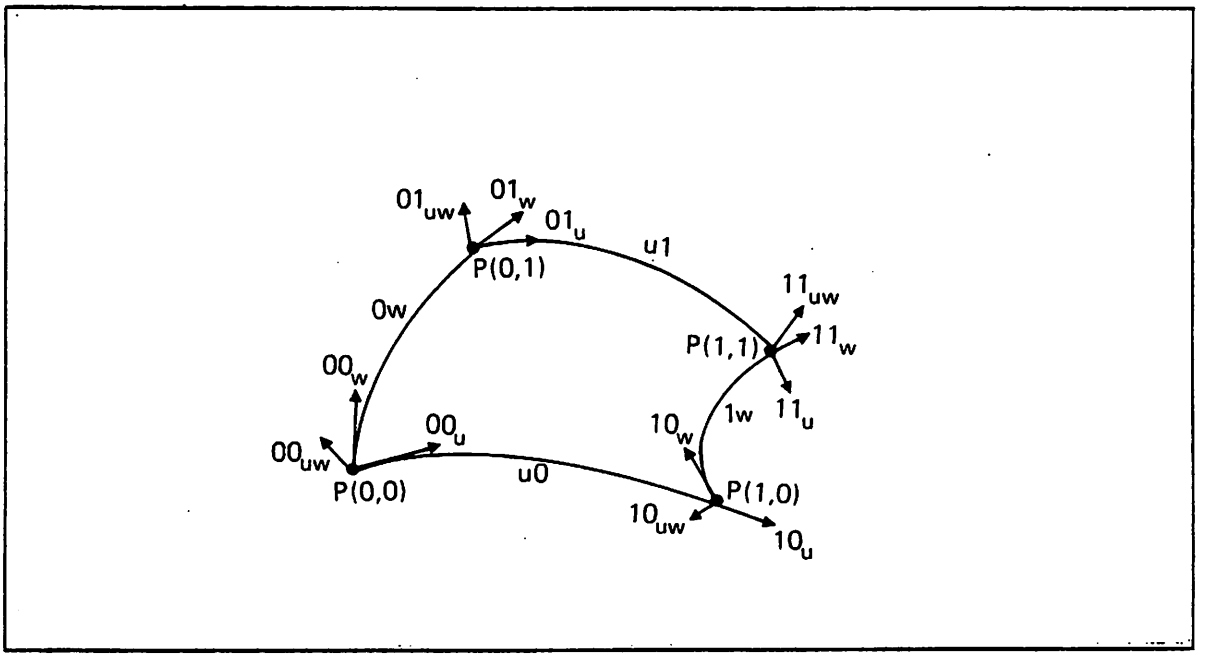

Figure 70. Generalized Coons surface patch.

An analysis of this equation shows the Coons patch to be the sum of an intrinsic surface and a slope correction surface. The blending functions F_0 and F_1 must satisfy the relationship $F_0 + F_1 = 1$ and they blend the intrinsic surface. Similarly the blending functions G_0 and G_1 blend the slope correction surface; however, they must satisfy two separate conditions:

(1) $G_0 + G_1 = 0$

(2) $G_0(0) = G_0(1) = G_1(0) = G_1(1) = 0.$

The first condition accounts for the fact that the opposing tangent vector curves are oriented in the same direction. One of them must be negated so that they both point toward

the interior of the surface patch. Both opposing tangent vectors are weighted equally in the interpolation of the slope correction surface. The second condition ensures that the slope correction surface leaves the boundary curves of the intrinsic surface intact - i.e. the surface which results from summing the intrinsic surface and the slope correction surface must still pass through the original boundary curves specified for the intrinsic surface. In order to meet this requirement, the boundary vectors of the slope correction surface must vanish.

