NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

Canada

Recognition of Overlapping Objects

by

Tayeb Damerji, B.A.Sc.

A thesis submitted to the
School of Graduate Studies and Research
in partial fulfilment of the requirements
for the degree

Master of Applied Science

Ottawa-Carleton Institute for Electrical Engineering
Department of Electrical Engineering
Faculty of Engineering
University of Ottawa
January 1994

ISBN 0-612-00525-9

Canada

UNIVERSITÉ D'OTTAWA
UNIVERSITY OF OTTAWA

To my parents

for their perpetual love and support

Abstract

This thesis describes the design and implementation of a model-based vision system for the recognition of partially occluded objects. The system can detect occlusion. It can also recognize and locate the objects forming the occlusion scene.

The system has two modes of operation: a learning mode and a recognition mode. In the learning mode the objects to be recognized are analyzed, their boundaries processed and a model representing each object is created and added to a list of object models. In the recognition mode the system analyzes the image which may contain several objects, the objects may occlude each other. The system processes the image, detects whether there is occlusion and then identifies and locates the objects present in the image.

The objects' models are based on the parameters of an auto-regressive (AR) filter. The parameters of the AR filter are obtained using the radial distances between the median point of the sub-part's[1] baseline and equi-spaced points on the sub-part's boundary. Each sub-part of the object is represented by a vector of AR filter parameters.

In the recognition mode the system first generates the scene sub-parts, then it goes through the three phases of the recognition: namely hypotheses generation, pruning and verification. It identifies the objects contained in the list of object models that have similar sub-parts and AR parameters vectors. Then, it eliminates the objects that have a small or no chance of being in the scene. Finally it checks the presence of the remaining hypothesized objects through affine transformations and

[1]Sub-parts of an object are sections of the boundary contained between two significant concavities. They are generated by dividing the boundary based on the location of the most concave points and other rules.

measurement of the distance between the objects' segments and the corresponding scene segments. The recognition process ends when all of the sub-parts in the scene are mapped to objects from the list of object models or when all the hypotheses have been checked.

Acknowledgements

The author wishes to express his sincere appreciation and gratitude to his professor, Dr. Dan Ionescu, for his continued support, guidance and understanding.

Financial assistance by the scientific mission of Tunisia and the University of Ottawa was greatly appreciated.

Finally, the author thanks his parents, his brother and his sisters for their love and constant support.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Introduction

Vision is our most powerful sense. In practice, it conveys useful information over many orders of magnitude changes in intensity of radiation, about scenes illuminated by a variety of light sources and populated by a broad range of surface types. The reflectance of most surfaces has a matte component so that the irradiance received by a camera is insensitive to slight shifts in the relative position of a light source, camera, and viewed surface. Other wavelengths, particularly sonar are less widely applicable. Vision is passive, in the sense that it does not require energy to be radiated to measure scene irradiance. This is important for many industrial and military applications. Finally vision is familiar to us, enabling the results of visual processing to be precisely specified in advance and to be easily and consistently interpreted.

For all these reasons, developments in visual processing have been exploited read-

ily throughout the past years by industry, the military, and for the office and laboratory. Military applications include target identification and tracking, passive navigation of a pilot-less vehicle, reconnaissance and sentry duty, and mapping. Industrial applications of vision have included inspection, part acquisition, visually guided processing, and warehouse sentry duty. Applications of vision in the office and laboratory are mostly concerned with automatic entry to a computer of documents or diagrams.

The number of systems and applications developed in these fields remains limited since many problems associated with computer vision have not been solved completely or adequately yet. Already some projects like the hand-held computers that perform some limited hand-writing recognition are proving the power and usefulness of computer vision.

1.2 Scope of the Research

This thesis addresses the series of problems encountered in developing and designing a vision system, specifically the problem of recognizing industrial objects in the presence of occlusion. The vision system has been implemented on an SIMD parallel processor, the AIS 5000. Many problems have been encountered in trying to optimize the feature extraction and understanding algorithms. The problem that will form the main body of this thesis is the detection and recognition of objects in the presence of occlusion.

To solve this problem in an efficient way, we explored a new scheme for detecting and recognizing objects in the presence of occlusion. Unlike previous efforts that tried to optimize only one aspect of the recognition process, in this work we have

tried to look at all the stages of the problem. Attempts have been made to obtain the best and most efficient boundary detection technique for this particular problem. We have also designed an efficient and extensible represenation of the boundary features. An algorithm for dividing the boundary into sub-parts has been implemented, an object modeling technique that relies on the auto-regressive filter parameters to represent the boundary is used for model storage and retrieval. The Auto-regressive filter was chosen since its parameters are invariant under rotation, translation and certain scaling factors. Finally, the recognition phase has been enhanced by relying on data driven hypotheses.

Since all the processing and recognition is based on local shape parameters and constraints apllied to these parameters, the algorithms used in the present system can be readily extended to other problems like hand-writing recognition, or 3-D vision. In the case of 3-D vision; objects are recognized without storing three-dimensional object models. Instead, objects are recognized by using combinations of two-dimensional views, since any view of a three-dimensional object can be approximated by the linear combination of a small number of its views.

Figure 4.1 illustrates the main tasks performed during the learning phase and Figure 4.2 shows the main stages the system goes through during the recognition phase.

The following section explains the approach followed in the implementation of the vision system. It also gives an overview of the algorithms and techniques used to improve and enhance the recognition engine.

1.3 Approach

The vision system starts by acquiring a gray level image of an object or a set of objects, the objects can occlude each other. The first processing step would be to extract a binary picture: i.e., to transform the 256 levels of gray into 2 levels of gray (0 and 1). This allows the isolation of the object from its background. Two methods have been set up to deal with this problem.

1. The first consists of applying a fixed threshold to the input image and to set all pixels having an intensity less than the threshold to 1 and those having a value higher than the threshold to 0. This assumes that the object is dark against a light background. If the background is darker then the assigned values will be the opposites.

2. The second method called adaptive threshold uses the same principle but before changing the pixels intensity values a threshold is computed by extracting a histogram[1] of the image. Then from the histogram we extract the intensity value that best divides the histogram into two poles representing the two intensity levels. Once the threshold value is extracted, we apply the procedure of the first method. This method can be made more robust by subtracting the background.

Once we obtain the silhouette, we extract the edge of the scene through morphological operators and then the chain code. In the chain code representation each pixel on the boundary of the scene will be encoded according to the direction of the boundary at that specific point. The values range from 1 to 8 and represent 8 different directions.

[1]The image histogram will be explained more fully in the next chapter.

We then extract the polygonal approximation of the scene's boundary. The polygonal approximation has the important property of representing the object with a minimum number of boundary points. It is obtained in a number of alternate ways. Among these we will describe the split and merge algorithm[34] and the scan-along algorithm.

The reduced set of pixels is then analyzed and the boundary is sub-divided into smaller components called sub-parts. Sub-parts are portions of the boundary separated by the most concave points.

We then estimate the auto-regressive (AR) coefficients of the polygonal approximations of the object's sub-parts. An *auto-regressive model* is a parametric equation that expresses each sample of an ordered set of data samples as a linear combination of previous samples from the set plus an error term. The model has been widely used in speech recognition, speech synthesis, data compression, and spectral modeling. The AR parameters will serve as descriptors of the object. It was selected for its simplicity, limited storage requirement, and the invariance of the parameters to scaling, certain degrees of rotation, and translation of the object. The invariance property simplifies of the hypothesis generation and verification operation. This simplification stems from the fact that we do not need to check every pose[2] of each stored model.

To improve the performance of the system and make dealing with occlusion simpler, rules to help detect occlusion and specify the area affected by it are applied to the image. The rules are mainly based on the following facts:

- occlusion creates concavities in the scene that are not present in the objects

[2]The pose of an object model is the transformation needed to map it from its own inherent coordinate system into agreement with scene data.

