MODEL-BASED RECOGNITION OF POLYHEDRAL OBJECTS

A Master's Thesis

Presented by

Hüseyin UZUNALİOĞLU

to

the Graduate School of Natural and Applied Sciences

of Middle East Technical University

in Partial Fulfillment for the Degree of

MASTER OF SCIENCE

in

ELECTRICAL AND ELECTRONICS ENGINEERING

MIDDLE EAST TECHNICAL UNIVERSITY

ANKARA

July, 1992

Approval of the Graduate School of Natural and Applied Sciences.

Prof. Dr. Alpay Ankara

Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Prof. Dr. Tuncay Birand

Chairman of the Department

We certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science in Electrical and Electronics Engineering.

Prof. Dr. Mete Severcan

Supervisor

Examining Committee in Charge:

Prof. Dr. Murat Aşkar (Chairman)

Prof. Dr. Y. Ziya İder

Prof. Dr. Mete Severcan

Assist. Prof. Dr. Mete Bulut

Prof. Dr. Neşe Yalabık (Computer Engineering Department)

ABSTRACT

MODEL-BASED RECOGNITION OF POLYHEDRAL OBJECTS

UZUNALİOĞLU, Hüseyin

M.S. in Electrical and Electronics Engineering

Supervisor: Prof. Dr. Mete Severcan

July, 1992, 84 pages.

In this study, a model-based object recognition system is developed. It consists of feature extraction, modeling and matching stages. The system is designed to recognize polyhedral objects. Linear features are used for object descriptions. Three linear feature extraction methods are investigated: A method utilizing corner points to detect lines, the classical Hough transform method and the rotation transform method. The first method is not satisfactory. The Hough transform method is better, but it needs some modifications. The rotation transform method gives satisfactory results. In this method, the parameters can be adjusted according to the properties of the lines to be detected. Finally, geometric hashing method is used for modeling and recognition processes. Each object is modeled using 2-D views taken from the viewpoints on the viewing sphere. A hidden-line elimination program is used to find these views from the wire-frame models of the objects. The recognition experiments yielded satisfactory results.

Keywords: Model-Based Object Recognition, Linear Feature Extraction, Polyhedral Object Recognition, Geometric Hashing.

Science Code: 619.02.05

ÖZ

POLİHEDRAL YAPILI NESNELERİN MODEL-TABANLI TANINMASI

UZUNALİOĞLU, Hüseyin

Yüksek Lisans Tezi, Elektrik ve Elektronik Mühendisliği Anabilim Dalı

Tez Yöneticisi : Prof. Dr. Mete Severcan

Temmuz, 1992, 84 sayfa.

Bu çalışmada, model-tabanlı bir nesne tanıma sistemi geliştirilmiştir. Sistem özellik çıkartma, modelleme ve tanıma aşamalarından oluşmaktadır. Sistem polihedral yapısı olan nesneleri tanımak için tasarımlanmıştır. Nesne tanımları için doğrusal özellikler kullanılmıştır. Üç adet doğrusal özellik çıkartma yöntemi incelenmiştir: Doğruları bulmak için köşeleri kullanan bir yöntem, klasik Hough dönüşümü yöntemi ve çevirme dönüşümü yöntemi. İlk yöntem iyi sonuç vermemiştir. Klasik Hough dönüşümü daha iyi sonuçlar vermesine ragmen bazı modifikasyonlar gerektirmektedir. Çevirme dönüşümü yöntemi tatmin edici sonuçlar vermiştir. Bu yöntemin parametreleri sezilecek doğruların özelliklerine göre ayarlanabilir. Son olarak, modelleme ve tanıma işlemleri için geometrik "hashing" yöntemi kullanılmıştır. Her nesne gözlem küresi üzerindeki gözlem noktalarından alınan iki boyutlu görüntülerle modellenmiştir. Bu görüntüler, nesnelerin tel-çerçeve modellerinden yararlanılarak görünmeyen doğruların yokedilmesini sağlayan bir programla bulunmuştur. Test nesnelerinin kullanıldığı tanıma deneyleri olumlu sonuçlar vermiştir.

Anahtar kelimeler: Model-Tabanlı Nesne Tanıma, Doğrusal Özellik Çıkartma, Polihedral Yapılı Nesne Tanıma, Geometrik Hashing.

Bilim Dalı Sayısal Kodu: 619.02.05

ACKNOWLEDGEMENTS

I would like to thank Prof. Dr. Mete Severcan for his friendly guidance and helpful comments.

In addition, I would like to acknowledge Assist. Prof. Dr. Mete Bulut for his help in taking the test images without which this thesis can not be prepared.

My thanks are also due for my family, my colleagues and my friends for their support and continuous understanding.

TABLE OF CONTENTS

LIST OF TABLES

Page

LIST OF FIGURES

CHAPTER I

INTRODUCTION

The purpose of computer vision is to extract information contained in a single image or multiple images. Recognition of 3-D objects using 2-D images is an important problem in computer vision. Determining the position and the orientation of the object relative to the viewer can also be included into this problem.

3-D object recognition systems have applications in automated manufacturing processes like object inspection (to detect defects or measure dimensions) and object manipulation by a factory robot. In a factory environment, the number of objects to be handled is limited and the objects are known in advance. Therefore, in industrial applications, model-based object recognition systems can be used. In model-based systems, object model extracted from the observed image is matched with a set of models taken from a database of possible objects. The three major components of the system are feature extraction, object modeling and matching. The system consists of training and recognition phases. In the training phase, a description for each object is formed and stored in the model database. In the classification phase, a description of the scene under investigation is found and compared to the object descriptions in the database. Output of the system is a decision and spatial information for the object to be recognized.

In this thesis, a 3-D model-based object recognition system is described. The system is designed for the recognition of polyhedral objects. Hence, the line

segments are used as the object features. For the recognition, geometric hashing method [7] is used. Algorithms have been implemented using C++ programming language on a 80386 PC (33 MHz CPU speed) with 80387 coprocessor. In the experiments, 256-gray-level images with the size of 256x256 pixels are used. Execution times for the implemented algorithms are given also. These values can be decreased further by optimizing the computer programs.

In Chapter II, 3-D object recognition systems are described. They are classified according to the dimension of the model description used. General features of each group are explained. Also, an example for each group is given.

In Chapter III, preprocessing stage of the system is described. Preprocessing consists of smoothing and edge detection operations. Two different edge detection methods are investigated. A fast algorithm for the Sobel edge detection process is also presented. Finally, an edge thinning algorithm is used to thin wide bands of points containing edge pixels.

Chapter IV is devoted to the line extraction algorithms. Since, the system is designed for the recognition of polyhedral objects, line segments are very important features for the system. Three line extraction algorithms are explained. Implementation results of each one are given.

In Chapter V, geometric hashing method [7] for 3-D object recognition is presented. After a discussion of general concepts, training and recognition stages are explained. Finally, experimental results are given.

CHAPTER II

3-D MODEL-BASED OBJECT RECOGNITION

Recognition of industrial objects and their orientation in the environment is a major task in computer vision. Industrial object recognition systems are mainly model-based. In this chapter, an overview of the model-based methods and some examples are presented. For a detailed survey, see reference [1]

In an industrial environment, objects to be recognized are known in advance and can be modeled efficiently. Model-based recognition consists of matching the input image with the models stored in the model database. Model database contains descriptions for each object and is prepared before the recognition process. Various types of descriptions can be used for the objects to be recognized. Choice depends on the types of the objects and the application area. The description chosen must represent the objects uniquely.

Industrial model-based recognition systems can be divided into three categories according to their object descriptions: 2-D, $2\frac{1}{2}$ -D and 3-D descriptions. This grouping is related to the dimension of the object representation space. Systems in each class usually make similar assumptions. For each group, different feature extraction, modeling and matching techniques are used.

2-D object descriptions are viewer-centered representations in "image space". For each distinct view of the models, shape descriptions are found using

binary or gray-level images. This representation is suitable for objects with small number of stable viewpoints. 2-D representations can be divided into three subgroups: (a) the global feature method (b) the structural feature method (c) the relational graph method.

3-D spatial descriptions use an object-centered coordinate system, define exact representations in "object space" and they are viewpoint independent volumetric representations. These representations enable computations at an arbitrary viewpoint and to a desired precision of detail.

$2\frac{1}{2}$ -D descriptions use properties of both 2-D and 3-D representations. Features are defined in "surface space". These descriptions are viewer centered. They use local surface properties in each view like surface orientation and range(depth). So, each possible view of the object must be included into the modeling process.

2.1. Models, Features and Matching

A typical model-based recognition system is shown in Figure 2.1. It consists of training and recognition phases. The three major components of the system are feature extraction, object modeling and matching. The sensor and feature extraction components of the training and recognition phases may be different. Below, general objectives of these three components are given.

2.1.1. Models:

Models are generally constructed using the geometric properties of the objects. Different shape features are used for 2-D, $2\frac{1}{2}$-D and 3-D representations.

2-D models can be constructed from a set of 2-D views of the object, one from each possible viewpoint. In industrial applications, the number of allowable viewpoints is limited. Objects' boundaries or edges are used as features. Extraction of these features is an easy task when compared to the 3-D model construction from a set of 2-D views of the objects. 2-D representations have the disadvantage of losing one dimension. Their completeness depends on the complexity of the object and the number and the positions of the possible viewpoints. One 2-D view may have more than one 3-D interpretation. This creates some limitations on such systems.

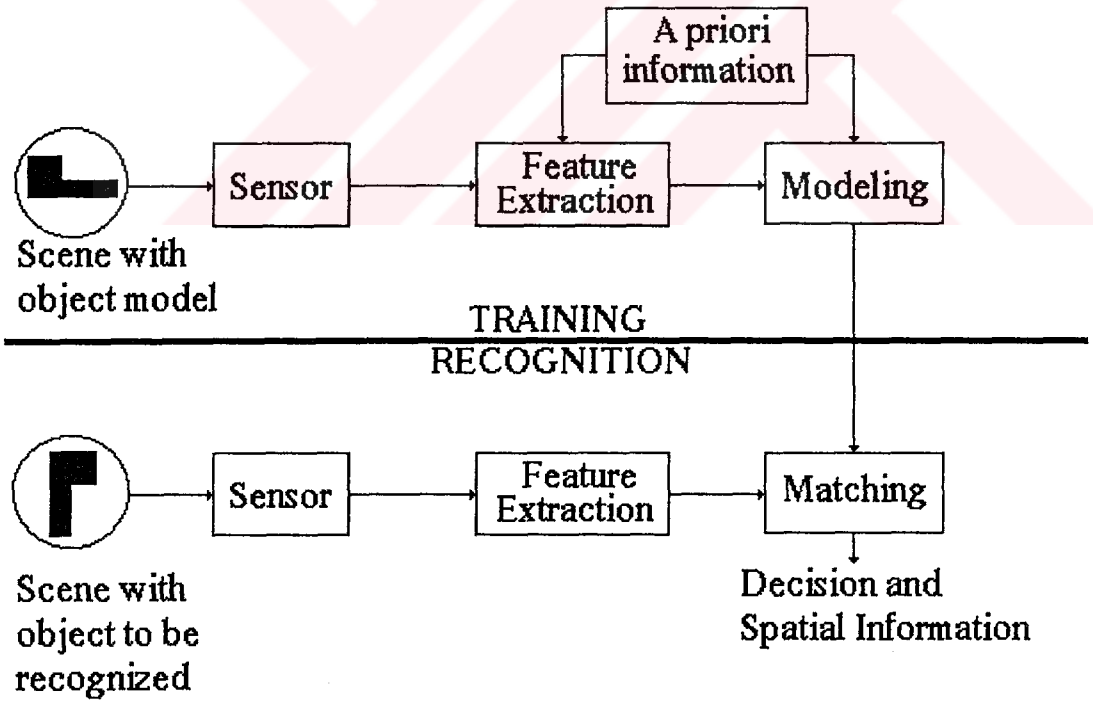


Figure 2.1. A general model-based object recognition system

$2\frac{1}{2}$ -D models use surface information. Hence, they carry more detailed information about the objects compared to 2-D models and therefore provide more complete information. The descriptions are viewer-centered. Hence, they use separate descriptions for each possible viewpoint. The extraction of surface information is an important problem for $2\frac{1}{2}$-D representations.

3-D representations are volumetric descriptions of the objects. They are independent of viewpoint. They carry most general and complete information about the objects. Object models can be formed directly using CAD-like systems. But, 2-D or $2\frac{1}{2}$ -D features extracted from an image can not be directly compared to 3-D models. Hence, 2-D to 3-D correspondence problem must be solved.

2.1.2. Features:

Object features are selected according to the model description and application area. Basic image features are edge, corner, line, curve, hole and boundary curvature. These features are combined to produce object descriptions. They carry high-level information. So, they are less sensitive to noise than the gray-level values of the image.

For industrial applications, generally, object boundaries and measurements derived from boundaries are used as features. These features can be categorized into three groups: global, local and relational features. Perimeter, centroid, distance of boundary points from centroid, curvature, area, moments of inertia are the examples of global features. Examples of local features are line segment, arc segment with constant curvature, corner point. Examples of relational

features include a variety of distance and relative orientation measurements interrelating substructures and regions of an object.

For the extraction of boundary features, object is placed in a high-contrast background with controlled lighting. This eliminates shadows and noisy background. Then, the image is thresholded and object is separated from background. Instead of thresholding, a sensor producing binary images can be used. Binary representation reduces the complexity and the amount of the data to be handled. Then, using this binary image, 2-D features are extracted.

Binary image features can be easily extracted. But, this kind of features has the disadvantage of using 2-D descriptions for 3-D objects. Also, they don't use the interior edge information. So, application areas for these systems are limited.

If, instead of binary images, gray-level images are used, more complex recognition tasks can be implemented. In addition to the boundary information, features like edge, corner, line and hole can be used. Feature extraction for gray-level images is more difficult and time consuming. It needs sophisticated feature extraction algorithms. Matching still depends on 2-D features. But, they give more reliable information about the objects compared to global features.

2.1.3. Matching:

Model-based recognition is the process of matching features extracted from an image with the features contained in the model database. Features of all model objects are stored in the model database before the recognition process. The

purpose of the matching process is to find a set of features in the given image that approximately matches with some features of one model object.

Matching process is highly dependent on the object representation. For 2-D global features, usually, statistical pattern recognition techniques are used [1]. Models represented by local features are related to syntactic matching methods. Graph-matching techniques are used for objects represented by relational features.

For $2\frac{1}{2}$-D models, sets of surfaces are compared. This comparison can be done directly by searching best-fitting regions between model surfaces and image surfaces or indirectly by comparing some features derived from the surfaces. If 3-D models are used, matching algorithm searches a 2-D projection of 3-D model that best matches with the image features. 3-D information about the unknown object can be extracted using techniques like shape from shading, range and stereo imaging. Here, matching process can be carried out in 3-D space.

2.2. 2-D Image Representations

Objects are modeled using their views taken from different viewpoints. Model database can be formed by training the system with each possible view of the object or by computing these views directly from the CAD description of the object. This representation deals with each viewpoint independently. So, 3-D object recognition becomes a 2-D recognition process. According to model descriptions and matching techniques, 2-D recognition methods can be divided into three categories: global feature method, structural feature method and relational graph method.

2.2.1. Global Feature Method:

This method uses global features like perimeter, area, center of gravity, moment of inertia and fourier descriptors. Extraction of these features is very easy. Models are the unordered lists of these features. Features can be thought as points in n-dimensional feature space. For recognition, feature vector of the unknown object is found and compared with the feature vectors of model objects using statistical pattern recognition methods. Decision rules may be parallel (Bayes classifier) or hierarchical/sequential.

These methods are fast because of the limited size and the number of feature vectors extracted from a given image. Global features are invariant to rotation, shift and size. On the other hand, the limitations of the system are; (1) a separate model must be used for each possible view of an object; (2) a single predefined threshold value is used to extract all objects in a given image; (3) objects are not allowed to touch or overlap each other and they must have no significant defects.

The SRI Vision Module [2] is an example of the global feature method. The user interactively selects a set of features which are used to construct an object model as a feature vector. System is trained by showing each distinct view of the object. Each connected region in the input binary image is extracted and analyzed independently. Some shape features like area, perimeter, number of holes, boundary chain code, compactness, number of corners and moment of inertia are used. Recognition is done using a decision-tree method. The tree is constructed using global feature vectors of the models as follows: The feature with largest separation in value between any pair of models is found. This feature forms the root node of the tree. A threshold value is chosen to distinguish these two models. All models with feature values less than or equal to this threshold are assigned to the left child and all models with feature

values greater than this value are assigned to the right child. This process is repeated recursively. A terminal node contains a single model. the decision-tree method is fast but it is not optimum.

Statistical matching methods like nearest neighbor classifier can be used for matching process also. In nearest neighbor classification, each model is represented by a point in n-dimensional feature space where n is the number of features. An unknown object is matched to a model that is closest in feature space.

2.2.2. Structural Feature Method

Object models are image features like line, arc and corner. These are local features. Each feature defines one portion of the object. Hence, even when an unknown object is occluded by another one, there is a chance for recognition. Models are formed by combining features in a highly structured way, such as an ordered list. Ordering is related to the boundaries of the object such that following the list sequentially is equivalent to tracing the boundary of the object. Matching uses a hypothesis-verification procedure. Locations of the objects in the scene are found using structured local features of the model. Then, object features are found using the hypothesis generated by the model. Finally, these features are matched to verify the hypothesis. Syntactic pattern recognition methods can also be used. In these methods, local features are transformed into primitives. Primitives are organized into sentences by some highly structured grammatical rules. Recognition is done by parsing.

