A MODIFIED KANADE MODEL FOR THREE-DIMENSIONAL OBJECT RECOGNITION

by

HENRY CHI-MING LAU, B.S. in E.E.

A THESIS

IN

INDUSTRIAL ENGINEERING

Submitted to the Graduate Faculty
of Texas Tech University in
Partial Fulfillment of
the Requirements for
the Degree of

MASTER OF SCIENCE

IN

INDUSTRIAL ENGINEERING

Approved

May 1990

ACKNOWLEDGMENTS

I would like to sincerely thank Dr. Roger G. Ford, the chairman of my committee, for his advice, guidance, patience, encouragement, and time.

I would like to give special thanks to both Dr. William J. Oldham and Dr. Milton L. Smith, for their ideas, patience, time, and constructive criticism.

I would like to dedicate this thesis to my father, my late mother, and my brother for their moral support, inspiration, and motivation which led to the completion of this research.

TABLE OF CONTENTS

ABSTRACT

The purpose of this research is to apply a modified Kanade model on visual information for three-dimensional object recognition. Kanade proposed a shape-recovery method to recover the three-dimensional shape of an object from a single view projected onto a two-dimensional picture plane. However, the labeling procedure of the search for consistent interpretation of objects was exhaustive. With objects that have more than one layer, the labeling procedure became complicated and required large memory spaces to store data information. In this paper, a simplified method is suggested to solve the above problem. Images of block samples from different views are taken. Edge detection is performed via the Sobel operator with smoothing performance. The data is then analyzed by the six-bit coding method and the parallelogram surface search method. Constructive solid geometry combines the information of the data and constructs the final image. The final sample can be displayed and simulated as if viewed at different locations and orientations. The two models are compared by the use of the same object for the three-dimensional object recognition. The results are that the modified Kanade model is more efficient in computer memory requirements saving and in the speed of three-dimensional object recognition.

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Problem Definition

The interpretation of line drawings was first introduced by Guzman (1968). He made the important observation that different junctions in a line drawing suggested possible associations of regions into objects. His heuristics were only related to the two-dimensional image domain and made no explicit treatment of three-dimensional space features (Cohen and Feigenbaum 1982). A more systematic approach was taken by Huffman (1971) and Clowes (1971) separately who emphasized the important distinction between the scene domain and the image domain. Then, Mackworth (1973) interpreted line drawings as three-dimensional scenes by reasoning about surface orientations based on the properties of the gradient space introduced by Huffman (1971). Using gradients, Kanade (1981) demonstrated multiple interpretation and quantitative shape recovery from line drawings. The labeling procedure tests the consistency surface orientations by using the gradient space constraints. Complicated labeling procedures are needed from those interpretations of line drawings. A large amount of computer memory is required to store the information of the

images. In this paper, a simplified algorithm is introduced for three-dimensional object recognition of trihedral vertices for solid objects. The algorithm requires less computer memory and simplifies the labeling procedure.

Problem Motivation

In this thesis, a simplified algorithm is designed for three-dimensional object recognition of trihedral vertices for solid objects. The algorithm includes the edge detection by using the Sobel operator with smoothing performance to reduce noise, a new six-bit coding method to save data in bit format, a new parallelogram surface search method to search for consistent parallelogram planar surface around selected vertices, and uses constructive solid geometry to construct the final object. The object is displayed in different locations and orientations with the properties of translation, rotation, zoom, and clipping. The parallelogram surface search method is based on the two heuristics developed by Kanade. A comparison between the Kanade model and the modified Kanade model in three-dimensional object recognition is also given.

Summary

In this project, a modified Kanade model has been developed for three-dimensional object recognition for trihedral vertices of solid blocks. This model simplifies the procedure of line labelings and is more dependent on the information provided by sensors. The six-bit code method saves the data of the edges around vertices in bit format without the use of large computer memory. The parallelogram search method searches the parallelogram surface with the selected vertices to obtain block information. Therefore, the modified Kanade model facilitates three-dimensional object recognition and reduces the computer memory requirement.

In Chapter II, background information on vision and artificial intelligence, the Kanade model for the three-dimensional object recognition, edge detection by using the Sobel operator, and the mathematical model for three-dimensional object viewing are presented. In Chapter III the problem is described in detail. In Chapter IV the modified Kanade model to solve the problems discussed in Chapter III is introduced. In Chapter V the algorithm is developed. In Chapter VI the results are analyzed and compared to the earlier model, and in Chapter VII the thesis is concluded with recommendations for further research.

CHAPTER II

BACKGROUND INFORMATION

In this chapter, the background information of the Kanade's cube model, the Sobel operator for edge detection, and the mathematical model for three-dimensional object displays as used in this thesis are discussed.

Vision and Artificial Intelligence

Vision is the information-processing task of understanding a scene from its projected images. An image is a two-dimensional function f(x,y), obtained with a sensing device, that records the value of an image feature at all points (x,y). Values might be binary for black-or-white images, gray level (intensity) for half-tone images, or vectors of color measures for color images. Images are converted into digital form for processing with a computer. An array $f_{i,j}$ of small picture-elements called pixels represents the image by recording the values of measurements at each pixel position.

The task of a computer-vision system is to understand the scene that an image (an array of pixels) depicts. With an image given, an image-understanding program builds a description not only of the image itself but also of the

scene it depicts. In artificial intelligence(AI) research, the term scene analysis is often used to emphasize the distinction between processing two-dimensional images and three-dimensional scenes. Image understanding requires knowledge about the task world, as well as sophisticated image-processing techniques. Usually, AI applies in the interpretation of the image. For example, vertex analysis determines the number of vertices to be formed from different planar areas. Line labeling describes the direction of the lines. Matching shape description represents a wide variety of shape-description notations for different surfaces and volumes (Brown 1982). Shape from line drawing is applied in this thesis for three-dimensional object recognition and is discussed in the later sections.

CAD, Vision, and Robotics

Computer-aided design (CAD), computer-aided manufacturing (CAM), computer vision, and robotics are considered integral to the factories of the future. The advances in the above technologies have been a driving force in manufacturing to provide higher quality products and to reduce production cycles by increasing the efficiency and productivity of design and production (Granky 1986). The main goal of a CAD system is to design new shapes suitable for automatically manufacturing objects to the desired

specifications in a cost-effective manner. CAD assists in the creation, modification, analysis, and optimization of a design. Computer graphics used in CAD are mainly concerned with realistic displays of objects from arbitrary viewpoints and under a variety of lighting conditions.

Vision systems can be applied in a number of manufacturing areas, especially with robots. Visual information is used to analyze objects already in existence for recognition and manipulation, and also to facilitate the efficient and automatic analysis of the sensed data. A summary of the Society of Manufacturing Engineers' Delphi forecast (Smith & Wilson 1982) is given in figure 2.1. As shown in the table, the basic computational model for object recognition and manipulation is strongly goal-directed. A variety of three-dimensional object models, with all the desired information for recognition and manipulation, is needed for intelligent robots to complete their tasks. In the next section, Kanade's origami world model to recover the three-dimensional shape of an object from a single view is discussed. Kanade's model is used to compare with the one that is developed in this thesis. Both models can be applied as an interface between CAD and vision systems.

	Forecasted Robot Characteristics by 1990	% of Robots sold in 1990 having the Forecasted Characteristics
1	Robots made of modular, standard components	63
2	Use of general-purpose hand	20
3	Self-propelled robots	15
4	Repeatability less than or equal to 0.001 inch	93
5	Use of integral sensors	60
6	Use of pattern recognition and feedback	22
7	Use of vision capability	25
8	Robots that will be programmed on-line	20

Figure 2.1. A summary of the SME Delphi forecast (Smith & Wilson 1982)

The Kanade Model for Object Recognition

A line drawing can be reviewed as the minimal representation of intensity discontinuities in an image corresponding to discontinuities of surface orientation, range, reflectance, and illumination. Due to such a rich amount of information contained in it, even a fledgling visual system is capable of inferring three-dimensional structure of a scene from a line drawing (Senze 1988). Line drawings are a natural early target for computer vision for the following reasons:

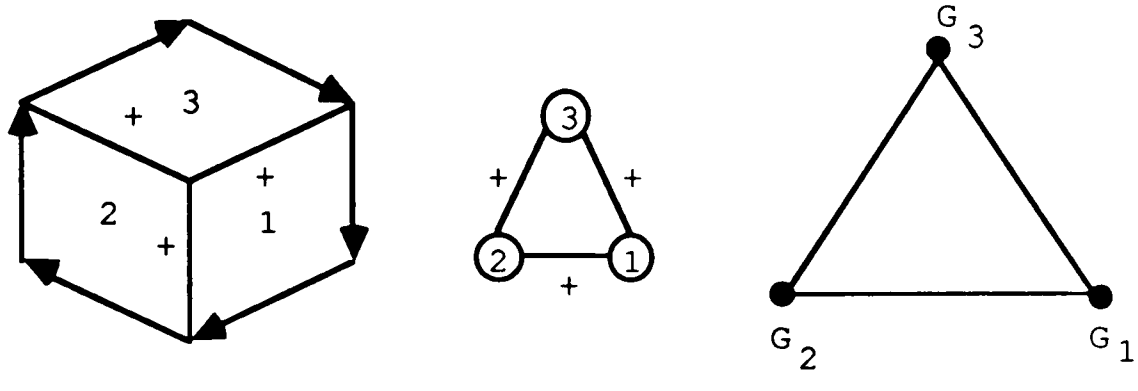
1. They are related closely to surface features of polyhedral scenes.

2. They may be represented exactly; the noise and incomplete visual processing that may have affected the "line drawing extraction" can be completely eliminated.

3. They present an interpretation problem that is significant but seems approachable (Brown 1982).

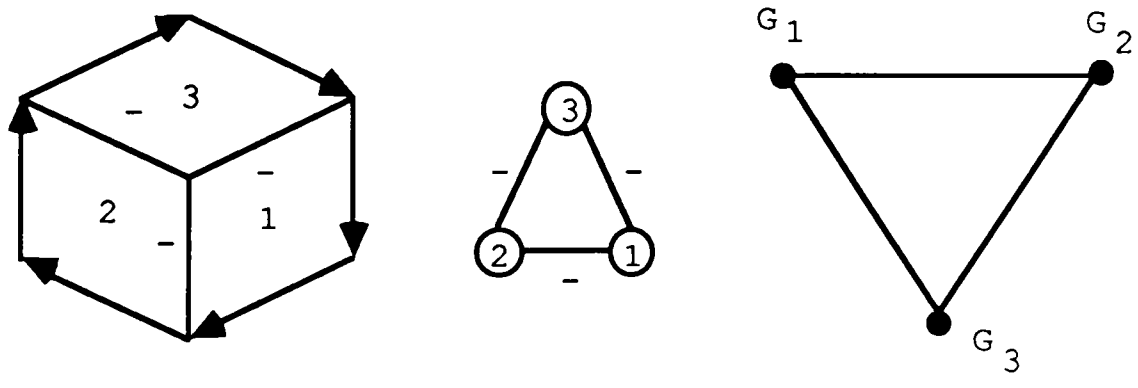
Kanade developed a model (Kanade 1981) for understanding line drawing in terms of plane surfaces and found their qualitative shape interpretations by assigning one of the three kinds of labels (+, -, ↑) to each line. "+" and "-" mean an edge at which two surfaces meet and form a convexity and a concavity, respectively. "↑" means an edge at which one surface occludes another. In his model, no more than three planar surfaces of different orientations meet at a vertex, and no more than three edges of different directions are involved at a vertex. Kanade generates different labels for up to three surface vertices. They are L, arrow, fork, T, K, X, and PSI junctions. The labeling procedure is combined with the Surface Construction Graph (SCG) and gradient constraints for reconstruction. The SCG is a labeled graph where a node represents surface and an arc represents a constraint between the surface. For example, in each of the figures 2.2-2.4, (a) shows the labeling, (b) the corresponding SCG, and (c) the

illustration of the constraints among the gradients represented by the SCG.



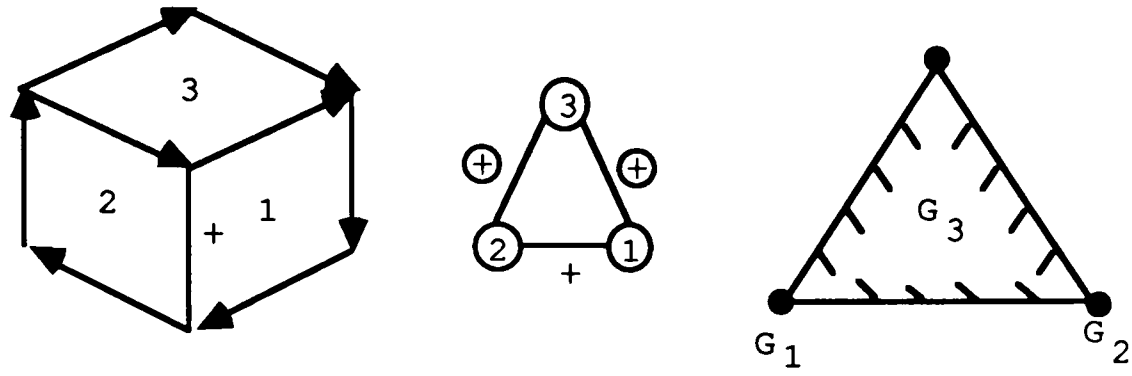
(a) (b) (c)

Figure 2.2. The first labeling of the 'cube' scene represents a convex corner



(a) (b) (c)

Figure 2.3. The second labeling of the 'cube' scene represents a concave corner



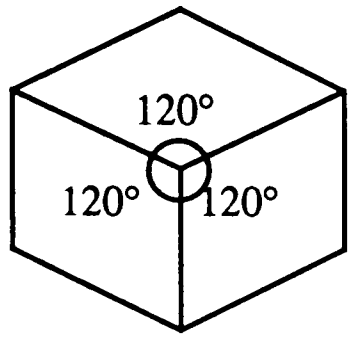
(a) (b) (c)

Figure 2.4. Third labeling of the 'cube' scene represents a convex edge, occluding the third place S3 partially

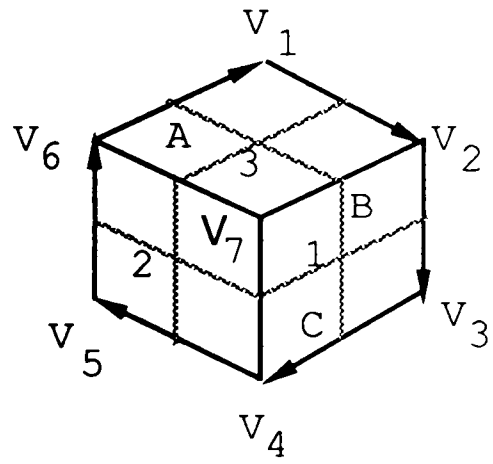
Kanade describes the heuristic of mapping the image regularities into constraints in the gradient space. He develops two theories in parallelism of lines and skewed symmetry. For the parallelism of lines and skewed symmetry, he explains as follows: "If two lines are parallel in the picture, they depict parallel lines in the scene." "A skewed symmetry depicts a real symmetry viewed from one (unknown) view direction" (Kanade 1981). The above theories are based on the assumption of orthographic projection. By combining SCG, labeling, and the constraints in the gradient space, quantitative shapes can be recovered by assigning a unique gradient to each surface. The cube described in the figure

2.2a is redrawn and is reproduced in the figure 2.5b. The labeling indicates that there are three totally visible surfaces, S_1(=V_3 V_4 V_7 V_2), S_2(=V_5 V_6 V_7 V_4), and S_3(=V_1 V_2 V_7 V_6), and that their gradients, G_1, G_2, and G_3, should form a triangle as shown in figure 2.2b. On the other hand, S_1, S_2, S_3 have skewed symmetries, their skewed symmetry axes and skewed transverse axes are shown in figure 2.5b as dotted lines. If these skewed symmetries are assumed to be projections of real symmetries, the hyperbola of each surface can be drawn in figure 2.5c.

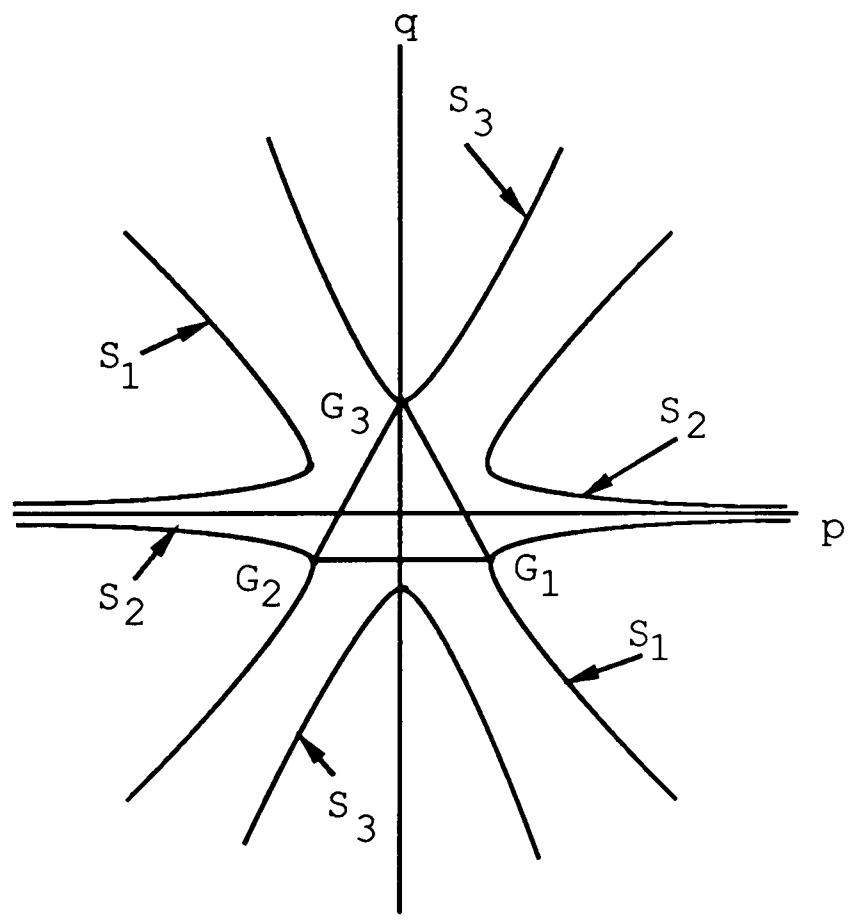
According to Kanade, four rules are needed to be followed for the assignment of gradients. First, gradients remain the same value when the sizes of the regions change. Second, the assignments of the surfaces are perpendicular to each other. Third, once the gradients have been computed, the plane formula in the three-dimensional space for each surface can be found. The three-dimensional coordinates of the vertices are found by assuming the "z" position of one point on the object to be known. Fourth, for a parallel quadrilateral, another skewed symmetry can be used by its diagonals. As shown in figure 2.6a, the diagonals are selected for each region to define its skewed symmetry. In figure 2.6b, the triangles are all the possible assignments of gradients.



(a)

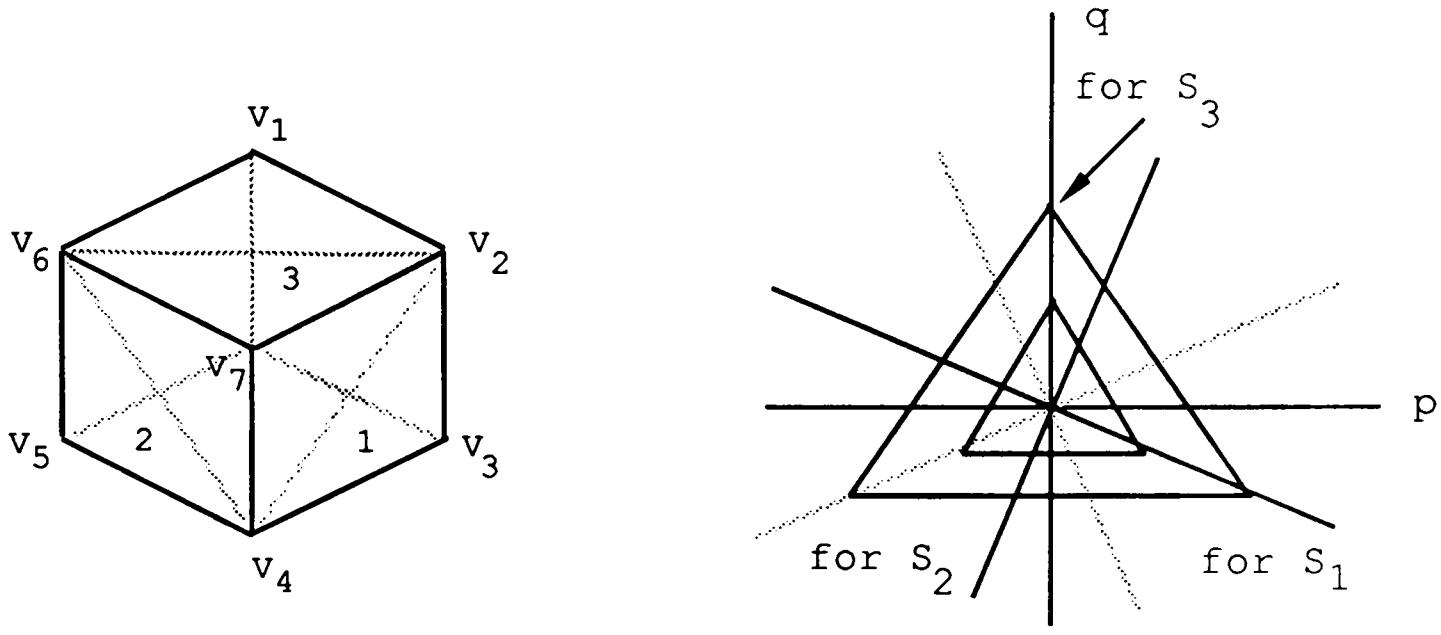


(b)



(c)

Figure 2.5. Quantitative shape recovery for the 'cube' scene for the labeling of Figure 2.2



(a) (b)

Figure 2.6. The parallel quadrilaterals in Figure 2.5b with other combinations of axes for skewed symmetries

Overall, Kanade's three-dimensional description for simple scenes can be achieved systematically from a single view by exploiting a few assumptions. His algorithm provides a labeling procedure which can recover qualitative shapes of line drawings together with constraints on surface orientations. The parallel-line and the skewed-symmetry heuristics map the picture properties into constraints in the gradient space. By putting all this information together, the surface orientations and the three-dimensional shape descriptions of the object can be determined easily.