- the number of branches emanating from the centroid of the scene would usually be higher than four
- the area and perimeter of the scene would increase proportionately to the number and characteristics of the objects forming the scene.

The results of the rules are assigned weights and combined linearly into one occlusion confidence value expressing the likelihood of occlusion. The probable region of occlusion, i.e., the intersection between the objects forming the scene can be deduced from the relative location of the most concave vertices.

After the occlusion detection phase, the image analysis phase starts by matching model features representing known sub-parts to sub-parts from the scene. Models representing known sub-parts are matched to the image according to a predetermined order, and some sub-parts from the scene are coupled if they satisfy certain geometric constraints. The sub-parts whose model features are close are assumed to be present and their corresponding objects are added to the list of object models to be verified.

The verification operation starts with the object that has the highest likelihood of being present and checks whether a sufficient number of its sub-parts exist in the scene. If so, its presence is confirmed and its corresponding sub-parts are removed from the scene. The verification resumes with the next hypothesized object until all the objects forming the scene are identified.

The general approach for the recognition of overlapped objects can be summarized in the following steps:

1. Image Acquisition: Acquire a 2-D grey level image using an RS-170 CCD camera. The image is stored on an image frame in the memory of the parallel processor.

2. Preprocessing: The two-dimensional image is digitized, filtered, thresholded, and stored as a 512x512 binary image on the memory of the parallel processor. Three different methods can be used in this step:

   (a) Simple thresholding with a fixed threshold.

   (b) Adaptive thresholding with a variable threshold value computed through the histogram of the gray level image.

   (c) Application of a detection algorithm: A difference of gaussian operator has been found to offer the best results.

3. Aspect ratio correction: correct the aspect ratio of the image by removing one column in each five-column band of the image. This step is optional and aspect ratio correction can be performed later by multiplying the y coordinates by the value of the aspect ratio[3] in the y direction.

4. Contour Extraction and chain code derivation: Apply morphological operations (erode, dilate, etc.) to extract one pixel wide boundary of the object and other operations (shift, or, and, etc.) to extract the chain code through a parallel algorithm. The detected boundary is represented by an array of points and an array of chain code values.

5. Polygonal Representation: Get the equivalent polygonal representation of the boundary. The chain code values are used to extract the error norms used to

[3]The aspect ratio specifies the ratio of pixel y spacing to pixel x spacing. An RS-170 CCD camera has a total aspect ratio of 3/4. On the AIS-5000, in the configuration used for this system, 384 pixels are extracted in the x direction by 244 lines in the y direction, so the aspect ratio is $= (3/244)/(4/384) = 1.1803$.

decide which points to select.

6. Decomposition: Get the angular characteristics of the vertices in the polygonal representation and divide it at concave vertices to produce the sub-parts. The decomposition method is especially geared to deal with occlusion and differs from previous methods for object decomposition that were mainly devised for syntactic pattern recognition.

7. Modeling: Calculate the AR parameters for each generated sub-part and for the total object boundary. store the AR parameters and the polygonal representation of the subparts.

   This is the last operation in the learning mode. In the recognition mode we add the following steps.

8. Hypotheses Generation: Compare the AR parameters of the scene sub-parts with the AR parameters of different models representing sub-parts of known objects. The result of this step is a list of hypothesized objects.

9. Pruning: Eliminate from the list the objects that do not satisfy the constraints extracted from the scene. The constraints are based on area and perimeter comparisons and on the number of concave vertices.

10. Verification: Confirm or reject the presence of a given hypothesis using a parallel template matching. In this step a parallel algorithm is used to test the validity of the match through a measurement of the common area between the two sub-parts. The matching tries to find enough of the object sub-parts in the scene before confirming that the object is present in the scene.

1.4 Conclusion

This chapter gave a brief overview of the work described in this thesis. The next chapter will present a review of known vision systems that tried to deal with the problem of occlusion detection and the recognition of occluded objects. Chapter 3 will outline the algorithms and methods used for the processing of the image, its filtering and the modeling of the objects. Chapter 4 deals with algorithms used in the recognition and verification of known objects in arbitrary scenes. In chapter 5 the implementation approach is explained and the implementation of some algorithms is presented. The results of the systems are presented and analyzed in chapter 6. Finally, the conclusion presents some lessons learned in developing the vision system on a parallel machine and the direction of future work.

Chapter 2

Review of the Literature

2.1 Introduction

In this chapter we review some approaches developed to deal with the problem of object recognition in the presence of occlusion.

Some computer vision systems have been devised to deal with this problem. The recognition techniques used in these systems were model- based. That is object models were used to guide the recognition process[1]. Almost all the vision systems that address occlusion use local features since global features are not invariant under the presence of occlusion. Local features used are: longest segments of polygon approximation [4, 29], holes, convex and concave corners [10], sub-templates in $\theta - s$ space [45], fixed length boundary segment [26], local lines and circular edge fragments [21]. Hypotheses are generated either by a Hough transform [45, 20] or by distance

[1]Models are descriptions of the objects, they can be based either on global features such as area, perimeter, centroid, etc., or on local features such as local boundary characteristics.

measures defined for feature pairs.

The matching approaches vary according to the type of features used, statistical pattern classification techniques are usually employed with global-feature based systems. Local-feature based systems generally use the hypothesize-verify paradigm and recognize the scene elements in two phases: in the first phase hypotheses are generated, in the second phase the hypotheses are tested and their validity is established. Matching can be conducted in several ways including: least squares distance between sequence of points, relaxation [8], search in interpretation space [20], sophisticated indexing of model features [29], growing clusters around matched privileged features [4, 10]. In the next section we will review some characteristics of these vision systems.

Bolles and Cain [10] used local features (holes, corners, and clusters of adjacent features) to create the object models. The cluster of features is obtained by locating the most significant feature in the image and "growing" it by adding neighboring features.

The matching process involves creating an association graph. Each node of the graph corresponds to a match (matches represent scene features similar to model features). An arc connects 2 structurally compatible matches. Hypotheses are generated by finding the largest connected sub-graph (maximal clique). Hypotheses are tested by checking for object boundary consistency.

The technique is not completely general since it assumes that model objects have sharp local features close to each other, and a cluster of local features must be visible for the part to be recognized. Furthermore, the method relies on algorithms for solving the NP-complete maximal clique problem, so their performance is likely to degrade significantly as the number of features increases[10].

Ayache and Faugeras [4] developed a recognition system called HYPER (Hypothesis Predicted and Evaluated Recursively). In this system objects are modeled by their polygonal approximations, the angles between successive sides, and salient sides. Hypotheses are generated by assigning salient sides of the models to compatible segments of the scene. The best hypotheses are verified by iteratively matching remaining model segments and updating the transformation matrices until a high quality measure of similarity is reached or enough hypotheses have been evaluated. Estimation of the affine transformation that maps some models onto the corresponding scene features is made recursively through the application of a Kalman filter.

The efficiency of this method depends on the correct choice of privileged sides and the visibility of the corresponding privileged sides in the scene.

Grimson and Lozano-Pérez[20] can locate an object in a cluttered scene from sparse position and orientation data measurements. The objects are modeled by local measurements of 3-D positions and surface normals. Models consist of polyhedral objects represented by their planar faces. The information about these faces (such as equations) and the relations between faces (such as distance) are also computed. Sparse range or tactile data of 3-D objects are used as scene features.

The matching process contains two steps; first an *interpretation* tree is generated by associating sides of the model with sensed points. Each path of the tree corresponds to an interpretation that is consistent with local geometric constraints. Interpretations inconsistent with local constraints are verified by a transformation test. An interpretation is accepted if it can be used to solve for a transformation that would place each sensed point on an object surface. After fast selection of an interpretation, a final test determines object position and validates the interpretation. To handle extraneous data points in a cluttered scene, sensed points can be ignored if no remaining side of a model is consistent with the interpretation and if

enough points have been interpreted. To limit the size of the combinatorial search only the complete interpretations in the tree are kept. This sometimes leads to identification errors.