The structural feature method is superior to the global feature method. But, it needs very sophisticated training and matching procedures. Gray-level image

features are more reliable than the thresholded image features. The number of local features is high. So, modeling and matching procedures must be good enough to avoid testing all correspondences of image and model features. Matching consists of a sequential process. First, a few features are tried to be matched. Then, this matching is used to constrain the search for other features. If the model is not appropriately designed, hypothesis-verification task will be very time consuming. As another advantage over the global feature method, occluded objects can be recognized. But, most of the object boundaries must be visible for a reliable recognition because most features are derived from the boundaries.

A method developed by Stockman et al. [3] is an example of structural feature method. Models are formed by organizing real vectors defining boundary segments and abstract vectors connecting primitive features (such as connecting two holes). The model is a line drawing version of the object, plus additional abstract vectors. Abstract vectors are used for increased precision and control over matching process. Feature extraction stage extracts point features. Then, suitable point pairs form vectors. Finally, holes are detected. Matching stage matches all possible image features with model features on the basis of local evidence. Rotation, scaling and translation transformation parameters are found for each possible pair of features. Then, a clustering operation is performed in transformation parameter sets. A cluster in the transformation space shows that many image features are matched with model features using corresponding transformation parameters.

2.2.3. Relational Graph Method

In this method, geometrical relationships between local features (edge, corner) have primary importance. Relational structure can be represented using a graph. Each node of this graph corresponds to a local feature. A list of properties related to the local feature are assigned to each node. Arcs represent the geometrical relations between the nodes. Also, lists of relation values are assigned to each arc. Recognition is a graph-matching process.

Local-feature-focus method developed by Bolles and Cain [4] uses relational-graph method. Holes and corners are used as local features. The position and the orientation of the local feature relative to the object's centroid are recorded. After learning each object, a cluster of local features that does not appear elsewhere in the object or in any other possible object is searched. A feature in this cluster is chosen as "focus" feature. In recognition stage, for each model object, focus feature related to that model is searched in the image. If it is found, other features in the clusters are searched. If they are found and their positions and orientations are consistent with the configuration found in the learning stage, the object is hypothesized to exist at that location. An object template is translated and rotated by the required amount and is matched to the image. If the match is good enough, the algorithm decides that the object exists at the location.

2.3. $2\frac{1}{2}$-D Surface Representations

This kind of representations is also viewer-centered. But they use physical characteristics of a single view of an object. They combine gray-level

information with intrinsic scene characteristics. In an intrinsic image, each pixel carries information like depth, surface orientation, reflectance, illumination, color or velocity of the object surface patch projecting to that pixel's position.

If multiple views of the object are required in the training stage, each view is processed as an independent model. Range maps can also be included into this group even they use 3-D data, because the models that use this data are viewer-centered, i.e., each model gives the description of the surface features that are visible from a single viewpoint. If the model describes a complete (viewpoint independent) 3-D object, it must be thought as a 3-D representation.

Extraction of surface information is a difficult task compared to the extraction of gray-level image features. Range maps can be found using ultrasonic and light time-of-flight measurement or structured light projection using a plane or grid of light. Instead of range map, surface orientation (needle) maps can be used. Surface shape can be found using needle maps. Surface orientation information of a scene can be found using shape from shading or stereo analysis techniques.

Oshima and Shirai [5] uses a relational-feature graph representation of the objects. For each separate view a different graph is found and treated independently by the matcher. Nodes of the graph are planar or smoothly curved surfaces extracted from a range map, and arcs represents the relations between the adjacent surfaces. Matching is done by comparing the relational graph of the observed image with a set of graphs for each viewpoint of each object modeled.

2.4. 3-D Object Representations

2-D and $2\frac{1}{2}$ -D representations are viewer centered and hence, they use separate models for each possible view of an object. If the object to be recognized is complex then the number of viewpoints and the number of models will increase. This problem can be solved using viewpoint-independent volumetric representations. A single model is used for each model. Model describes implicitly all possible views of the object. 3-D representations can be divided mainly into two groups: (1) exact representations (2) multiview feature representations. The first one specifies an object's spatial occupancy completely, the second only describes selected visible 2-D or $2\frac{1}{2}$ -D surface features.

2.4.1. Exact Representations:

These are complete, volumetric and object centered representations based on surface patches, sweeps or volume primitives. Surface models describes an object by its boundaries or surfaces using primitives like edge or face. Volume representations uses solids such as generalized cylinders, cubes, spheres and rectangular blocks. This class of representations provide an exact object-centered description of the model object. But it is difficult to use this kind of representations in real-time applications. Because, matching of 2-D features extracted from a given image with a 3-D model requires 2-D to 3-D or 3-D to 2-D projections which are computationally expensive.

3DPO (three-dimensional part orientation system) is developed by Bolles et al. [6] to generalize their local-feature-focus [4] method to 3-D. An object model

consists of an augmented CAD model and a set of features-classification networks. Augmented CAD model describes edges, surfaces and vertices and their relations. The feature classification network classifies visible features by type and size (i.e. surface elements that have the same normal direction and cylinders that have a common axis). Range data is used to detect surface discontinuities. Matching process is similar to the matching operation of local-feature-focus method. First, system searches for features that match some model's feature like a cylinder with a given radius. If it is found, other features are used for verification.

2.4.2. Multiview Feature Representation:

This class of representations combines 2-D or $2\frac{1}{2}$ -D descriptions to form a single model of an object. Two approaches can be used for multiview object representations. The first approach uses topologically distinct stable views of an object. These views are such that small changes in a viewing position do not affect the topological structure of the set of visible object features (i.e. point and line singularities). Aspect graph representations uses this approach. Second approach for multiview feature representation is "discrete view-sphere representation". Object is viewed from a set of points on a sphere around the object. 2-D viewer-centered descriptions for each viewpoint is stored. This process can be done using a complete volumetric description at training phase. The descriptions must be compatible with the features extracted from an image at recognition phase.

Geometric hashing technique proposed by Lamdan et al. [7] uses multiview feature representation. Two images of a scene taken with the same viewing angle are in a similarity correspondence. Similarity transformation consists

of rotation, translation and scale. Parameters of this transformation can be found by matching a point pair of the first image to a point pair of the second image. Models are constructed by forming a new basis using an ordered model point pair and finding the coordinates of the other model points in this new coordinate system. This process is repeated for each possible point pair and view taken from the viewpoints lying on a sphere around the object. New coordinate of each point (after a proper quantization) serves as an index entry for a table. View number, object name and basis pair is recorded into this table. This training phase is performed off-line. In recognition stage, an ordered image point pair is used for defining a basis and coordinates of other points are found using this basis. These coordinates are used to index the table and the corresponding view numbers, object names and basis pairs are recovered from the table. A voting table is used to see how many times a view, object and basis pair is recovered from the table. An object and a view with the maximum number of votes are chosen as a candidate match. This match is verified using corner and line information of the scene and the model. Details of this method and implementation results are given in Chapter V.

CHAPTER III

PREPROCESSING

Gray-level images are composed of several regions. Each region corresponds to a surface in 3-D space. Hence, region boundaries are very important features in image analysis. Regions can be found using either global or local methods [8]. Global methods like thresholding, region growing, and region splitting and merging use similarity of the gray-level values inside the regions. Local methods use local gray-level discontinuities. Edges are image points where the light intensity change abruptly. So, given an image, boundaries can be obtained by combining the edge pixels. To detect edges accurately, image must be smoothed first. Smoothing removes the small intensity discontinuities inside a region while preserving the discontinuities in the region boundaries. Also, to improve the speed of the higher-level processing stages, detected wide edge segments must be thinned.

In the next section, a smoothing technique is explained. Then, some edge detection methods and a thinning algorithm are presented. Finally, experimental results are given.

3.1. Image Smoothing

The purpose of smoothing operation is to enhance the quality of the input image that is often degraded by noise due to imaging system. This degradation can

be due to lighting conditions. In edge detection, light intensity values of image pixels are used. So any degradation in the image may result in erroneous edges. Smoothing removes the small intensity discontinuities inside a region. Appropriate smoothing may increase the success of the edge detection process.

One simple smoothing technique is the spatial low-pass filtering of the image. In this method, image f(i,j) is transformed to a smoothed image g(i,j) whose gray level at point (i,j) is found by averaging the gray values of the pixels of f in a predefined neighborhood of (i,j). This can be shown by the relation

$$g(i,j)=\frac{\Sigma\, w(k,l)\, f(i-k,j-l)}{\Sigma\, w(k,l)} \qquad (3.1)$$

Summation limits depend on the size of the window w. Generally, a 3x3 window is used. If entries of w are chosen to be equal to unity, this smoothing operation is called as "neighborhood averaging". If entries take various values, this task is called as "weighted neighborhood averaging". In this work, both window types are implemented. Window values are shown in Figure 3.1.

1	1	1
1	1	1
1	1	1

(a)

$\frac{1}{16}$	$\frac{1}{8}$	$\frac{1}{16}$
$\frac{1}{8}$	$\frac{1}{4}$	$\frac{1}{8}$
$\frac{1}{16}$	$\frac{1}{8}$	$\frac{1}{16}$

(b)

gure 3.1. Two smoothing windows (a) Neighborhood averaging (b) Weighted neighborhood averaging.

3.2. Edge Detection

In 3-D computer vision, one of the most important image features is the "edge" or boundary line of a uniform region. Edge detection is generally a preliminary step for linear feature extraction. Edge pixels are points at which pixel values (light intensity) change abruptly. Therefore, the most common edge detection methods make use of local derivative operators. Below, first, a different method which uses image histogram is explained. Then, edge detection using local derivative operators is presented.

3.2.1. Edge Detection Using a Global Method

This method [9] uses the similarity of the gray-values inside a region instead of using local intensity discontinuities. It classifies the edges according to their visibility in the scene. For example, "primary edges" are visible at a glance. They form the overall outline of the object. "Secondary edges" are less visible and so on. In [9], it is proposed that natural vision systems percept edges in this manner. Hence, each pixel has an edge perception value which shows the classification of a pixel as an edge element.

In this method, after smoothing, image edges are found by decimating the different intensity regions using intensity histogram of the image. It is assumed that the gray-values between two successive minima of the smoothed histogram belong to a region. Decimation of two adjacent intensity regions can be done by transforming intensity level of one region to 0 (dark) and the other region to 1 (maximum brightness). Some special sinusoidal functions are used for these

transformations. After the transformation, minima of the histogram take the value of 0.5. The midpoints between the minima become 0 or 1. The other gray values are mapped into the range [0,0.5] (dark) or the range [0.5,0] (light). Then, for each pixel, an "edge perception" value is found. This is done by finding the difference between the transformed value of that pixel and the minimum value of the pixels in the local neighborhood of that pixel in the transformed image. Finally, the pixels with the edge perception values larger than a threshold value are chosen as edge pixels.

This method was implemented and tested. The results were unsatisfactory. Proposed mapping functions may be erroneous. Many false edges were detected. Also, the execution time was several minutes. Hence, another method is used for the tasks in the remaining of this thesis.

3.2.2. Edge Detection by Gradient Operators

Since edges are the points at which the intensity values change abruptly, magnitudes of the local derivatives at these points are high. So, if a local derivative operator is applied to the image, result will be an edge enhanced image. Then, by thresholding this edge enhanced image, edges can be found.

Local gradient at a pixel can be found by calculating a weighted average of the pixels in the neighborhood of that pixel. Usually, a template with the size of the local region is created and is filled with the coefficients that corresponds to the weights. Then, the whole image is convolved with this template. The result is an edge enhanced image. Every pixel value is a measure of the edge strength of the

corresponding pixel in the original image. A simple pixel thresholding gives image edges (i.e. pixels with high edge strength value).

In literature [10], a various number of edge detection kernels are proposed. In this study, Sobel filters are implemented. Sobel kernels are shown in Figure 3.2. For each pixel, horizontal and vertical gradient values , f_x and f_y respectively, are found by applying the corresponding Sobel kernel. Edge strength and direction for a given pixel is found by

$$S(x,y)=\sqrt{f_x^2 + f_y^2} \tag{3.2a}$$

and

$$\psi(x,y)=\arctan\left(\frac{f_y}{f_x}\right) \tag{3.2b}$$

where f_y and f_x are the vertical and horizontal gradients of the image f(x,y) calculated by Sobel operator, respectively.

Here, the calculation of S(x,y) requires two multiplication, one addition and one square root operation. Instead of this time consuming equation,

$$S(x,y)= | f_x | + | f_y | \tag{3.3}$$

can be used.

Finally, simple pixel thresholding is applied to the edge strength image S(x,y). Pixels with values greater than a predefined threshold are said to be "edge pixels" and the resultant image consisting of these pixels is called as "edge image". In the next section, a fast algorithm for the Sobel edge detection is explained.

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

(a)

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

(b)

Figure 3.2. Sobel kernels in (a) vertical direction (b) horizontal direction

Another edge detection operator is the Laplacian operator which is an approximation to the mathematical Laplacian

$$L(f(x,y)) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \tag{3.4}$$

This operation can be implemented digitally by convolving the image with the kernel shown in Figure 3.3.

The edge directions can not be found using the Laplacian operator. Also, since it is a second order derivative operator, it is very sensitive to noise. This restricts the usage of the Laplacian operator.

0	1	0
1	-4	1
0	1	0

Figure 3.3. The Laplacian edge detection kernel.

The zero-crossing method is another edge detection method which uses a smoothing operation before applying the Laplacian operator. In this method, the image is convolved with the Gaussian operator and then the Laplacian operator is applied. The zero-crossings of the resultant image give the edge points of the image. But, small light intensity variations inside the regions may also produce zero-crossings. So, if the difference of the image gray-level function f(x,y) across the zero crossing is small, the zero crossing must be rejected as an edge point.

3.2.3. A Fast Algorithm For Edge Detection

Sobel kernels shown in Figure 3.2 requires four multiplication and eight addition operations for every image pixel. Digital multiplication requires longer time compared to addition. So the use of Sobel kernels directly is computationally expensive. Figure 3.4 shows a decomposition of Sobel operators into simpler kernels.

It is clear that the new kernels P and Q require no multiplication and hence, the use of these kernels is computationally more efficient [11].

3.3. Edge Thinning

Sobel operation gives the edge strength and direction at each image pixel. Simple thresholding extracts possible edge pixels. But, generally, this is not sufficient. A nonmaximum suppression technique must be used for thinning the wide bands of points containing edge pixels. In this technique, a pixel is considered as an

edge element if its edge strength is greater than the edge strengths of its neighbors in a direction normal to the edge direction of the pixel in a local window. An edge pixel which does not satisfy this condition is eliminated even if its edge strength is larger than the threshold.

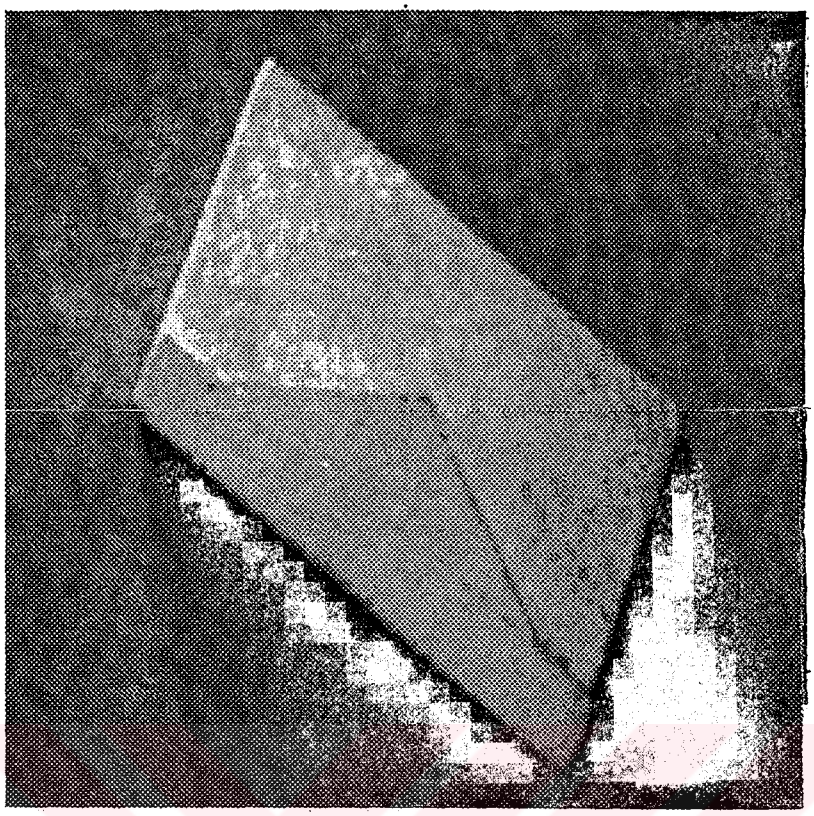
1	2	1		1	1	0		0	1	1		
0	0	0	=	1	0	-1	+	-1	0	1		
-1	-2	-1		0	-1	-1		-1	-1	0		
K1			=	P			+	Q				

1	0	-1		1	1	0		0	-1	-1		
2	0	-2	=	1	0	-1	+	1	0	-1		
1	0	-1		0	-1	-1		1	1	0		
K2			=	P			-	Q				

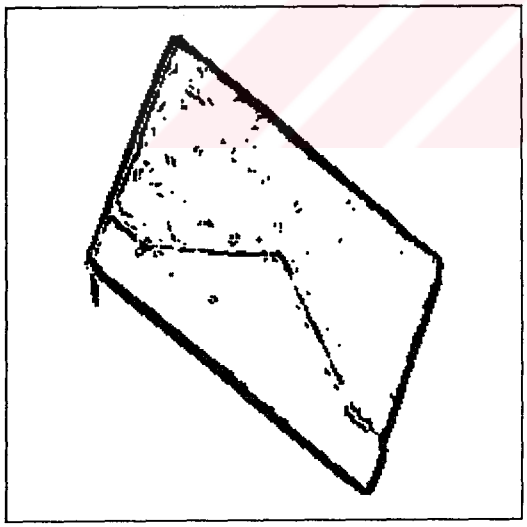
Figure 3.4. Decomposition of the Sobel Kernels

3.4. Experimental Results

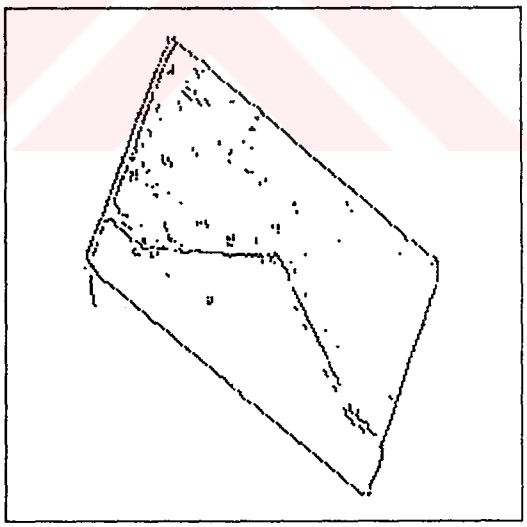
In this study, smoothing, Sobel edge detection and edge thinning algorithms are implemented. For Sobel operator, fast algorithm described on Section 3.2.3. is used. Execution times are 5 seconds for smoothing operation and 9 seconds for edge detection and thinning operation. Figures 3.5 and 3.6 show results of these operations for two gray level images. According to different threshold values, detected edges vary. If the threshold is chosen too low, some erroneous edge points appear in the edge image. On the other hand, if it is chosen too high, some real edge points does not appear in the edge image. So selection of a suitable threshold is very difficult and image-dependent. For images taken in similar conditions, a suitable threshold value can be chosen by trial-and-error. For images shown in Figures 3.5a and 3.6a, edge strength threshold is chosen as equal to 100. As it is seen from the Figures 3.5b and 3.6b, points near to the object's outer boundary form wide edge areas. But, for edges inside the outer boundary, thinner edge areas are found. Some edge points are missed while some erroneous points are detected as edge points. Nonmaximum suppression technique is applied to these images. Resultant images (which will be called as "edge image" in the remaining of this thesis) consists of narrow strips of points on the locations of possible lines in the original image. Figures 3.5c and 3.6c show images after nonmaximum suppression technique is applied. Line detection algorithms will use such images as their inputs.