When equation (8) is rewritten in the following form, it is clear that the Coons patch is the sum of an intrinsic surface and a slope correction surface.

$$
\begin{aligned}
uw = {} & [F_0u \quad F_1u] \begin{vmatrix} 0w \\ 1w \end{vmatrix} + [u0 \quad u1] \begin{vmatrix} F_0w \\ F_1w \end{vmatrix} \\
& -[F_0u \quad F_1u] \begin{vmatrix} 00 & 01 \\ 10 & 11 \end{vmatrix} \begin{vmatrix} G_0w \\ G_1w \end{vmatrix} \\
& +[G_0u \quad G_1u] \begin{vmatrix} 0w_u \\ 1w_u \end{vmatrix} + [u0_w \quad u1_w] \begin{vmatrix} G_0w \\ G_1w \end{vmatrix} \\
& -[G_0u \quad G_1u] \begin{vmatrix} 00_{uw} & 01_{uw} \\ 10_{uw} & 11_{uw} \end{vmatrix} \begin{vmatrix} G_0w \\ G_1w \end{vmatrix} \\
& -[G_0u \quad G_1u] \begin{vmatrix} 00_u & 01_u \\ 10_u & 11_u \end{vmatrix} \begin{vmatrix} F_0w \\ F_1w \end{vmatrix} \\
& -[F_0u \quad F_1u] \begin{vmatrix} 00_w & 01_w \\ 10_w & 11_w \end{vmatrix} \begin{vmatrix} G_0w \\ G_1w \end{vmatrix}
\end{aligned}
\tag{9}
$$

Notice that there are two additional factors in equation (9). These factors must be subtracted because the tangent vectors at the corners have been counted twice in the slope correction equation.

There is a restricted form of the Coons patch equation which does not involve the boundary tangent vector curves.

$$
\begin{aligned}
uw = {} & [F_0u \quad F_1u] \begin{vmatrix} 0w \\ 1w \end{vmatrix} + [u0 \quad u1] \begin{vmatrix} F_0w \\ F_1w \end{vmatrix} \\
& -[F_0u \quad F_1u] \begin{vmatrix} 00 & 01 \\ 10 & 11 \end{vmatrix} \begin{vmatrix} F_0w \\ F_1w \end{vmatrix} \\
& +[G_0u \quad G_1u] \begin{vmatrix} 00_{uw} & 01_{uw} \\ 10_{uw} & 11_{uw} \end{vmatrix} \begin{vmatrix} G_0w \\ G_1w \end{vmatrix}
\end{aligned}
\tag{10}
$$

This form assumes the boundary tangent vectors are zero and the intrinsic surface is augmented by an interpolation of the corner twist vectors. Despite the absence of boundary tangent vectors, a large variety of surface shapes are representable with the restricted Coons equation.

The generalized Coons surface patch has a very nice property which is not possessed by the other patch forms; it is referred to as the "smooth join" property. Under certain conditions two Coons patches may be joined along a common boundary in a "curvature continuous" fashion.