In the next section, edge-detection for the images is discussed.

Edge Detection--The Sobel Operator

Edge detection is an important step in segmenting an image. This method is used to locate boundaries of meaningful regions that may be defined by a relatively uniform color, gray level, or texture. By finding sudden discontinuities of such image features, edges can be easily detected. "Threshold" is a value given to a certain gray level. Edge elements are selected by thresholding the output of the edge detector. If the output of an edge detector at a pixel exceeds a certain threshold value, that pixel is determined to be an edge element and given a value of one; otherwise, it is given a value of zero.

The Sobel operator is an example of an edge detector which is employed in this thesis. The Sobel operator is a nonlinear computation of the edge magnitude at (m,n) defined by Cohen & Feigenbaum (1982):

$$\text{grad } S(m,n) = (d_x^2 + d_y^2)^{1/2} , \qquad (2.1)$$

where

$$d_x = (f_{m-1,n-1} + 2f_{m,n-1} + f_{m+1,n-1}) - (f_{m-1,n+1} + 2f_{m,n+1} + f_{m+1,n+1}), \qquad (2.1a)$$

$$d_y = (f_{m+1,n-1} + 2f_{m+1,n} + f_{m+1,n+1}) - (f_{m-1,n-1} + 2f_{m-1,n} - f_{m-1,n+1}). \qquad (2.1b)$$

An edge image is created by calculating the two-dimensional derivative of the source image. The algorithm calculates this derivative by looking at each pixel in the source image and its eight nearest neighbors. The pixel of interest and its neighbors are treated as a 3x3 array (figure 2.7). The algorithm averages the intensities of the pixels on either side of each of these lines. Then the algorithm replaces the central pixel with the largest absolute difference of the four averaging operations (Fu 1986). After computing a new edge image, the image is passed through a threshold filter. The filter examines each pixel and assigns a "0" or "1" to it, depending on whether its value is below or above a given threshold value, respectively. In the next section, the mathematical model in three-dimensional translation for the display of objects used in this thesis is discussed.

-1	0	1		1	2	1
-2	0	2		0	0	0
-1	0	1		-1	-2	-1

Figure 2.7. A 3x3 array of weights in Sobel algorithm

Mathematical Model in Three-Dimensional Translation

Projection of Three Dimensions into Two Dimensions

The conceptual model of the three-dimensional viewing process is based on projection. Projection transforms three-dimensional objects onto a two-dimensional object plane. Usually, a view volume in the world, a projection onto the projection plane, and a viewport on the view surface are specified. The contents of the window, which are themselves the projection of the view volume onto the projection plane, are then mapped into the viewport for display. This class of projections is defined as a planar geometric projection because the projection is onto a plane and uses straight line projections (Foley & Dam 1982). Two basic kinds of projections are perspective and parallel.

For perspective projection, the center of the projection is at a finite distance. For parallel projection, the center of projection is infinitely distant. The perspective projection is discussed here because this kind of projection creates a visual effect similar to that of a photographic system and the human visual system with some degree of realism. In perspective projection, there is a vanishing point which refers to the convergence of any set of parallel lines not parallel to the projection plane. Perspective projections are categorized by the number of

principle vanishing points they have and by the number of axes the projection plane cuts (figure 2.8 and figure 2.9) (Foley & Dam 1982). In figure 2.9, this is a-one-point projection because lines parallel to the x- and y-axes do not converge and only lines parallel to the z-axis do so. This one point projection is used throughout the program in this project. In the next section, the mathematical model is described for viewing three-dimensional objects.

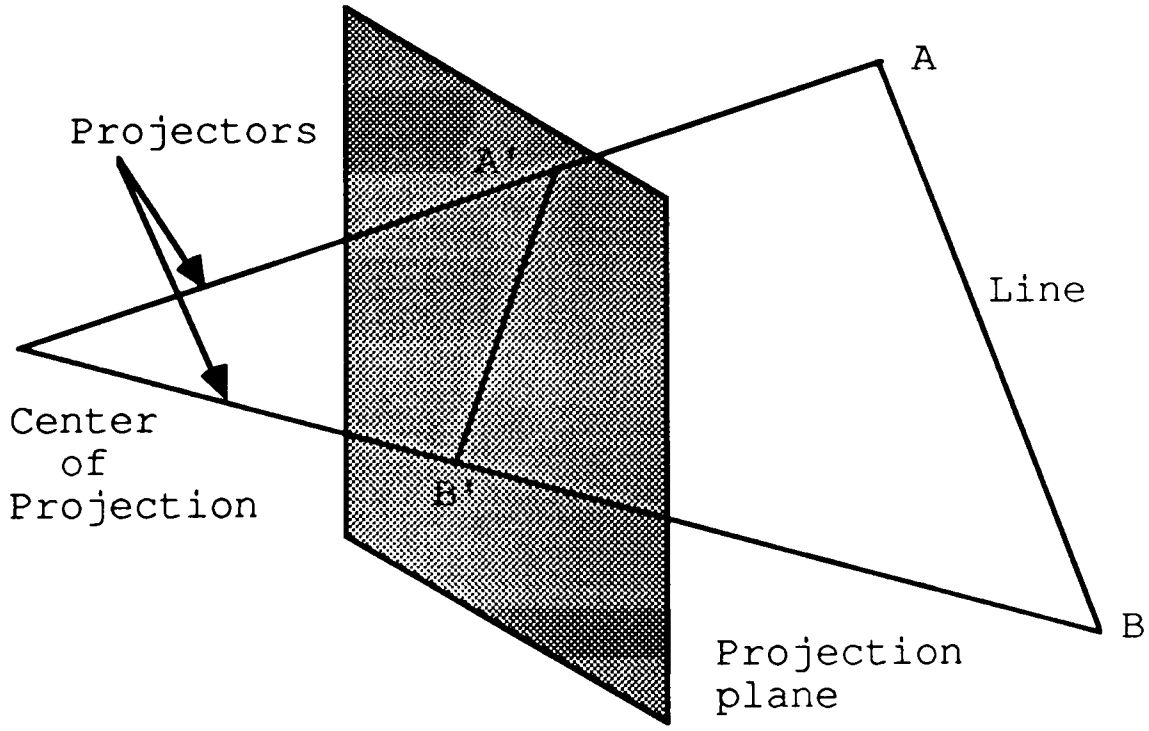


Figure 2.8. Line AB and its perspective projection A'B' (Foley & Dam 1982)

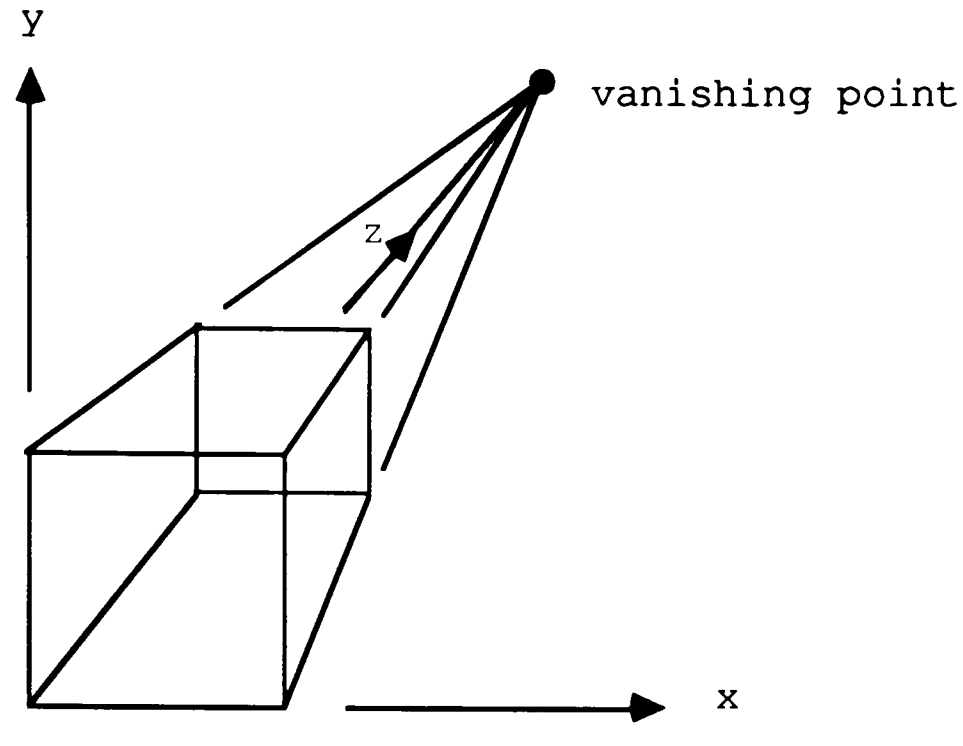


Figure 2.9. An example of one-point perspective projection (Foley & Dam 1982)

Mathematics of Planar Geometric Projections

The viewing coordinate system is left-handed with x to the right, y upwards, and z into the screen. The positive rotation is clockwise for a left-handed coordinate system. The representation of the three-dimensional points (x,y,z) is represented in homogeneous coordinates as (wx, wy, wz, w), with "w" not equal to zero. Transformation, rotation, clipping, and zoom discussed here are at the heart of the graphics application. The equations (Foley & Dam 1982) for three-dimensional translation are:

$$T(dx, dy, dz) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ dx & dy & dz & 1 \end{bmatrix} , \qquad (2.2)$$

and for three-dimensional rotation about the z-axis, y-axis, and x-axis (Foley & Dam 1982) are:

$$Rz(\theta) = \begin{bmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} , \qquad (2.3a)$$

$$Ry(\theta) = \begin{bmatrix} \cos\theta & 0 & -\sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} , \qquad (2.3b)$$

$$Rx(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & \sin\theta & 0 \\ 0 & -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} . \qquad (2.3c)$$

Assume the location of the camera or the eye is located in world coordinate called "from (F)" point, the camera or eye is looking directly toward a location called "at (A)" point, and also the orientation of the viewing plane with relation to the coordinate system is called "up (U)" point (figure 2.10). The above "R" matrix can be represented as an arbitrary matrix (Foley & Dam 1982),

$$A = \begin{bmatrix} r_{11} & r_{12} & r_{13} & 0 \\ r_{21} & r_{22} & r_{23} & 0 \\ r_{31} & r_{32} & r_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} , \qquad (2.4)$$

where $A_1 = (r_{11}, r_{12}, r_{13})$, $A_2 = (r_{21}, r_{22}, r_{23})$,

$A3 = (r_{31}, r_{32}, r_{33})$.

To compute A_1, A_2, A_3, we let A' be a vector where A' = A - F, U' = A - U, and

$$A_3 = \frac{A'}{\| A' \|} , \qquad (2.5a)$$

$$A_1 = \frac{A' \times U'}{\| A' \times U' \|} , \qquad (2.5b)$$

$$A_2 = \frac{(A' \times U') \times A'}{\| (A' \times U') \times A' \|} . \qquad (2.5c)$$

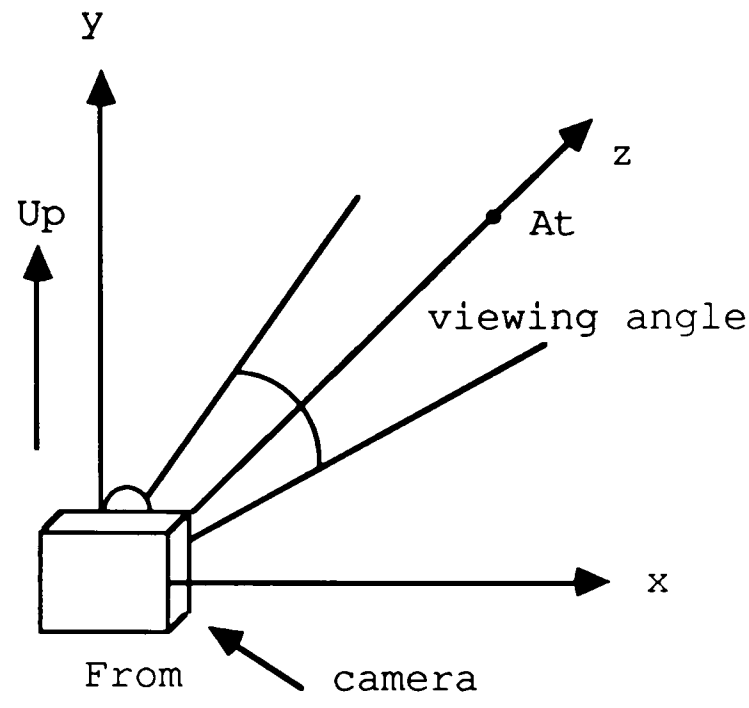


Figure 2.10. Viewing parameters appeared in eye coordinates

In addition, we will apply the matrix (Foley & Dam 1982),

$$\mathbf{M} = \begin{bmatrix} D & 0 & 0 & 0 \\ 0 & D & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} , \qquad (2.6)$$

where $D = \dfrac{1}{\tan(\text{viewing angle}/2)}$,

to adjust the view according to the viewing angle so that the object extends between the lines $y = z$, $y = -z$, $x = z$, and $x = -z$, and makes clipping a lot easier. Clipping will be discussed in the following session.

Clipping Image against a Canonical View Volume

Clipping means to excise any part of the objects not in view of a given size of a window. Given the specialization of a view volume and a projection, we could clip lines against the view volume by first calculating their interactions with each of the planes that define the view volume. Lines remaining after the clipping would be projected onto the view plane by calculating the interaction of the projectors through the endpoints with the view plane. The coordinates would then be transformed from three-dimensional world coordinates to two-dimensional device coordinates. The canonical view volume is defined by the planes (Foley & Dam 1982),

$$y = z,\ y = -z,\ x = z,\ x = -z,\ z = zmin,\ z = 1.$$

To clip against the canonical volume for perspective projection, we use a four-bit code, with the bits defined by:

bit 1 - point is about view volume: $y_v > z_v$;

bit 2 - point is below view volume: $y_v > -z_v$;

bit 3 - point is right of view volume: $x_v > z_v$;

bit 4 - point is left of view volume: $x_v > -z_v$.

As in two dimensions, a line is accepted if both endpoints have a code of all zeros and rejected if the bit-by-bit logical "and" of the codes is not all zeros. Otherwise, the process of line subdivision begins. Up to four intersections may have to be calculated. The intersection calculations make use of the parametric representation of a line from $P_1(x_1, y_1, z_1)$ to $P_2(x_2, y_2, z_2)$:

$$x_v = (x_2 - x_1) \cdot t + x_1 \quad ; \qquad (2.7a)$$

$$y_v = (y_2 - y_1) \cdot t + y_1 \quad ; \qquad (2.7b)$$

$$z_v = (z_2 - z_1) \cdot t + z_1 \quad ; \qquad (2.7c)$$

and calculate the intersection of lines with the sloping planes, $y_v = z_v$, $y_v = -z_v$, $x_v = z_v$, $x_v = -z_v$. For example, let us consider the $y_v = z_v$ plane for which

$$(y_2 - y_1) \cdot t + y_1 = (z_2 - z_1) \cdot t + z_1,$$

then

$$t = \frac{z_1 - y_1}{(y_2 - y_1) - (z_2 - z_1)} \quad , \qquad (2.8a)$$

$$x_v = \frac{(x_2 - x_1)(z_1 - y_1)}{(y_2 - y_1) - (z_2 - z_1)} + x_1 \quad , \qquad (2.8b)$$

$$y_v = \frac{(y_2 - y_1)(z_1 - y_1)}{(y_2 - y_1) - (z_2 - z_1)} + y_1 \quad . \qquad (2.8c)$$

The sequences of processes that implement the three-dimensional object for viewing are shown in Figure 2.11. In the next chapter, the algorithm for three-dimensional object recognition will be discussed.

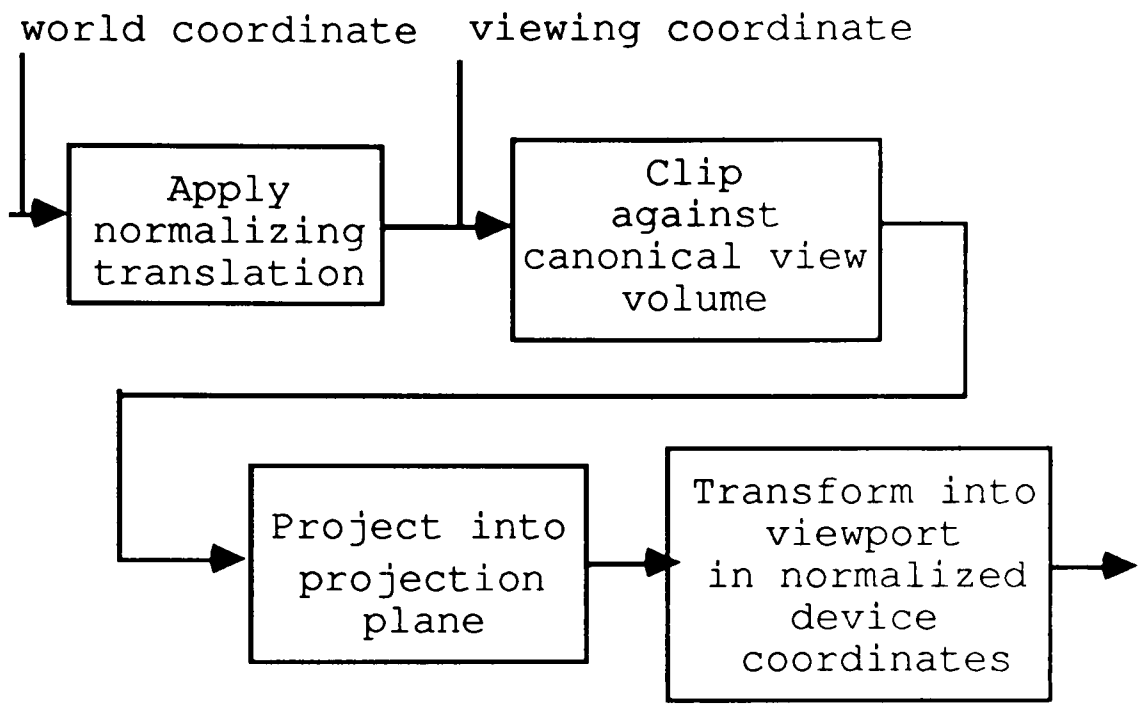


Figure 2.11. Implementation of three-dimensional viewing

CHAPTER III

PROBLEMS DEFINED IN KANADE MODEL

This chapter is to present the problems that are solved in this paper. They are the labeling procedure, the requirement for computer memory and the computing time.

Complicated Labeling Procedure

There are three major steps for the Kanade labeling procedure. The first step is to assign a single label (+, -, ↑) along each line of its entire length, and then a junction label (L, arrow, fork, T, K, X, PSI) assigns on each junction. The above procedure is previously described by Waltz (1975) but is expanded by Kanade. The second step is to construct a Surface Construction Graph (SCG) where a node represents surface and an arc represents a constraint between the surfaces. It indicates that surfaces are connected to other surfaces with constraints. The third step is to assign a unique gradient to each surface by using the principles of the parallelism of lines and the skewed symmetry. However, the search for consistent interpretations was exhaustive (Cohen & Feigenbaum 1982). The presence of the large number of junction label delays the process of

assigning the correct junction label on a line. Although the constraints in the gradient space facilitate the interpretation of a line drawing, the speed to label a correct junction is still very low. As the scene complexity increases, the required time increases. In addition, if a vertex cannot be labeled, the neighboring vertex is also affected. Then, the labeling procedure will fail with data of wrong labels. Due to the combination rule (Waltz 1975), a label is a possible description for a junction if and only if there is at least one label which "matches" its assigned to each adjacent junction. Two junction labels "match" if and only if the line segment which joins the junctions gets the same interpretation from both of the junctions at its ends. The rule leads the labeling procedure resulting in multiple labeling of a single line drawing (Cohen & Feigenbaum 1982.) For example, a cube scene has three interpretations as shown in figure 3.1. All labeling is equally natural in terms of geometrical realizability, the multiple labeling complicates the interpretation of the scene.