(a)

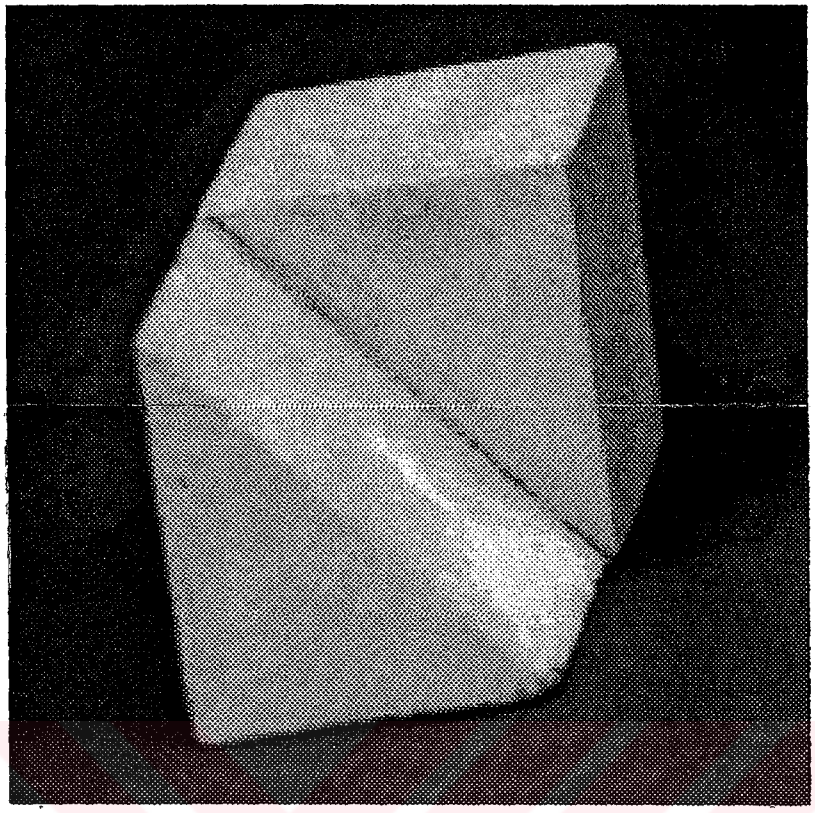


(b)

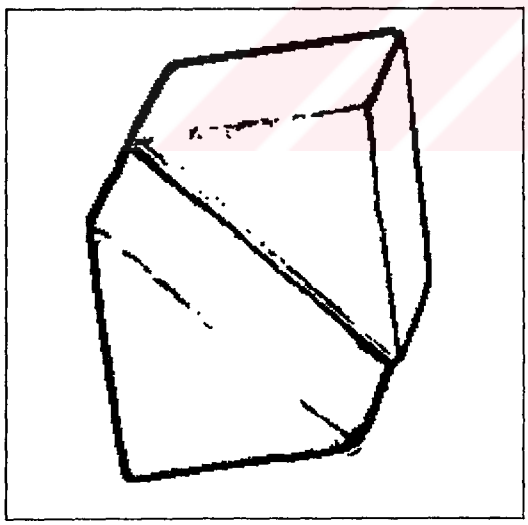


(c)

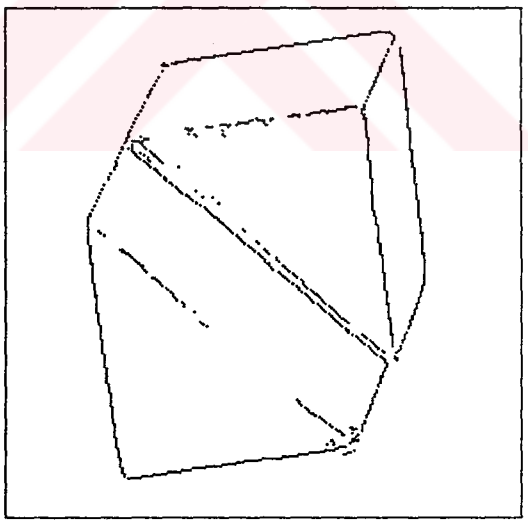
Figure 3.5. Edge detection and thinning (a) Original image (b) Thresholded edge points (c) Thinned edge points



(a)



(b)



(c)

Figure 3.6. Edge detection and thinning (a) Original image (b) Thresholded edge points (c) Thinned edge points

CHAPTER IV

LINE DETECTION ALGORITHMS

In image analysis, line segments are important features, especially, if the image consists of polyhedral objects. Hence, line detection is an important area in computer vision. Extracted line drawings can be used not only for recognition tasks but also for stereo ranging and dynamic scene analysis. It must be noted that successful line drawing extraction simplifies high-level processing.

In this chapter, three line extraction algorithms are described. Experimental results are given also.

4.1. Line Detection Using Corner Points

In this algorithm [12], line segments are found using corner points detected by a corner detector. Corner points are the intersection points of the lines. Hence, line search process is concentrated on local regions centered at corners. After finding the possible line segments, a false line and corner elimination algorithm is used also. Below, the algorithm and the implementation results are given.

4.1.1. Algorithm

Corners are image points at which the edge direction changes by a large amount. In [12], a corner detection algorithm developed by Fang and Huang [17] is proposed. According to this algorithm, "cornerness" of an image point is proportional to gradient of $\psi(x,y)$ (where $\psi(x,y)$ is the edge direction of the point). This quantity attains a local maximum at a corner point. At every image pixel, the product of the "edginess" (which is equal to the magnitude of the gradient of the image) and the cornerness is calculated and if this value is greater than a predefined threshold, the pixel is chosen as a corner point. Mathematically,

$$\psi(x, y) = \arctan\left(\frac{f_y}{f_x}\right) \tag{4.1}$$

$$\begin{pmatrix}\psi_x \\ \psi_y\end{pmatrix} = \frac{1}{P}\begin{pmatrix} f_{xx} & f_{xy} \\ -f_{xy} & f_{yy}\end{pmatrix}\begin{pmatrix}-f_y \\ f_x\end{pmatrix} \tag{4.2}$$

where $P = (f_x^2 + f_y^2)^{1/2}$. ψ_x and ψ_y are the horizontal and vertical gradients of $\psi(x,y)$. f_x and f_y are the horizontal and vertical gradients of the gray value function $f(x,y)$ of the image, respectively. f_{xx}, f_{xy} and f_{yy} are the second order gradients of $f(x,y)$. Sobel operator is used for the calculation of these gradients. For noise suppression, local maxima of $P*C(x,y)$ are chosen as candidates of corner points where $C(x,y)=(\psi_x^2 + \psi_y^2)^{1/2}$. Also, candidates smaller than a predefined threshold are eliminated. Remaining points are said to be corner points.

After finding the corner points, line directions emerging from these corners must be found. This can be done by investigating the distribution of edge points on a local window centered at each corner point. The aim is to find enough number of edge points in a certain direction. Directions are quantized to a certain number of angles. For example, if N is the number of directions, then a line can make an angle of $i*2\pi/N$ with the horizontal where i gives the quantized direction as in Figure 4.1. For each corner, quantized line directions are found.

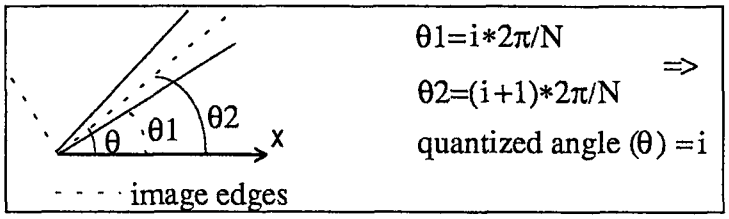


Figure 4.1. Quantized direction of a line.

Line directions are used for finding lines forming the objects in the image. A search algorithm tries to find possible lines between every pair of corner points. Given a pair, if the distance between two corners is larger than a predefined threshold, following condition is applied : If there is a line direction for each corner which is consistent with the direction of the line joining two corner points, there is a line between these two points. If the distance is smaller than the threshold, following condition is applied : If the number of edge points lying on a local neighborhood of the line combining two corners is larger than a predefined percentage of the distance, there is a line joining two corners.

The output of the line search algorithm is a set of lines. False corners and lines must be removed from this set. Three kinds of corners must be eliminated. These are: 1)Isolated points: There is no line emerging from these points.

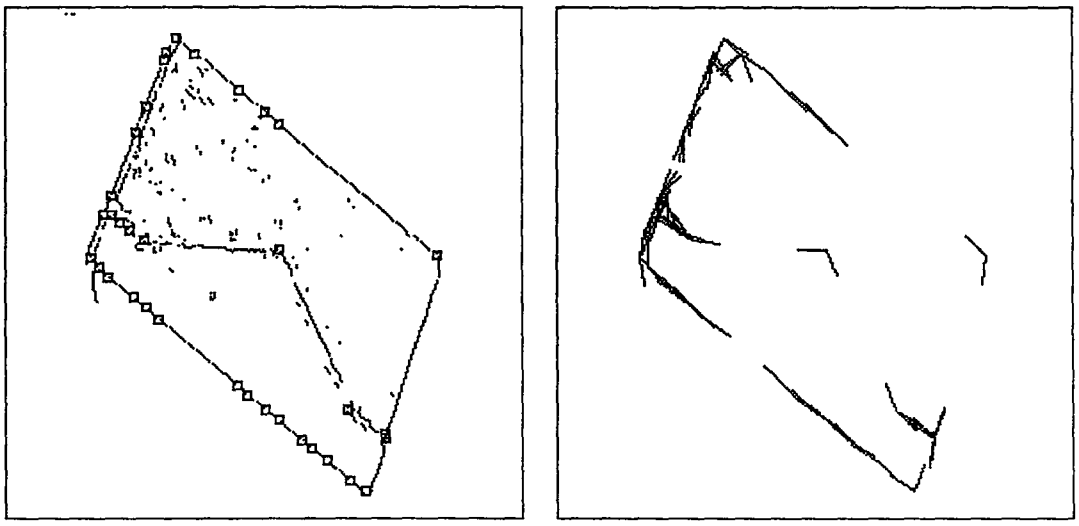
2)Termination points: There is only one line connected to these points.

3)Noncorner points: There are two lines radiating from these corners but the angle between them is nearly 180^{o}.

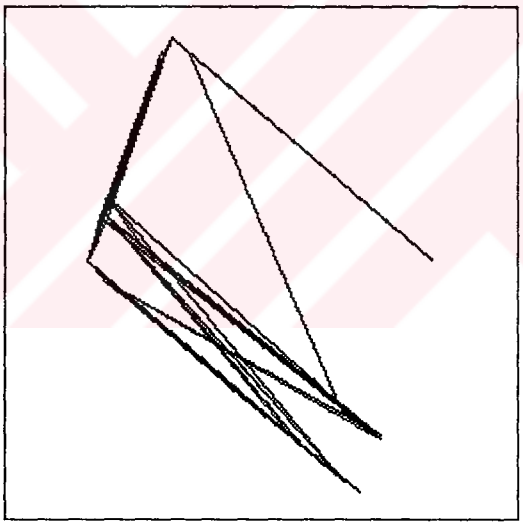
This corner and line elimination algorithm must be applied recursively until no elimination occurs.

4.1.2. Experimental Results

Algorithm is tested using two images shown in Figures 3.5a and 3.6a. Execution time of this algorithm is 4 seconds. Obtained results are shown in Figures 4.2 and 4.3. In Figures 4.2a and 4.3a small squares indicate the positions of the corners detected by corner detector. Threshold value for corner detector is $6*10^{4}$. It is seen that corner detector detects a lot of false corner points. On the other hand, it misses some real corner points. Also, positions of some detected corner points are misplaced by small amounts. Hence, performance of the corner detector is very poor. Figures 4.2b and 4.3b show the results of the line clustering algorithm. If the corner detector misplaces the corner point by a little amount, direction finder produces erroneous results. It misses some line directions and finds some line directions for false corners. Finally, detected lines, shown in Figures 4.2c and 4.3c, are not satisfactory. Line extraction algorithm uses these false corners and line directions as its inputs. Hence, Line detector produces a lot of false lines while missing some real ones. False corner and line elimination algorithm is not implemented, because it is not adequate for finding these false lines and corners and missing lines. These results show that this algorithm can be used only for images taken in very controlled environments.

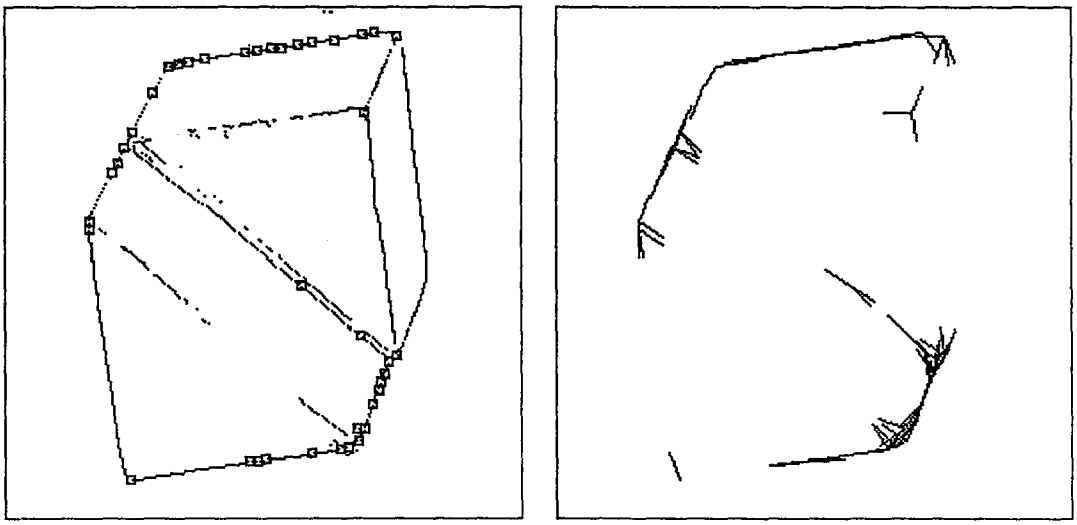


(a) (b)

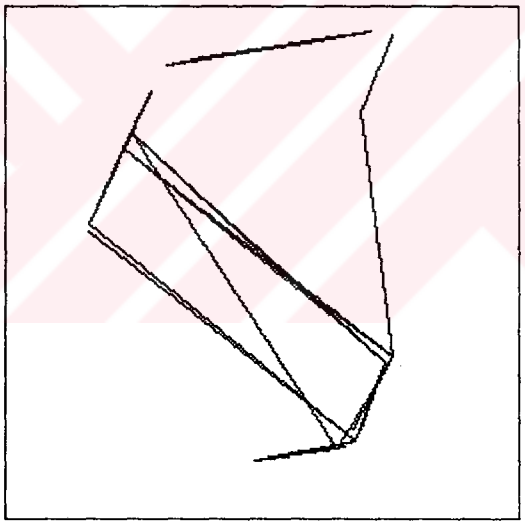


(c)

Figure 4.2. Line detection using corners (for Figure 3.5a) (a) Detected corners (b) Line clusters for each corner (c) Extracted line drawing.