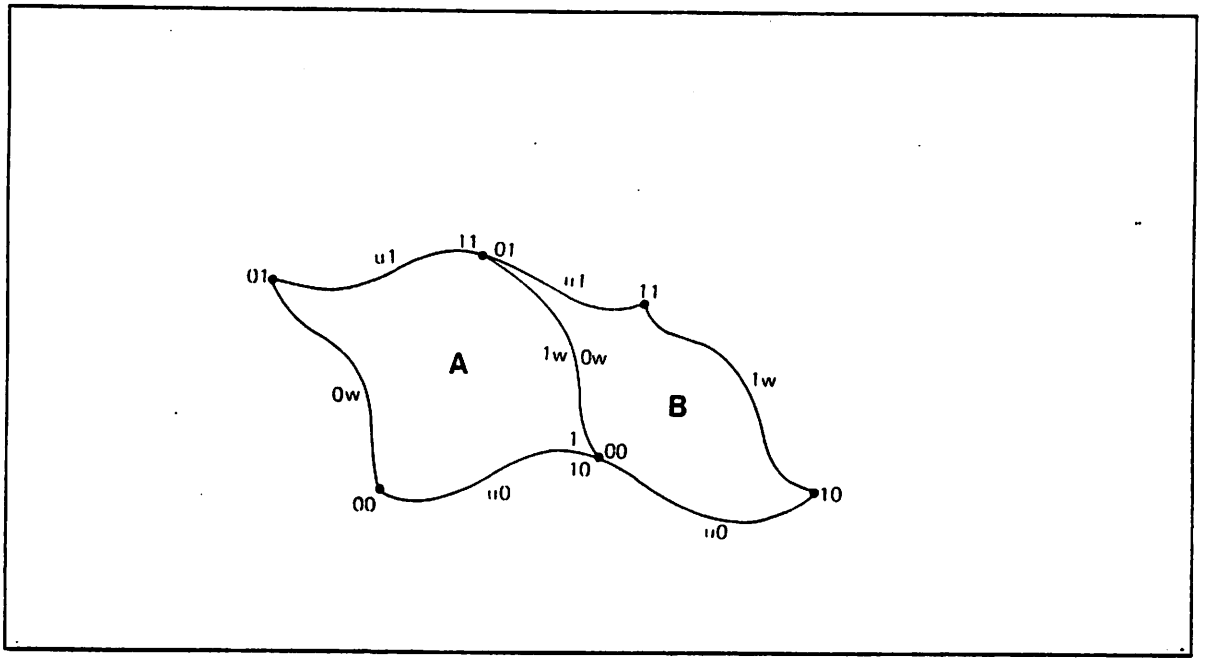

Figure 71: Smooth join of two Coons patches.

In order for the Coons patches A and B (Figure 71) to be smoothly joined along the common boundary A(1w)=B(0w), the following conditions must be satisfied:

(1) the blending functions F_0,F_1 must satisfy:

$F_0(0)=1$	$F_0'(0)=0$	$F_0''(0)=0$
$F_0(1)=0$	$F_0'(1)=0$	$F_0''(1)=0$
$F_1(0)=0$	$F_1'(0)=0$	$F_1''(0)=0$
$F_1(1)=1$	$F_1'(1)=0$	$F_1''(1)=0$

(2) the blending functions G_0, G_1 must satisfy:

$G_0(0)=0$	$G_0'(0)=1$	$G_0''(0)=0$
$G_0(1)=0$	$G_0'(1)=0$	$G_0''(1)=0$
$G_1(0)=0$	$G_1'(0)=0$	$G_1''(0)=0$
$G_1(1)=0$	$G_1'(1)=1$	$G_1''(1)=0$

This set of conditions can be written more compactly as a matrix equation,

$$\begin{vmatrix} F_ij & F_i'j & F_i''j \\ G_ij & G_i'j & G_i''j \end{vmatrix} = \begin{vmatrix} \delta_{ij} & 0 & 0 \\ 0 & \delta_{ij} & 0 \end{vmatrix}$$

where F_ij denotes $F_i(j)$ and i,j=0,1. The primes indicate differentiation with respect to the parametric variable and δ_{ij} denotes the "Kronecker delta",

$$\delta = \begin{cases} 0 & \text{if } i=j \\ 1 & \text{otherwise} \end{cases}$$

This condition is called the delta condition for obvious reasons and Figure 72 shows examples of functions which meet this condition.

(3) The boundary curves are continuous in slope in the proper parametric direction (in this case,the u-direction) at the ends of the contiguous boundary between A and B -- i.e.

$$A(10_u) = B(00_u)$$

and

$$A(11_u)=B(01_u)$$

When these two conditions are met, Coons (Coons 1967,p.11-13) has proved that the two patches A and B are joined in a curvature-continuous and hence slope-continuous fashion. Thus,

$$A(1w_u)=B(0w_u) \quad \text{(slope continuity)}$$
$$A(1w_{uu})=B(0w_{uu}) \quad \text{(curvature continuity)}$$

The smooth join property is a very powerful feature which is obtained at the small expense of an additional restriction on the blending functions.