Requirement of Large Computer Memory

It is obvious that computer memory plays a very important rule in determining the effectiveness of a program and the ensuing cost of searching. The labeling method

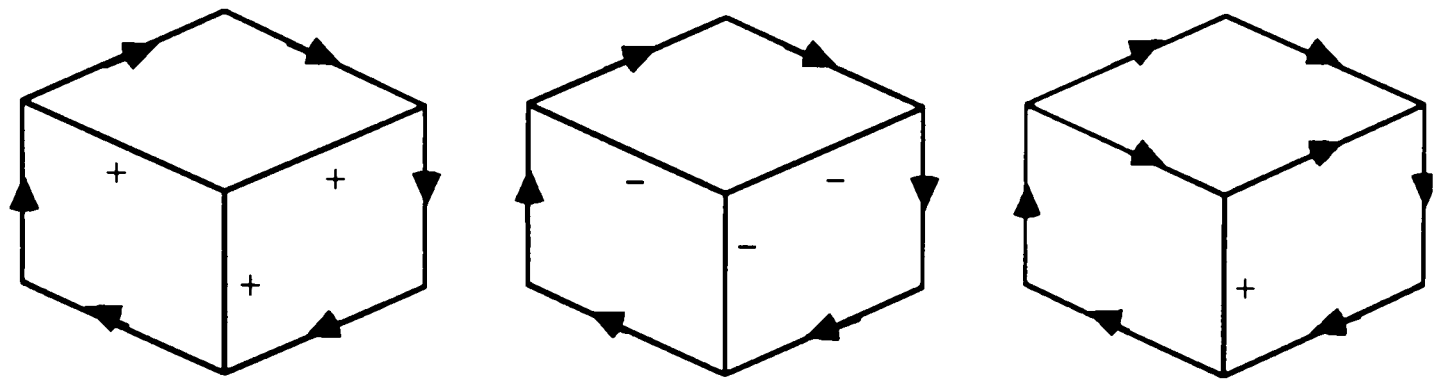


Figure 3.1. Interpretation of the "cube" line drawing

discussed above requires a large amount of computer memory space to run the program due to the searching of corrected junction labels. A certain amount of memory is needed to save the label junctions, the label of each line, and the information obtained from the Surface Construction Graph even though Kanade's model uses the constraints of the gradients to decrease the process of the labeling procedure. The algorithm will be less cost-effective if the algorithm is applied in manufacturing areas of CAD or vision systems with three-dimensional object recognition, in which both systems are usually expensive. The requirement of the

computer memory increases proportionally to the cost of the systems.

Requirement of Long Computing Time

Despite the description of the labeling process in the first session involving the long computing time, the third step of the Kanade model has the same problem, too. By obtaining the gradients of each surface, simultaneous equations are solved. At least three simultaneous equations are required to solve the three-dimensional shape of an object. The more complex of the scene, the more computing time is required. Less computing time is required in this new process. This will be demonstrated in Chapter VI.

In the next chapter, a systematic approach will be introduced to solve the above problems.

CHAPTER IV

MODIFICATION OF KANADE MODEL

In this chapter, a modified Kanade model is introduced. Two algorithms, the six-bit coding method and the parallelogram search method, are designed. The six-bit coding method reduces the need for computer memory for storage of edge information. The parallelogram search method searches the consistent planar surfaces to get the object information. Finally, Constructive Solid Geometry is applied to construct the final image.

Six-Bit Coding Method

The six-bit coding method is the method of putting the edge information into bits so that each vertex has a value of the surrounding edge information. To develop the theory, we first look at the Kanade labeling model. He showed that twelve quadrant planes could be obtained by intersecting three full planes as primitives (figure 4.1). The vertices that can be generated by those primitives are called up-to-three-surface vertices (Kanade 1981). All legal junction labels can be enumerated by fixing the eye position in one of the eight octants bounded by the quadrant planes and generating all possible combinations of occupied and vacant

quadrants. Instead of generating all junction labels from these quadrant planes, we can get the information of up to six edges around a vertex connected to it.

Martelli (Brown 1982) defines edges as having a direction even though there is no gray level change (figure 4.2.) An edge is a straight boundary of a plane surface. A line is an orothgraphic projection of an edge to the picture plane. A vertex is a point where edges of the surface(s) meet. A junction is a point in the picture where lines meet. Assume there is a point at the center of the image grid structure; an example of edge and vertex formation is shown in figure 4.3.

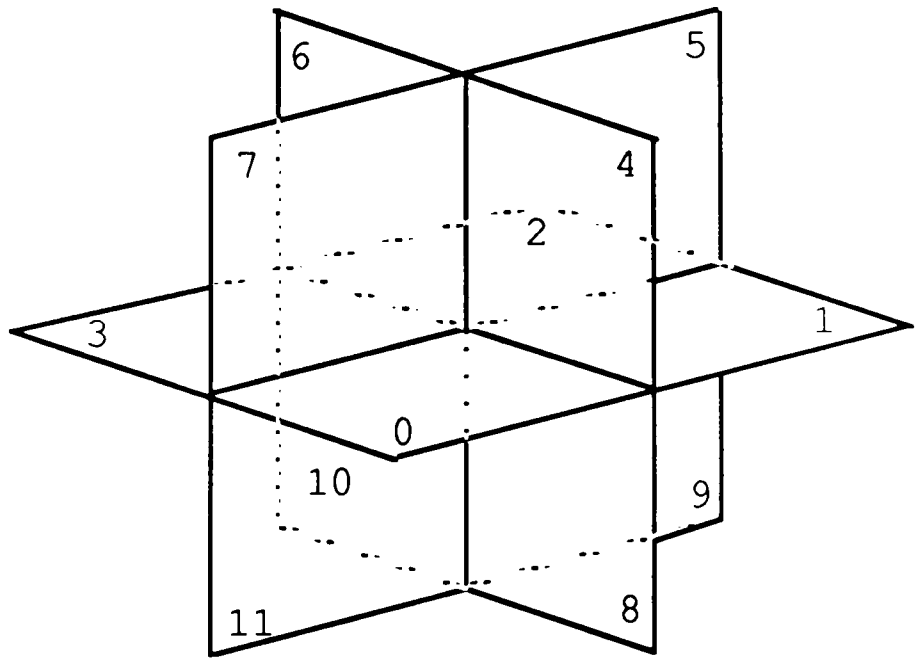


Figure 4.1. Twelve quadrant planes to generate up-to-three-surface vertices

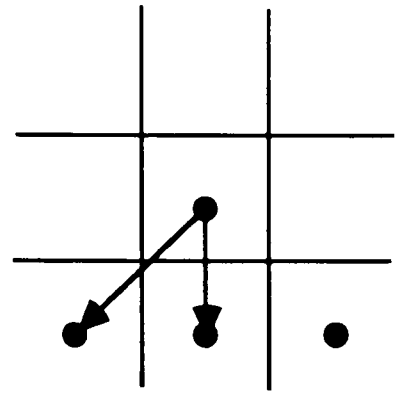
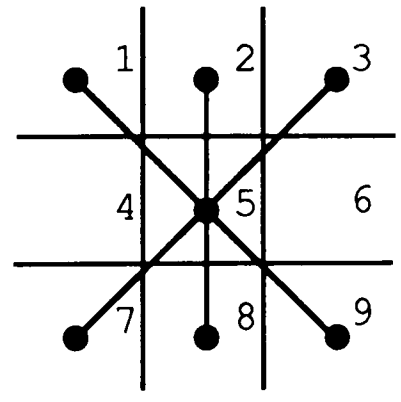


Figure 4.2. Edge direction as described by Martelli



edge (pixels 2,5,8, pixels 3,5,7, etc)

vertex (pixels 2,5,3, pixels 9,5,8, etc)

Figure 4.3. An example of edge and junction formations

By using the above information, the six-bit coding method is designed as follows. Blocks, which are treated as orthogonal polyhedrons, may have up to six different edges connected to a vertex. The six different edges are up, down, left up, left down, right up, right down, which are corresponding to "up," "dn," "lu," "ld," "ru," "rd" slots, respectively (figure 4.4). The vertex type has a six-bit code, in which each bit with a given value of "1" and "0" denotes whether or not the corresponding type of edge exists along the direction, respectively. The vertex type value shows those directions to which the edges are connected. The six-bit code is described as follows:

bit 0 = 1 vertical up (up) edge connected to a vertex;
= 0 otherwise;

bit 1 = 1 vertical down (down) edge connected to a vertex;
= 0 otherwise;

bit 2 = 1 left up (lp) edge connected to a vertex;
= 0 otherwise;

bit 3 = 1 left down (ld) edge connected to a vertex;
= 0 otherwise;

bit 4 = 1 right up (rp) edge connected to a vertex;
= 0 otherwise;

bit 5 = 1 right down (rd) edge connected to a vertex;
= 0 otherwise.

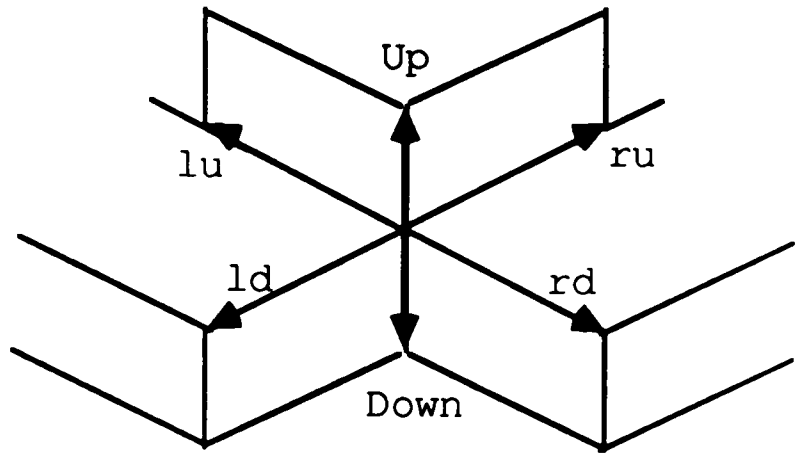


Figure 4.4. Six different edges of vertex directions

The binary digits of the bits can be transformed into any other form of representation. The notation or representation of a Number(N), can be written as follows (Fletcher 1982):

$$(N)_r = \sum_{j=-m}^{n-1} a_j \, r^j \quad . \qquad (4.1)$$

For example, if ru = 1, ld = 1, down = 1, up = 1, rd = 0, lu = 0, the vertex and edge descriptions are as follows:

edge	rd	ru	ld	lu	down	up
bit	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
Vertex	0	1	1	0	1	1

therefore, the value of the vertex in decimal form is:

$= 2^0 + 2^1 + 2^3 + 2^4$

$= 27.$

As a result, the edge information can be stored in bit format rather than a whole value. The amount of the computer memory required will be decreased.

Parallelogram Search Method

Another algorithm developed in this thesis is called the parallelogram search method. The constraints in the gradient space, as discussed by the parallel-line heuristic and the skewed-line heuristic by Kanade are used for developing this algorithm. A planar graph search method for parallelograms is used to find the two-dimensional planar surface of blocks. A graph is said to be planar if there exists some geometric representation of the graph which can be drawn on a plane such that no two of its edges intersect (Deo 1987). The planes that are searched are only parallelograms because of the parallelogram shape of the block surface and the images taken at 45-degree angles. A vertex can have three different plane formations, on the left and on the right (figure 4.5). With the information of gradient, length, and the corresponding bit of the edge provided, another vertex can be found along that direction. Then, the search will be followed with another vertex, until a parallelogram is completed. The different shapes and the directions of the parallelograms between the left and right

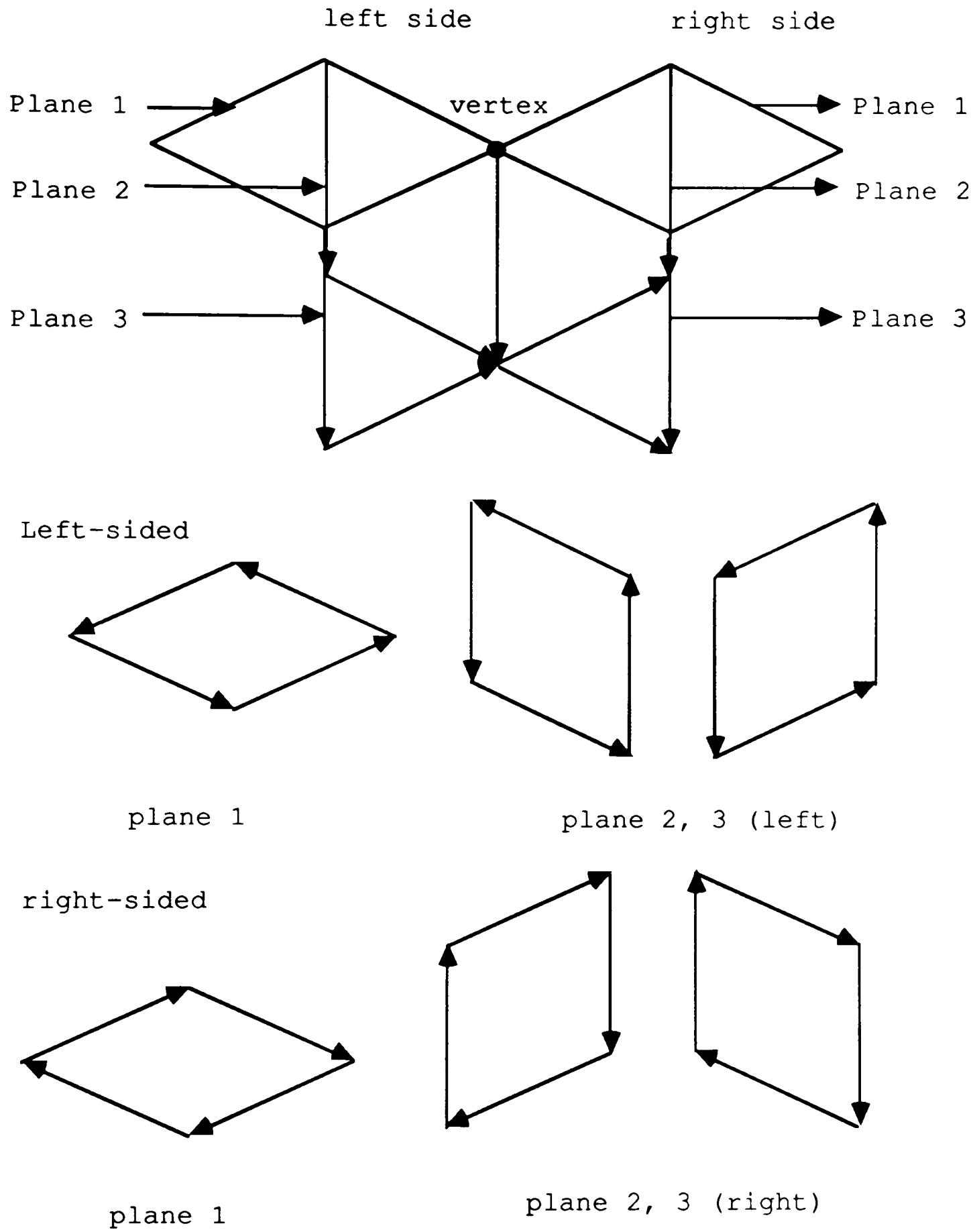


Figure 4.5. Plane surfaces in parallelogram form and its corresponding search

sides of the vertex are shown in figure 4.5. The height of a block can be obtained from the parallelogram search when the corresponding bit of edge information of "up" is one.

The purpose of this search is to find the consistent parallelogram surfaces for the object information. From the search, the following information is obtained; vertex location (x,y), height, length of left-up side, left-down side, right-up side, right-down side, and other vertices connected to this vertex. The height, length of the left-up side and the right-up side are to describe a block because this information is directly related to the height, width, and length of a block for recognition. In addition, the shape of the parallelogram is formed by blocks. The search of a parallelogram is restricted to the direction as listed in figure 4.5 to provide a uniform searching for the parallelogram surface.

Construction of the Final Sample

After the information of the block sample has been obtained, the constructive solid geometry (CSG) scheme is applied to construct the final sample. CSG, which was designed by Boyse (Boyse 1979), represents an object as a binary tree where each leaf represents an instance of a primitive and each node represents an operation of its decendents (Requicha & Voelcker 1982, Boyse 1979).

Primitives, such as blocks, spheres, cylinders, and cones, are combined by union, intersection, or difference from the bottom to the top of the tree. The final block samples are represented as a composition of blocks, which adds the data together from the database of the blocks starting from the lowest level to the upper levels. An example is shown in figure 4.6.

In this chapter, the six-bit coding method and the parallelogram search method were introduced for three-dimensional object recognition. Constructive Solid Geometry then applies to construct the final image. For the next chapter, an algorithm will be developed to link with the six-bit coding method, the parallelogram search method, and the Constructive Solid Geometry together.

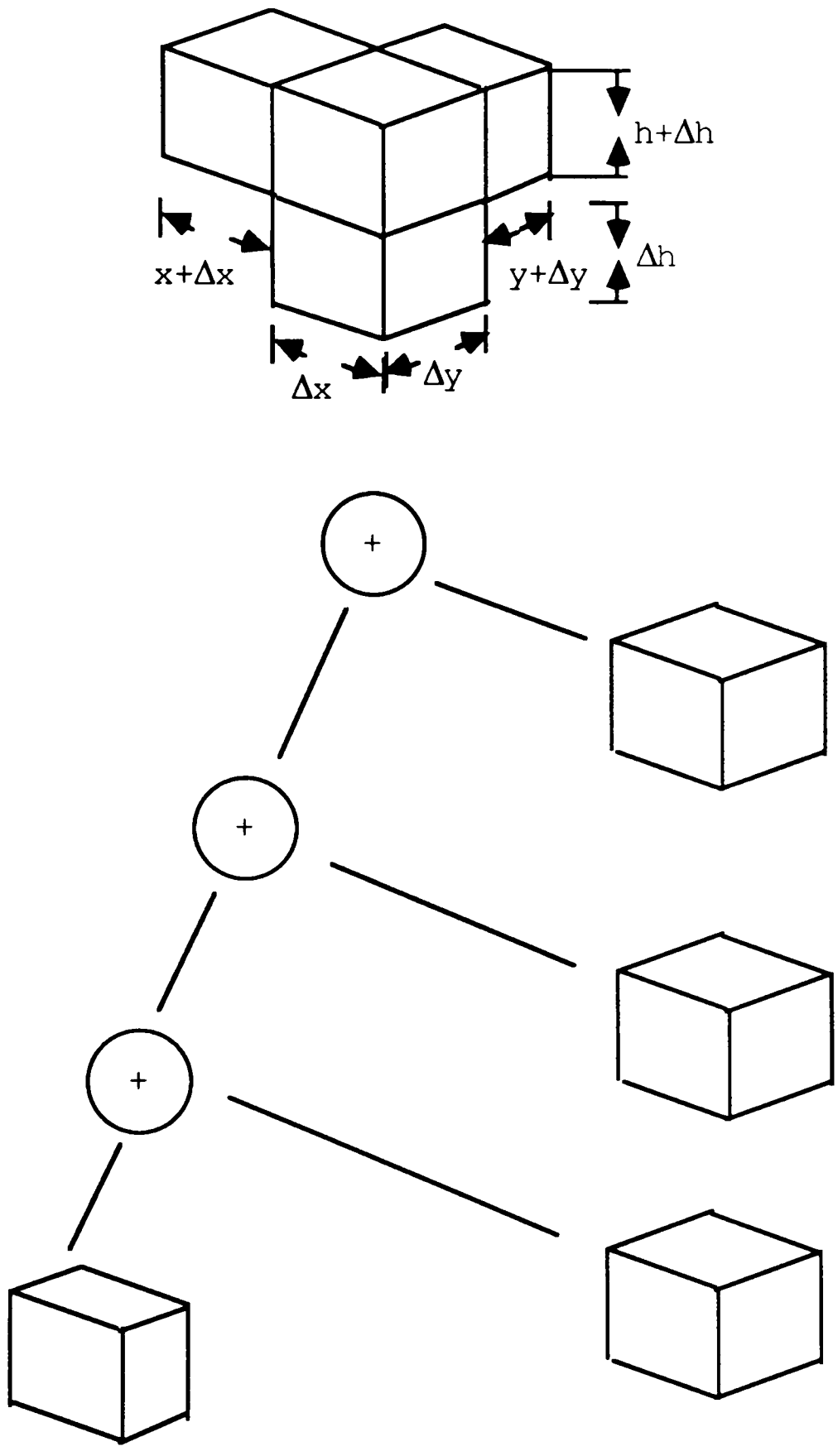


Figure 4.6. An example of constructive solid geometry (CSG) representation

CHAPTER V

ALGORITHM DEVELOPMENT

This chapter introduces the algorithm for three-dimensional object recognition. This algorithm uses the techniques discussed in Chapter IV. More specifications are linked with the techniques for the operation of the algorithm. A flow chart is also provided at the end of this chapter.