(a) (b)



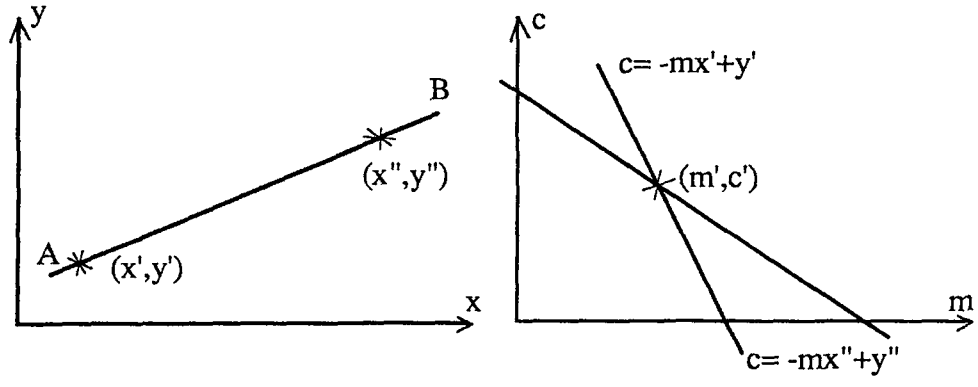
(c)

Figure 4.3. Line detection using corners (for Figure 3.6a) (a) Detected corners (b) Line clusters for each corner (c) Extracted line drawing.

4.2. Line Detection Using Hough Transform

Given a set of points, Hough transform tries to find the parameters of a curve that fits to the points. When it is used for the line detection purposes, it gives the line parameters. This is done by a voting process.

Consider the point (x',y') in Figure 4.4a. There is an infinite number of lines passing through this point. Each line satisfies the condition y'=mx'+c. This condition is equivalent to a line in m-c space, or in "parameter space", if x' and y' are kept constant. A second point (x",y") has also an associated line in the parameter space. These two lines intersect at the point (m',c') which corresponds to the line AB connecting these two points. Actually, all the points in the line AB have related lines in the parameter space which intersect at the point (m',c') as shown in Figure 4.4b. Hough transformation algorithm [10] uses this relationship between the image and the parameter spaces for line detection.



(a) (b)

Figure 4.4. A line in (a) image space (b) parameter space.

4.2.1. Algorithm

First, parameter space is quantized. This quantized space is called as "accumulator array". Each cell of the accumulator array is initially set to zero. For each edge point, parameters of all lines passing through that point are found. The content of the corresponding accumulator cell is incremented by 1. Collinear points in an edge image increase the value of a particular cell in the accumulator array. This cell gives the parameters of the line passing through these points. Straight lines can be found by searching the local maxima of the accumulator array. One disadvantage of this algorithm is that the values of the slope and intersection can be infinite. In order to solve this problem, an angle-radius parameter space can be used instead of slope-intersection space.

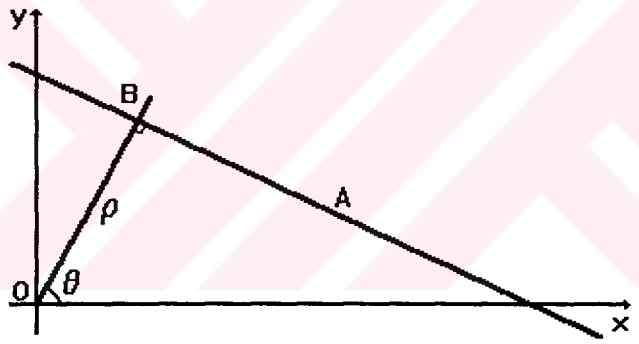


Figure 4.5. Line representation using θ - ρ parameters.

In Figure 4.5, straight line "A" is represented using two parameters: (1) θ, angle between X-axis and line "OB" which is perpendicular to line "A" and passes through the origin and (2) ρ, distance of the line "A" from the origin. This can be shown by the relation

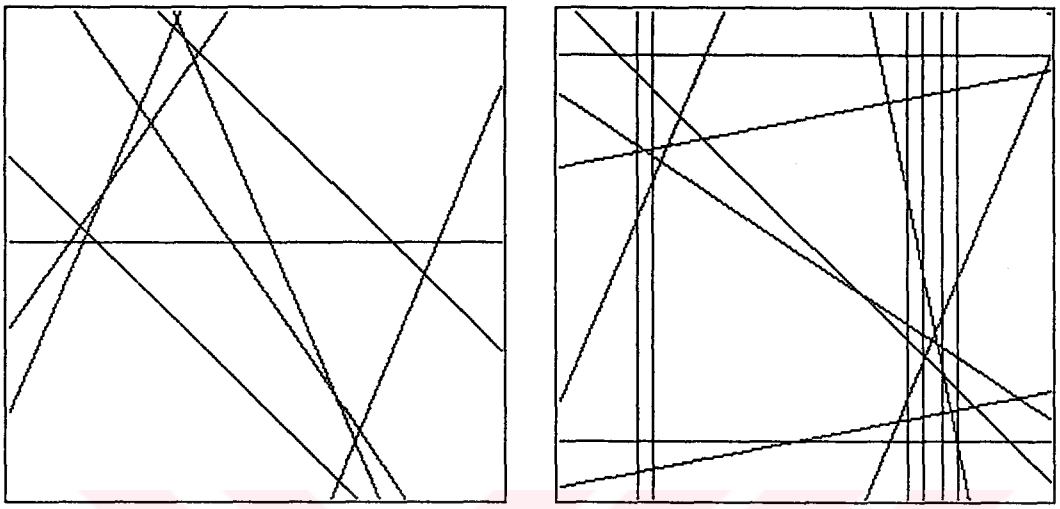
$$x \cos\theta + y \sin\theta = \rho, \theta : [0, \pi]. \quad (4.3)$$

Here, a line passing through the point (x,y) in the image plane can be represented by θ and ρ in parameter space. Each of the straight lines passing through an image point is mapped to various points (θ_i , ρ_i) on a sinusoidal curve in θ - ρ space. Collinear points in the image contribute to the accumulator cell related to the line passing through these points. So, the contents of these kinds of cells increase. Local maxima of θ - ρ space exceeding a predefined threshold give the parameters of lines in the image. In this parameter space, values of θ and ρ are bounded. To implement this algorithm on a computer, θ and ρ values must be quantized. To decrease the computation time, edge directions can be used as an approximation for the line directions [13]. Instead of finding θ and ρ values for every θ in the range of $[0,\pi]$, a smaller interval, $\theta_e-\phi<\theta<\theta_e+\phi$, can be used. Here, θ_e is the edge direction of the pixel and ϕ is a tolerance angle due to erroneous edge directions.

4.2.2. Experimental Results

Algorithm described in the previous section is implemented and tested. Execution time is 8 seconds for the images considered. Extracted line drawings are shown in Figure 4.6. It is difficult to find a proper threshold value to find the local maxima for existence of an image line. If the threshold value is chosen too high, lines with small lengths will be lost. On the other hand, if it is chosen to be too low, some erroneous lines around the real long lines will be detected. For the test images, the threshold value is chosen to be equal to 14. Extracted line drawing of Figure 3.5 which is shown in Figure 4.6a has two false lines. This result can be thought as

satisfactory and false lines can be removed by postprocessing. But extracted line drawing of Fig. 3.6 which is shown in Figure 4.6b has a lot of false lines and it is very difficult to remove these lines.



(a) (b)

Figure 4.6. Extracted lines by Hough transform (a) For Figure 3.5 (b) For Figure 3.6.

This algorithm gives the parameters of each line with infinite length. The two end points of the lines must be found using edge image. This postprocessing step requires extra work. It may lead to some erroneous starting and end points also. Some collinear points lying on different lines cause false lines passing through these points. It is difficult to detect these kinds of errors. Consequently, a better method must be used to extract line segments.

4.3. Line Detection Using Rotation Transformation

In this algorithm [14], edge image is rotated around its center by a certain angle step. Then, rotated image is searched in vertical and horizontal directions to find possible end points of horizontal and vertical line segments. If any end point pair is found, they are rotated backwards the same amount to obtain the corresponding line segments in the original image. This process is repeated until image is rotated by 90^{o}.

4.3.1. Algorithm

In Figure 4.7., line "AB" makes an angle of θ with X-axis. If the image is rotated about the center of the image plane by θ to X-axis, line "AB" becomes "A' B'" in the rotated image. Line "A' B'" is parallel to X-axis and it is very easy to find end points of this line by a horizontal search. End points of line "AB" are found by rotating backwards (rotation by $-\theta$) the detected end points of line "A' B'".

Each edge pixel in the edge image must be rotated by θ according to the Equation (4.4)

$$x' = x \cos\theta + y \sin\theta \qquad (4.4a)$$

$$y' = -x \sin\theta + y \cos\theta , \qquad (4.4b)$$

where (x',y') is the rotated coordinates of (x,y). Inverse transformation is given by the Equation (4.5).

$$x = x' \cos\theta - y' \sin\theta \tag{4.5a}$$

$$y = x' \sin\theta + y' \cos\theta \tag{4.5b}$$

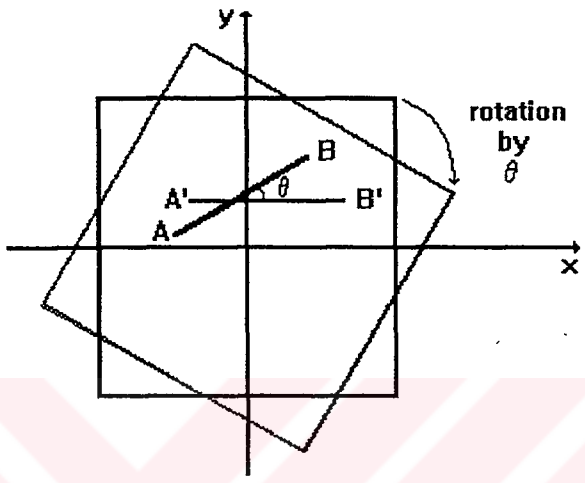


Figure 4.7. Rotation of the image plane by θ around the origin.

Straight line segment extraction using rotation transform algorithm can be explained as follows:

1) Initialize a 2D accumulator of size $(\sqrt{2}\, M/E) \times (\sqrt{2}\, N/E)$, where MxN is the size of the image which are equal to 256 for our images and E is the quantization interval which is identical to ρ in the Hough transformation.

2) Rotate each edge element e_i (x_i , y_i) by θ_j , where i= 1,2,...,N (number of edge elements) and j=1,2,...,L (number of quantization levels of θ) using Equation (4.4) and quantize the rotated coordinates (x',y') with multiples of E. Increase the content of the corresponding cell of the accumulator array by one. Repeat this step for every edge element of the original edge image.

3) Search each row of the accumulator array in the horizontal direction to detect the end points of any line segments. Rules for search process are as follows : If the content of an array cell is larger than a predefined threshold (LE, line element), the cell is considered as a line element. If there exists more than a number (ML, minimum line length) of line elements which are not separated by a certain gap (SG, separation gap) or larger, these line elements form a line segment. The first and the last line elements are regarded as the beginning and end points of this line segment, respectively. Explanations for parameters mentioned above are as follows:

LE : minimum content of an array cell which can be considered as a line element.

SG : minimum separation gap between two line segment lying on the same imaginary line.

ML : minimum number of line elements which form a line segment, or minimum line length.

4) Rotate backwards the end points of the line segments found in step 3 using Equation (4.5) to find the end points of the corresponding line segments in the original edge image.

5) Repeat step 3 and step 4 for the vertical direction.

6) Increase the rotation angle θ by the quantization step and repeat steps 1-5. The step size for rotation angle is identical to that of Hough transformation and must be chosen carefully.

4.3.2. Relationship Between Rotation Transformation and Hough Transformation

Rotation transformation is actually the generalization of the Hough transform. Equations related to both kinds of transformations are given below for convenience.

Rotation transform: $$x' = x \cos\theta + y \sin\theta \quad (4.6a)$$

$$y' = -x \sin\theta + y \cos\theta \quad (4.6b)$$

Hough transform: $$\rho = x \cos\theta + y \sin\theta \,. \quad (4.7)$$

x' and ρ are the same. On the other hand, y' is equal to the distance when the angle is equal to $\theta + 90^{o}$. Let this distance be called as ρ'. This equality can be shown as

$$\rho' = x \cos(\theta + 90) + y \sin(\theta + 90)$$

$$= -x \sin\theta + y \cos\theta$$

$$= y'. \quad (4.8)$$

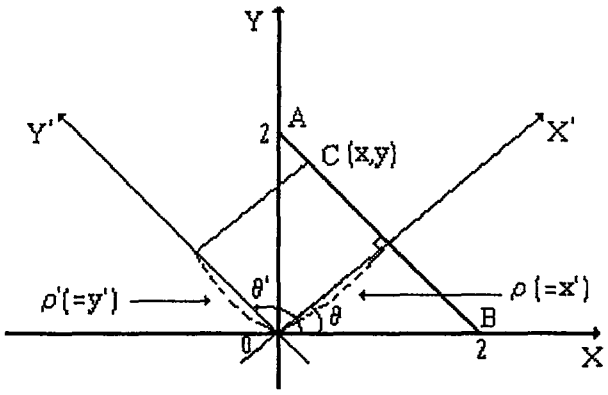


Figure 4.8. Parameters of Hough transform and rotation transform.

In Figure 4.8, ρ' is shown as the distance between the origin and the projection of point "C" onto the y'-axis. Variable ρ' has relative positional information about the edge elements which compose the straight line "AB". This information can be used for finding the end points of any straight line in the edge image. ρ, ρ' and θ are orthogonal to each other. Hence, a space called rotation transform or generalized Hough transform space can be defined using these variables. Any point (x,y) in 2D image plane is mapped to a curve in 3D rotation transform space. In Figure 4.8, parameters of the rotation transform and the Hough transform of a point "C" in the image plane are shown for a fixed θ. For rotation transform, instead of rotating point "C" around the origin coordinate system is rotated inversely.

A point $e_i(x,y)$ in the edge image is mapped to point $e_r(\theta,x',y')$, i.e. $e_r(\theta_i, \rho_j, \rho_k')$ in the rotation transform space, and to point $e_h(\theta_i, \rho_j)$ in the Hough transform space. If R_{ijk} and H_{ij} are the contents of the cell $(\theta_i, \rho_j, \rho_k')$ of the rotation transform accumulator array and the cell of (θ_i, ρ_j) of the Hough transform accumulator array, respectively, following equality can be written

$$H_{ij} = \sum_{k=-l}^{l} R_{ijk} \tag{4.9}$$

where l is the smallest integer greater than (image size x $\sqrt{2}$ / interval of ρ). This shows that Hough transform is the projection of rotation transform onto $\rho' = 0$ plane. The rotation transform maps all edge elements to a cell (θ_i, ρ_j) in the Hough transform space according to the value of ρ', distance parameter for θ' $(=\theta + 90^o)$. By searching ρ_k' for a straight line (θ_i, ρ_j), necessary information related to the line (end points, line length, separation gap) can be extracted easily. Rotation transform

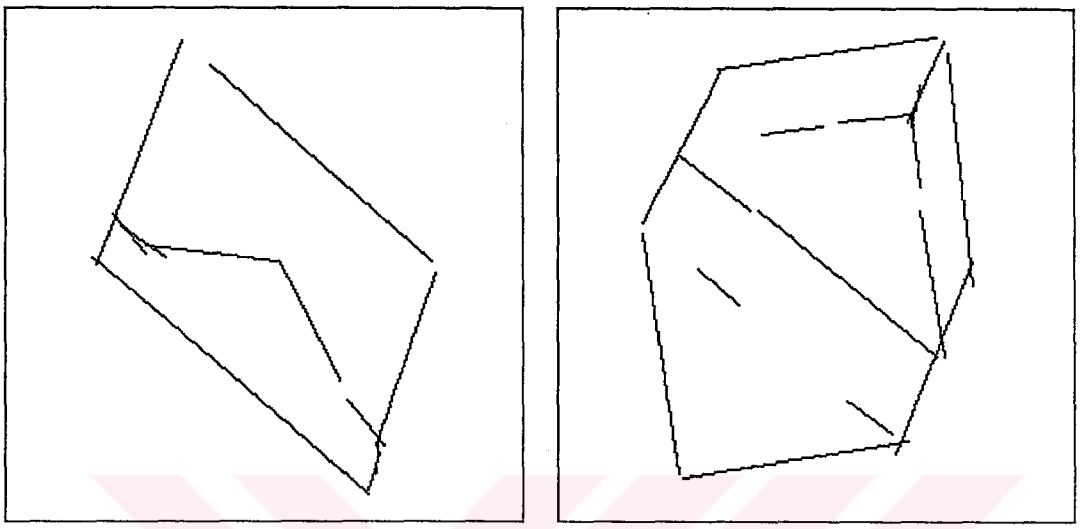
may be thought as the generalized Hough transform to which structural information of a straight line is added.

4.3.3. Experimental Results

Line detection using rotation transform algorithm is tested using the test images. Execution time is 17 seconds. Resultant images are shown in Figure 4.9. According to the parameters chosen, detected lines change. "LE" (minimum content value of an array element which can be considered as a line element) is chosen to be equal to 1. Minimum line length "LE" can be chosen from 7 to 15 pixels. If it is chosen to be too long, short line segments are missed where long line segments are detected successfully. When "LE" is small, line segments with small lengths are detected easily. But, some false lines can be detected because of noise. Also, for long line segments, besides the actual lines, some shorter line segments are detected. A postprocessing step has been applied for these kinds of lines. If two line segments are very near to each other, these two lines are combined to form a single line segment. It works successfully. Minimum separation gap between two line segments which lie on the same imaginary line, "SG", can be chosen from 1 to 3. When it is greater than 1, the probability of detecting false lines is very high. On the other hand, if "SG" is equal to 1, algorithm can miss some lines or detect lines with lengths smaller than the actual lengths of the corresponding lines in the image. Quantization interval of ρ , "E", is chosen as 2. Quantization interval for θ is equal to 7.5^{o}. If it is chosen too large, some details could be lost.