Turney, Mudge, and Volz [45] use boundary curves for matching. They divide a model, or *template*, of length n pixels into $n/2$ sub-templates, each of length h . Each sub-template of all models is compared against the entire image boundary. To improve the system's efficiency, the matching of a sub-template against the boundary is performed by a least-squares fit in $\theta - s$ space, where θ represents tangent slope and s represents the arc-length along the curve. This method works well for overlapping objects, but it is slow since the number of sub-templates is large.

Grosky and Mehrota [29] used a feature index organized as a search tree. In the recognition phase, a feature from the analyzed portion of the scene data is taken as a test feature. The feature index is searched to find the best matching model feature. The model/location list associated with the model feature is used to hypothesize the identities and location in the image of the possible objects. The hypotheses are checked to decide their validity. The hypotheses verification is done by finding matching points in the scene data for selected points on the transformed boundary of a hypothesized object. The selected points belong to privileged strings. Each privileged string starts at a sharp vertex. The sharpness of vertices can change and result in the selection of incorrect privileged strings and therefore low accuracy of recognition.

Kalvin et al. [24] generate footprints of boundary subsections. A footprint of an object is an n-dimensional curve derived from the boundary of that object. This curve is dependent on a starting point on the boundary. The footprint of a subsection is hashed to select a set of candidate models having similar footprints. Each of the candidate models is matched against the boundary segment under consideration

to select the best candidate and to decide its location. The process of footprint generation is computationally complex and the number of candidate models found by hashing is usually large in case of noisy and complex scenes.

Koch and Ragasam [27] use special vertices of the polygon to detect and locate the model in the scene. Polygon moment calculation lead to estimates of the dissimilarity between the scene and model corners, and determine the model corner location in the scene.

Bhanu and Faugeras [8] use the segment lengths, slope of the segment, the angle between two segments, and the inter-vertex distance as features. A shape measure indicates the goodness of a match between a model and a scene segment. A stochastic labeling scheme is then used to label each of the model segments as either a scene segment or NIL (no match). This method is computationally intensive and requires a good estimate of the initial assignment of the labels for the convergence of the algorithm and validity.

Price [35] developed a simple approach to the occlusion problem. He used the line segments and the polygon approximation as features. In the matching process, every model segment is compared to every scene segment. If the inter-segment angle and the length are within certain thresholds, the model segment is said to be compatible with the model segment, and their orientation difference is stored in an array known as a disparity array. Since segments are of an object are arranged sequentially along the object contour, segments between the model and the scene are likely to be matched in sequence. The longest consecutive sequence of matched segments between the model and the scene corresponds to the longest compatible consecutive diagonal entries of the disparity array that have similar orientation differences. A transformation that aligns the model segments with the matched scene segments is evaluated. Applying this transformation to the model segments, disparity values

based on the segment positions and orientations are updated and stored in the disparity array. The final matches between the model and the scene segments are decided by finding the longest compatible consecutive diagonal entries of the new disparity array.

Price's procedure is simple, but not computationally efficient. The technique is also sensitive to scale variation since the features used are scale dependent.

Bhanu and Ming [9] suggest an approach similar to Price's but using a different matching process. The matching process first applies the *K-mean* clustering algorithm iteratively on the disparity array until the optimal number of clusters is found. It then checks for the elements of each cluster that are in sequential order, and finds the sequences. Several heuristics are included to determine the sequences. The process then clusters the sequences' averages using the same clustering algorithm. The cluster that contains the largest number of sequences decides the final matches between the model and the scene segments. A confidence value that is the ratio of the cumulative length of the segments in the final match to the local length of the model segments is evaluated to verify the final match. This approach is computationally expensive due to the iterative nature of the matching algorithm.

2.2 Conclusion

This section has presented some of the systems that have been devised to deal with detection and recognition of objects in the presence of occlusion. We have only reviewed a representative sample since this problem is very popular and some of the other solutions are not very different from those reviewed here. The timing characteristics have not been listed since they can be sometimes misleading, the computers

used vary from small PCs to powerful vector and massively parallel computers. so no conclusive comparisons can be drawn between the various performance parameters listed.

The system reviewed in this thesis varies from the previous systems in that it has been implemented and designed for a parallel machine. It also uses a combination and structural and statistical pattern recognition methods based on the AR parameters of the object sub-parts.

Chapter 3

Image Analysis, and Modeling for Occluded Object Identification

This chapter deals with the techniques used for image analysis, boundary description, and object modeling. These techniques enable the extraction of meaningful information from the 256 grey level representation obtained through the CCD camera. In this stage the following operations are performed on the image:

- filtering and thresholding,
- extraction of the boundary and derivation of the chain code and the polygonal representation,
- the decomposition of the polygonal representation into smaller components called sub-parts,
- and finally the calculation of the auto-regressive parameters.

In the next section we will describe these operation in more detail and elaborate on the specific techniques and algorithms that were used.

3.1 Filtering and Thresholding

A 2-D grey level image is acquired using an RS-170 CCD camera. The CCD camera output consists of a grey scale image with 256 levels of grey. This image is saved in memory, then it is processed to extract a binary array and to isolate the main object or group of objects present. This operation is called *thresholding*. It would result in the separation between the pixels that compose the object from those that belong to the background. If the object is darker than the background, image pixels with intensity values lower than the given threshold are assigned the value "1" and pixels with intensity values higher than the threshold are assigned "0".

To deal with variations in lighting and image granularity an adaptive thresholding algorithm was used. Adaptive thresholding operates with a threshold value that is extracted from the image to be operated upon. The threshold value is extracted by computing a histogram for the image, then from the histogram we extract the intensity value that best divides the histogram into two poles representing the two intensity levels. Once the threshold value is extracted, we apply the procedure of the first method.

3.2 Contour Extraction

Once the binary picture is obtained, we need to extract the contour or boundary of the object or objects that compose the image. The contour is obtained by shrinking the image through an erosion operation and then subtracting the result from the original image. Once the boundary points are extracted, we need to represent it in a format that would be concise and computationally efficient. One such format is the chain code that encodes the boundary pixels as a chain of vectors. The following section will give details on the merits of this technique and its characteristics.

3.3 Chain Code Derivation

Chain codes are used to represent a boundary as a set of straight line segments of specified length and direction. To generate the chain code of a given boundary, we subdivide the boundary into vectors of equal length. The vectors have a direction value from the allowed chain-code directions. Figure 3.1 shows the possible directions of the chain code representation.

The advantage in using the chain code lies in the possibility of obtaining many descriptors of the shape through integer arithmetic. In fact, a number of parameters can be derived directly from the chain code. They include area, perimeter, shortest distance between two vertices in chain code links, and the derivative of the chain code. The derivative of the chain code is useful because it is invariant under boundary rotation. The derivative, defined as a first difference *modulo 8*, is simply another

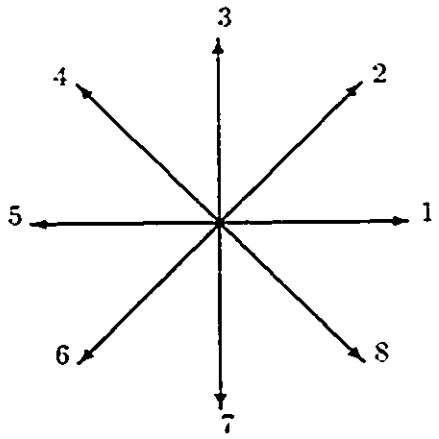

Figure 3.1: The Chain code representation.

sequence of numbers indicating the relative direction of chain code segments.