Three-Dimensional Object Recognition

The three-dimensional object recognition is initiated by taking images from multiple views of 45-degree angles. The Sobel operator is then applied for edge detection. After that, the data will be analyzed by the six-bit coding method. A vertex with the surrounding six-edge information is stored in bit format of a memory location. When the corresponding edge is present, the bit has a value of one; otherwise, its value is zero. Since images are taken at different views of 45-degree angles, the two pixels 4 and 6, as seen in figure 4.3, are not considered. The above condition is still coincident to the six-bit code method with only a maximum of six edges connected to a vertex. When the vertices and the edge information are obtained, vertices

are re-arranged by sorting, starting from the minimum location of y to the maximum location of y. The gradients of the left and the right side of a block are constant. Moreover, the height of a block, the length from one vertex to the left and to the right of the other vertices are constant in orthographic projection of the image, too. The gradients then become the constraints in the image scene. It is also important to consider the vertices that may connect to other vertices of the same block or different blocks so that the number and the type of blocks from each layer can be estimated. There are three major types of vertices needed to be considered: the "Y" type and the two collinear "T" types (figure 5.1). The connection of blocks in a layer can be found by the "Y" type or the two collinear "T" types, while the connection of blocks from one layer to another layer can be found by the two collinear "T" types. For example, in figure 5.2, there are four vertices that have the "Y" type and two vertices that have the collinear "T" types which indicate the connections with other blocks. However, further examination of the block connections is necessary.

After the vertices with the "Y" type or the two collinear "T" types are selected, the information of the connections between vertices and a primary shape of the

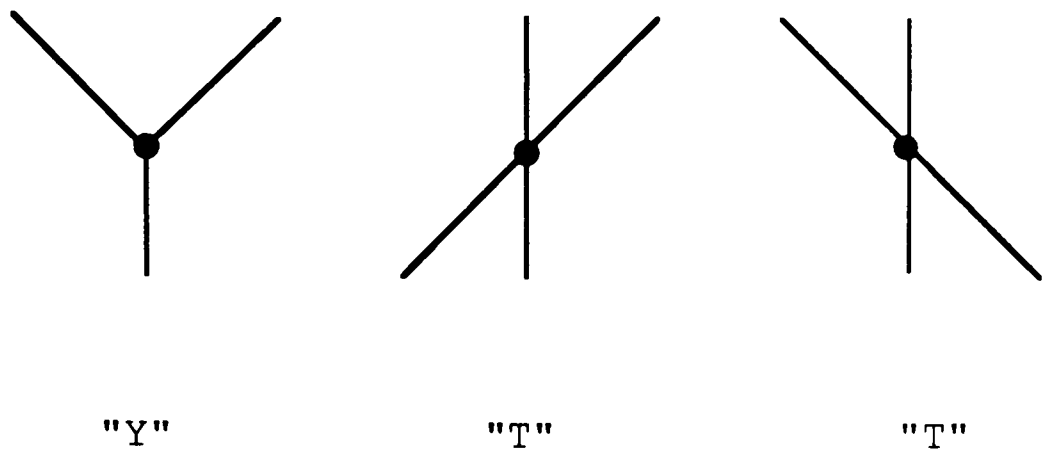


Figure 5.1. "Y" and two
collinear "T" type vertices

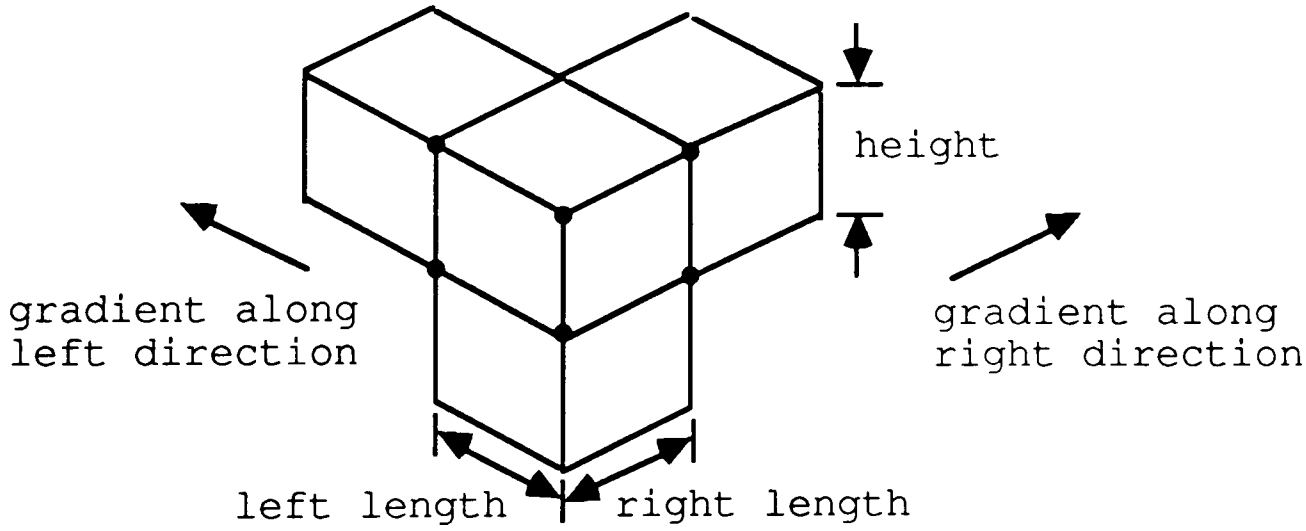


Figure 5.2. Vertices with "T" and "Y"
types selected(•)

block sample is needed to be solved. The parallelogram search method is employed on vertices to search for the consistent parallelogram along a given direction. The vertex location (x,y), height, width, and length of a block are found in this process. The next step is to classify the number of levels and the blocks which belong to these levels.

The data information obtained from above shows that one vertex is connected to another vertex, and this vertex is connected back to the original vertex. Therefore, the data are needed to put it together and to re-arrange according to the levels so that the data are more meaningful and organized. At this point, the number of levels, type of blocks, height, left-up and right-up side are put into a data file for further investigation. For two collinear "T" types, one of the left-up side or right-up side is unknown because of the occluding by the other surfaces. As a result, another view of the block sample may be used to collect the information and compare with the first image. For a complicated block sample, four different views of the block sample will be needed. For a simple block sample, two views from the opposite corners are enough. By comparing two images from opposite corners, three constraints need to be considered.

(1) When length is different from the same vertex of the two images, the longest length is chosen. It is because part of the line is occluded by another surface.

(2) Vertices are not consistent when the parallelogram search method fails. Usually, at least one parallelogram planar surface is found from the selected vertex.

(3) No block is connected on that side of the vertex when the parallelogram search method fails on that side.

After the images of height, the left-up side, and the right-up side have been compared, this information is put together to obtain the type of the blocks. The positions of the blocks can be estimated from the image and from the database of the predefined blocks. The constructive solid geometry (CSG) scheme is then applied to construct the final image sample. A reference vertex of a block sample is chosen at the x-, y-, z-directions. The final image sample is represented as a composition of blocks, which adds the data together from the database of the blocks starting from the lowest level to the upper levels. The mathematical equations that are discussed in Chapter III are used to display the final block sample. A flow chart for the whole procedure is shown in figure 5.3.

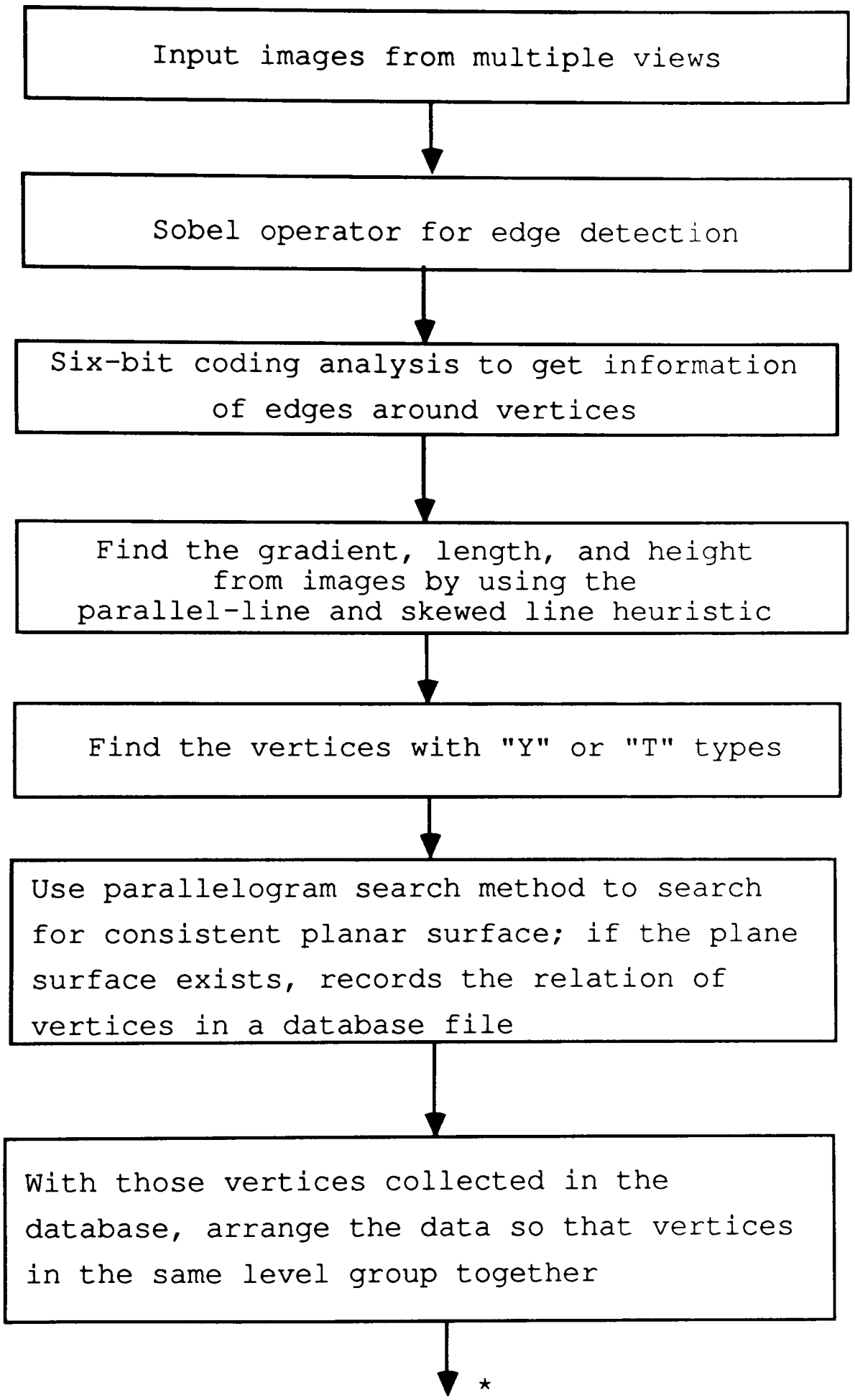


Figure 5.3. Simplified flow chart for three-dimensional object recognition

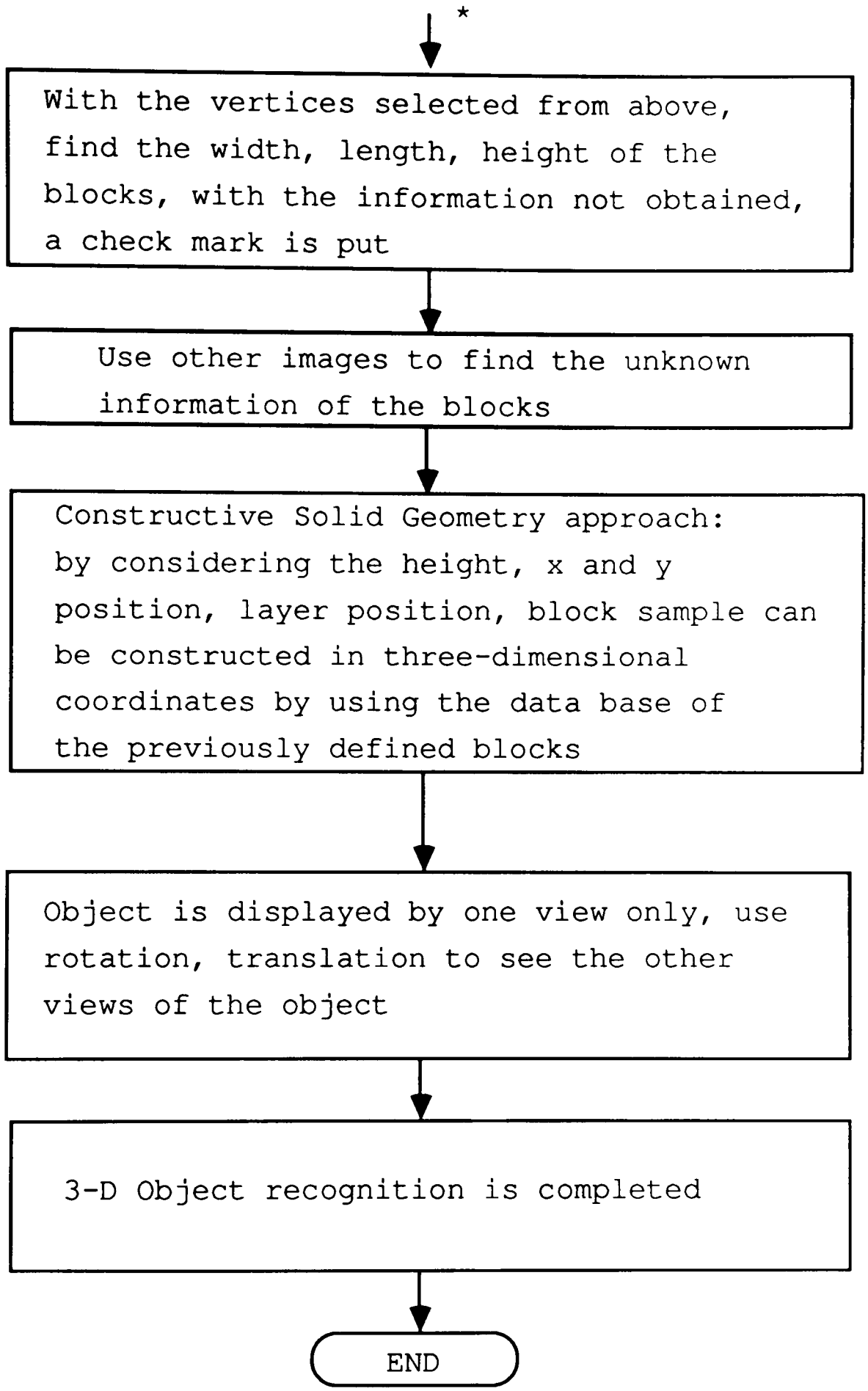


Figure 5.3. Continued

CHAPTER VI

VERIFICATION

In this chapter, the results of using the algorithm discussed in the previous chapter for three-dimensional object recognition are analyzed. The algorithm is compared with the model suggested by Kanade. There are two programs written for this project. The first program is to obtain the images and display them in graphics with edge detection characteristics. Another program is used to perform the three-dimensional object recognition and to display the final image in different views. Three basic blocks from Fischer Technik® are used (figure 6.1). Since the blocks have regular height, width, and length, the block sample can be recognized as different types of blocks with layers. The two-dimensional images from four viewpoints (North-East, South-East, North-West, South-West) of the sample are taken from a video camera. The following sections will explain all the procedures in detail.

Obtaining Images with Edge Detection

The images are analyzed with a Compaq personal computer-based digitizing system. The imaging device used is

a COHU 4810 series solid state CCD camera. The camera is attached to an EPIX silicon video acquisition and display board with one megabyte of on-board memory. The eight-bit digitizer board allows the storage of four fields of 512x480 pixels. These data are then transferred to a 32-bit Compaq computer system with direct memory access to the video memory for storage. The digital image can be displayed on a PVM-1271Q Sony studio monitor. The COHU video camera has 754(H) by 512(V) active picture elements, which allows the capture of a 512(H) by 480(V) pixel image. The vertical to horizontal pixel size ratio is 1.173913. The whole set-up is shown in figure 6.2.

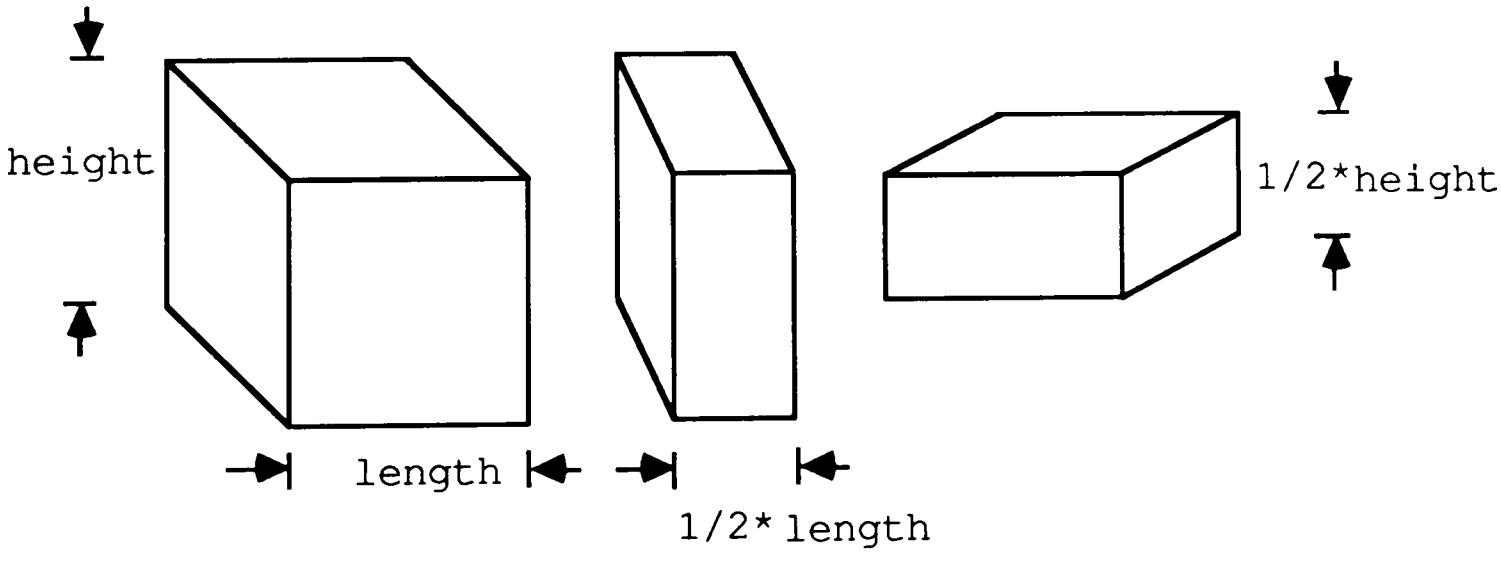


Figure 6.1. Three types of Fischer Technik® blocks

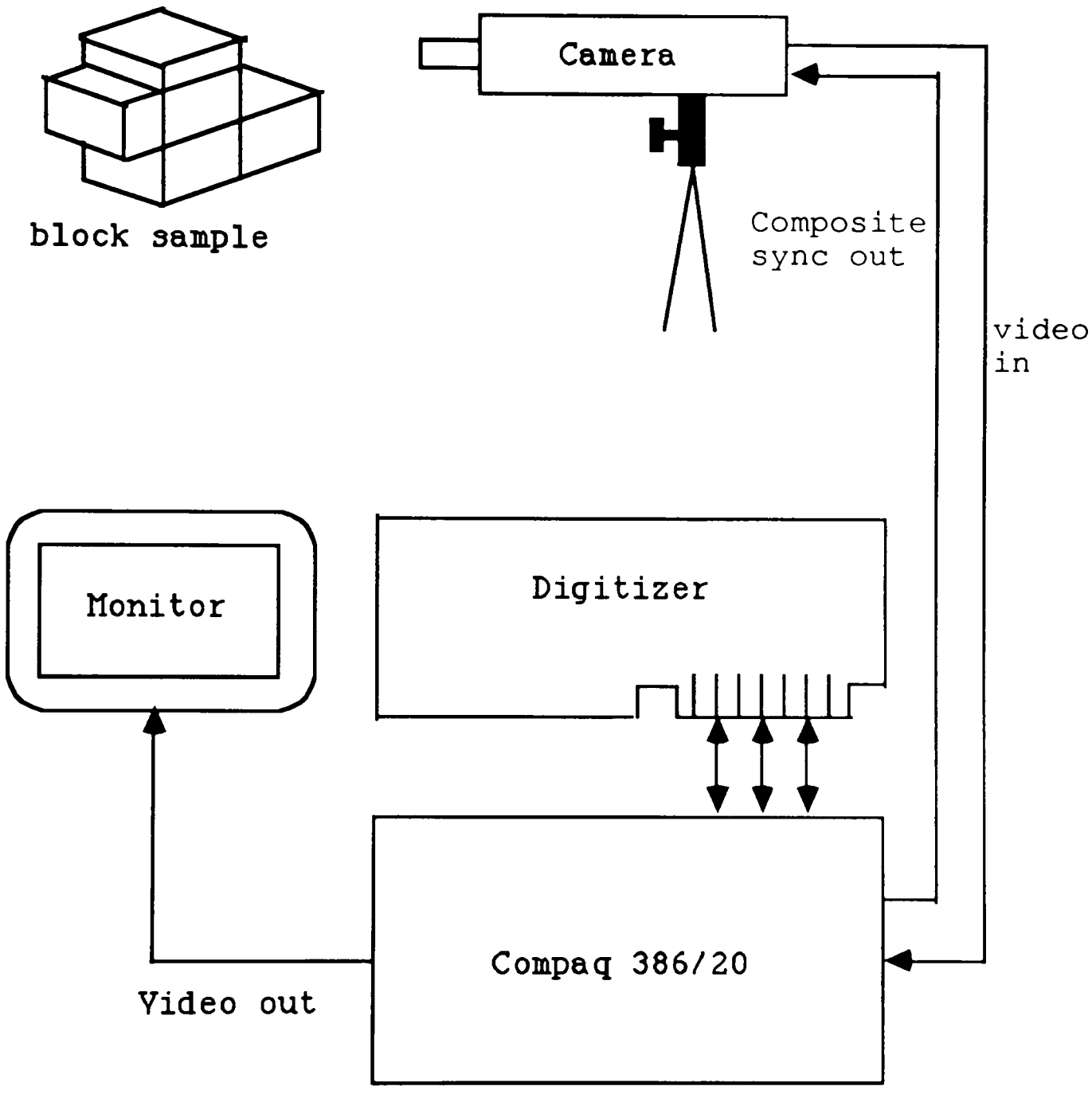


Figure 6.2. Set-up of obtaining the computer images

A program written in C language is used to display the image via TI Business-Pro computer with color graphics adapter (CGA). Because of the sizes of the images, which are 256(H) by 240(V), the vertical axis is greater than the size of the CGA with high resolution (600x200), the images are reduced by half to 128(H) by 120(V), by taking the averages of the 2x2 pixels. The Sobel operator is then applied to the images for edge detection. Figure 6.3 is one of the basic blocks. Figure 6.4 is the image of the block sample used for the three-dimensional object recognition. Figure 6.5 and figure 6.6 are the results of the images after the Sobel operator. The threshold value is selected at the value of "80" for both figure 6.5 and figure 6.6.