Satisfactory results have been obtained using this algorithm. Choice of parameter values gives great flexibility to the algorithm. Extracted line drawing is

still not perfect. This is mainly due to the edge detection process. Performance of the algorithm depends on the edge detection process. It works satisfactorily for a given edge image.



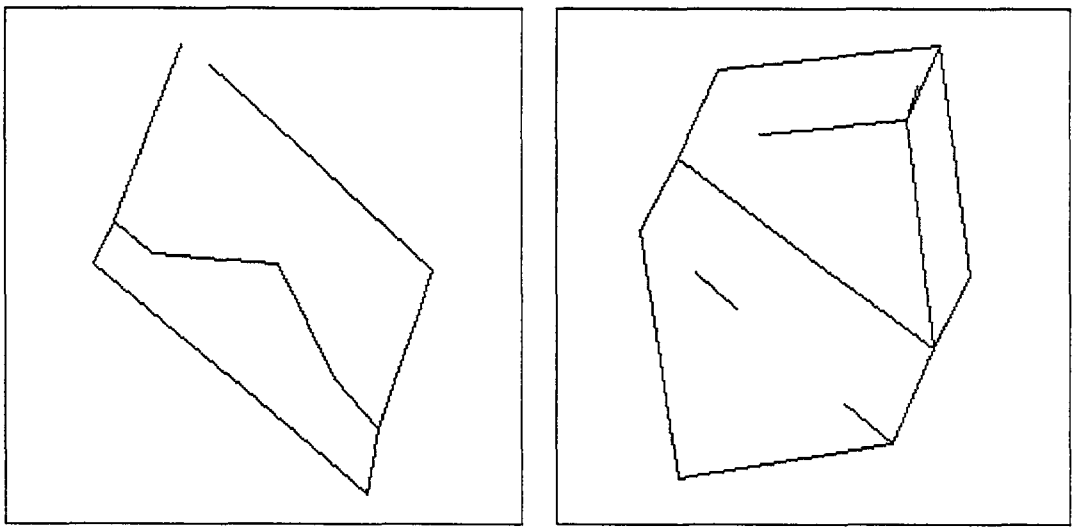
(a) (b)

Figure 4.9. Extracted lines by rotation transform (a) For Figure 3.5 (b) For Figure 3.6.

As it can be seen from Figure 4.9, the rotation transformation method gives satisfactory results. Finally, a postprocessing step must be applied. This is done for removing double lines, connecting broken lines and finding corners. If the endpoints of the two lines are close to each other and they are nearly parallel, the lines are combined. If the endpoints of the two lines are near to the intersection points of the lines, the endpoints are changed to that position.Figure 4.10 shows final line drawings of the test images.

Line drawings in Figure 4.10 are not perfect. There are some missing lines. Endpoint locations of the line segments are not very accurate. Hence,

higher-level processing stages that will use these line drawings must be capable of handling this situation.



(a) (b)

Figure 4.10. Final line drawings (a) For Figure 3.5 (b) For Figure 3.6.

CHAPTER V

POLYHEDRAL OBJECT RECOGNITION BY GEOMETRIC HASHING

Geometric hashing [7] is a model-based object recognition method for the recognition of 3-D objects with unknown 3-D position and orientation from a single 2-D image of the objects. Local features like corners and lines are used. Models are formed using geometric relations of the selected features. In training phase, object descriptions are found and a hash table is formed. This process is done before recognition. 3-D objects are modeled using 2-D views of the objects taken from points on a sphere around the objects. Recognition phase uses a single 2-D image of an unknown object and the hash table which is prepared in the training phase.

5.1. Problem Definition

The aim is to find a solution to the recognition of a 3-D object from a 2-D view taken from an unknown viewpoint. To solve this problem, first, imaging system must be modeled.

Imaging systems use perspective projection for 3-D to 2-D transformation. The parameters of the perspective projection can be computed from six pairs of a model and image points. But, it is very difficult to solve perspective

projection parameters. Generally, parallel (orthographic) projection is used instead of perspective projection. In orthographic projection, the direction of the projection is orthogonal to the image plane. So, object size remains constant as it moves further from the camera. For a good approximation to the perspective projection, a scale factor must be added to the orthographic projection. If the object is deep with respect to its distance to the viewpoint, a single scale factor will be insufficient for a good approximation. If z_{min} is the distance of the closest point of the object to the camera, and z_{max} is the farthest point of the object from the camera, the necessary condition for a good approximation to the perspective projection is $z_{max}-z_{min} << (z_{max}+z_{min})/2$.

Imaging transformation consists of a 2-D translation (in the plane orthogonal to viewing direction), a 3-D rotation and scaling. The aim of the object recognition system is to find the parameters of this transformation.

Geometric hashing method solves this problem using multiview approach. If the viewing angle is kept constant, then two different images of the object will be in a similarity correspondence, i.e. 2-D rotation, translation and scaling. Hence, 3-D recognition problem becomes a 2-D matching problem. However, since the viewing angle is arbitrary, training phase uses views taken from several viewpoints. Viewpoint information is added to the model description. A discrete set of viewing angles which is obtained by tessellating the viewing sphere is used.

Similarity transformation can be represented by a 2-D matrix A and a 2-D translation vector b. Each point x in the first image is translated to the point Ax+b in the second image. Now, the problem becomes the matching of the model points to the scene points. Some special points must be used for this operation. These

points must be invariant under 2-D rotation, translation and scaling and will be called as 'interest points'. Choice of the interest points depends on the model database. For example, line endpoints can be used for polyhedral objects. On the other hand, sharp convexities, deep concavities and zero curvature points can be used for curved objects.

The parameters of a similarity transformation can be found using the correspondence of a pair of points in the model with a pair of points in the object. So every such correspondence can be thought as a candidate for a similarity transformation. Each candidate must be verified using the full descriptions of the model and the scene.

This scheme is computationally very expensive for object recognition systems. If m is the number of the model interest points and n is the number of the scene interest points, the worst-case number of hypothesis to be verified is $2\binom{m}{2}\binom{n}{2}$. Thus, the recognition complexity in the worst-case is $O(m^2n^2t)$ where t is the complexity of verifying the model against the scene [7]. If m and n are of the same magnitude and t is at least the magnitude of n, then the worst-case complexity will be $O(n^5)$. There are more than one view for an object. Also, the number of models may be larger than one. These will increase the complexity further.

Geometric hashing method aims at reducing the complexity of the recognition task by dividing the algorithm into two independent stages. The first one is the training or preprocessing stage which is applied to the model points. This stage does not use any information related to the scene points. Hence, it is performed off-line before the recognition task. The second stage is the recognition stage in which points extracted from the scene and the data prepared in the training stage are used. The execution time of this stage gives the actual recognition time.

5.2. Similarity Invariant Representation of 2-D Point Sets

To reduce the recognition complexity, model and scene points must be represented independently and this representation must allow comparison of point pairs. A new coordinate system which is defined by an ordered pair of points is used for this purpose. This pair is used as the unit vector defined in the x direction. One of the points will have the (0,0) coordinate and the other will have the (1,0) coordinate as in Figure 5.1. The unit vector in the y direction is the orthogonal vector of the same length pointing 90 degrees in the counter-clockwise direction. These two vectors define 2-D space uniquely. The point pair will be called as 'basis pair'. After defining the new coordinate system, coordinates of all other points are found using new basis vectors.

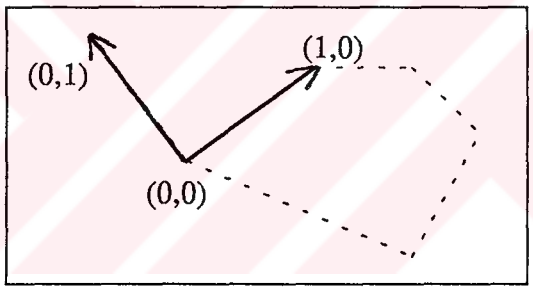


Figure 5.1. A polygon with an edge chosen as a basis.

Any similarity transformation applied to the point set preserves the coordinates of the points if the same ordered pair is chosen as basis pair. This can be shown as follows: Let u and v be the basis vectors related to chosen basis pair. Any point p can be represented as

$$p=\alpha.u+\beta.v \tag{5.1}$$

(α,β) is the new coordinate of the point p. If a similarity transform T is applied to the point set, p will be transformed to

$$T.p=\alpha.Tu+\beta.Tv \qquad (5.2)$$

If the same ordered pair of points is used as basis pair, basis vectors will be Tu and Tv. Hence, point T.p will have the same coordinates (α,β) in the new coordinate system after the transformation.

5.3. Recognition of 3-D Objects

5.3.1. Preprocessing:

Assume that a view of a model contains m interest points. For each ordered pair of these points, the coordinates of the remaining m-2 points are found in the new coordinate system defined by this basis pair. After quantization, these coordinates are used as index entries for a hash table. For each coordinate , a triple (model, viewing angle, basis pair) is recorded in the specified location of the hash table. Flowchart of this stage is given in Figure 5.2.

This redundant representation gives robustness to the algorithm and reduces the recognition time. Actually, coordinates calculated using a single basis pair is enough for recognition. But, in some cases (like occlusion or imperfect interest point extraction) corresponding point pair in the scene may be invisible. Hence, recognition fails. To overcome this problem, all possible model interest point pairs are used in the preprocessing stage.

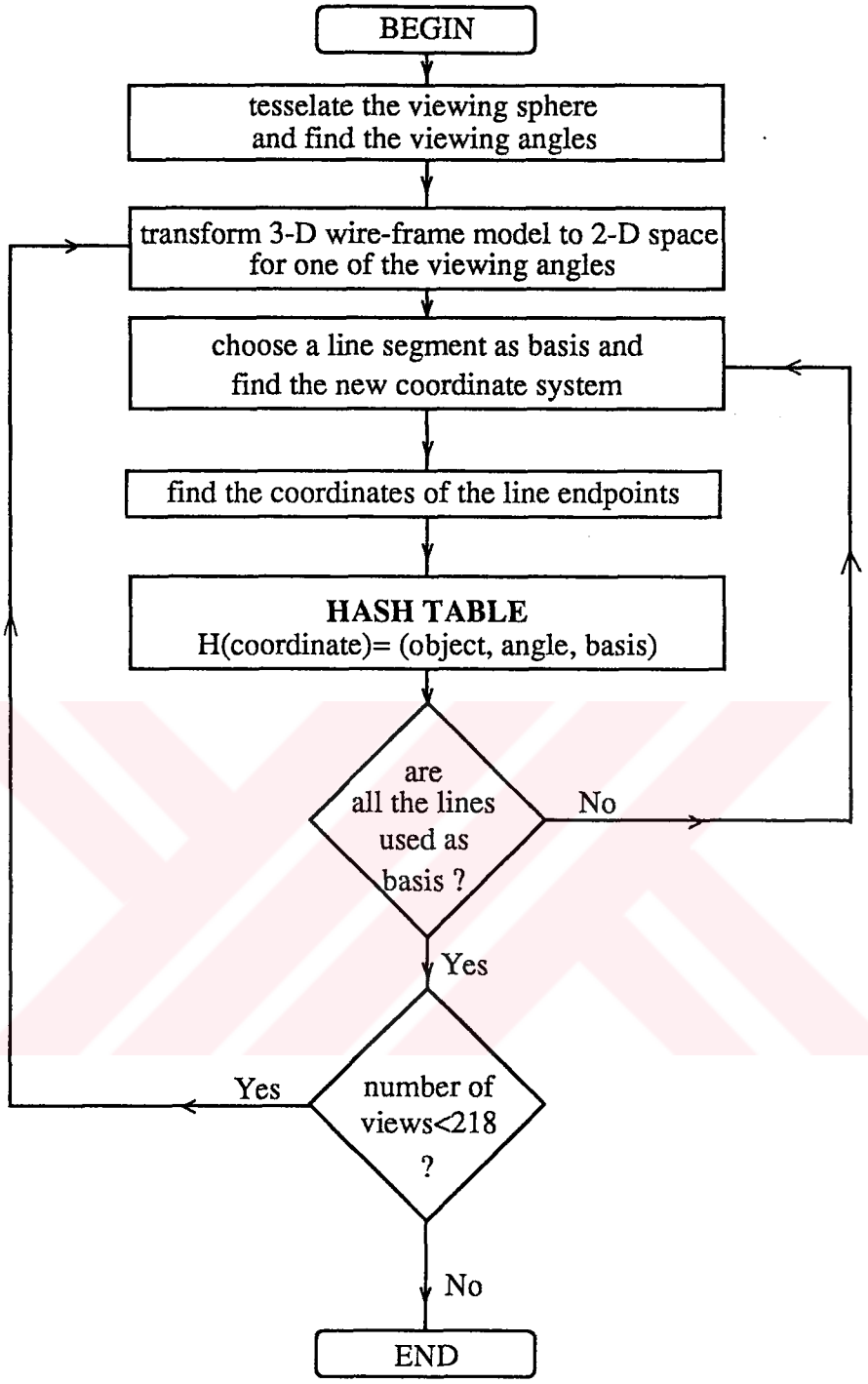


Figure 5.2. Flowchart of the preprocessing stage.

Complexity of this preprocessing stage for one view of a model is $O(m^3)$ [7]. Separate viewpoints and models bring a constant factor. Any model object which will be added to the model database is processed independently from the models that are already in the database. The preprocessing is done off-line and does not affect the recognition time.

5.3.2. Recognition

First, interest points in the given image are extracted. Assume that there are n interest points extracted. An ordered pair of these points is chosen as basis pair and the coordinates of all the remaining n-2 points are found using this basis. For each such coordinate (after quantization), corresponding entry in the hash table is checked to see the triples which are recorded in the preprocessing stage. For each triple (model, angle, basis) appearing in the specified location, the score of that triple is incremented.

After this process, if one of the triples reaches a large score, it is chosen as a matching candidate. Because of occlusion or imperfect interest point extraction, there may be no matching candidate. If this situation occurs, the same process is repeated using a different basis pair.

When a matching candidate is found, it is verified using other model and scene points. The transformation between the point pairs defines a candidate transformation between the scene and the model. Correspondence of two point pairs generally does not give the transformation between the scene and the model very accurately. But, it gives some further point correspondences. Using these, model points are aligned against to scene points by means of best-least squares matching

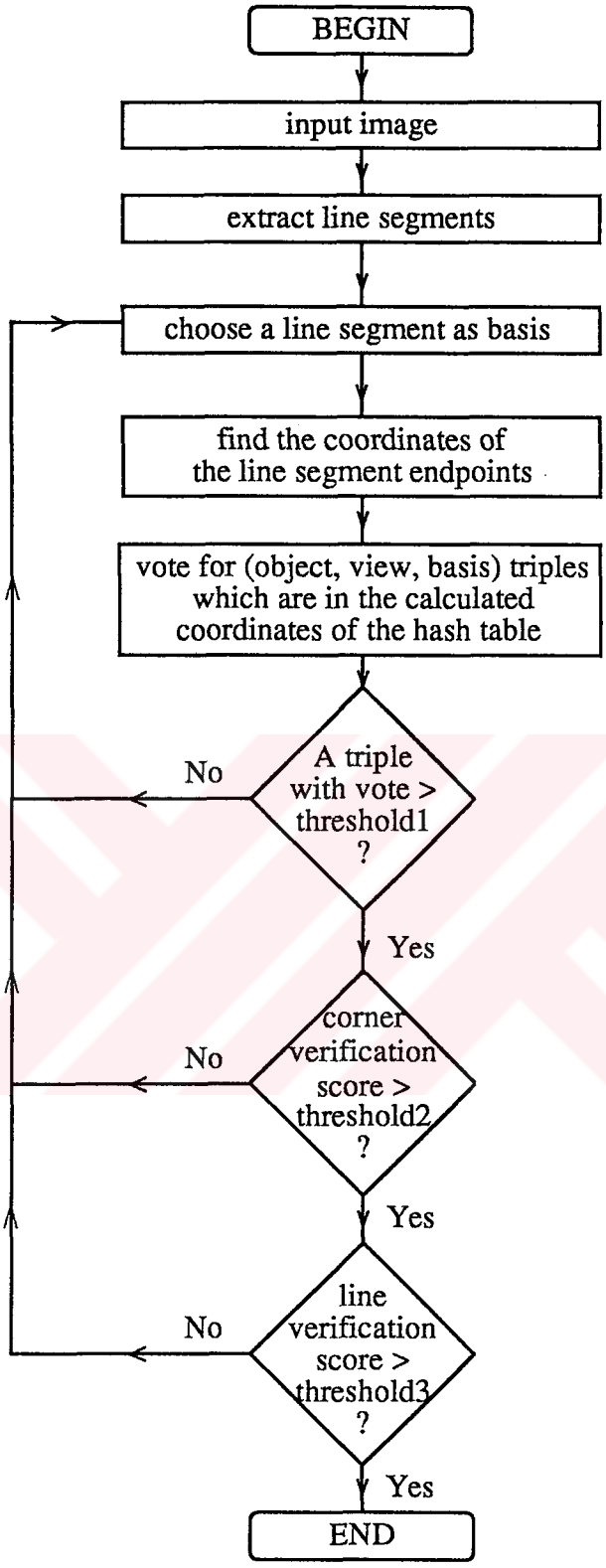


Figure 5.3. Flowchart of the recognition stage.

criterion. After this, model corners and lines are verified using scene corners and lines. If the verification result is satisfactory, hypothesis is decided to be a correct match. If it is unsatisfactory, whole process is repeated using a different basis pair. Flowchart of the recognition stage is given in Figure 5.3.

If a correct basis in the scene is chosen, correct model takes a high number of votes. Voting process for a basis pair is linear in the number of scene interest points. If there is one object in the scene and the object belongs to the model database, it will be recognized after first trial with a complexity of O(n). On the other hand, if the model does not belong to the model database it will be rejected after $O(n^2)$ trials. Hence, the worst-case complexity is $O(n^3)$ [7]. In most situations, the recognition complexity is between O(n) and $O(n^3)$.