The chain code is obtained in a process that involves three steps:

- Parallel extraction of the chain code: in this step the boundary of the image undergoes a series of neighborhood operations to isolate the pixels according to their directions. Each pixel direction (north, northeast, east, etc) is isolated by shifting the image frame in the corresponding direction and then subtracting the shifted image from the original image. The values are encoded according to Table 3.1. Table 3.2 lists the logical operations that determine the bit values of different bit planes in the *boundary* image. Each boundary image consists of four bit planes. The value of the 4 bit planes at location (x, y) encodes the direction (from 1 to 8) of a given pixel at that location.

- Boundary following according to the retrieved directions: we start with an arbitrary pixel, usually a pixel with the smallest x coordinate then we extract the pixel value from the boundary plane. This value will determine the x and y coordinate to be applied the tracing coordinates. Table 3.1 list the increments

Direction	*Dir. value*	*X Incr.*	*Y Incr.*
North	1	1	0
North East	2	1	-1
East	3	0	-1
South East	4	-1	-1
South	5	-1	0
South West	6	-1	1
West	7	0	1
North West	8	1	1

Table 3.1: Chain code values and increments for different directions.

Bit plane	N	NE	E	SE	S	SW	W	NW	Logical Operation
0	1	0	1	0	1	0	1	0	E or S or W or N
1	0	1	1	0	0	1	1	0	NE or E or W or SW
2	0	0	0	1	1	1	1	0	SE or W or S or SW
4	0	0	0	0	0	0	0	1	NW

Table 3.2: Logical Operations to determine the chain code values.

Direction	*X Incr.*	*Y Incr.*
North	-1	0
North East	-1	1
East	0	1
South East	1	1
South	1	0
South West	1	-1
West	0	-1
North West	-1	-1

Table 3.3: Chain code increments for back tracing.

applied to x and y coordinates for the different chain code values.

- Error correction and handling of special cases: in some cases boundary quantization and other noise sources cause the loss of continuity in the boundary tracing process. In this case, we look around the neighborhood of the last non-zero pixel for non-detected pixels that have a chain code value different from zero. When this pixel is found we resume the boundary tracing. If we are unable to find any non-zero pixel in the immediate neighborhood of the current pixel, we go back to the first pixel selected and try to carry the boundary tracing in the reverse direction, in this case the x and y coordinate increments are different from those used in the first step; they are given in Table 3.3.

The contour tracing is terminated when the current pixel reaches the initial one.

Two object characteristics that can be derived from the chain code and that proved useful in subsequent operations of the vision system are the area deviation between two pixels and the segment length. The two measures will be used in the

selection of the dominant points and the derivation of the polygonal representation. The following sections will explain the concepts of area deviation and segment length and their derivation from the chain code values.

3.3.1 Area Deviation

The area deviation[46] of a contour between two boundary points is defined as the surface that separates the segment (P_0, P_i) from the portion of the contour between P_0 and P_i (see Figure 3.3.1). The variation of the area deviation is easy to compute when one of the endpoints is shifted to its neighbor.

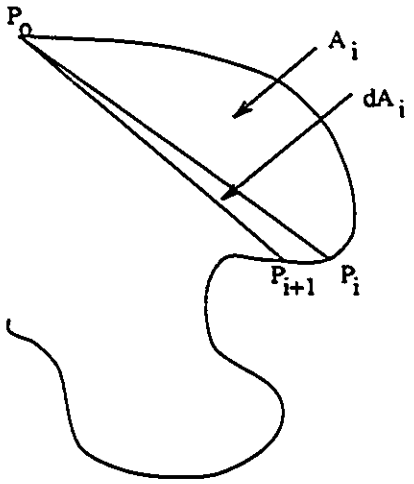

Figure 3.2: The derivation of area deviation.

Let A_i be the area deviation between P_0 and P_i, and A_{i+1} the area deviation between P_0 and P_{i+1}, P_i and P_{i+1} being two neighbors. A_{i+1} can be expressed as:

$$A_{i+1} = A_i + dA_i$$

From Figure 3.3.1 we can see that dA_i corresponds to that of the triangle (P_0, P_i, P_{i+1}). From Figure 3.3.1 we have

$$dA_i = \frac{1}{2} \mid \overrightarrow{P_0 P_{i+1}} \times \overrightarrow{P_i H} \mid$$

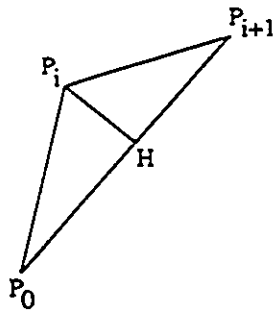

Figure 3.3: Area deviation between two pixels.

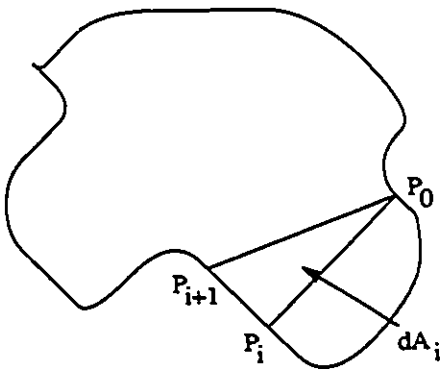

Figure 3.4: Area computation.

where $\times$ is the vector product operator and (x_i, y_i) and (x_{i+1}, y_{i+1}) are respectively the coordinates of $\overrightarrow{P_0 P_i}$ and $\overrightarrow{P_0 P_{i+1}}$. This leads to:

$$dA_i = \frac{1}{2}(dx_i . y_i - dy_i . x_i)$$

where

$$dx_i = x_{i+1} - x_i$$

$$dy_i = y_{i+1} - y_i$$

Direction	dx_i	dy_i	dA_i
1	1	0	$-y_i$
2	1	-1	$-y_i - x_i$
3	0	-1	$-x_i$
4	-1	-1	$-x_i + y_i$
5	-1	0	y_i
6	-1	1	$x_i + y_i$
7	0	1	x_i
8	1	1	$x_i - y_i$

Table 3.4: Area deviation for different chain code values.

Since P_i and P_{i+1} are neighbors on the boundary, dx_i and dy_i can only take the values $0, -1$, and 1. The area deviation can then be computed using only subtractions and additions of integer numbers (x_i and y_i). The area deviation dA_i is mapped directly as a function of all chain code values in Table 3.4

The Area deviation between two boundary points P_m and P_n can be computed using the:

$$A_{m,n} = \sum_{i=m}^{n} dA_i$$

3.3.2 Segment Length

Segment Lengths can also be computed in the same way as area deviation[46]. Let L_i and L_{i+1} be the length of the segment joining the points P_0 and P_{i+1}, and P_i and P_{i+1} respectively. We have:

$$L_{i+1}^2 = \| \overrightarrow{P_0 P_{i+1}} \| = x_{i+1}^2 + y_{i+1}^2$$

$$L_i^2 = \| \overrightarrow{P_0 P_i} \| = x_i^2 + y_i^2$$

Direction	dx_i	dy_i	dL_i
1	1	0	$1+2x_i$
2	1	-1	$2+2(x_i-y_i)$
3	0	-1	$1-2y_i$
4	-1	-1	$2-2(x_i+y_i)$
5	-1	0	$1-2x_i$
6	-1	1	$2-2(x_i-y_i)$
7	0	1	$1+2y_i$
8	1	1	$2+2(x_i+y_i)$

Table 3.5: Segment Length deviation for different chain code values.

We want to calculate L_{i+1}^2 as a function of L_i^2. Let us assume that:

$$L_{i+1}^2 = L_i^2 + dL_i^2$$

since:

$$dx_i = x_{i+1} - x_i$$

$$dy_i = y_{i+1} - y_i$$

we can formulate dL_i^2 as:

$$L_{i+1}^2 = (x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2$$

$$L_{i+1}^2 = x_i^2 + y_i^2 + 2x_i dx_i + 2y_i.dy_i + dx_i^2 + dy_i^2$$

$$L_{i+1}^2 = L_i^2 + 2x_i dx_i + 2y_i.dy_i + dx_i^2 + dy_i^2$$

Hence,

$$dL_i^2 = 2x_i dx_i + 2y_i dy_i + dx_i^2 + dy_i^2$$

Table 3.5 list the values of segment length difference for different chain code values.