The Effect of the Sobel Operator

The Sobel Operator is used to enhance the image by increasing the values of the pixels along the edges of the line of the images. This enhancement is achieved by an approximation of the Sobel gradient operator which replaces the intensity of each pixel by the equation discussed in Chapter II. The Sobel operator is chosen because it combines with smoothing operation, which enhances edges with less noise. The smoothing operation is to remove isolated edges corresponding to noise in the background. This is

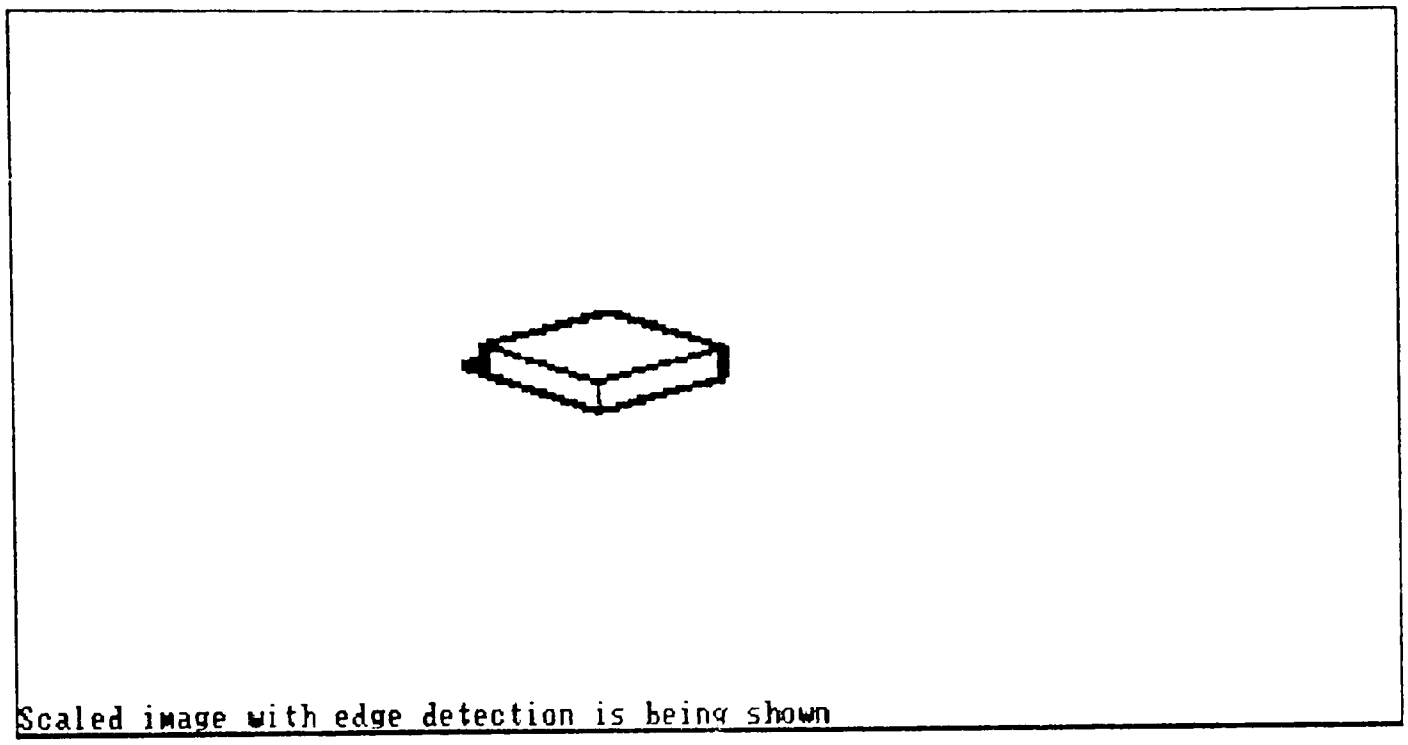


Figure 6.5. Edge detection of the block in Figure 6.3

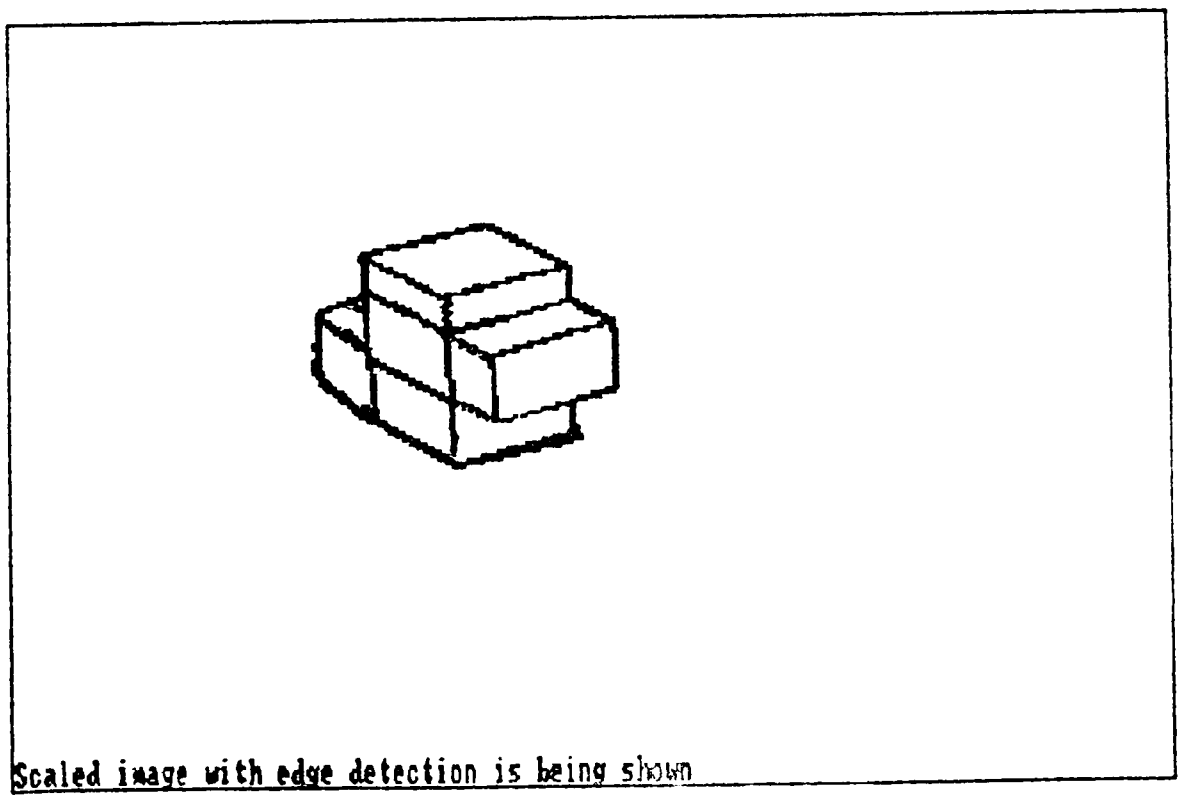


Figure 6.6. Edge detection of the block sample

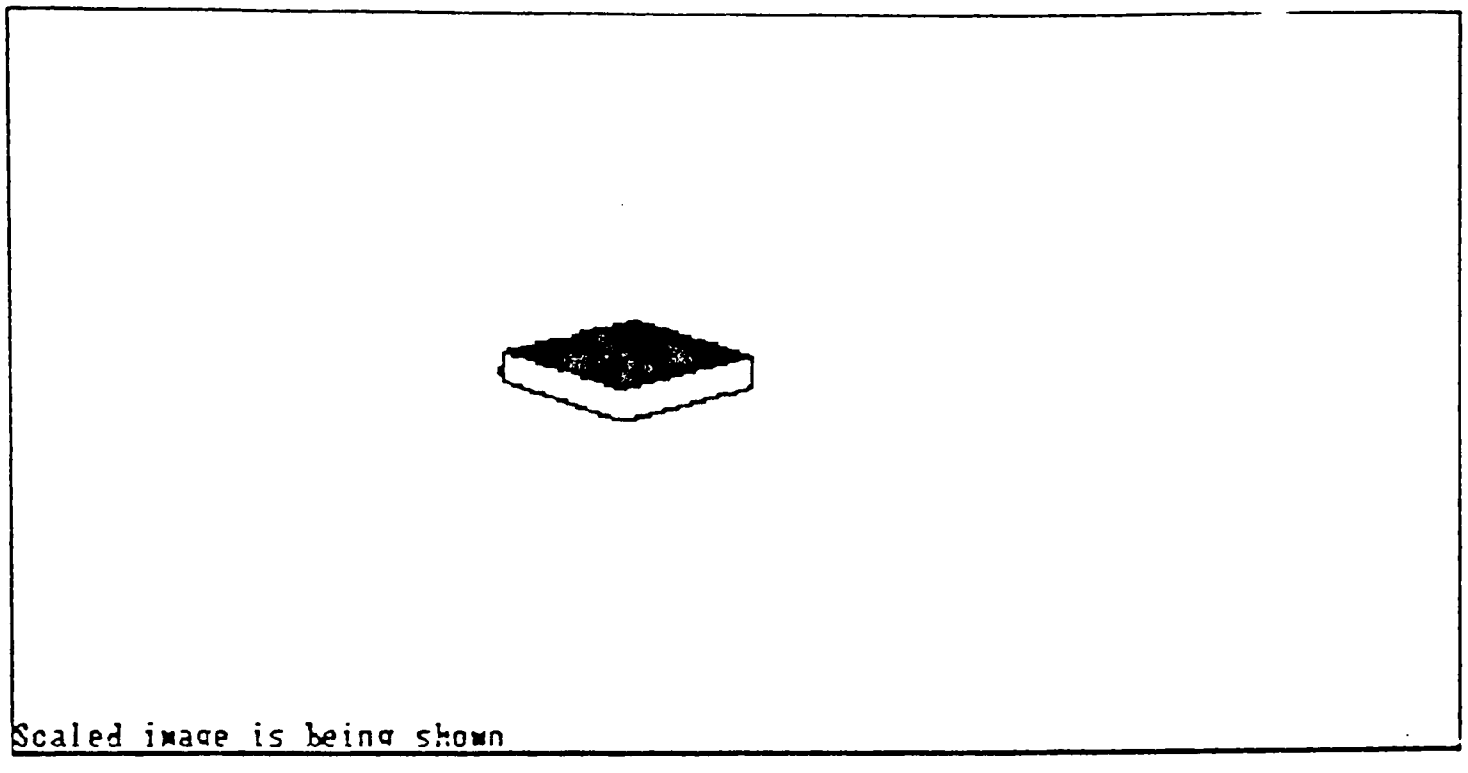


Figure 6.3. Image of one of the basic blocks

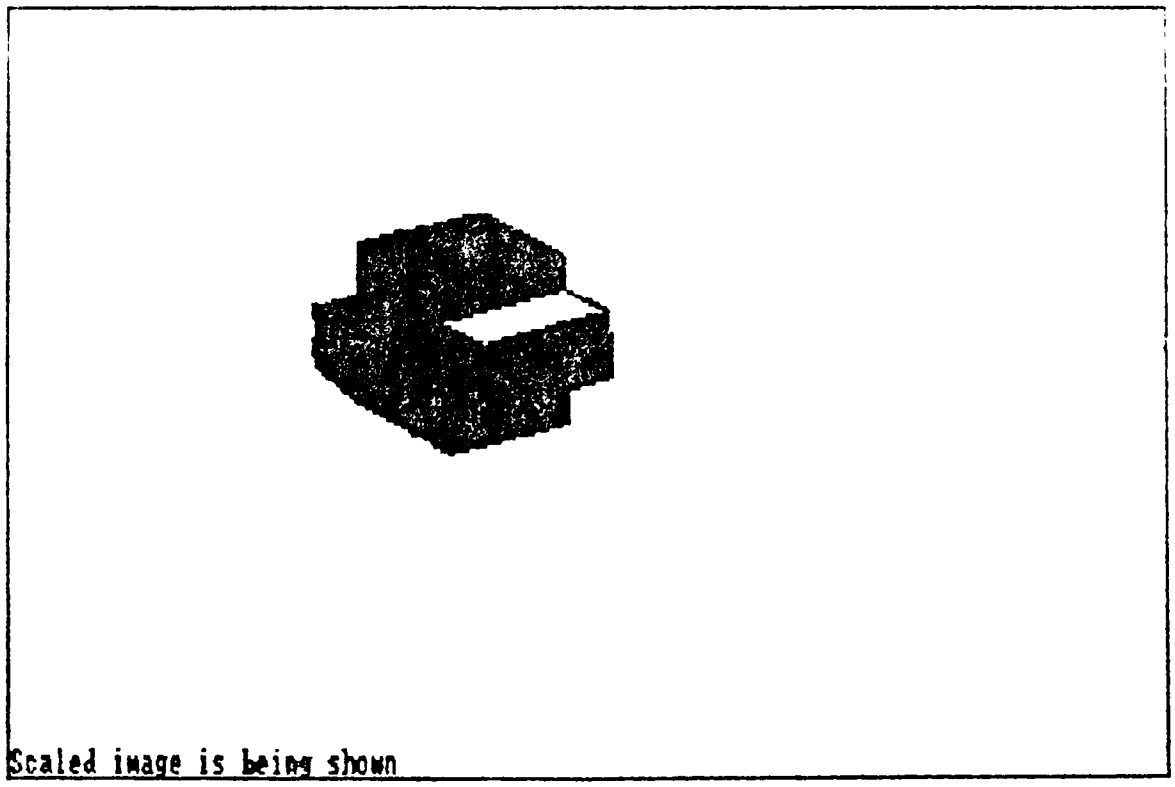


Figure 6.4. Image of the block sample

accomplished by replacing the value of every pixel with the average of the 3x3 pixels of its neighborhood (Chang 1987.) The procedure is shown as follows by the 3x11 ideal image,

3x11 ideal image,

100	100	100	50	50	50	50	100	100	100	100
100	100	100	50	50	50	50	100	100	100	100
100	100	100	50	50	50	50	100	100	100	100

after the Sobel operator, and

0	0	200	200	0	0	200	200	0	0	0
0	0	200	200	0	0	200	200	0	0	0
0	0	200	200	0	0	200	200	0	0	0

after smoothing,

0	0	67	133	133	67	67	133	133	67	0
0	0	67	133	133	67	67	133	133	67	0
0	0	67	133	133	67	67	133	133	67	0

and threshold at 60,

0	0	1	1	1	1	1	1	1	1	0
0	0	1	1	1	1	1	1	1	1	0
0	0	1	1	1	1	1	1	1	1	0

therefore, the image is getting smoothed.

Display the Final Sample

After edge detection, 20 to 30 pieces of data of a line are first selected. The Least Square Fitness Test (Banks & Carson 1982) is then applied to the data to obtain the gradient of a line. A vertex is found by solving the equations of two lines. The whole block sample is then redrawn and displayed on a TI Business-Pro computer with CGA graphics monitor.

The six-bit coding method obtains the edge information around the vertices. The parallelogram search method searches for the consistent parallelogram surfaces. Constructive solid geometry is then applied to construct the final image. Since perspective projection is used to display all the objects onto a viewing plane, the objects will appear distorted but give a natural sensation of depth. The program has a main menu with the following functions:

r(read blocksample) - use to display the block sample;

m(move from point) - move the eye coordinates toward or away from the object by pressing up arrow or down arrow;

a(enter at point) - enter the location of the object;

f(enter from point) - enter the location of the eye coordinates;

u(enter up point) - enter the orientation of the viewing plane with the related to the coordinate system;

z(enter view angle) - magnify the object (when view angle smaller) or demagnify the object (when view angle larger);

d(get world coordinates) - get world coordinates of the block sample;

i(read image) - read input image, perform 6-bit code analysis around vertices;

s(sample analysis) - find out the number of layers, the number of blocks in a layer and the location of blocks;

c(compare images) - compare image from different views and get an overall data file for the final sample ready for "read blocksample" function to display.

The above program is written in C language and is run on TI Business-Pro computer with color graphics adapter. For other types of graphics adapters, the specification of the type of graphics adapter and to the display of the images must need to be changed inside the program.

A pop-up window will appear in the center of the screen promoting the users to enter filenames or change the point locations. The data after the three-dimensional object recognition are stored into two separate files. The first file contains the number of vertices and the location in x, y, z dimensions. The second file contains the information of connections between vertices. An example of these two data files is shown in appendix B. When "read blocksample" is selected, the two data files are read and the objects will then be displayed. Other data files including the analysis after the six-bit coding and the parallelogram search method

are also saved for users easily to check. The final display of the sample in different views is shown in figure 6.7, figure 6.8, figure 6.9, and figure 6.10. Figure 6.11 shows the enlarged block sample with clipping after using "move From point" function since the object moves towards the eye-coordinates. Figure 6.12 shows the appearance of the block sample with the view angle of 30 degrees compared with the view angle of 60 degrees in figure 6.11. All "From" point, "At" point, and "Up" points locations are shown at the right-hand corner of the screen. In addition, a reference vertex is chosen with the minimum value of "x" from the original two-dimensional image. Then, the "x" position is set at "0" position and the "y" position is dependent on its layer position of the reference vertex. The bottom layer of the block sample is the position where the y-axes equal to zero. The next section is to discuss the Kanade model used in three-dimensional object recognition.