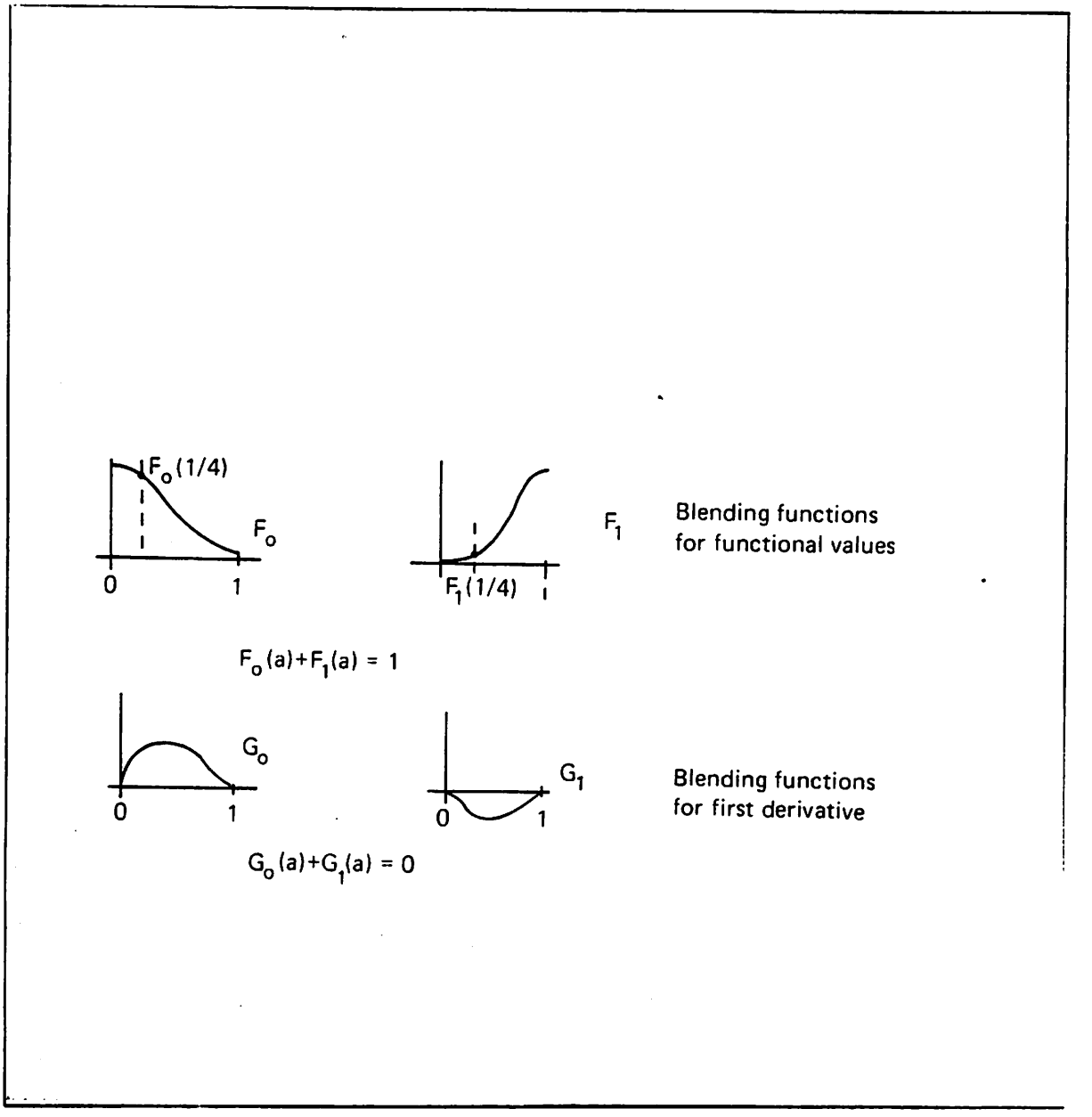

Figure 72. Blending functions for smooth join of Coons patches

Appendix III A SAMPLE SHAPE DATABASE

III.1 LTM NETWORK STRUCTURE

The network structure for shape knowledge in a small sample data base contained 9 spaces, 158 nodes, approximately 1500 in-pointing edges and 1500 out-pointing edges. A summary of the graph is presented in a set of tables, since the information can be presented in a more compact form.