Once we obtain the chain code representation, we will need to prune this representation even further in order to generate a more concise representation of the boundary. The polygonal representation provides a mechanism of selecting the points that occur at significant boundary curvature changes. It has been chosen as an intermediate representation that will be used in subsequent processing stages.

The following section describes the polygonal representation and how the measures derived from the chain code serve in the selection of the significant points.

3.4 Polygonal Approximations

A digital boundary can be approximated with arbitrary accuracy by a polygon. For a closed curve, the approximation is exact when the number of segments in the polygon is equal to the number of points in the boundary so that each pair of adjacent points defines a segment in the polygon. In practice, the goal of polygonal approximation is to capture the "essence" of the boundary shape with the fewest possible polygonal segments[34].

A common function shared by most polygonal approximation algorithms is the collinearity test that checks if points along a boundary portion are collinear with respect to a straight line. Collinearity is usually determined by the maximum perpendicular distance from a point of the boundary portion to the straight line. Consider a boundary portion between two points, A and C (see Figure 3.4) , we compute the maximum perpendicular distance from the boundary portion to the straight line AC. If the distance is within tolerance, that boundary portion is approximated by the straight line segment AC. Otherwise, the point along the boundary portion that yields the maximum distance becomes a new break point, say B, and the boundary

portion is approximated by two line segments AB and BC. The algorithm used to extract polygonal approximations is the split-and-merge algorithm.

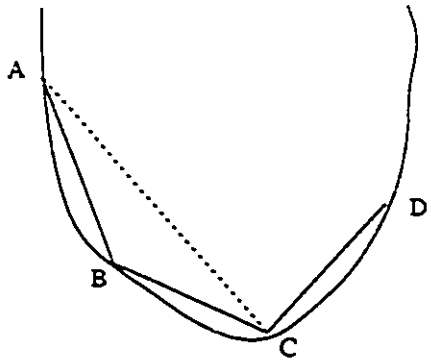

Figure 3.5: Split and merge concept.

Traditional implementations of the split and merge algorithm are time consuming due to the following factors:

- If the initial position is chosen arbitrarily the number of splits and merges will be large.
- The error norms used, either the maximum Euclidean distance or the integral square error, require extensive computation time.

In order to remedy the first problem, a rough, but very fast scan-along polygonal approximation [46] precedes the split and merge algorithm. Vertices resulting from this scan along procedure serve as initial partition for the split and merge algorithm.

The second drawback is addressed by using area deviation per length as error norm. This norm originally used for the scan along technique allows the split and merge algorithm to have a better execution time.

A brief outline of the resulting split-and-merge algorithm is given below:

1. Assign an arbitrary number of points along the boundary as the initial set of break points. The initial approximated polygon is formed by joining the sequence of break points along the original boundary with straight lines.

2. For each pair of adjacent break points, determine the points along the boundary portion that yield the maximum perpendicular distance to the straight line segment joined by the two break points. If the maximum perpendicular distance is greater than a given tolerance, that point becomes a new break point; i.e., the line segment is replaced by two line segments. This is the splitting part of the algorithm.

3. For each three consecutive adjacent points, say A, B, and C, compute the maximum perpendicular distance from the boundary portion between A and C to line AC. If the distance is within tolerance, break point B is removed. That is, line segments AB and BC are replaced by line segment AC. Note that each replacement is immediately tested for merging with the next line segment. This is the merging part of the algorithm.

4. Repeat Steps (2) and (3) until an equilibrium is reached; i.e., no more splitting and merging is necessary.

The number of points and their locations along the boundary varies if the boundary is at a different orientation or a different scale. To make the points obtained insensitive to the change of orientation and scale an extra processing step is needed. An approach to making the break points more stable is explained in the following section.

3.5 Gaussian Filtering

Due to the discrete boundary representation and quantization error, false local concavities and convexities along a boundary are introduced. Smoothing is thus necessary to reduce these false concavities and convexities. It has been shown that a Gaussian filter is an ideal smoothing filter for numerical differentiation [44]. We use the approach of smoothing the image frames through calls to the image filtering routines provided with the AIS software.

The new polygonal approximation algorithm can be summarized by the following procedure :

1. Remove all one-pixel wide protrusions which may result due to the discrete boundary representation and quantization error. This is a parallel operation performed on the image stored in parallel memory.
2. Smooth the boundary with the Gaussian filter. This is also a parallel operation.
3. Find the set of positive maximum and negative minimum curvature points along the Gaussian smoothed boundary.
4. The points along the original boundary that correspond to the set of points found in step (3) are used as the starting set of breakpoints for the polygonal approximation.
5. Use the split-and-merge polygonal approximation algorithm.

Two parameters, ω of the Gaussian filter and the tolerance for collinearity, must be set in using the algorithm. There is a trade-off when choosing the value of ω. A

large value will remove small details of the boundary curve, while a small value will permit false concavities and convexities. Since we only want to estimate a starting set of break points for the split-and-merge algorithm, a flexible range of values for ω is feasible. The tolerance is scale dependent: i.e., using the same tolerance for a boundary at a different scale may yield a different result. It is usually chosen by trial and error or based on a priori knowledge about the scale of the boundary.

3.6 Boundary Decomposition

Decomposing polygons is a useful operation when dealing with occlusion since the resulting decompositions can be used as features to describe the object even if some of its parts are not visible. The advantage in using sub-parts as object models resides in the ability to recover parts of the original model in case of occlusion or when the original object is not completely visible.

In the decomposition process we divide the object into a number of sub-parts. The decomposition can be carried out according to a number of techniques, good surveys are available in the literature [23, 18]. Each decomposition technique is driven by certain criteria, which can be adapted to suit the application at hand. In the case of overlapping objects it is desired to group the points located between two concave vertices in the same sub-part to prevent having points belonging to two different objects associated in the same sub-part.

We have devised a special decomposition method that is well adapted to the problem of occluded objects identification. In the learning phase the decomposition method tries to create sub-parts that are stable, informative and significant. By significant we mean that each sub-part has enough points and covers at least

a minimum area this prevents the generation of very small or thin sub-parts that might be due to false concavities and quantization noise and do not have a good differentiating capability. By informative we mean that each sub-part contains peculiar features of the original object. By stable we mean that the method is able to cope with some changes in scaling, orientation and moderate noise levels. In the recognition phase the decomposition method divides the shape representing the occluded objects into sub-parts. It is required in this stage that the sub-parts generated obey certain criteria. First each sub-part has to come from only one object model, this would help in the generation of the hypothesis. It is also desirable that some sub-parts have more than a certain (usually 64) number of points on their boundary; this criteria is needed to generate hypotheses through the AR classifier. By relying on a robust method for the identification of concave and convex arcs and a simple pairing algorithm that pairs compatible concave arcs into valid sub-parts, the method is able to create sub-parts that are significant, informative and stable.

The first stage of the method is to identify concave and convex arcs. A concave arc is a sequence of concave vertices uninterrupted by convex vertices (see Figure 3.6).

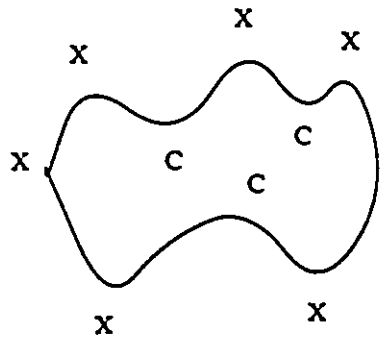

Figure 3.6: Division of the boundary into concave and convex arcs.