Three-Dimensional Object Recognition by Kanade Model

In this section, the same object is applied to the Kanade model for three-dimensional object recognition. The recognition is based on the three major steps from the Kanade labeling procedure. The first step is to assign a

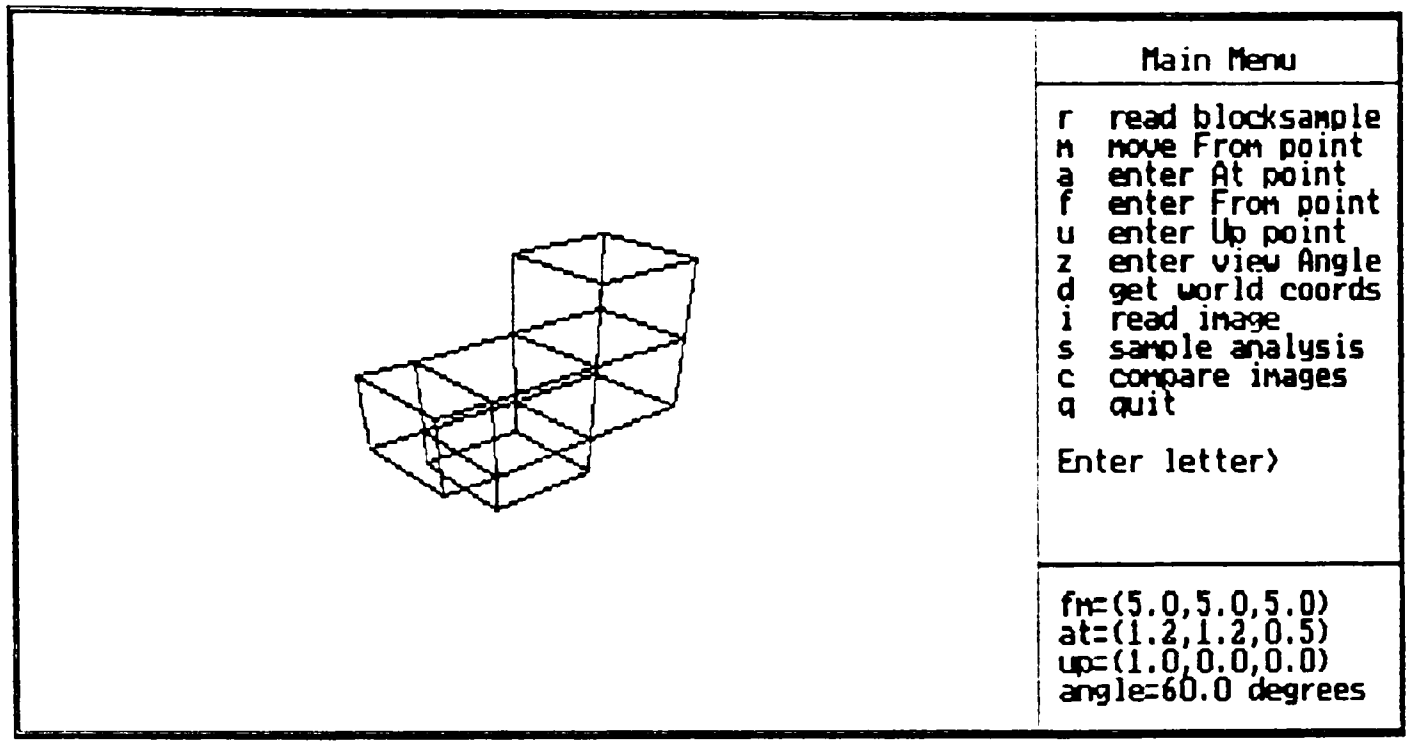


Figure 6.7. The view of the object with "Up" function at (1, 0, 0)

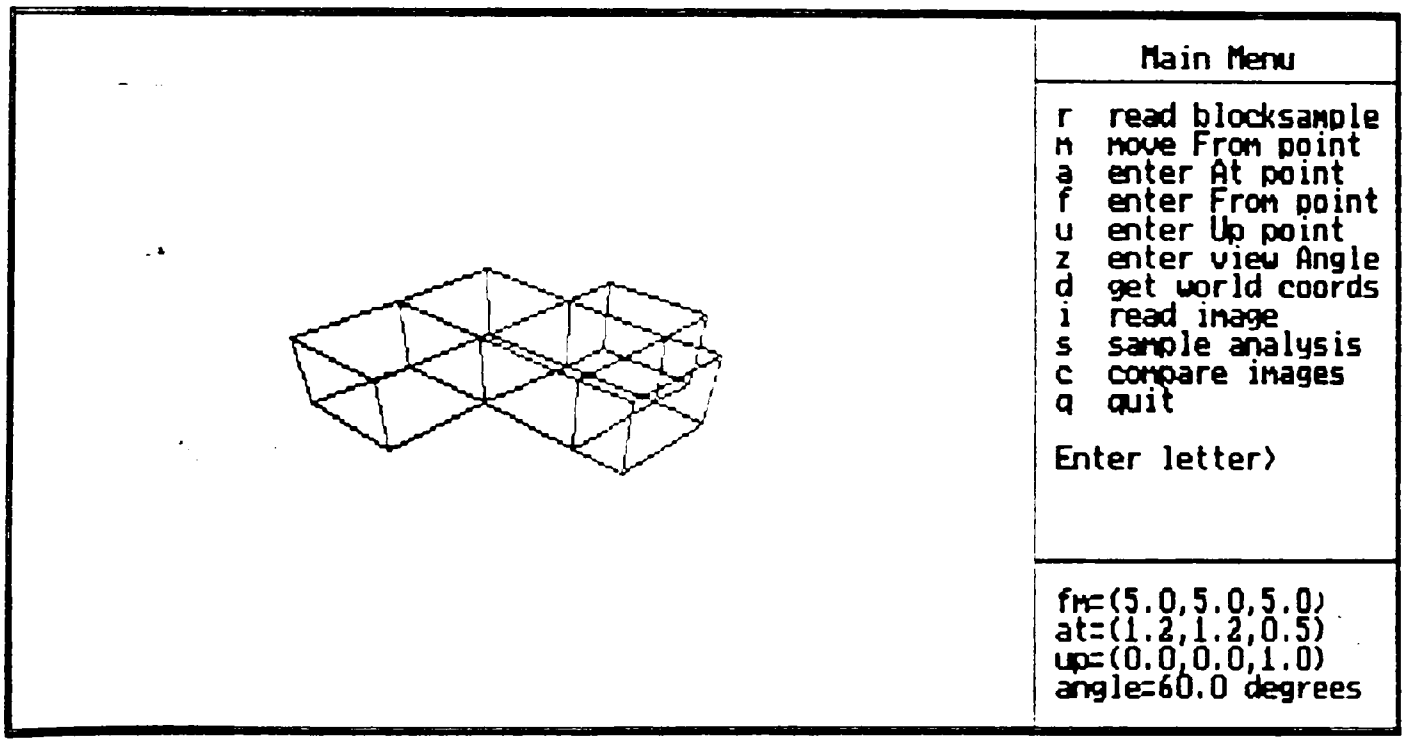


Figure 6.8. The view of the same object with "Up" function at (0, 0, 1)

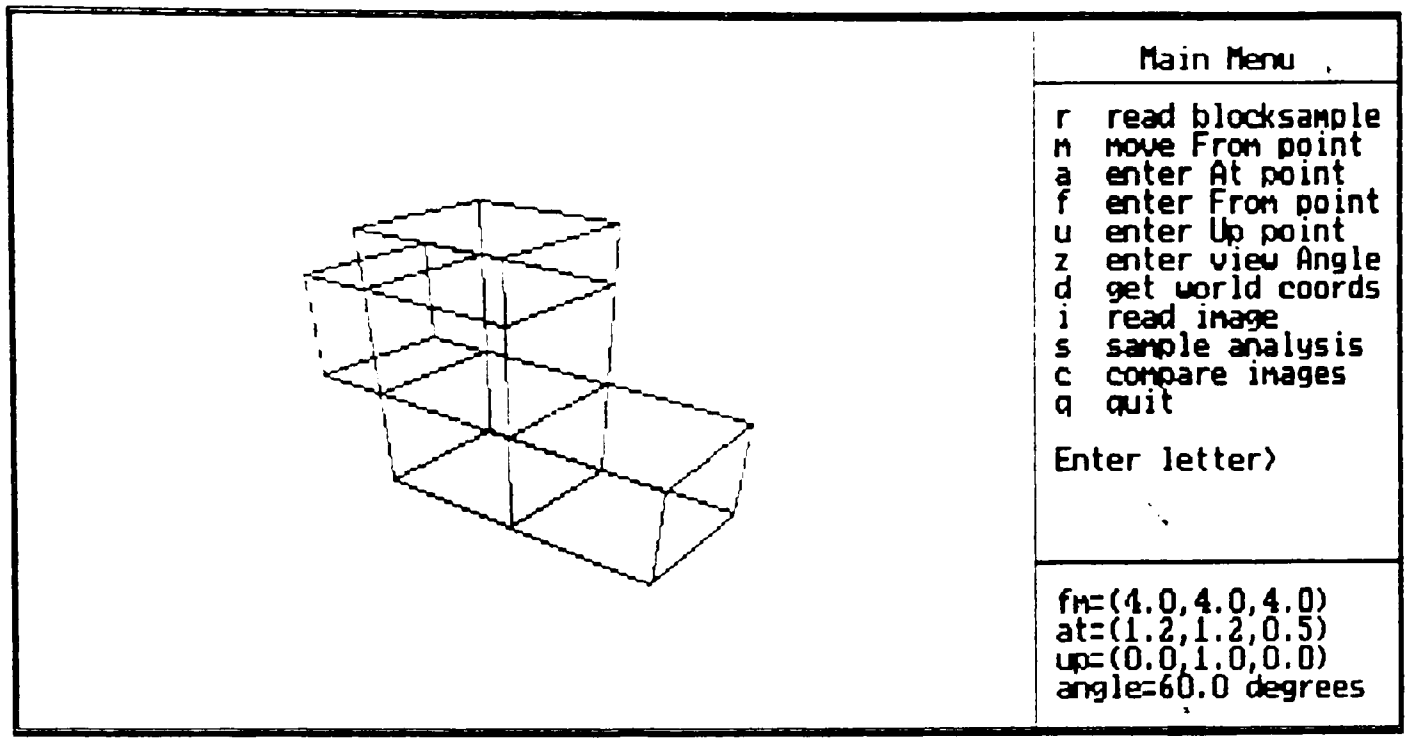


Figure 6.9. The view of the same object with "Up" function at (0, 1, 0)

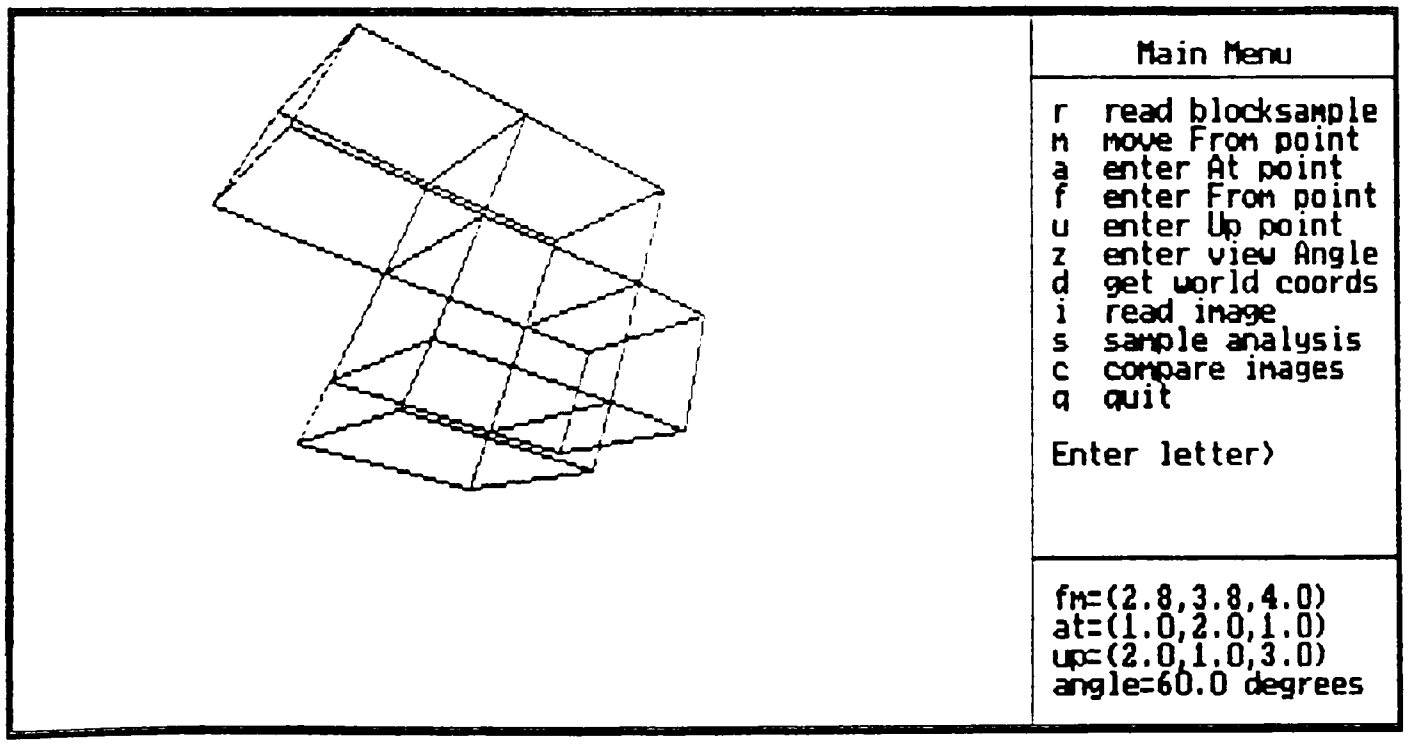


Figure 6.10. The view of the same object with "Up" function at (2, 1, 3)

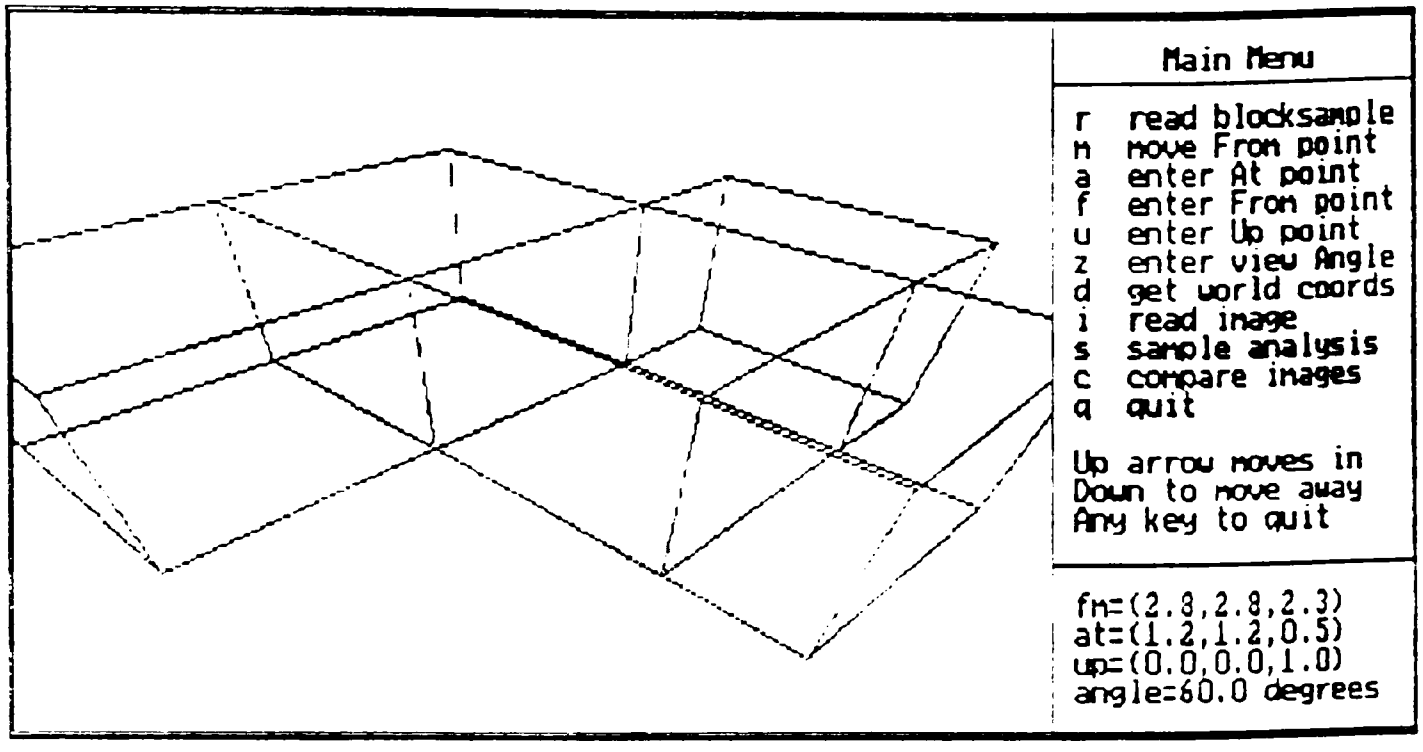


Figure 6.11.The view of the object after "move From point" function applied

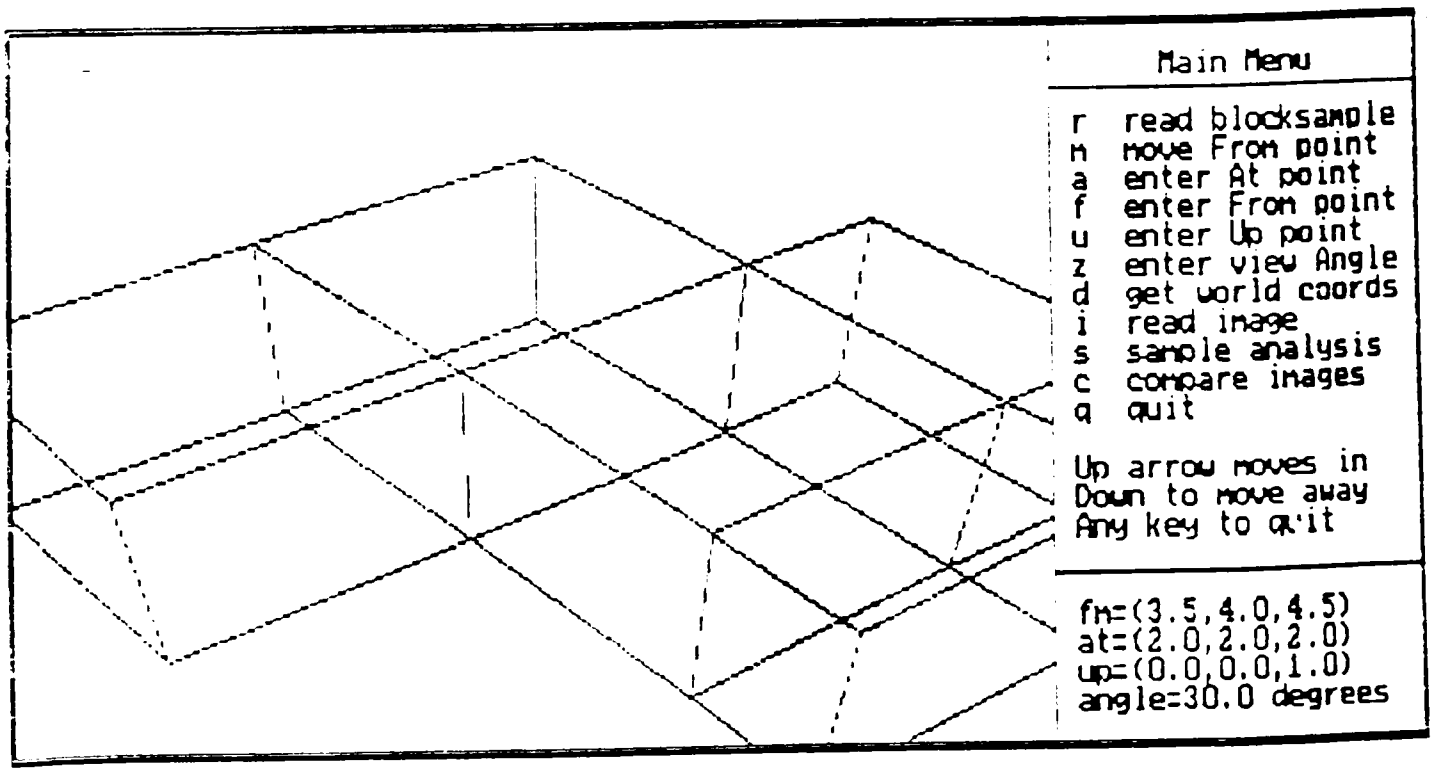


Figure 6.12. The view of the object after a smaller view angle applied

single label (+, -, ↑) along each line. A "+" line represents a convex edge with both of its planes visible from the camera. A "-" line represents a concave edge with both of its planes visible from the camera. A "↑" line represents an occluding edge: a convex edge with both of its planes on the same side of the edge as viewed from the camera, one occluding the other. A junction label (L, arrow, fork(Y), T, K, X, PSI) is then assigned on each junction. Figure 6.13 shows the interpretation of line drawing on the object. Figure 6.14 displays the table with the type of junction corresponding to the junctions. Five types of junctions are used. They are "L," "Y," "arrow," "T," and "X." Junction-M has the choice of different types of junctions with "Y," "T," "K," or "X."

The second step is to construct a Surface Construction Graph to show the connection between surfaces with the constraints (figure 6.15.) The "+" and the "-" labels become the constraints on the intersection line of surfaces. When the intersection line of surfaces is occluded, a label of "⊕" is denoted.