III.1.1 Feature Spaces

Table 19 Curve-Features

NODE-NAMES	NODE-NAMES
ANG-VARIABILITY-3D	NO-TURNS
CLOSURE	NO-VERTICES
COMPACTNESS-C	NT-OVER-NV
MAX-ANG-TURN	PLANARITY
MAX-LEG-OVER-MIN-LEG	POLY-LEN-OVER-BASELINE
MULT-VERTICES	SMOOTHNESS
MV-OVER-NV	

Table 20 Patch-Features (V/S)

NODE-NAME	NODE-NAME
COMPACTNESS-P	VOL-SURF-AREA-EDGE
IDENT-MAP	PLANARITY-P
PVA-00-01	TVMR-00U-0010
PVA-00-10	TVMR-00W-0001
PVA-01-11	TVMR-01U-0111
PVA-10-11	TVMR-01W-001
PVMR-00-01	TVMR-10U-0010
PVMR-00-10	TVMR-10W-1011
PVMR-01-11	TVMR-11U-0111
PVMR-10-11	TVMR-11W-1011
TVA-00	TWVA-00
TVA-01	TWVA-01
TVA-10	TWVA-10
TVA-11	TWVA-11
TWVMR-00	
TWVMR-01	
TWVMR-10	
TWVMR-11	

III.1.2 Class Spaces

Table 21 Object Class

NODE-NAME	NODE-NAME
CORVETT HOOD	TABLE
DRINKING CUP	TELEPHONE-POLE
PENCIL	TELEPHONE-RECEIVER
RUNNING SHOE	WASTE-BASKET
SCREWDRIVER BLADE	TABLE-TOP
SCREWDRIVER HANDLE	TABLE-LEG
SPOON	

Table 22 Patch (V/S) Class

NODE-NAME
CYLINDER
RECTANGULAR SOLID
TETRAHEDRON
WEDGE

Table 23 Curve Class

NODE-NAME
CIRCLE
POINT
STRAIGHT-LINE
RECTANGLE
EQUILATERAL-TRIANGLE
ISOSCELES-RIGHT-TRIANGLE
CUBIC
N-GON

III.1.3 Instance Spaces

The object instance space contains nodes labelled O1 through O18. The patch instance space contains nodes labelled P1 through P17 and the curve instance space contains nodes labelled CU1 through CU50. Since this sample data base contains mostly one-patch objects, the connection structure can be captured in a single table.

Table 24 Partial Network Connection Structure

Patch-instance	u0	u1	0w	1w	object-instance	object-class
P1	CU1	CU2	CU3	CU4	O1	CORVETTE-HOOD
P2	CU5	CU6	CU7	CU7	O2	DRINKING-CUP
P3	CU8	CU9	CU10	CU10	O3	PENCIL
P4	CU11	CU12	CU13	CU13	O4	RUNNING-SHOE
P5	CU14	CU15	CU16	CU16	O5	SCREWDRIVER-BLADE
P6	CU17	CU18	CU19	CU19	O6	SCREWDRINER-HANDLE
P7	CU20	CU51	CU52	CU53	O7	SPOON
P8	CU21	CU22	CU23	CU23	O9	TELEPHONE-POLE
P9	CU24	CU25	CU26	CU26	O10	TELEPHONE-RECEIVER
P10	CU27	CU28	CU29	CU29	O11	WASTE-BASKET
P11	CU30	CU31	CU32	CU32	O12	TABLE-TOP
P12	CU33	CU34	CU35	CU35	O13	TABLE-LEG
P13	CU36	CU37	CU38	CU38	O14	TABLE-LEG
P14	CU39	CU40	CU41	CU41	O15	TABLE-LEG
P15	CU42	CU43	CU44	CU44	O16	TABLE-LEG
P16	CU45	CU46	CU47	CU47	O17	(TETRAHEDRON)
P17	CU48	CU49	CU50	CU50	O18	(WEDGE)

Note that O8 appears to be missing from Table 24. Actually O8 is an instance of the object, table. It consists of a table top and four table legs. Since O8 is a multi-patch object instance; it does not have a direct representation as a single patch. Tetrahedron and wedge appear in parenteses in Table 24 to indicate that these patch instances may be treated as abstract geometrical objects. Thus, the name of the class for the abstract object is the same as the name of the patch class.

One additional table will complete the picture of this portion of the graph. This table relates curve instances to curve classes.