If X is used to denote a convex corner and C a concave corner then the angular

characteristic of the shape S is defined as

$$AC(S) = x_1 x_2 ... x_n$$

where x_i is either C or X. A sequence of C's will correspond to a concave arc. A shape S is decomposable if its angular characteristic $AC(S)$ contains at least two concave arcs. Such a decomposition can map a picture into a graph by having nodes corresponding to sub-parts and branches linking nodes if the respective sub-parts share a boundary. Figure 3.6 illustrates this approach the lines in the figure are the dividing lines.

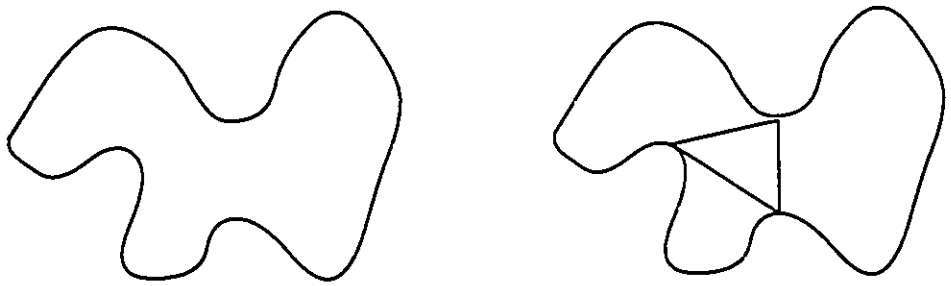

Figure 3.7: An object and its sub-parts.

A vertex which forms with its neighboring vertices an angle that that exceeds a certain threshold (say 170°) would be regarded as convex vertex whereas a vertex whose angle which is less than 170° and higher than 10° would be regarded as a concave vertex.

The decomposition procedure uses lines joining vertices belonging to adjacent concave arcs as dividing lines. Intuitively, a set of points which should be grouped together as a simple part of the curve will be joined by interior line segments. A set of points joined by external segments represent an intrusion to the object. Sets of

points represented by an intersecting line have, in general, no spatial relationship. When a line joining two concave vertices from different arcs is found, it is a candidate for a baseline provided that it satisfies the following four criteria:

1. The line must not lie outside the boundary: this is a necessary condition since if the line lies outside the boundary (see Figure 3.9) the enclosed vertices will not represent a sub-part of the object.

2. The line must not intersect the boundary in one or more points: this is also a necessary condition since if the line cuts the boundary the vertices enclosed by the two endpoints of the dividing line will not represent a continuous sub-part (see Figure 3.10).

3. the area covered by the sub-part is above a certain threshold: this is an optional condition which is needed to prevent the creation of very thin sub-parts that do not contain enough discriminating information.

4. the contour of the boundary contains at least N points, where N is bigger than the minimum contour length threshold: this is also an optional condition needed to make sure that the sub-parts created contain enough vertices (64 at least) to form the time series which will be used to generate the AR parameters.

The first condition can be checked by comparing the oriented angles as shown in Figure 3.6. The concave angle I between the original sides is greater than the angle O between the line pp' and either of the sides when pp' lies within the polygon, while the opposite is true if pp' is outside.

The second condition can be verified by checking the orientation of boundary traversal. Given three points A, B, and C we want to know whether in traveling

from A to B to C we turn clockwise or counterclockwise. Then to decide whether two segments (p_1, p_2) and (p_3, p_4) intersect, we check whether both endpoints of each line are on different "sides" of the other, that is we turn in different directions in going from p_1 to p_2 to p_3 and from p_1 to p_2 to p_4 or in going from p_3 to p_4 to p_1 and from p_3 to p_4 to p_2.

The third condition is verified by computing the area contained between the two points, this can be done in a number of ways. We use the following formula:

$$A = \frac{1}{2}\sum_{i=p}^{p'}(x_{i+1}y_i - x_i y_{i+1})$$

For the fourth condition we just count the number of pixels on the part of the boundary between the two sub-part endpoints.

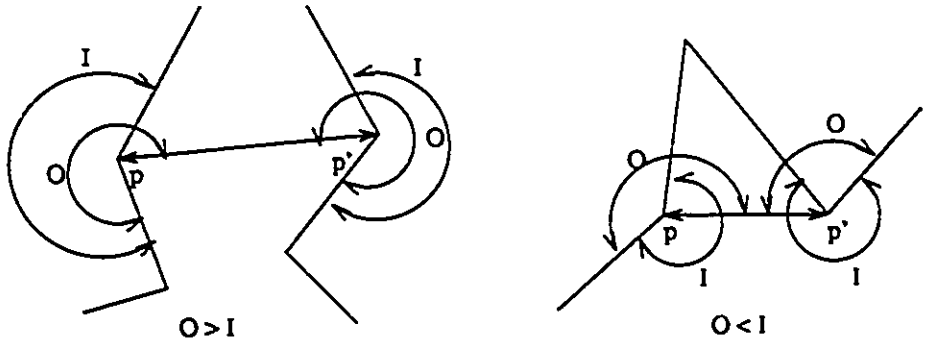

Figure 3.8: Comparison of the angles O and I shows whether the line lies inside.

Figure 3.6, Figure 3.6, and Figure 3.6 illustrate the three different types of connections that can be established between disjoint points on the boundary of an object.

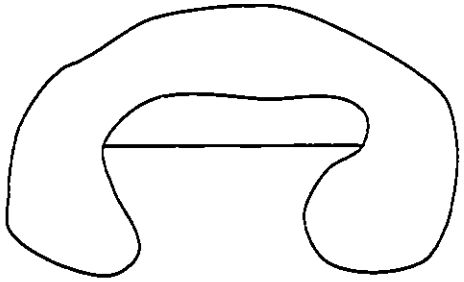

Figure 3.9: Invalid dividing line. The line is located outside the object.

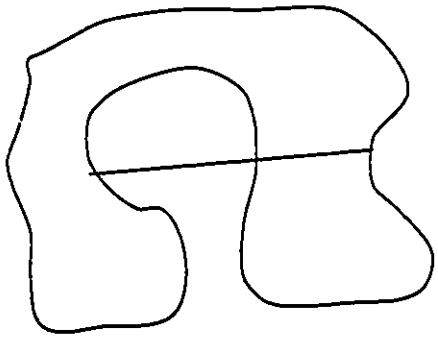

Figure 3.10: Invalid dividing line. The line intersects with the boundary.

The points of the contour that lie between these two concave vertices form a separate sub-part. Since almost all occlusions start at concave vertices, the risk of having a sub-part contain vertices from two different objects is reduced. Thus sub-parts can form a good starting point for generating hypotheses in the recognition process.

The decomposition of the boundary has many desirable properties; it is translation and rotation invariant and, to a large extent, size invariant. The invariance property stems from the fact that all the processing operations involved in the creation (from boundary extraction to sub-part division) of the sub-parts are not affected by changes in the orientation, scaling or the position of the object. This will ease the process of finding an invariant description of the object's boundary. One such description is given by the parameters of an auto-regressive model.

Once the boundary is divided into sub-parts we proceed by modeling each sub-part by a set of AR parameters. This compresses the volume of data used to characterize each sub-part and provide a description that is invariant to changes of starting point, translation, scaling and certain degrees of rotation. The next section will explain the theory behind the Auto-regressive modeling technique.

3.7 The Auto-regressive Model

A model is a description of an object. Statistical models characterize an image in terms of its statistical properties, such as correlation or co-occurrence. Structural models describe an image by its structural primitives and their placement rules. But if a structural model is not also statistical, the images it describes are too regular to be of interest, and if a statistical model cannot reveal the image's basic structure, it is too weak to be useful. A combination of statistical and structural modeling is necessary for efficient representation.