5.3.3. The Best-Least Squares Match

Correspondence of model and scene point pairs gives an initial estimation for the transformation between the model and the scene. If, in preprocessing stage, CAD-like systems are used, models are noiseless. But, in recognition stage, interest point extraction can be very noisy. So, the transformation which is found by the alignment of the model point pair against to the scene point pair can be erroneous. But, some further point correspondences can be found using this transformation. Then, using this information, a more accurate transformation can be found by the best-least squares error criterion. The approach below is used in [15] in which an application of geometric hashing method for the recognition of planar objects is described.

Let $(u_i)_{i=1}^n$ and $(v_i)_{i=1}^n$ be the sequence of points to be matched. It is required to find a transformation T such that the square of the distance between $(Tu_i)_{i=1}^n$ and $(v_i)_{i=1}^n$ will be minimized. T is defined as follows:

$$Tu_i = Au_i + b \qquad i = 1,2,\ldots,n \tag{5.3}$$

where

$$A = \begin{pmatrix} a_{11}a_{12} \\ a_{21}a_{22} \end{pmatrix}, \quad b = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}.$$

The aim is to find the parameters of the matrix A and the vector b such that the error δ defined as

$$\delta = \min_T \sum_{i=1}^n |Tu_i - v_i|^2 \tag{5.4}$$

will be minimized.

Without loss of generality, $(u_i)_{i=1}^n$ can be translated such that $\sum_{i=1}^n u_i = 0$.

Then,

$$\begin{aligned} \delta &= \min_{A,b} \sum_{i=1}^n |Au_i + b - v_i|^2 \\ &= \min_{A,b} \left[\sum_{i=1}^n |b - v_i|^2 + \sum_{i=1}^n |Au_i|^2 + 2\sum_{i=1}^n b.Au_i - 2\sum_{i=1}^n Au_i.v_i \right] \end{aligned} \tag{5.5}$$

However,

$$\sum_{i=1}^n b.Au_i = b.A\left(\sum_{i=1}^n u_i\right) = 0 \tag{5.6}$$

Hence, A and b can be found independently. b can be found as follows:

$$\frac{\partial}{\partial b}\left[\sum_{i=1}^{n}|b-v_i|^2\right]=\sum_{i=1}^{n}2(b-v_i)=0 \Rightarrow b=\frac{1}{n}\sum_{i=1}^{n}v_i \tag{5.7}$$

If g(A) is defined as

$$g(A)=\sum_{i=1}^{n}|Au_i|^2-2\sum_{i=1}^{n}Au_i.v_i \tag{5.8}$$

To find the minimum of g(A), following system of four equations must be solved.

$$\frac{\partial g(A)}{\partial a_{ij}}=0 \quad i=1,2 \quad j=1,2 \tag{5.9}$$

g(A) is a quadratic function in each of its unknowns. So, Equation (5.9) is a system of four linear equations. Actually, it is two independent sets of two linear equations with two unknowns.

$u_i=(u_i^1,u_i^2)$ and $v_i=(v_i^1,v_i^2)$ are two dimensional vectors for each i=1,2,...,n. Solution will be shown using four n-dimensional vectors which are defined as below:

$$\begin{aligned} U^j &= (u_i^j)_{i=1}^n \\ V^j &= (v_i^j)_{i=1}^n. \end{aligned} \qquad j=1,2.$$

Then, the solution of Equation (5.9) is as follows:

$$a_{11} = \frac{(U^1.V^1)(U^2.U^2) - (U^2.V^1)(U^1.U^2)}{\Delta}$$

$$a_{12} = \frac{(U^1.U^1)(U^2.V^1) - (U^1.V^1)(U^1.U^2)}{\Delta}$$

$$a_{21} = \frac{(U^1.V^2)(U^2.U^2) - (U^1.U^2)(U^2.V^2)}{\Delta}$$

$$a_{22} = \frac{(U^1.U^1)(U^2.V^2) - (U^1.V^2)(U^1.U^2)}{\Delta}$$

where

$$\Delta = (U^1.U^1)(U^2.U^2) - (U^1.U^2)(U^1.U^2) \qquad (5.10)$$

((.) shows the vector dot product.)

5.3.4. Verification

In recognition stage, after the voting process, the triple (model, view, basis pair) with vote larger than a predefined threshold is chosen as a candidate for a possible match. Best-least squares algorithm makes the transformation more accurate. Finally, candidate transformation between the model and the object must be verified. This verification is done by comparing the model corners and lines with the scene corners and lines. In the implementation, two verification stages are used. First stage is relatively simple and rapid. It eliminates many false matches. The second stage is more accurate and slower. In the first stage, line segment endpoints of the scene are compared with the corners of the model. If there are a sufficiently large number of correct matches, the model passes this verification step. A model corner matches with a scene line endpoint if the distance between them is sufficiently small. The threshold values used are given in the next sections.

Second verification stage compares the line segments of the scene with the line segments of the model. This stage is rather time consuming but reduces the likelihood of accepting a false match as a real one. According to the spatial relations of the model and the scene lines, a verification score is incremented or decremented. If a transformed model line is sufficiently close and parallel to a scene line and the endpoints of these two lines are sufficiently close to each other, verification score is incremented by 2. If only one endpoint of the model line is close to the scene line endpoints, score is incremented by 1. All model lines are checked in this manner. Then, for each unmatched scene line the score is decremented by 2 and for each unmatched model line the score is decremented by 1.

A model and a view are said to be a correct match if the verification score is larger than a predefined threshold value. When this happens, recognition algorithm terminates. If the score is small, algorithm returns to the voting process again.

5.4. Implementation

Hash table method is implemented for a database of polyhedral objects. Ordered line segments are used as basis vectors. Hence, if n is the number of edge segments, 2n basis vectors are used for the hash table formation and the recognition tasks. If the number of the corners is close to n, the complexity of the preprocessing stage is $O(n^2)$. The worst-case recognition complexity is $O(n^2)$. This shows that the use of line segment information reduces the recognition and the preprocessing complexity by a factor of n [7].

In preprocessing stage, separate views of the models are needed. They are obtained using wire-frame models of the objects. For a given viewpoint, parameters of the 3-D to 2-D transformation are found. After the transformation, a hidden line elimination algorithm [16] is used. The computer program given in [16] is used with some modifications. Some sample outputs of the program are given in Figure 5.4. Viewpoints are found by tessellating the viewing sphere using square patches. This is done by first imposing a regular grid on the faces of a cube. Then, the grid elements are transformed radially onto the viewing sphere. In the experiments, 6 by 6 grids on each face of the cube with a total of 218 views give accurate results. If the number of viewpoints is increased more accurate results may be obtained. But, this increases the storage requirements of the hash table. Hence, 218 views are enough. If more localization accuracy is required, obtained viewing angle can be used as an initial estimation for another method. Best-least squares error criterion is used for this purpose.

Two models are used for the experiments. Separate hash tables are used for each of them. Actually, this is not a desired situation. It increases the recognition time. But, the size of the hash table for a single object is very large. Because of the memory restrictions of an IBM compatible PC, a single hash table can not be used for several objects. Even when the number of corners and lines of an object is large, it is not possible to use a single hash table for that object. In the experiments, for one of the objects, two hash tables are used. Each one has different views of the object.

Line segments in a given image are extracted using image processing tools which are described in the previous chapters. First, the image is smoothed. Second, edges are detected using Sobel operator. Lines are detected using rotation transformation. Then the aspect ratio of the extracted line drawing is corrected. Resultant line drawing is not perfect. Some lines are missing while some false lines

are detected. Also, the locations of the line endpoints are not very accurate. This situation will be referred as "image noise" in the remaining of this chapter.

A nonlinear quantization scheme is used for the size of hash table bins. This is the approach used in (15). It is assumed that if the coordinate value is large, the effect of the image noise on this value is large. So, the size of the hash table bins becomes larger when the coordinate value increases. The amount of growth is linear in each coordinate value. A point with coordinates (x,y) belongs to bin (i,j) such that $X(i-1)\leq x<X(i)$ and $Y(j-1)\leq y<Y(j)$. X(.) and Y(.) are defined as:

$$X(0)=Y(0)=0,$$

$$X(i)-X(i-1)=Y(i)-Y(i-1)=i\delta \quad i=1,2,...,$$

$$X(i)-X(i+1)=Y(i)-Y(i+1)=i\delta \qquad i=-1,-2,... . \tag{5.11}$$

In the experiments, $\delta=0.005$ is used.

Voting is done by choosing an ordered line segment as a basis. Then, the hash table bins which correspond to the coordinates of the scene points are found. Vote of the each triple in the chosen hash table bin is incremented by 1. Also, the neighboring bins of the chosen bin are used for voting. This reduces the loss of some votes because of the image noise. Triples with votes larger than 0.3 times the number of the line endpoints of the image are verified using corner and line endpoint information. The threshold for corner verification score is chosen as 0.6 times the number of line endpoints of the image and the threshold value for the line verification score is chosen as 0.8 times the number of line segments in the image. These values are found by trial-and-error.

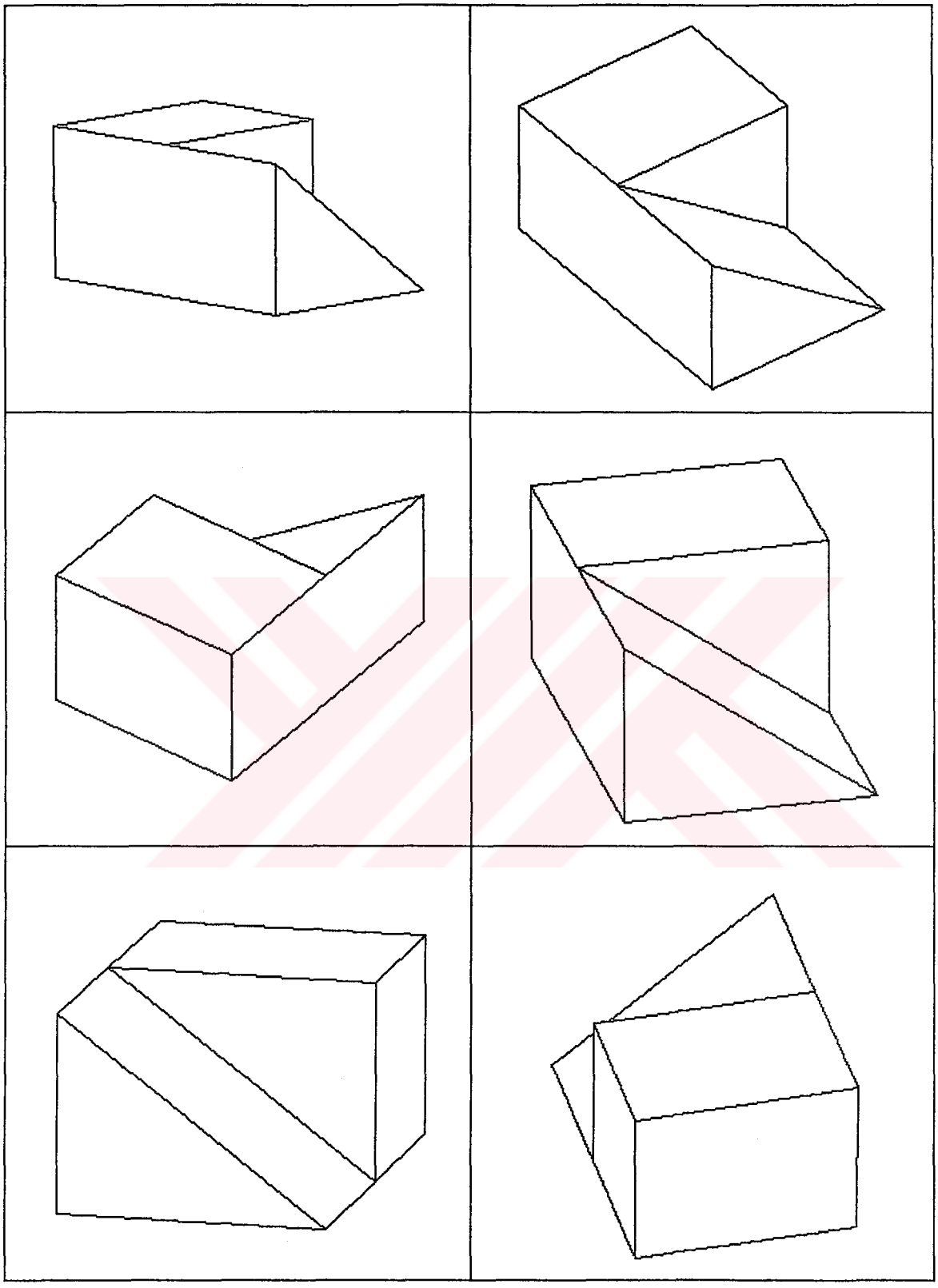


Figure 5.4. Sample outputs of hidden-line elimination program.

5.5. Experimental Results

Two objects (which will be called as 'Object1' and 'Object2' in the remaining of this chapter) with two images of each are used in the experiments. Images are shown in Figure 5.5 and Figure 5.6. Hash tables are formed for each model. The recognition algorithm is tested using the extracted line drawings of the images and the hash tables.

When the hash table of Object1 is used as the model, two images of Object1 are successfully recognized. Model is aligned to the image in two steps. First, using hash table votes, basis vectors of the image and the model are aligned. Then, the transformation which is found using the best-least squares criterion is applied to the model. Figure 5.7 shows the results of these two alignments. For one of the images, two separate model views were found. They are very close to each other as it is seen from Figures 5.8 and 5.9. On the other hand, two images of Object2 are tested using all possible basis vectors. But, they are rejected as a model object. Some statistical results related to the recognition process for this hash table are given in Table 5.1a.

For Object2, two hash tables with different views are used. Two images of Object2 are recognized successfully. The images of Object1 are decided to be unknown objects. The recognition results are given in Figures 5.10 and 5.11 and Table 5.1b.