Table 25 Curve Class/Instances Relationships

CIRCLE -	CU9,CU14,CU17,CU18,CU21, CU22,CU24,CU25
POINT -	CU8,CU46,CU52,CU53
STRAIGHT-LINE -	CU7,CU13,CU19,CU23,CU29,CU32,CU35, CU38,CU44,CU47,CU50
RECTANGLE -	CU15,CU30,CU31,CU33,CU34,CU36 CU37,CU39,CU40,CU42,CU43
EQUILATERAL-TRIANGLE -	CU45
ISOSCELES-RIGHT-TRIANGLE -	CU48,CU49
CUBIC -	CU1,CU2,CU3,CU4,CU5,CU6,CU11, CU12,CU20,CU51
N-GON(n>5) -	CU10,CU16,CU26,CU27,CU28

In the following pages selected portions of the GRASPER data structure for LTM are listed; the complete listing requires several hundred pages and is too voluminous to include here.

SPACES

<< CURVE-CLASS >>

<< CURVE-FEATURES >>

<< CURVE-INSTANCES >>

<< OBJECT-CLASS >>

<< OBJECT-INSTANCES >>

<< PATCH-FEATURES >>

<< PATCH-INSTANCES >>

<< PATCH-CLASS >>

Node CIRCLE in CURVE-CLASS

Edges

-- edge -> node indicates outpointing edge

<- edge -- node indicates inpointing edge

-- has-instance -> cu14

-- has-instance -> cu17

-- has-instance -> cu18

-- has-instance -> cu21

-- has-instance -> cu22

-- has-instance -> cu24

-- has-instance -> cu25

-- has-instance -> cu9

<- isa -- cu14

<- isa -- cu17

<- isa -- cu18

<- isa -- cu21

<- isa -- cu22

<- isa -- cu24

<- isa -- cu25

<- isa -- cu9

Node CU9 in CURVE-INSTANCES

Edeges

-- av-3d-> ang-variability-3d

value = (0.0,0)

-- comp-c -> compactness-c

value = 12.57

-- isa -> circle

-- maxat -> max-ang-turn

value = 144

-- maxl/minl -> max-leg-over-min-leg

value = 1

-- mv/nv -> mv-over-nv

value = 0

-- mv -> mult-vertices

value = 0

-- nt/nv -> nt-over-nv

value = 1

-- nt -> no-turns

value = 10

-- nv -> no-vertices

value = 10

-- oc -> closure

value = 1

-- pgl/bl -> poly-len-over-baseline

value = 0

-- pl -> planarity

value = 0

-- sm -> smoothness

value = 0

-- u1 -> p3

<- has-instance -- circle

<- av-3d -- ang-variability-3d

value = (0 0 0)