In this system the boundary of the object's sub-parts (structural primitives) are represented by an auto-regressive model. An *auto-regressive model* is a parametric equation that expresses each sample of an ordered set of data samples as a linear combination of a specified number of previous samples from the set plus an error term. The model has been widely used in speech recognition, speech synthesis, data compression, and spectral modeling.

In this work, a different time series is created for each sub-part of the object. The time series consists of the radial distances $r(n)$ of N angularly equispaced radius vectors projected between the middle point of the line delimiting the sub-part and the boundary.

Rectangular coordinate representations are not appropriate in this situation since they are not rotation invariant. The boundary approximation can be improved by increasing the number of radius vector projections, N. The radius vector lengths r are a function of the angle of projection $\phi = t2\theta/N$, where $t = 1, ..., N$, and the function $r(\phi)$ forms a one dimensional boundary approximation. The ordered set of numbers r can also be expressed as a "time series" $r(n)$, with the parameter n describing the position of a radius vector at equiangular increments from the starting point. For example, $r(1) = r(2\theta/N)$ and $r(N) = r(2\theta)$. The requirement of sub-part closure results in a periodic time series.

This time series is represented by an AR process

$$s_i(n) = \sum_{k=1}^{m} a_i(k) s_i(n-k) + e_i(n)$$

$$r_i(n) = s_i(n) + \mu_i, \quad i = 1, 2, ..., p$$

where m is the model order, $r_i(n)$ is the radial distance from vertex n to the center of sub-part i, p is the number of sub-parts. Here $s_i(n)$ is a zero mean stationary random sequence, μ_i is the set mean of $r_i(n)$, and $e_i(n)$ is a uncorrelated sequence with zero mean and variance β_i^2. The AR model parameters $a_i(k)$, β_i^2, and μ_i can be considered as features of the given set of sub-parts.

The features $a_i(k)$ are invariant under translation, scaling, and rotation. This is because the underlying correlations of the sequences $r_i(n)$, which determine the $a_i(k)$, are also invariant under these transformations. β_i^2 and μ_i are sensitive to scaling, but the term $\mu_i/|\beta_i|$ is not. The feature vector for sub-part i is given by $(a_i(1), ..., a_i(m), \mu_i/|\beta_i|)$.

3.7.1 Parameter Determination

Given a particular boundary or sub-part sequence s_n.

$$s_n = -\sum_{k=1}^{p} a_k s_{n-k} + e_n \tag{3.1}$$

the problem is to determine the particular a_k.

Method of Least Squares

Let the approximate to s_n be $\tilde{s}_n$:

$$\tilde{s}_n = -\sum_{k=1}^{p} a_k s_{n-k} + e_n \tag{3.2}$$

The error between the actual value s_n and the predicted value $\tilde{s}_n$ is :

$$e_n = s_n - \tilde{s}_n = s_n + \sum_{k=1}^{p} a_k s_{n-k} \tag{3.3}$$

e_n is also known as the *residual*. In the method of least squares the parameters a_k are obtained as a result of the minimization of the mean or total error squared with respect to each of the parameters. The squared error is given by

$$E = \sum_n e_n = \sum_n (s_n + \sum_{k=1}^{p} a_k s_{n-k})^2 \tag{3.4}$$

E is minimized by setting

$$\frac{\partial E}{\partial a_i} = 0 \quad 1 \le i \le p \tag{3.5}$$

from (3.4) and (3.5) we obtain the set of equations

$$\sum_{k=1}^{p} a_k \sum_n s_{n-k} s_{n-i} = -\sum_n s_n s_{n-i} \tag{3.6}$$

Equations (3.6) are known as the normal equations. The minimum total squared error, denoted by E_p, is obtained by expanding (3.4) and substituting (3.6). The

result can be shown to be

$$E_p = \sum_n s_n^2 + \sum_{k=1}^{p} a_k \sum_n s_n s_{n-k} \tag{3.7}$$

The error E in (3.4) is minimized over a finite interval, say $0 \leq n \leq N$. Equations (3.6) and (3.7) then reduce to

$$\varphi_{0i} = - \sum_{k=1}^{p} a_k \varphi_{ki} \tag{3.8}$$

$$E_p = \varphi_{00} + \sum_{k=1}^{p} a_k \varphi_{0k} \tag{3.9}$$

where

$$\varphi_{ik} = \sum_{n=0}^{N-1} s_{n-i} s_{n-k} \tag{3.10}$$

is the covariance of the signal s_n in the given interval. The coefficients φ_{ki} form a covariance matrix. This matrix is symmetric Toeplitz[1], i.e., $\varphi_{ik} = \varphi_{ki}$, and the terms along each diagonal are equal, since

$$\varphi_{i+1,k+1} = \varphi_{ik} + s_{-i-1} s_{-k-1} - s_{N-i-1} s_{N-k-1} \tag{3.11}$$

Since s_n is periodic in N, we have

$$s_{-i-1} s_{-k-1} = s_{N-i-1} s_{N-k-1}$$

So

$$\varphi_{i+1,k+1} = \varphi_{ik} \tag{3.12}$$

[1] a Toeplitz matrix is one where all the elements along each diagonal are equal

3.7.2 Computation of the Predictor Parameters

Direct Methods

The predictor parameters can be computed by solving a set of p equations with p unknowns, There exists several standard methods for performing the necessary computations, e.g., the Gauss elimination method. The general methods require $\frac{p^3}{3} + O(p^2)$ operations (multiplications or divisions) and p^3 storage locations.

Reduction in computing time can be achieved by writing the set of equations (3.8) as

$$\begin{pmatrix} \varphi_0 & \varphi_1 & \varphi_2 & \cdots & \varphi_{p-1} \\ \varphi_1 & \varphi_0 & \varphi_1 & \cdots & \varphi_{p-2} \\ \varphi_2 & \varphi_1 & \varphi_0 & \cdots & \varphi_{p-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \varphi_{p-1} & \varphi_{p-2} & \varphi_{p-3} & \cdots & \varphi_0 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_p \end{pmatrix} = - \begin{pmatrix} \varphi_1 \\ \varphi_2 \\ \varphi_3 \\ \vdots \\ \varphi_p \end{pmatrix} \tag{3.13}$$

since this is a Toeplitz matrix, and the column vector on the right comprises the same elements found in the covariance matrix, a method attributed to Durbin [28] can be used to solve the system of equations. This method requires $2p$ storage locations and $p^2 + O(p)$ operations. The Durbin recursive procedure can be specified as follows :

$$\begin{aligned} & E_0 = \varphi(0) \\ & k_i = -[\varphi(i) + \textstyle\sum_{j=1}^{i-1} a_j^{(i-1)} \varphi(i-j)]/E_{i-1} \\ & a_i^{(i)} = k_i \\ & a_j^{(i)} = a_j^{(i-1)} + k_i \, a_{i-j}^{(i-1)} \quad 1 \le j \le i-1 \\ & E_i = (1 - k_i^2) E_{i-1} \end{aligned}$$

These equations are solved recursively for $i = 1, 2, ..., p$ the final solution is given by

$$a_j = a_j^p \quad 1 \le j \le p \tag{3.14}$$

In obtaining a solution for a predictor of order p one actually computes the solutions for all predictors for order less than p.