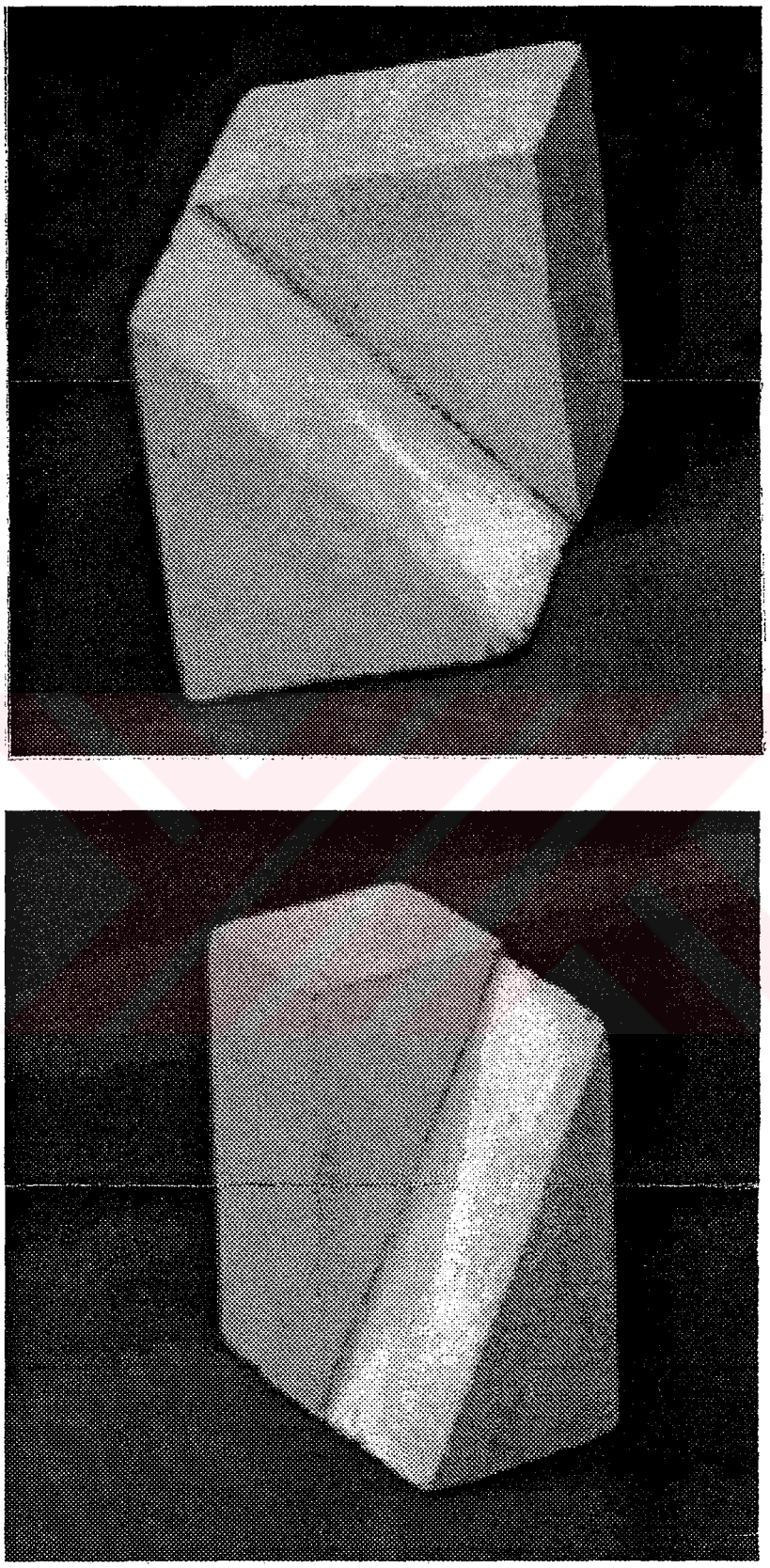


Figure 5.5. Original images of Object1

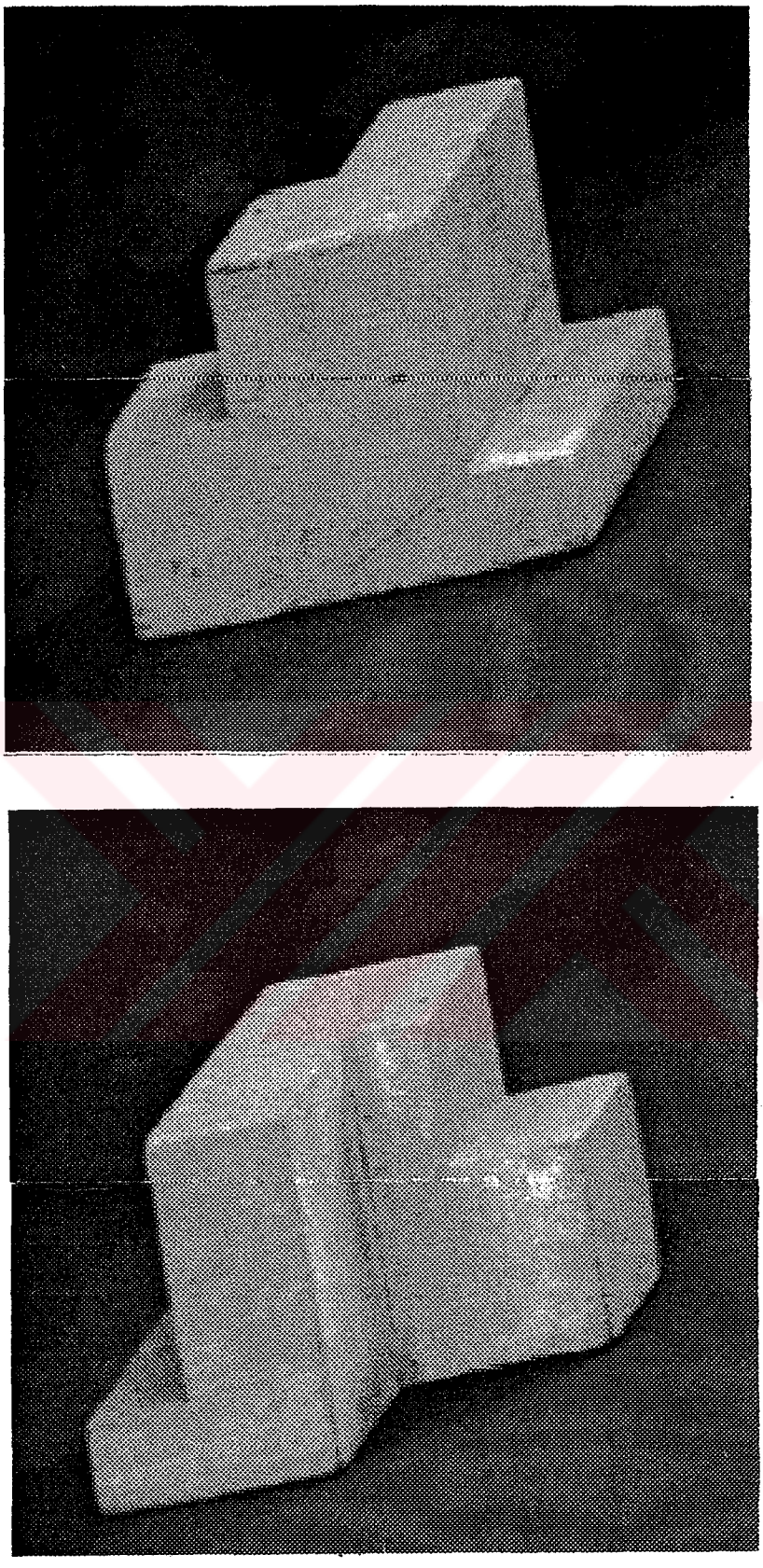


Figure 5.6. Original images of Object2

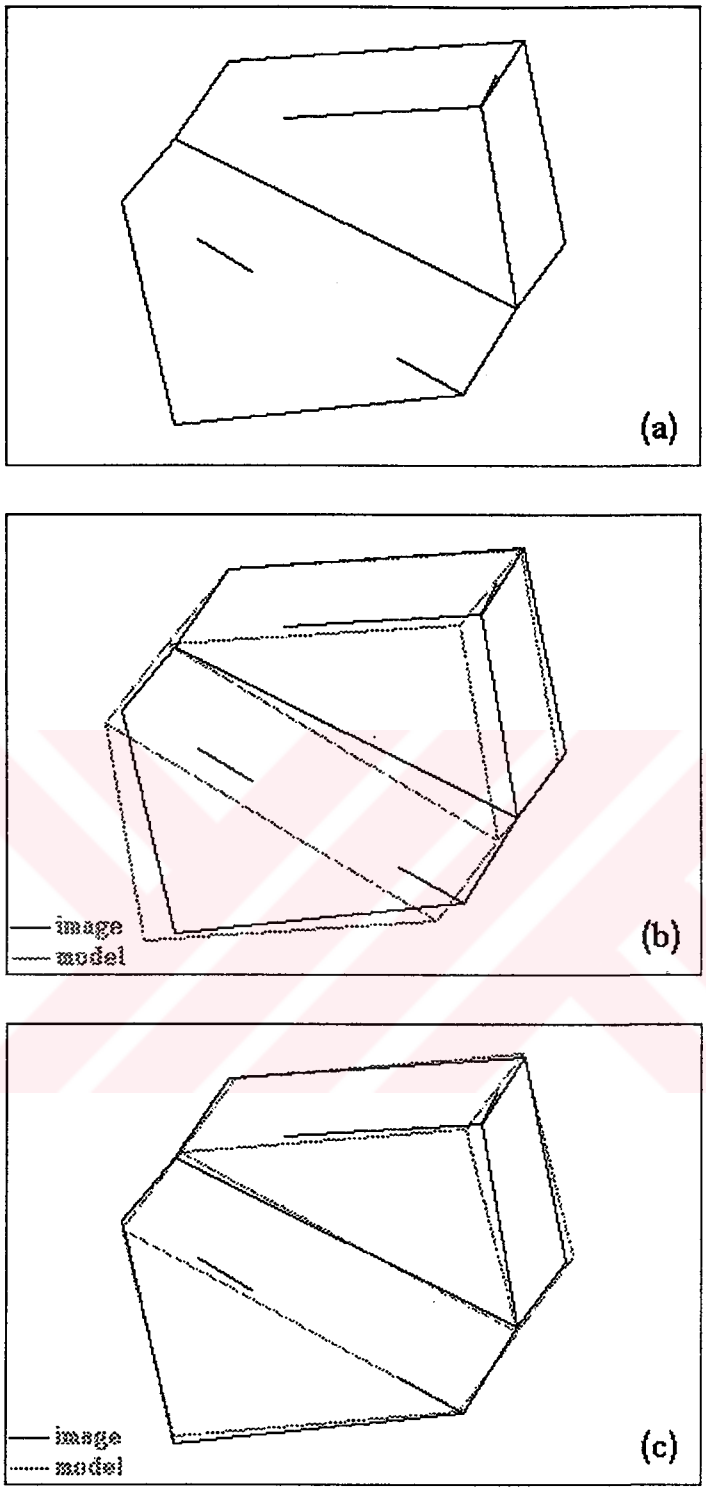


Figure 5.7. Experimental results for the first image of Object1 when the hash table of Object1 is used. (a) image (b) alignment of the image and the model using basis vectors (c) alignment after the best-least squares error transformation.

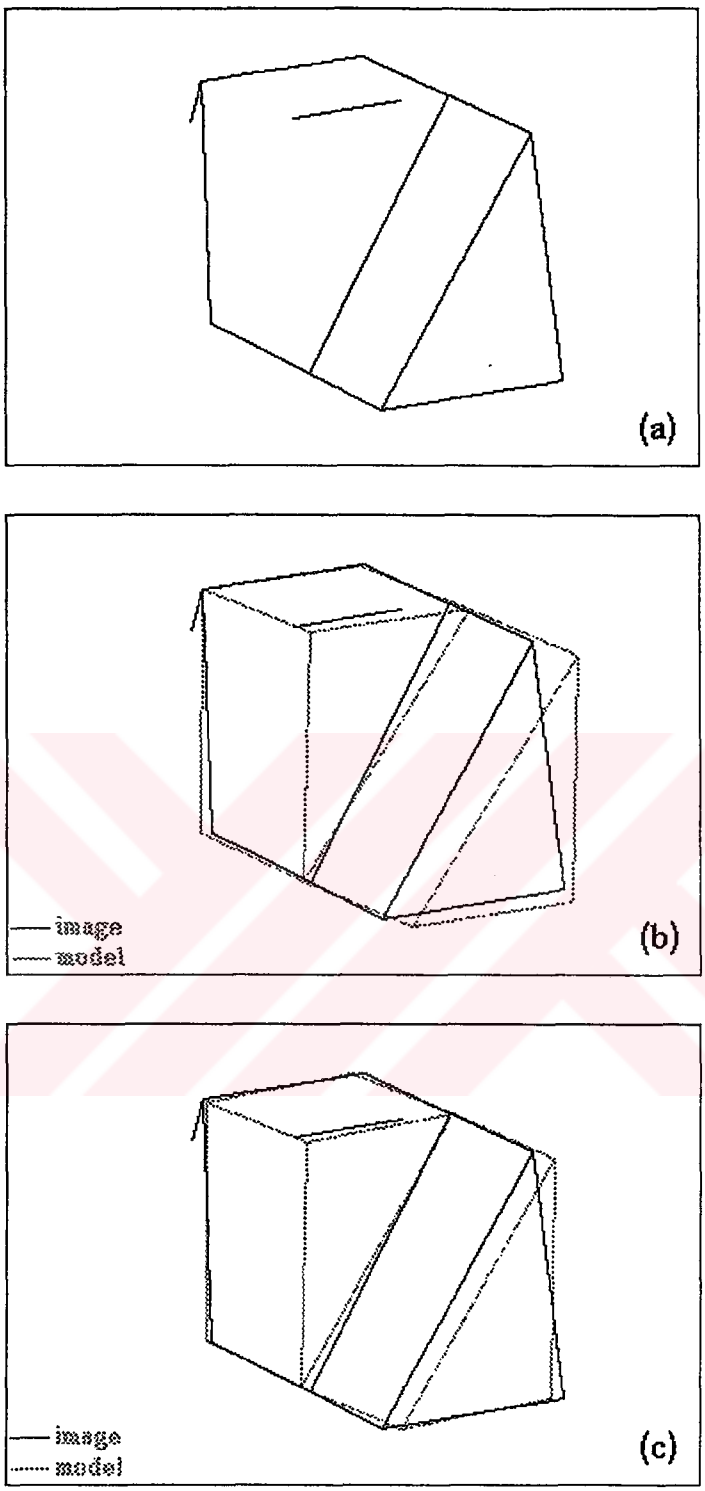


Figure 5.8. Experimental results for the second image of Object1 when the hash table of Object1 is used. (a) image (b) alignment of the image and the model using basis vectors (c) alignment after the best-least squares error transformation.

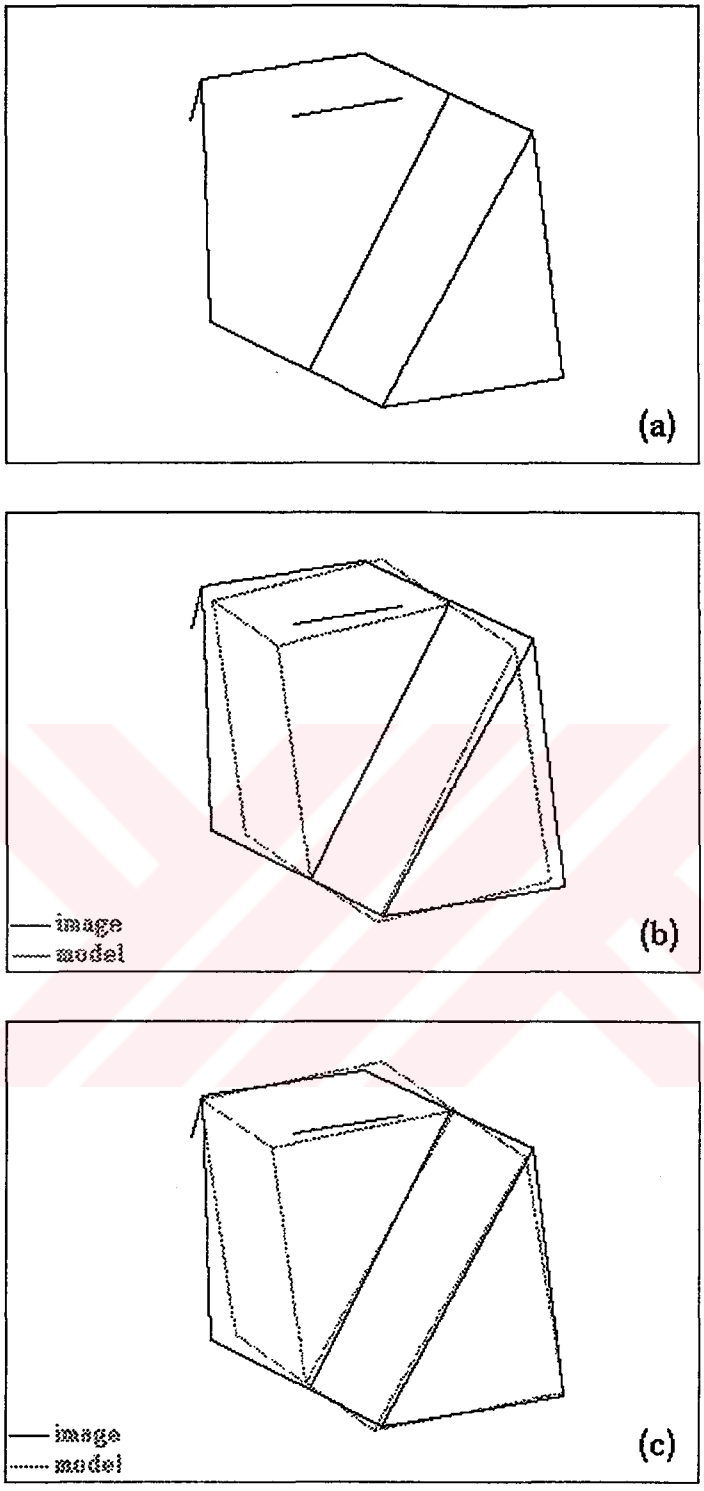


Figure 5.9. Experimental results for the second image of Object1 when the hash table of Object1 is used. (a) image (b) alignment of the image and the model using basis vectors (c) alignment after the best-least squares error transformation.

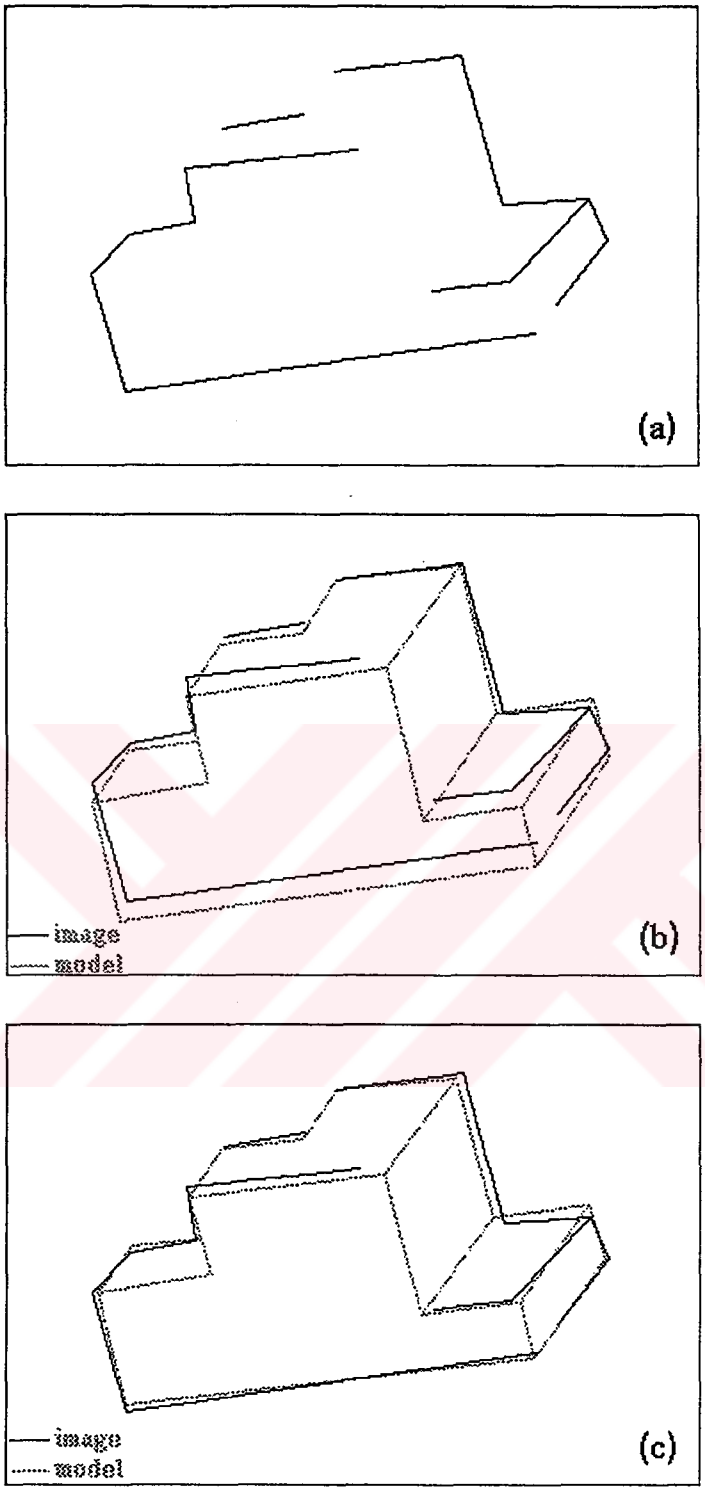


Figure 5.10. Experimental results for the first image of Object2 when the hash table of Object2 is used. (a) image (b) alignment of the image and the model using basis vectors (c) alignment after the best-least squares error transformation.