<- comp-c -- compactness-c

value = 12.57

<- maxat -- max-ang-turn

value = 144

<- maxl/minl -- max-leg-over-min-leg

value = 1

<- mn/nv -- mv-over-nv

value = 0

<- mv -- mult-vertices

value = 0

<- nt/nv -- nt-over-nv

value = 1

<- nt -- no-turns

value = 10

<- nv -- no-vertices

value = 10

<- oc -- closure

value = 0

<- pgl/bl== poly-len-over-baseline

value = 0

<- pl -- planarity

value = 0

<- sm -- smoothness

value = 0

<- u1 -- p3

Node O10 in OBJECT-INSTANCES

Edges

-- has-patch -> P9

-- isa -> telephone-receiver

<- has-instance -- telephone-receiver

<- part-of -- P9

Node P9 in PATCH-INSTANCES

Edges

-- 0w -> cu26

-- 1w -> cu26

-comp-p -> compactness-p

-im -> ident-map

-- part-of -> O10

-- pl-p -> planarity-p

-- pva0001 -> pva-00-01

-- pva0010 -> pva-00-10

-- pva0111 -> pva-01-11

-- pva1011 -> pva-10-11

-- pvmmr0001 -> pvmr-00-01

-- pvmr0010 -> pvmr-00-10

-- pvmr0111 -> pvmr-01-11

-- pvmr1011 -> pvmr-10-11

-- tva00 -> tva-00

-- tva01 -> tva-01

-- tva10 -> tva-10

-- tva11 -> tva-11

-- tvmr00u0010 -> tvmr-00u-0010

-- tvmr00w0001 -> tvmr-00w-0001

-- tvmr01u0111 -> tvmr-01u-0111

-- tvmr01w0001 -> tvmr-01w-0001

-- tvmr10u0010 -> tvmr-10u-0010

-- tvmr10w1011 -> tvmr-10w-1011

-- tvmr11u0111 -> tvmr-11u-0111

-- tvmr11w1011 -> tvmr-11w-1011

-- twva00 -> twva-00

-- twva01 -> twva-01

-- twva10 -> twva-10

-- twva11 -> twva-11

-- twvmr00 -> twvmr-00

-- twvmr01 -> twvmr-01

-- twvmr10 -> twvmr-10

-- twvmr11 -> twvmr-11

-- u0 -> cu24

-- u1 -> cu25

-- v/sae -> vol-surf-area-edge

<- 0w -- cu26

<- 1w -- cu26

<- comp-p -- compactness-p

<- has-patch -- 010

<- im -- ident-map

<- pl-p -- planarity-p

<- pva0001 -- pva-00-01

<- pva0010 -- pva-00-10

<- pva0111 -- pva-01-11

<- pva1011 -- pva-10-11

<- pvmr0001 -- pvmr-00-01

<- pvmr0010 -- pvmr-00-10

<- pvmr0111 -- pvmr-01-11

<- pvmr1011 -- pvmr-10-11

<- tva00 -- tva-00

<- tva01 -- tva-01

<- tva10 -- tva-10

<- tva11 -- tva-11

<- tvmr00u0010 -- tvmr-00u-0010

<- tvmr00w0001 -- tvmr-00w-0001

<- tvmr01u0111 -- tvmr-01u-0111

<- tvmr01w0001 -- tvmr-01w-0001

<- tvmr10u0010 -- tvmr-10u-0010

<- tvmr10w1011 -- tvmr-10w-1011

<- tvmr11u0111 -- tvmr-11u-0111

<- tvmr11w1011 -- tvmr-11w-1011

<- twva00 -- twva-00

<- twva01 -- twva-01

<- twva10 -- twva-10

<- twva11 -- twva-11

<- twvmr00 -- twvmr-00

<- twvmr01 -- twvmr-01

<- twvmr10 -- twvmr-10

<- twvmr11 -- twvmr-11

<- u0 -- cu24

<- u1 -- cu25

<- v/sae -- vol-surf-area-edge

Query 5

Feature	Relation	Value
CLOSURE	=	0

Comments:

This query returns all open curves in the data base as well as all patches and objects in which those curves participate. Since every object instance in the data base contains a patch with at least one open curve in its boundary, all object instances in the data base satisfy this query.

Results:

Curves:
(STRAIGHT-LINE (CU7 CU13 CU16 CU19 CU23
CU26 CU29 CU32 CU35 CU38
CU41 CU44 CU47 CU50))
(N-GON (CU10))

Curve-instances to Patch-intances:
(P1 (CU1 CU2 CU3 CU4))
(P2 (CU7))
(P3 (CU10))
(P4 (CU13))
(P5 (CU16))
(P6 (CU19))
(P8 (CU23))
(P9 (CU26))
(P10 (CU29))
(P11 (CU32))
(P12 (CU35))
(P13 (CU38))
(P14 (CU41))
(P15 (CU44))
(P16 (CU47))
(P17 (CU50))

Patch-instances to object instances and class
(PENCIL (O3 (P3)))
(RUNNING-SHOE (O4 (P4)))
(SCREWDRIVER-BLADE (O5 (P5)))
(SCREWDRIVER-HANDLE (O6 (P6)))
(TELEPHONE-POLE (O9 (P8)))
(TELEPHONE-RECEIVER (O10 (P9)))
(WASTE-BASKET (O11 (P10)))
(TABLE-TOP (O12 (P11)))
(TABLE-LEG (O13 (P12) O14 (P13) O15 (P14) O16 (P15)))
(TETRAHEDRON (O17 (P16)))
(WEDGE (O18 (P17)))

Query 6

Feature	Relation	Value
PLANARITY	=	0
SMOOTHNESS	=	0

Comments:

This query retrieves all of the smooth, planar curves from the database as well as all patches and objects in which they participate. For our current purposes, points are considered to be smooth and planar. Circles are obviously smooth and planar. CU3 and CU4 are the open, smooth, planar curves at the front and back of the patch representing the corvette hood. CU11 and CU12 are the closed, smooth, planar curves representing the sole and the neck of the running shoe. Straight line segments are not considered planar, because an infinite number of planes pass through a given straight line.