The third step is to assign a unique gradient to each surface. By using the same object as shown in figure 6.13, the gradients G_1, G_2, and G_3 can be found from the surfaces

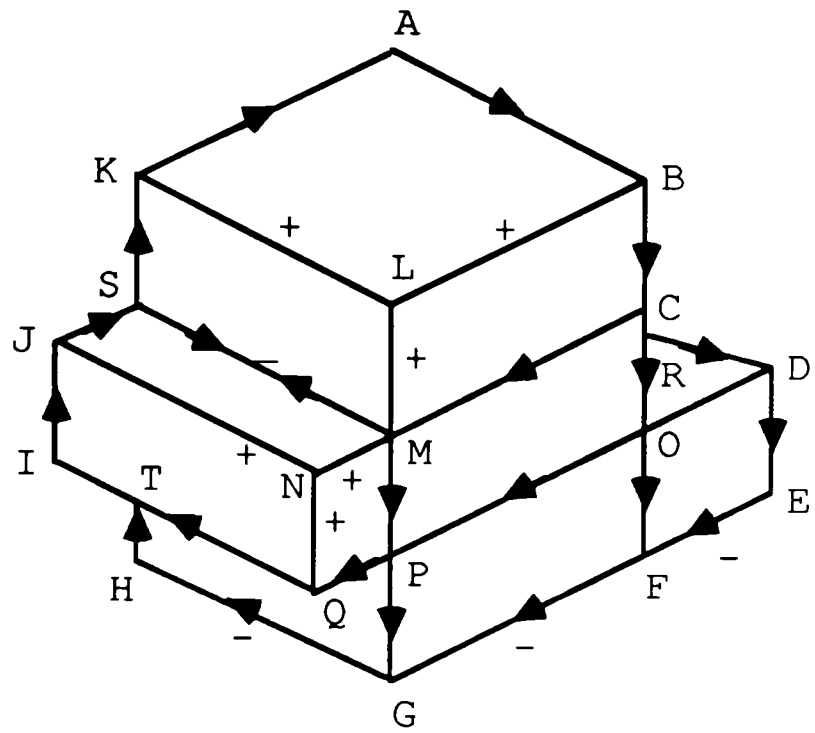


Figure 6.13 Interpretation of the block sample by line drawing

Junction \ Type	L	Y	ARROW	T	X
	A	L	B	C	P
	E	S	D	R	O
	H		G	F	
	I		J		
			K	T	
			N	Q	

Figure 6.14. Types of junction assigned on each vertex

Surfaces	Vertices	Surfaces	Vertices
S1	ABLK	S2	BLCM
S3	LMSK	S4	SMNJ
S5	NQIJ	S6	MPQN
S7	COPM	S8	DOR
S9	DEFO	S10	OFGP
S11	PGHTQ		

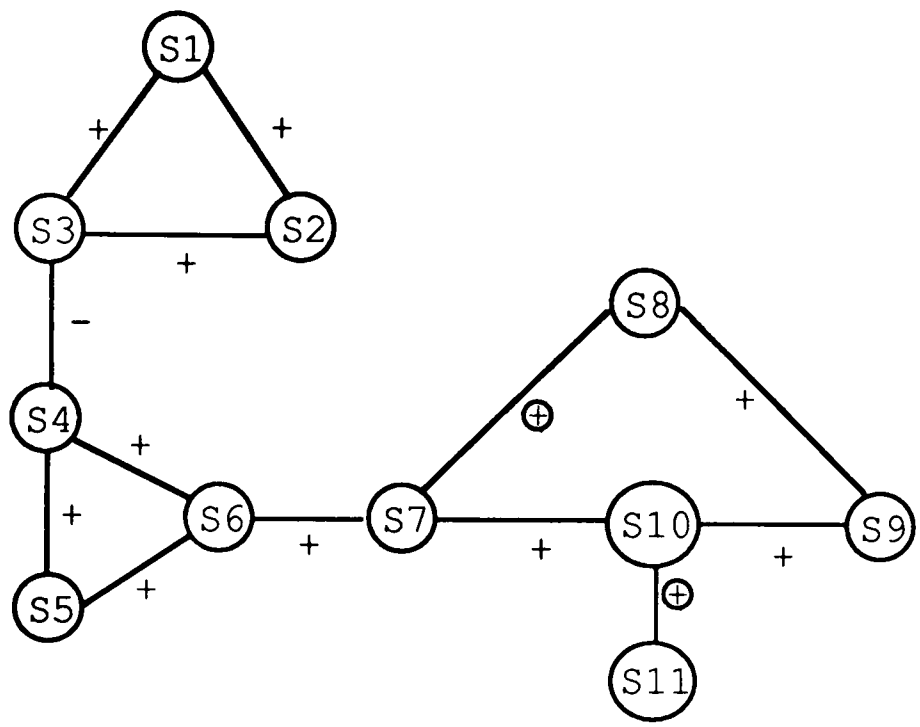


Figure 6.15. Surface Connection Graph to show the connection between surfaces and the constraints

S_1(ABLK), S_2(KLMS), and S_3(BCLM). Assume the block sample is made of cube with one unit on each side for easy calculation; the origin of the x-y position is assumed to be located at the vertex L with the given of z position equal to zero; then we have,

$$S_1 = \sqrt{3/2}\,x - \sqrt{1/2}\,y + z = 0$$

$$S_2 = -\sqrt{3/2}\,x - \sqrt{1/2}\,y + z = 0$$

$$S_3 = \sqrt{2}\,y + z = 0 \ .$$

The three-dimensional coordinates of the vertices for the three surfaces are found from the above equations. Other vertices can be found in a similar way as gradients remain unchanged along the direction of the lines. At this point, the analysis of the three-dimensional object recognition by Kanade is completed.

Comparisons of the Two Models

The two models for three-dimensional object recognition are analyzed from the previous sections. The following is the comparison between the two models. The labeling procedure for the modified Kanade model is simple. It requires two different types of junctions, "Y," and "T," rather than seven different types of junctions. From the analysis in the last section, there is still a problem of labeling junction-M among the junction labels of the Kanade model. The parallelogram search method applied to "Y" and "T" vertices searches for parallelogram surfaces; this is in contrast with the Surface Construction Graph which searches

for planar surface with constraints from labeling. Besides, the comparison of the two models is also from the labeling procedure, computing time, and computing memory requirement.

By checking at the junction table (appendix A) of the Kanade model, the search of the consistent labeling for vertices of the picture scene needs a certain amount of time. In the Kanade model, the search for the corrected junction type to each vertex is in the range provided from the junction table (appendix A). An example which demonstrates the filtering algorithm with the possible labeling for each junction in the context of the entire picture is shown in appendix A. The labeling algorithm compares adjacent pairs of junctions, sees if their constraints can be satisfied, and strips out inconsistent labels as it proceeds. However, the six-bit coding method is used to find the presence of lines around vertices. For every vertex, there is only a maximum of six lines for searching. Two important points need to consider for the long computing time in Kanade model rather than in the modified Kanade model. First, from appendix A, most of the time is spent in the iterative process of finding the consistent label junctions to each junction. The possible labels are checked with other label junctions to find the consistency. From the appendix A, filtering inconsistent

label junctions is started by initial constraints of putting arrows in a clockwise direction along the outer boundary of the picture scene. Although only those combinations which give consistent labels to lines are to be explicitly selected, the remaining combinations of junctions to be filtered can have up to 1x4x1x4x1x4x19 = 1,216. Comparing with the six-bit coding method, the searching of the number of lines is dependent on the number of vertices. By using the same example, there are only 7x6 = 42 lines for searching. As a result, more amount of time in the Kanade model is contributed to search and save the possible labeling than in the modified Kanade model. Second, the surface connection graph is used to provide information of connections between surfaces. The information is stored in the data file. However, the parallelogram surface searching method uses the data obtained from the six-bit coding method, which has all the line information around a vertex in one data. Therefore, the latter method provides a faster access to the information and has a faster speed to execute the program.

Other significant computer time for the Kanade model is used to solve the simultaneous equations in order to get the three-dimensional coordinates of the block sample; in the modified Kanade model, the three-dimensional coordinates are

found by the pre-defined blocks from the database, using the principle of constructive solid geometry. The addition of data for the three-dimensional object coordinates in the modified Kanade model is faster than the solving of simultaneous equations in the Kanade model since solving equations involves the matrix manipulation.

Computer memory can be compared by the use of the two models, too. In the modified Kanade model, each vertex is represented by a computer word with line information on it. The amount of memory used is proportional linearly to the number of vertices. In the Kanade model, one computer word represents a line label or a junction type. With a simple scene, the amount of memory saved is not significant. But, as the complexity of the scene increases, more amount of memory can be saved. The smaller the amount of memory utilized to represent the information, the faster the speed to execute the program. Assume each line of the picture must be assigned a single label along its entire length. Therefore, the number of line labels is equal to the number of lines in the picture. Each line and junction label is represented by one computer word. From the surface connection graph of the Kanade model, the information is explicitly stored by means of links, which connect a pair of related regions and which include description of the

constraints on their gradients. From figure 6.16, three computer words are used to name the link and to indicate the link between the two surfaces R2 and R3. Another two computer words are also used to link the constraints of the gradient (in this case, +) and the given link. The number of computer words used from three types of different picture scenes is shown in figure 6.17. Note that the estimation of the memory space used in the Kanade model does not include the memory used for storage of other possible labeling of junctions. Only a unique labeling for each junction in the picture is selected. As shown from figure 6.17, more than 75% of the memory is saved in the modified Kanade model.

Moreover, the modified Kanade model is especially good in recognizing objects with multiple layers because of its simplified recognition of the object; the Kanade model becomes complicated in this case as only one single view of the image is taken. The comparisons of the two models are summarized in figure 6.18. In the next chapter, the modified Kanade model will be concluded with recommendations.

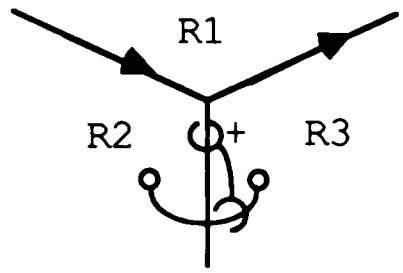


Figure 6.16. A link between surfaces in the "Y" junction

type / computer words used in each case	Cube	Two cubes of different size contacted together	Image used from Fig.6.13
line label	9	18	27
junction label for vertices (database of storage)	22	65	100
six-bit coding method	7	15	20
% of memory saved by using six-bit coding method over the Kanade model	24/31x100% = 77.42%	68/83x100% = 81.93%	107/127x100% =84.25%

Figure 6.17 Comparisons of different amount of memory used from the two models in terms of computer words

	Kanade model	modified Kanade model
line information	labeling by (+,-, ↑) and junctions	six-bit code method
storage of line information	large data file	bit format
type of junctions	at least five	only two
surface searching	planar surface	parallelogram surface of vertices "Y" and "T"
gradient known	required	required
reference vertex	required	required
three-dimensional coordinates found	by gradients and planar equations	addition of the world coordinates by database of predefined blocks using CSG principle
computer memory required	modified Kanade model saves memory more than 75% of Kanade model	
computer time	relatively long computer time in Kanade model in the following conditions: (1) large combination of junctions after initial constraints e.g. cube case= 1,216	Searching of possible lines e.g.cube case =42
	(2) relatively large amount of memory used,speed is slower	small amount of memory utilized, speed is faster
	(3) find 3D coordinates by solving simultaneous eqns	find 3D coordinates by addition of data

Figure 6.18. Summaries of the comparisons between the two models

CHAPTER VII

CONCLUSIONS AND RECOMMENDATIONS

Review of the Modified Kanade Model

As seen from the previous chapter, the modified Kanade model first makes use of the visual information for edge detection. The modified Sobel operator with smoothing performance for edge detection reduces the noise of the images. The six-bit coding method then is applied to store the information of the surrounding vertex information with the purpose of saving computer memory. The parallelogram search method makes the recognition easy since the Kanade model has more labeling type junctions and the search for consistent interpretations is exhaustive. The final object is then reconstructed from the primitive blocks. By using perspective projection with the mathematical model discussed in Chapter III, different locations and orientations of the objects can be displayed with a natural sensation of depth. As a result, the modified Kanade model can be considered as an effective and capable model in terms of computer memory requirement to save data and computing time for three-dimensional object recognition.

There are constraints of using this algorithm. The constraints of this algorithm are that cubic and rectangular blocks are contacted at least by one surface, either with the floor or with other blocks. The algorithm works well and is faster when the blocks are put in uniform direction; otherwise, the gradients have to be changed to deal with different block directions. The algorithm is also applied to recognize trapezoidal blocks. Because of the gradient constraints, solid blocks and trapezoidal blocks can be distinguishd (figure 7.1). In the case of multiple layers, when the upper layer of the block is larger than the lower layer of the block, the algorithm fails as parallelogram surfaces cannot be found from each corner direction. Figure 7.2 shows the condition when the algorithm fails. With other kinds of blocks like pyramids or circles, the parallelogram surface searching fails and other algorithms will have to be used.

Further Recommendations

The modified Kanade model is used for three-dimensional object representations of solid objects with trihedral vertices. Two basic considerations can enhance the process. They are to extend the present program and to apply off-line programming for robots.

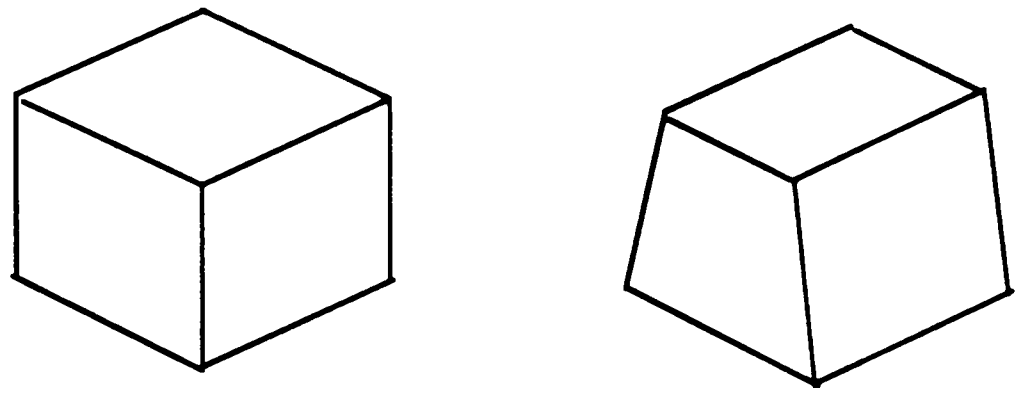
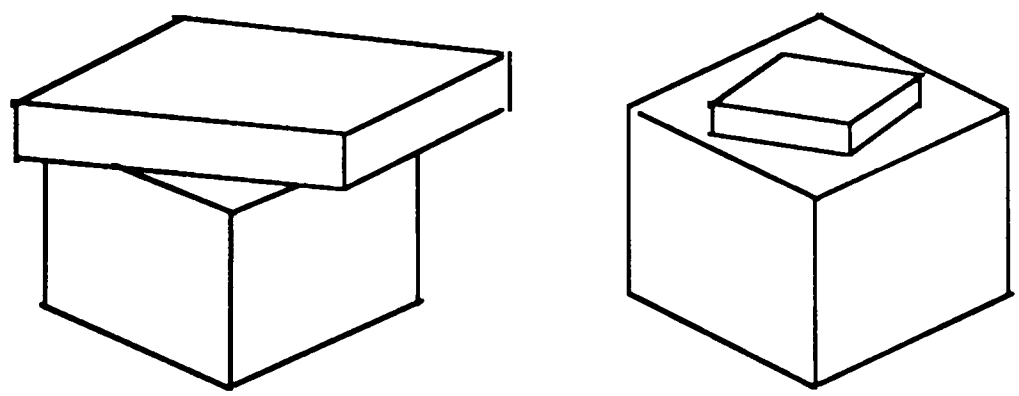


Figure 7.1. Gradient constraints are used to distinguish solid block and trapezoidal block



(a) (b)

Figure 7.2 The algorithm cannot be applied on (a) but on (b)

There are several enhancements that can be added to the three-dimensional viewing package. Functions that interactively draw three-dimensional objects may be put into the program. Several viewports that show the objects at different perspectives on the screen at the same time can

help the design of the objects in the future. Other possible extentions of the program would be to add color, or fill patterns to the objects so that they look more realistic. The hidden surface removal and the shading models (Foley & Dam 1982) can be attached to the display for the objects too. Three-dimensional object recognitions other than solid objects with trihedral vertices can be added to the algorithm for further research.

On the other hand, robots are applied widely in the flexible manufacturing areas. The manipulation of the objects by robots is done in real time. However, the three-dimensional object recognition can be done in off-line programming. Manipulators can be programmed in simulation by developing a set of instructions on an independent computer system, and then the instructions can be used to control the manipulators at a later time. The off-line programming does not require the physical use of the robots, and production, therefore, is not disturbed. The off-line programming can be accomplished in a sophisticated CAD system that links vision and robots together (figure 7.3). In the future, robots and vision systems will require a significant amount of integration with existing CAD systems. The three-dimensional object recognition becomes one of the major contributions of this research in the flexible manufacturing area.

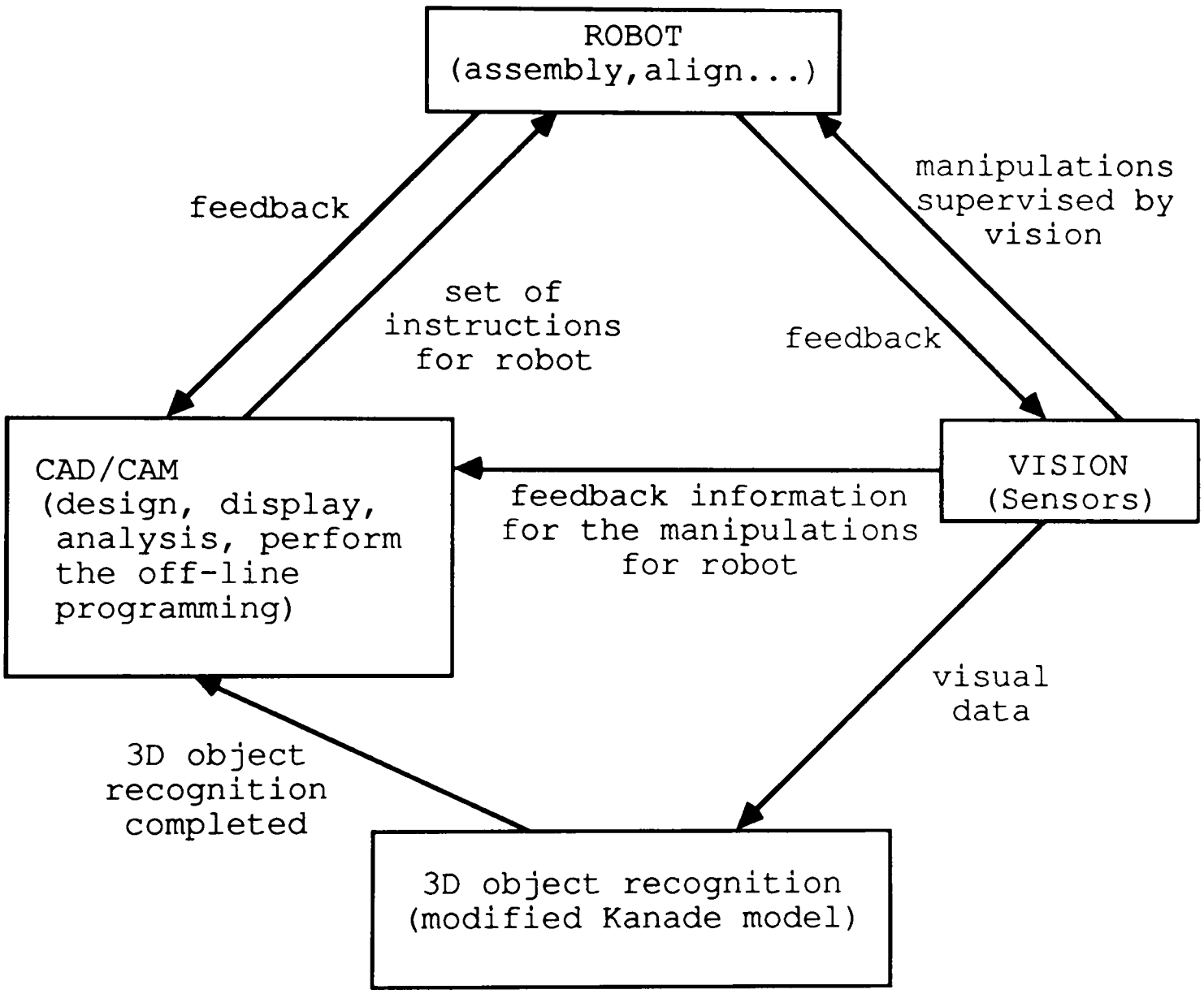


Figure 7.3. The modified Kanade model applied in CAD, vision, and robots

BIBLIOGRAPHY

Banks, Jerry & Carson, John S. (1982), Discrete-Event System Simulation, Prentice-Hall.

Berthold, Klaus & Horn, Paul (1986), Robot Vision, MIT Press.

Boyse, J.W.(1979), "Data structure for a solid modeller," NSF Workshop, Univ. Pennsylvania, May.

Brown, C. (1982), Computer Vision, Princeton.

Browne, Arthur (1986), Vision and Information Processing for Automation, Plenum.

Chang, Heng H. (1987), "Image Thresholding for Blueprint Image Extraction," SME Machine Vision'87.

Charniak, Eugene & McDermott, Drew (1985), Introduction to Artificial Intelligence, Addison-Wesley.