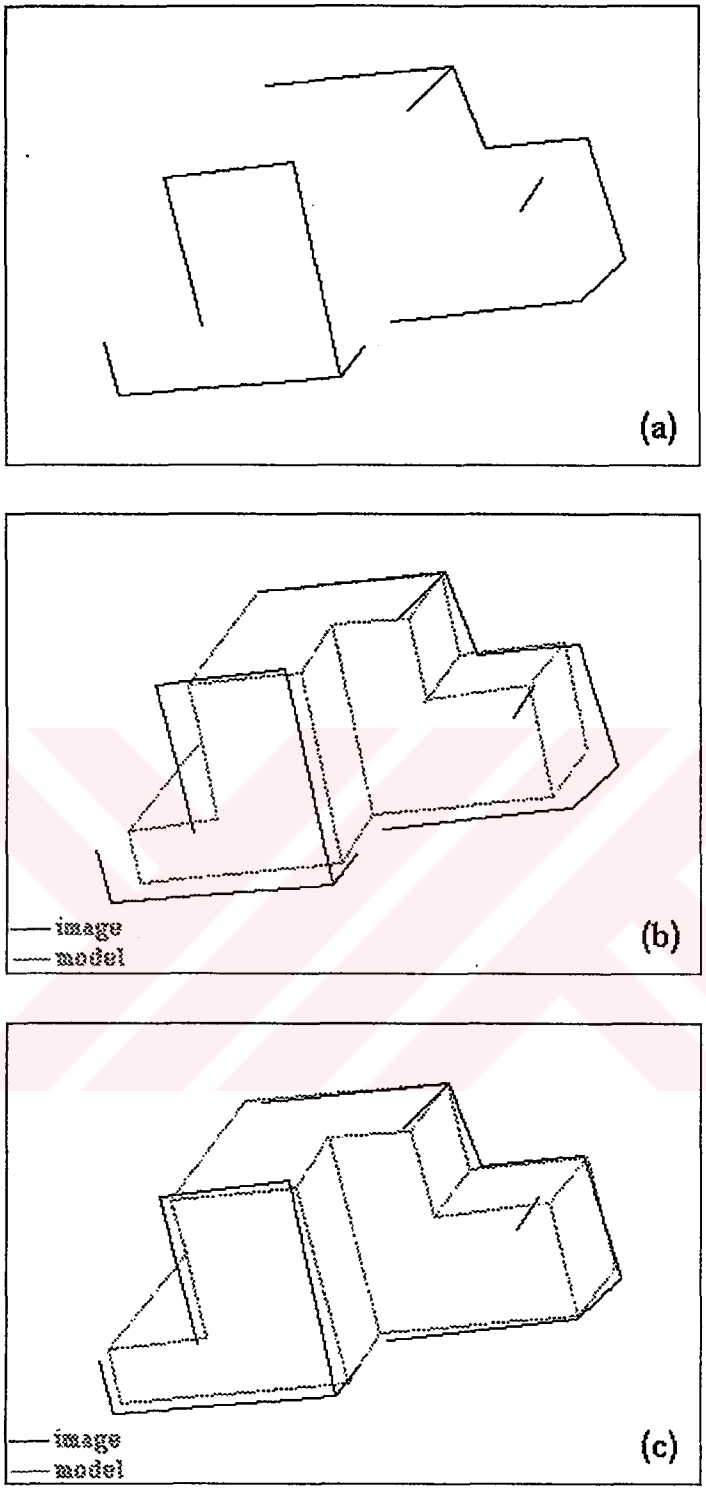


Figure 5.11. Experimental results for the second image of Object2 when the hash table of Object2 is used. (a) image (b) alignment of the image and the model using basis vectors (c)alignment after the best-least squares error transformation.

Table 5.1. Voting scores and execution times (scores under the fractions are related to false matches with highest verification scores).

(a)

hash table of Object1	hash table vote	corner verification score	line verification score	execution time (seconds)
1st image of Object1	5/5	9/8	20/2	1
2nd image of Object1	4/4	7/6	17/8	1
1st image of Object2	6	5	X	44
2nd image of Object2	6	5	X	45

(b)

hash table of Object2	hash table vote	corner verification score	line verification score	execution time (seconds)
1st image of Object1	5	9	-15	92
2nd image of Object1	4	7	-4	100
1st image of Object2	10/9	15/9	21	4
2nd image of Object2	12/9	15/11	16/2	39

In Table 5.1, for different images, voting scores are given. For a recognized object, scores related to a false match with the highest verification score is shown also. If the object is rejected, highest verification scores are given. Recognition and rejection times are also presented.

Table 5.1 shows that a wrong model view can take votes approximately equal to the votes of a correct view. This is also true for the corner verification scores. But, there is a great separation between the line verification scores of the correct view and a false view. This makes the method reliable. The probability of accepting a wrong view as the correct view is very small. The threshold values must be low enough to avoid missing a correct view. When they are too low, the number of the model views to be verified becomes high. The best value is dependent on the models and the performance of the line extraction stage.

According to the tables, it is clear that if the image belongs to the model object, it is recognized and aligned in several seconds. This duration does not include the feature extraction time. Recognition time related to the second view of Object2 is very large compared to the others. This is due to the use of two separate hash tables for that object. If a single hash table were used, the recognition time could fall to several seconds. Actually, this image is recognized in 2 seconds when the hash table with the corresponding model view is used. Rejection times of the images are large compared to the recognition times as expected. Because, an image is rejected only after all the edge segments are used as basis vectors. This takes relatively large time. Execution times mainly depend on the sizes of the hash tables.

In this algorithm, for a single image, more than one model object may pass all the verification stages. Current implementation chooses the first matching model as the correct model. If a single hash table is used for all the model objects,

the model with the largest number of votes can be chosen after the hash table voting process. To implement the same operation when separate hash tables are used, the voting process can be repeated for each hash table. Then, the model with largest number of votes can be chosen to go through the corner and line verification stages.

Some views of the model objects can be the same. This may lead to wrong decision about the object. This problem can be eliminated by searching such views and adding this information to these views.

CHAPTER VI

CONCLUSION

A model-based 3-D object recognition system has been developed and tested. The system consists of preprocessing, feature extraction, modeling and matching stages. For each stage, some algorithms are implemented and tested.

Preprocessing covers smoothing and edge detection operations. Neighborhood averaging scheme has been used for smoothing. Edges were detected using Sobel operator. Standard Sobel kernels were divided into two simpler kernels to increase the speed of the operation. Sobel operator produces an edge enhanced image. A threshold value is found by trial-and-error to extract edge pixels from the edge enhanced image. This value was suitable for the test images that are taken using similar lighting conditions. If the lighting changes, it may be necessary to change the threshold value. To eliminate this problem, an adaptive thresholding technique may be applied.

For the polyhedral object recognition, most natural features are the line segments forming the object boundaries. Three line segment extraction algorithms have been implemented. The performance of the first method [12] that uses corner points is very bad. Corner detection algorithm does not work properly. It detects a lot of false corners while missing some real ones. Also, some of the detected real corners are misplaced. Since the line extraction algorithm uses the detected corner points, this algorithm misses some real lines and detects many false lines.

Second line extraction method was the classical Hough transformation method. It gave better result than the first one, but there are also some problems related to this method. First of all, the threshold selection to find the local maxima of the accumulator array bins is very difficult. If it is chosen too high, some lines are missed. If the threshold value is low, a lot of lines are detected. Secondly, line segment endpoints can not be found using this method. Only the parameters of a line with infinite length can be computed. Thirdly, some collinear edge points that belong to separate lines may lead to the detection of false lines. Some postprocessing algorithms can be applied to remove some of these disadvantages. But these algorithms may produce erroneous results also. Therefore, the classical Hough transform method is not suitable for the line segment extraction. The algorithm may be modified to eliminate the disadvantages.

The third line extraction algorithm was the rotation transformation method. There are several threshold values in this method, but the selection of them is not very critical. In fact, they give the algorithm robustness. The algorithm misplaces some of the edge segments because of the imperfect edge detection. Some of the line segment endpoints are misplaced by several pixels in the postprocessing stage. In spite of these two disadvantages, the method can be thought as satisfactory.

Finally, the geometric hashing method was used for object modeling and matching tasks. The experiments show that the method is successful in recognizing and finding the pose of an unknown object viewed from an unknown viewpoint. The disadvantage of this method is the size of the hash table. This makes the use of a single hash table for all the model objects impossible. Even when the number of feature points of an object is large, two separate hash tables with different views are needed. This increases the recognition time. When the unknown object is the same as the model, the recognition takes a couple of seconds. If they are not the same, the

rejection time is around a minute. These values do not include the feature extraction times. To decrease the rejection times, search for a hash table can be stopped after using a couple of basis pair. After the rejection, another model hash table must be used. This must be repeated until the object is recognized or all the hash tables are used. If none of the models is the same as the unknown object, the object is rejected. Therefore, the total rejection time is the summation of all the rejection times related to each hash table. If the object is recognized, the recognition time is the summation of all the rejection times related to each hash table checked and the recognition time when the hash table of the matching model is used. These values may be very large because of the usage of several hash tables instead of a single one. This problem can be eliminated by grouping the features. This decreases the number of features to be stored in the hash table. Another alternative can be the elimination of some views. Instead of using all the views defined by the tessellation of viewing sphere, some of them can be used. These views correspond to the stable positions of the objects.

The current implementation can be used in industrial operations involving the manipulation of a single part. A specified object can be chosen from a group of different objects. Also, the implementation can be helpful in the size measurements of the objects produced in a production line.

As a future research direction, occlusion case can be investigated. In this study, it was assumed that a single object is present in the scene. To handle occlusion, some modifications in the verification stage can be adequate if the hash table score threshold is low enough.

In this thesis, the recognition is limited to polyhedral objects. The method can be modified for the recognition of curved objects also. This can be achieved by changing the feature extraction and verification stages.

REFERENCES

[1] R. T. Chin and C. R. Dyer, "Model-Based Recognition in Robot Vision", Computing Surveys, 18(1), (1986).

[2] G. J. Gleason and G. J. Agin, "A Modular System for Sensor-Controlled Manipulation and Inspection", Proc. of the 9th International Symposium on Industrial Robots, Society of Manufacturing Engineers, (1979).

[3] G. C. Stockman et al., "Matching Images to Models for Registration and Object Detection via Clustering", IEEE Trans. Pattern Anal. Mach. Intell., 4(3), (1982).

[4] R. C. Bolles and R.A. Cain, "Recognizing and Locating Partially Visible Objects: The Local Feature Focus Method", Int. J. of Robotics Res., 1(3), 57, (1982).

[5] M. Oshima and Y. Shirai, "Object Recognition Using Three-Dimensional Information", IEEE Trans. Pattern Anal. Mach. Intell.,5(4), 353,(1983).

[6] R. C. Bolles et al., "3DPO: A Three-Dimensional Part Orientation System", In Robotics Research: The 1st International Symposium, edited by M. Brady and R. Paul, 413, (1984).

[7] Y. Lamdan and H. J. Wolfson, "Geometric Hashing: A General and Efficient Model-Based Recognition Scheme", 2nd International Conference on Computer Vision held in Florida, edited by R.Bajcsy and S. Ullman, 238, (1988).

[8] R. C. Gonzalez and P. Wintz, Digital Image Processing, Addison-Wesley, Massachusetts, (1987).

[9] M. M. Gupta and G. K. Knopf, "Theory of Edge Perception for Computer Vision Feedback Control", Journal of Intelligent and Robotics Systems, 2, 123, (1989).

[10] Y. Shirai, Three-Dimensional Computer Vision, Springer-Verlag, Berlin, (1987).

[11] K. H. Hedengren, "Decomposition of Edge Operators", 9th Int. Conf. on Pattern Recognition held in Rome, 2, 963, (1988).

[12] W. K. Gu and T. S. Huang, "Connected Line Drawing Extraction from a Perspective View of a Polyhedron", IEEE Trans. Pattern Anal. Machine Intell., 7, 422, (1985).

[13] E. R. Davies, Machine Vision: Theory, Algorithms, Practicalities, Academic Press, London, (1990).

[14] C. W. Kang et al., "Extraction of Straight Line Segments Using Rotation Transform: Generalized Hough Transformation", Pattern Recognition, 24(7), 633, (1991).

[15] Y. Lamdan et al., "Affine Invariant Model-Based Object Recognition", IEEE Trans. Robotics and Automation, 6(5), (1990).

[16] L. Ammeraal, Programming Principles in Computer Graphics, John Wiley and Sons, Great Britain, (1986).

[17] J. Q. Fang and T. S. Huang, "A Corner Finding Algorithm for Image Analysis and Registration", Amer. Ass. Artiff. Intell. Nat. Conf. held in Pittsburgh, (1982).

[18] H.Schildt, Advanced Turbo C, Osborne Mc Graw-Hill, (1987).

APPENDIX

APPENDIX A

DESCRIPTIONS OF THE COMPUTER PROGRAMS

In this thesis, computer programs have been written using C++ programming language. Programs have been designed to handle 256-gray-level images with the size of 256x256 pixels. Below, information on the file formats and the programs are given.

A.1 File Format for Image Files and Line Drawing Data

All the 256-gray-level images are byte-formatted, i.e. each pixel value is a byte long. The image files contain only these values. Pixel values are ordered row by row in the file. No header file is used. The size of these files is 65536 bytes.

Line drawings consist of corner points linked with lines. In the programs, following file format is used: First, some characters are written to identify the file. Then, the number of corners and coordinates of the corners are recorded. Finally, a matrix representing the corner connections comes. (i,j)th entry of the matrix is one if the corner i is connected to corner j. It is zero otherwise.

APPENDIX A

A.2 Edge Detection Using Image Histogram

The Program "Edge_percept":

This program finds image edges using image histogram and some mapping functions. Name of the input image file is given by the user. The resultant edge image is displayed on the screen.

A.3 Line Extraction Using Corner Points

The Program "Line_cor.C":

This program extracts lines in an image using corner points. It uses a data file called "corner.dat" to obtain the line directions. When the program is executed, the name of the image file to be processed is asked. Then, smoothing, edge detection, corner detection, line direction finding and line finding operations are performed successively. For each operation (except smoothing), a threshold value is asked and the resultant image is shown. Since the threshold value is very large, one hundredth of this value must be entered.

A.4 Line Extraction Using Hough Transformation

The Program "Hough.C":

This program uses Hough transformation method to extract lines.The program asks the name of the image file to be handled first. Then, it finds the edge

image. Next, the Hough transformation is applied to the edge image. The threshold value for the accumulator array bins is asked from the user. Extracted lines are displayed on the screen.

A.5 Line Extraction Using Rotation Transformation

The Program "Rott.C":

This program uses rotation transformation for line segment extraction. When the program is executed, name of the image file to be handled is asked. After the edge detection, function "RT" is called. Parameters of the rotation transformation are in the declaration part of the function RT. Any change in the value of these parameters can be done by finding the declaration part of RT. Parameters are as follows:

l= the number of quantization levels of θ

e= quantization interval of ρ

ml= minimum line length

sg= separation gap.

When the final line drawing is found, it is shown on the screen. Also, the line drawing in formation is recorded into the file whose name is given by the user.

APPENDIX A

A.6 Hidden-Line Elimination

The Program "Hidlin.C":

This program finds the 2-D projections of a 3-D object when viewed from points found by the tessellation of the viewing sphere using square patches. This is done by first imposing a regular grid on the faces of a cube. Then the grid elements transformed radially onto the viewing sphere.

The input of the program is a data file that carries the 3-D information of the object. The program is executed by entering the command "hidlin <name of the data file>" at the DOS prompt. The format of the data file is as follows: On the first line, coordinates of the center of mass of the object is given. Then the number of each vertex and the 3-D location are given (one vertex in one line). Following this, the word "Faces" is written and the object faces are coded using vertex numbers. Each face is a polygon. Vertex numbers are listed counterclockwise when the polygon is viewed from the outside of the object. Each polygon definition is ended by the character "#".

Output file contains the line drawing information for each view as described previously. Name of this file is asked from the user.

APPENDIX A

A.7 Preprocessing for Object Recognition by Geometric Hashing

The Program "Modeller":

This program creates hash table for the given object. When it is executed, name of the file that contains the 2-D views of the model object is asked. This file must be created by hidden line elimination program previously. After reading the file name, the program constructs the hash table. Finally, the name of the hash table file is asked and the table is stored into that file. Quantization interval of the table which is called as "delta" is declared in the parameter declaration part of the program. It must be the same as that of the program "matcher".

A.8 Recognition by Geometric Hashing

The Program "Matcher:"

This program finds the best matching model to the line drawing of the unknown object. When a matching model is found, it is shown on the screen after aligning the model against image.

When the program is executed, the number of views, name of the hash table, the name of the file to be recognized and the name of the file that contains the line drawings of the model object is asked. The file to be recognized must be created by the program "Rott.C" previously using the image file of the unknown object. The

APPENDIX A

file which contains the line drawings of the model object must be created by the program "Hidlin.C" before the recognition. The number of views must be the same as the number of views contained in the hash table. Verification score thresholds can be adjusted by changing the values of the variables "t1", "t2" and "t3" in the parameter declaration part of the program. These are multiplication factors related to the hash table vote, corner verification score and line verification score, respectively.