Results:

Curves:
(CIRCLE (CU9 CU14 CU17 CU18 CU21
CU22 CU24 CU25))
(POINT (CU8 CU46))
(CUBIC (CU3 CU4 CU11 CU12))

Curve-instances to Patch-intances:
(P1 (CU3 CU4))
(P3 (CU8 CU9))
(P4 (CU11 CU12))
(P5 (CU14))
(P6 (CU17 CU18))
(P8 (CU21 CU22))
(P9 (CU24 CU25))
(P16 (CU46))

Patch-instances to object instances and class
(CORVETTE-HOOD (O1 (P1)))
(PENCIL (O3 (P3)))
(RUNNING-SHOE (O4 (P4)))
(SCREWDRIVER-BLADE (O5 (P5)))
(SCREWDRIVER-HANDLE (O6 (P60)))
(TELEPHONE-POLE (O9 (P8)))
(TELEHPONE-RECEIVER (O11 (P9)))
(TETRAHEDRON (O17 (P16)))

Query 7

Feature	Relation	Value
PLANARITY	=	0
SMOOTHNESS	=	1

Comments:

This query retrieves all of the planar, non-smooth curves from the data base, as well as all of the patches and objects in which they participate.

Results:

Curves:
(RECTANGLE (CU15 CU30 CU31 CU33 CU34
CU36 CU37 CU39 CU40 CU42 CU43))
(EQUILATERAL-TRIANGLE (CU45))
(ISOSCELES-RIGHT-TRIANGLE (CU48 CU49))
(N-GON (CU10 CU26 CU27 CU28))

Curve-instances to Patch-intances:
(P3 (CU10))
(P5 (CU15))
(P9 (CU26))
(P10 (CU27 CU28))
(P11 (CU30 CU31))
(P12 (CU33 CU34))
(P13 (CU36 CU37))
(P14 (CU39 CU40))
(P15 (CU42 CU43))
(P16 (CU45))
(P17 (CU48 CU49))

Patch-instances to object instances and class
(PENCIL (O3 (P3)))
(SCREWDRIVER-BLADE (O5 (P5)))
(TELEPHONE-RECEIVER (O10 (P9)))
(WASTE-BASKET (O11 (P10)))
(TABLE-TOP (O12 (P11)))
(TABLE-LEG (O13 (P12) O14 (P13)
O15 (P14) O16 (P15)))
(TETRAHEDRON (O17 (P16)))
(WEDGE (O18 (P17)))

Query 8

Feature	Relation	Value
CLOSURE	=	1
COMPACTNESS-C	<=	20

Comments:

This query retrieves all of the closed curves with a compactness value less than 20, resulting in the retrieval of all the circles and rectangles in the data base.

Results:

Curves:
(CIRCLE (CU9 CU14 CU17 CU18 CU21
CU22 CU24 CU25))
(RECTANGLE (CU15 CU30 CU31 CU33 CU34
CU36 CU37 CU38 CU39 CU40 CU42 CU43))

Curve-instances to Patch-intances:
(P3 (CU9))
(P5 (CU14 CU15))
(P6 (CU17 CU18))
(P8 (CU21 CU22))
(P9 (CU24 CU25))
(P11 (CU30 CU31))
(P12 (CU33 CU34))
(P13 (CU36 CU37))
(P14 (CU39 CU40))
(P15 (CU42 CU43))

Patch-instances to object instances and class
(PENCIL (O3 (P3)))
(SCREWDRIVER-BLADE (O5 (P5)))
(SCREWDRIVER-HANDLE (O6 (P6)))
(TELEPHONE-POLE (O9 (P8)))
(TELEPHONE-RECEIVER (O10 (P(9)))
(TABLE-TOP (O12 (P11)))
(TABLE-LEG (O13 (P12) O14 (P13)
O15 (P14) O16 (P15)))