Clowes, M.B. (1971), "On seeing things," Artificial Intelligence , vol 2, p79-116.

Codd, E.F. (1970), "A relational model for large shared data banks," Commum ACM 13, No 6, p377-387.

Cohen, Paul R. & Feigenbaum, Edward A. (1982), The Handbook of Artifical Intelligence, vol 3, Kauffman, Inc.

Deo, Narsingh (1987), Graph Theory with Applications to Engineering and Computer Science, Prentice-Hall.

Hall, Ernest L. (1986), Computer Image Processing & Recognition, Academic Press.

Hall, Ernest L. (1988), "Machine Vision Research," Manufacturing Engineering, vol 101, No 1, p50-55, July.

Fletcher, W.I. (1982), An Engineer Approach to Digital Design, Prentice-Hall.

Foley, J.D. & Dam, A.V. (1982), Fundamentals of Interactive Computer Graphics, Addison-Wesley.

Fu, King-Sun (1986), Handbook of Pattern Recognition and Image Processing, Academic Press.

Granky, Paul (1986), Computer Integrated Manufacturing, Prentice-Hall.

Gunnaskon, Kristjan T. & Prinz, Friedrich B. (1987), "CAD Model-Based Localization of Parts in Manufacturing," IEEE Computer, p66-76, July.

Guzman, A. (1968), "Decomposition of a visual scene into three-dimensional objects in a visual scene," AFIPS Fall Joint Conferences, vol 33, p291-304.

Huffman, D.A. (1971), "Impossible objects as nonsense sentences," Machine Intelligence, vol 6, p295-323.

Inoue, H. & Inaba, M. (1984), "Hand-eye Coordinate in Rope Handling," Robotics Research, MIT Press, Cambridge, Massachusetts.

Kanade, T. (1981), "Recovery of the three-dimensional shape of an object from a single view," Artificial Intelligence, vol 17, p409-460.

Mackworth, A.K. (1973), "Interpreting pictures of polyhedral scenes," Artificial Intelligence , vol 4, p121-137.

Martelli, Alberto (1976), "An application of heuristic search method to edge and contour detection," Commum ACM, No 19, p73-83.

Minsky, Marvin (1975), "A Framework For Representing Knowledge," The Psychology of Computer Vision, McGraw-Hill.

Offen R.J. (1985), VLSI Image Processing, McGraw-Hill.

Paul, R.P. (1981), Robot Manipulators, Cambridge, Massachusetts, MIT Press.

Requicha, A.A.G. & Voelcker, H. (1982), "Solid Modeling: A Historical Summary and Contemporary Assessment," IEEE Computer Graphics and Applications, p9-24, March.

Roberts, L.G. (1965), "Machine perception of three-dimensional solids," Optical and Electro-optical Information Processing, MIT Press.

Senze, J.C. (1988), Advance in Machine Vision, Springer-Verlag.

Shi, J. & Prat, R. (1988), "3D object localization and automatic tracking," Advanced Sensor Technology, IFS Ltd, Bedford, UK.

Shirai, Y. & Inoue, H. (1973), "Guiding Robot by Visual Feedback in Assembling Tasks," Pattern Recognition, No 8, p99-108.

Smith, D.N. & Wilson, R.C. (1982), "Industrial Robots: A Delphi Forecast of Markets and Technology," Society of Manufacturing Engineers, Dearborn, Mich.,.

Voelcker, H.B. & Requicha, A.A.G., (1977), "Geometric modelling of mechanical parts and processes," IEEE Computer, vol 10, p48-57, December.

Waltz, David (1975), "Understanding line drawings of scenes with shadows," The Psychology of Computer Vision, McGraw-Hill, p19-93.

Wright, P.K. (1988), Manufacturing Intelligence, Addison-Welsey.

APPENDIX A

A DETAILED DESCRIPTION OF THE KANADE MODEL AND THE MODIFIED KANADE MODEL IN RECOGNITION PROCESS

The following program shows how the Kanade model interprets the labeling procedure. The interpretation of L, fork, arrow, and T of the Kanade junction is shown in figure A.1. The terminologies that are used here are edge, line, and junction. An edge can be classified according to its three-dimensional physical meaning of the scene. The following terms and labels are used:

convex	+	edge along which two surfaces meet and form a convexity,
concave	-	edge along which two surfaces meet and form a concavity,
occluding	-> or <-	edge along which one surface occludes another.

A line can therefore be labeled with one of the labels (+, -, -> or <-), according to its physical meaning of the corresponding edge. For example, a line in convexity depicts a convex edge. Junctions are classified according to the number of lines meeting at the junctions and their geometrical configurations in the picture. The step numbers refer to figure A.2, which summarizes the label sets assigned to each junction.

Steps of labeling procedure of figure A.2:

1. Each junction is given a set of possible labels drawn from figure A.1 according to its junction types. Assume

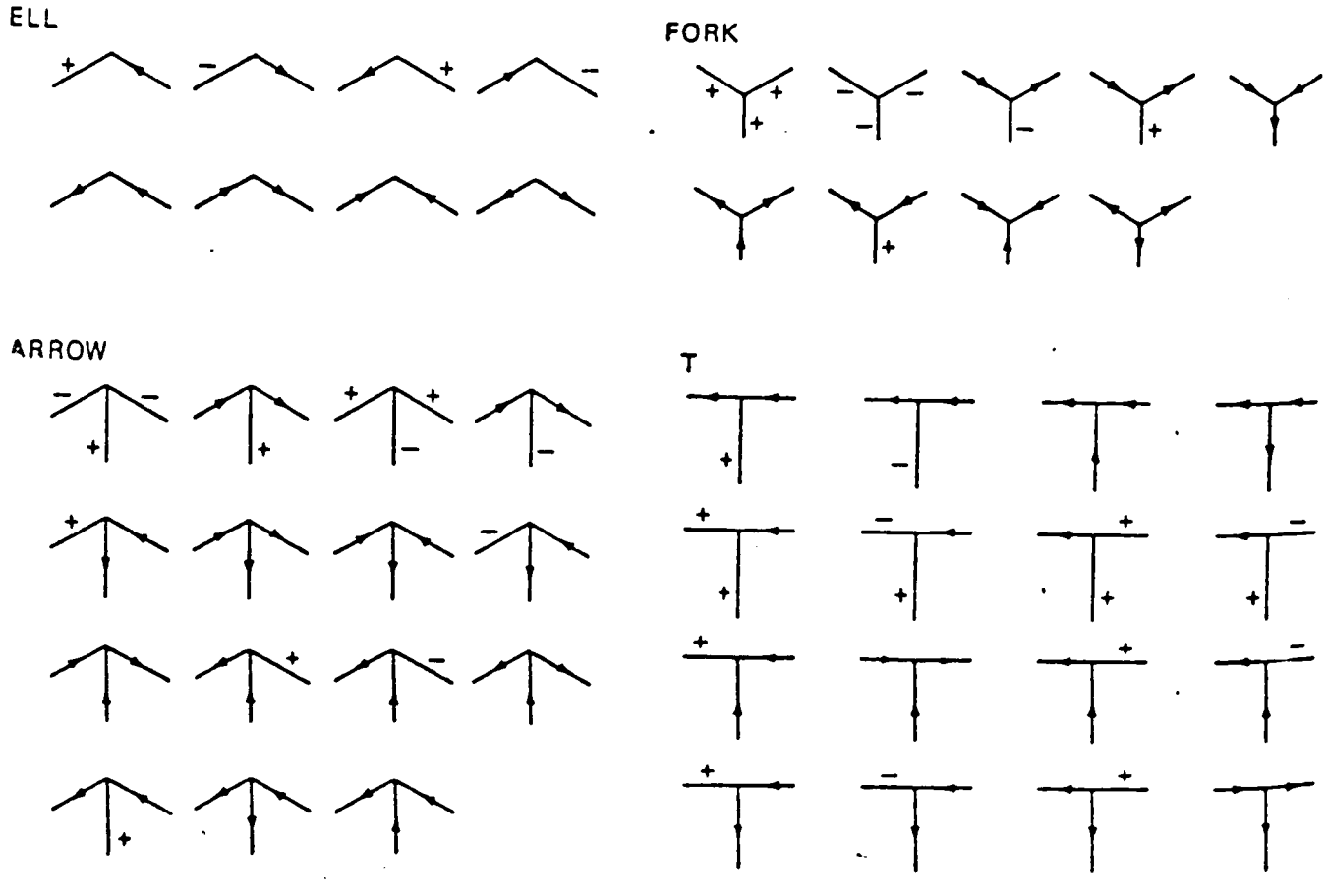


Figure A.1. The interpretation of L, fork, arrow, and T of the Kanade junction

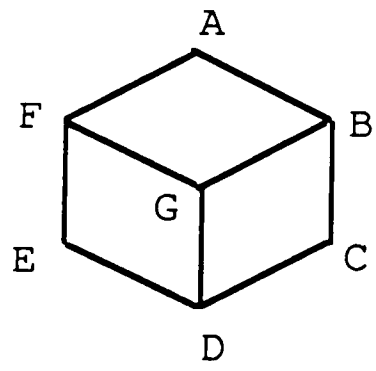


Figure A.2. A cube to explain the Kanade labeling process

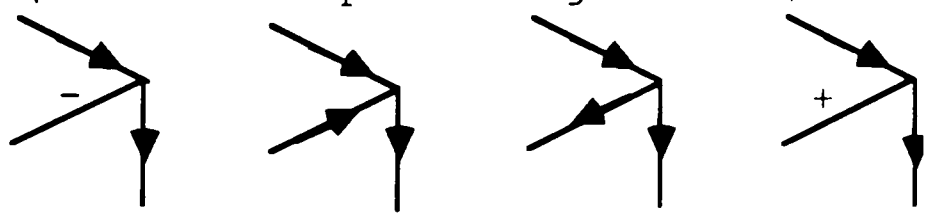
the procedure starts with junction A of figure A.2 and follows by junction B, C, D, E, F, G. The junction type belonging to each junction can be labeled in any of these ways as listed in figure A.1.

Junction	Type of junction
A	L
B	Arrow
C	L
D	Arrow
E	L
F	Arrow
G	Y

2. Initial constraints are that the outer boundaries should have the label, -> or <-; the arrows surround the object clockwise. Under these conditions, the possible label for each junction is/are:

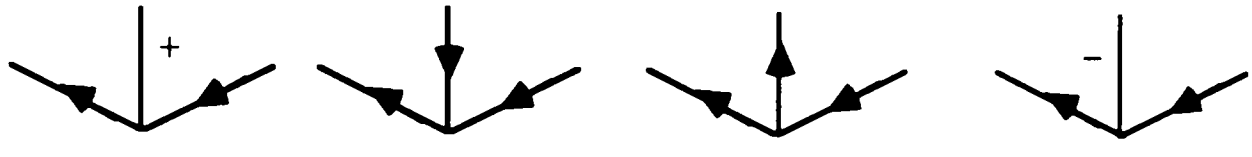
junction A (1 out of 8 possible junctions)

junction B (4 out of 15 possible junctions)



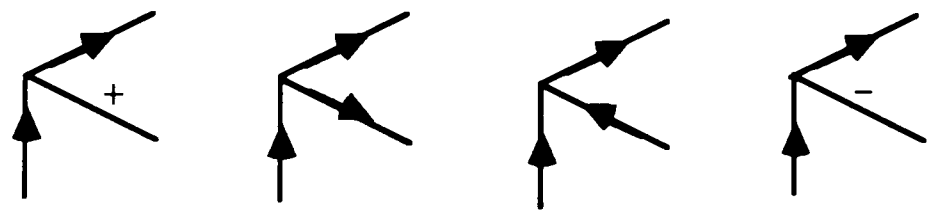
junction C (1 out of 8 possible junctions)

junction D (4 out of 15 possible junctions)



junction E (1 out of 8 possible junctions)

junction F (4 out of 15 possible junctions)



junction G (19 possible assignments exist since a Y junction is rotational)

all "Y" type of labels of figure A.1.

Each junction now has a set of possible junction labels which have been filtered. Only those combinations which give consistent labels to lines are to be explicitly selected. For the first junction, one junction label is selected from the set of possible junction labels for it. For the second junction, one junction label is selected which is consistent

with the labels given to the first junction. Similarly, the third junction is followed. If no more consistent label exists for the present junction, the search backtracks.

3. The search is combined with the surface connection graph (SCG) to perform further filtering of the junctions. A pair of nodes are connected by an arc if a junction involving the corresponding pair of regions has been given a junction label which has a link between them. The gradient space is used to check the consistency of label. If the link of the present junction label adds an arc and results in the formation of a new loop in the SCG, the spanning angle of this arc is checked against paths which connect the same pair. If they have a null intersection, the partial interpretation is inconsistent. For example, in the following configuration figure A.3, the corresponding partial SCG of junction B consists of a single arc (surface 2 -> surface 1). Then, junction F is given a junction label. Two new arcs are added to the SCG: first (surface 2 -> surface 3), and then (surface 3 -> surface 1). Its gradients are found and are along the spanning angle as shown in figure A.4. Since both the spanning angles do not have an overlapping area, the labeling junctions turn out to be inconsistent; otherwise, the intersection is not null, the labeling junctions are consistent. Therefore, if any inconsistency in the configuration is detected, either in

the combination of junction labels or in surface orientations, the search process backtracks one step and searches for the next combination.

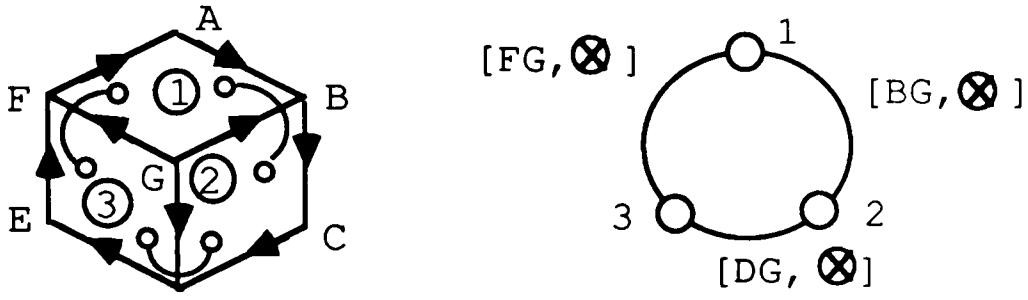


Figure A.3. The labeling junction of figure A.2 and its SCG

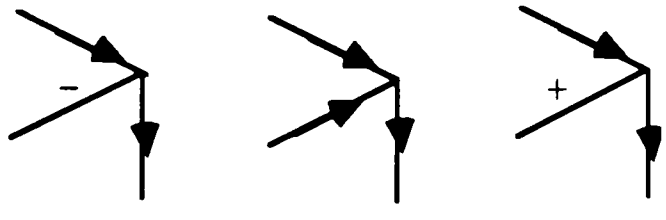
(a) Surface (2->3->1) (b) Surface (2->1)

Figure A.4. Gradients with spanning angles

4. The resulting labeling junctions are obtained as follows and the interpretation of "cube" line drawing is shown in figure A.5.

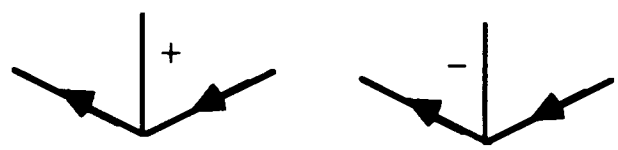
junction A

junction B



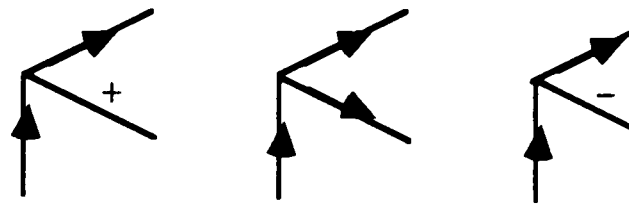
junction C

junction D

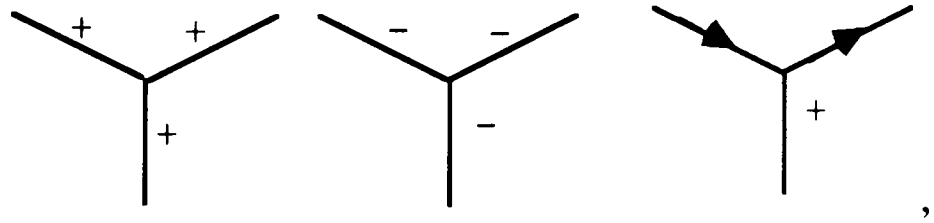


junction E

junction F



junction G



,

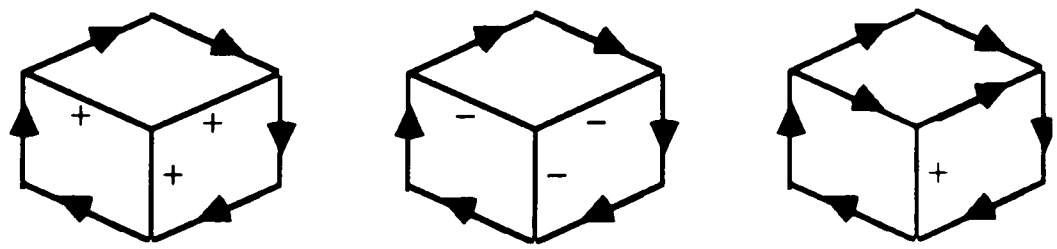


Figure A.5. Interpretation of "cube" line drawing

For the modified Kanade model, the information is stored by the use of the six-bit coding method. It is done by searching for the presence of lines along vertices. The analysis is shown in figure A.6. The parallelogram search method is used to search for the presence of parallelogram surface by using the information provided from figure A.6. Please refer to Chapter V for more details.

Junction \ line	rgtdn*	rgtup*	lup*	ldown*	up	down	value
A	1	0	1	0	0	0	40
B	0	0	1	1	0	1	13
C	0	0	1	0	1	0	10
D	0	1	0	1	1	0	22
E	1	0	0	0	1	0	33
F	1	1	0	0	0	1	49
G	0	1	0	1	0	1	21

* rdtdn = right down
rdtup = right up
lup = left up
ldown = left down

Figure A.6. Analysis of line information of figure A.2 by using six-bit coding method

APPENDIX B

DATA OF THE VERTICES AND THEIR CONNECTIONS

```
/*  Data of vertices in x, y, z direction       */
40
1.500000 1.000000 1.000000
1.500000 1.000000 0.000000
1.500000 0.000000 0.000000
1.500000 0.000000 1.000000
0.500000 1.000000 1.000000
0.500000 1.000000 0.000000
0.500000 0.000000 0.000000
0.500000 0.000000 1.000000
2.500000 1.000000 1.000000
2.500000 1.000000 0.000000
2.500000 0.000000 0.000000
2.500000 0.000000 1.000000
1.500000 1.000000 1.000000
1.500000 1.000000 0.000000
1.500000 0.000000 0.000000
1.500000 0.000000 1.000000
0.500000 2.000000 1.000000
0.500000 2.000000 0.000000
0.500000 1.000000 0.000000
0.500000 1.000000 1.000000
0.000000 2.000000 1.000000
```

```
0.000000 2.000000 0.000000
0.000000 1.000000 0.000000
0.000000 1.000000 1.000000
1.500000 2.000000 1.000000
1.500000 2.000000 0.000000
1.500000 1.000000 0.000000
1.500000 1.000000 1.000000
0.500000 2.000000 1.000000
0.500000 2.000000 0.000000
0.500000 1.000000 0.000000
0.500000 1.000000 1.000000
1.500000 2.500000 1.000000
1.500000 2.500000 0.000000
1.500000 2.000000 0.000000
1.500000 2.000000 1.000000
0.500000 2.500000 1.000000
0.500000 2.500000 0.000000
0.500000 2.000000 0.000000
0.500000 2.000000 1.000000
```

```
/*  Data that shown the connections bteween vertices                    */

120

1     5     8     -4    5     6     7     -8    6     2     3     -7
1     4     3     -2    8     7     3     -4    6     5     1     -2
9     13    16    -12   13    14    15    -16   14    10    11    -15
9     12    11    -10   16    15    11    -12   14    13    9     -10
17    21    24    -20   21    22    23    -24   22    18    19    -23
17    20    19    -18   24    23    19    -20   22    21    17    -18
25    29    32    -28   29    30    31    -32   30    26    27    -31
25    28    27    -26   32    31    27    -28   30    29    25    -26
33    37    40    -36   37    38    39    -40   38    34    35    -39
33    36    35    -34   40    39    35    -36   38    37    33    -34
```

PERMISSION TO COPY

In presenting this thesis in partial fulfillment of the requirements for a master's degree at Texas Tech University, I agree that the Library and my major department shall make it freely available for research purposes. Permission to copy this thesis for scholarly purposes may be granted by the Director of the Library or my major professor. It is understood that any copying or publication of this thesis for financial gain shall not be allowed without my further written permission and that any user may be liable for copyright infringement.

Disagree (Permission not granted)	Agree (Permission granted)
_____	Lori Henry
Student's signature	Student's signature
_____	3/14/90
Date	Date