3.7.3 The Burg Method

The Burg algorithm may be viewed as a constrained least squares minimization. Assuming a wide sense stationary process, the forward linear predictor

$$e_{pn} = s_n - \hat{s}_n = \sum_{k=0}^{p} a_{pk} s_{n-k} \tag{3.15}$$

where

$$a_{p0} = 1$$

is defined for $p \leq n \leq N-1$ and the backward error is given by

$$b_{pn} = \sum_{k=0}^{p} a_{pk}^{*} s_{n-p+k} \qquad p \leq n \leq N-1 \tag{3.16}$$

To obtain estimates of the AR parameters, Burg minimized the sum of the forward and backward prediction error energies,

$$\mathcal{E}_p = \sum_{n=p}^{N-1} |e_{pn}|^2 + \sum_{n=p}^{N-1} |b_{pn}|^2 \tag{3.17}$$

Subject to the constraint that the AR parameters satisfy the Levinson recursion

$$a_{pk} = a_{p-1,k} + a_{pp} a_{p-1,p-k}^{*} \tag{3.18}$$

for all orders 1 to p. This constraint is to ensure a stable AR filter (poles within the unit circle). By substituting

$$e_{pn} = e_{p-1,n} + a_{pp} b_{p-1,n-1}$$

and

$$b_{pn} = b_{p-1,n-1} + a_{pp}^{*} e_{p-1,n}$$

into the above recursion, $\mathcal{E}_p$ becomes a function of only the unknown prediction errors at order $p-1$, which are assumed known. Thus one need estimate only a_{ii} for $i = 1, 2, \ldots, p$. Setting the derivative of $\mathcal{E}_i$ with respect to a_{ii} to zero then yields

$$a_{ii} = \frac{-2\sum_{k=i}^{N-1} b^*_{i-1,k-1} e_{i-1,k}}{\sum_{k=i}^{N-1}(|b_{i-1,k-1}|^2 + |e_{i-1,k}|^2)} \tag{3.19}$$

$|a_{ii}| \leq 1$

A recursion relationship for the calculation of the denominator is given by:

$$DEN(i) = DEN(i-1)[1 - |a_{i-1,i-1}|^2] - |b_{i-1,N-i}|^2 - |e_{i-1,i}|^2 \tag{3.20}$$

The Burg algorithm can be improved upon if one minimizes $\mathcal{E}_p$ in equation (3.17) with respect to all the a_{pk} for $k = 1, \cdots, p$, this means that the Levinson recursion constraint is removed. To obtain the p normal equations for the forward-backward algorithm, determine the minimum of $\mathcal{E}_p$ by setting the derivatives of $\mathcal{E}_p$ with respect to all the AR parameters a_{p1} trough a_{pp} to zero. This yields

$$\frac{\partial \mathcal{E}_p}{\partial a_{pi}} = 2\sum_{j=0}^{p} a_{pj} r_p(i,j) = 0 \tag{3.21}$$

for $i = 1, \cdots, p$, where $a_{p0} = 1$ by definition and

$$r_p(i,j) = \sum_{k=0}^{N-p-1} (x_{k+p-j} x^*_{k+p-i} + x_{k+i} x^*_{k+j}) \tag{3.22}$$

for $0 \leq i, j \leq p$. The minimum prediction error may be determined to be

$$\mathcal{E}_p = \sum_{j=0}^{p} a_{pj} r_p(0,j) \tag{3.23}$$

the above expressions can written in matrix form as

$$R_p A_p = E_p \tag{3.24}$$

where

$$A_p = \begin{pmatrix} 1 \\ a_{p1} \\ \vdots \\ a_{pp} \end{pmatrix}, E_p = \begin{pmatrix} \mathcal{E}_p \\ 0 \\ \vdots \\ 0 \end{pmatrix}, R_p = \begin{pmatrix} r_p(0,0) & \cdots & r_p(0,p) \\ \vdots & & \vdots \\ r_p(p,0) & \cdots & r_p(p,p) \end{pmatrix} \tag{3.25}$$

3.7.4 Burg Algorithm

Initialization

$$\mathcal{E}_0 = \sum_{k=1}^{N} |x_k|^2$$

$$DEN(0) = 2\mathcal{E}_0$$

Compute the reflection coefficients

$$a_{ii} = \frac{-2 \sum b^*_{i-1,k-1} e_{i-1,k}}{DEN(i)}$$

Levinson Recursion

$$a_{ki} = a_{k-1,i} + a_{kk} a^*_{k-1,k-i}$$

$$\sigma_k^2 = (1 - |a_{kk}|^2)\sigma_{k-1}^2$$

Update prediction errors

$$e_{pn} = e_{p-1,n} + a_{pp} b_{p-1,n-1}$$

$$b_{pn} = b_{p-1,n-1} + a^*_{pp} e_{p-1,n}$$

Increase Order by 1 and recalculate coefficients.

3.7.5 Selection of AR Model Order

One of the most important aspects of using the AR model is the selection of the order p. As a general rule, if we select a model with a low order, we obtain a highly

smoothed spectrum. On the other hand, if p is selected too high, we run the risk of introducing spurious low-level peaks in the spectrum. Two of the better known criterion for selecting the model order have been proposed by Akaike [1]. With the first, called *final prediction error (FPE) criterion*, the order is selected to minimize the performance index

$$FPE(p) = \hat{\sigma}^2_{wp}\frac{N+p+1}{N-p-1}$$

where $\hat{\sigma}^2_{wp}$ is the estimated variance of the linear prediction error. This performance index is based on minimizing the mean-square error for a one-step predictor.

The second criterion proposed by Akaike [2], called the *Akaike information criterion (AIC)*, is based on selecting the order that minimizes

$$AIC(p) = ln\hat{\sigma}^2_{wp} + \frac{2p}{N}$$

Note that the term $\hat{\sigma}^2_{wp}$ decreases and hence $ln\hat{\sigma}^2_{wp}$ also decreases, as the order of the AR model is increased. However, $\frac{2p}{N}$ increases with an increase in p. Hence a minimum value is obtained for some p.

An alternative information criterion proposed by Rissanen [36], is based on selecting the order that *minimizes the description length (MDL)*

$$MDL(p) = N\ln(\hat{\sigma}^2_{wp}) + p\ \ln(N)$$

A fourth criterion proposed by Parzen [31] called the *criterion auto-regressive transfer (CAT)*, function and is defined as

$$CAT(p) = (\frac{1}{N}\sum_{k=1}^{p}\frac{1}{\bar{\sigma}^2_{wk}}) - \frac{1}{\hat{\sigma}^2_{wp}}$$

where

$$\bar{\sigma}^2_{wk} = \frac{N}{N-k}\hat{\sigma}^2_{wp}$$

The order p is selected to minimize $CAT(p)$.

3.8 Conclusion

In this chapter we have described the theory behind the techniques used to create object models. The models will be used by the algorithms described in the next chapter in the recognition of and labeling of objects in arbitrary images containing objects that have been previously modeled.

The next chapter will describe the algorithms used for occlusion detection, hypotheses generation, pruning and verification. It will also show how the sub-part division helps in the detection of occlusion and the recognition of objects in the presence of occlusion. We will also show the usefulness of using the auto-regressive filters parameters in the hypotheses generation phase and how they speed up the recognition process. Finally we will describe the algorithms used for the verification of the pruned hypotheses.

Chapter 4

Object Recognition Algorithms

The previous chapter dealt with the techniques used in the extraction of the boundary, the derivation of the polygonal representation and its decomposition, and finally the computation of the auto-regressive (AR) model parameters for each sub-part. The techniques described in this chapter aim to produce an efficient and accurate recognition engine that will be used to recognize and locate objects. The recognition engine can tolerate occlusion, moderate noise intensity and missing object parts. Its main tasks will be: to detect occlusion, identify and find the pose of objects that are present in the scene. The main aids in the recognition process are the characteristics of the sub-parts.

The next section will describe the recognition engine and the pattern recognition algorithms it uses.

4.1 Scene Analysis

The system can be used to recognize known objects under different orientations, translations, and scalings and in the presence of occlusion.

The techniques used to extract scene features are identical to the ones applied in the learning stages. The boundary of the scene is extracted and segmented, then the points composing the boundary are processed to generate the polygonal representation and the objects' sub-parts. Finally, the AR parameters are computed for each blob in the scene and for each detected sub-part. Figure 4.1 illustrates the processing steps in the learning mode and Figure 4.2 the steps the system goes through in the recognition mode.

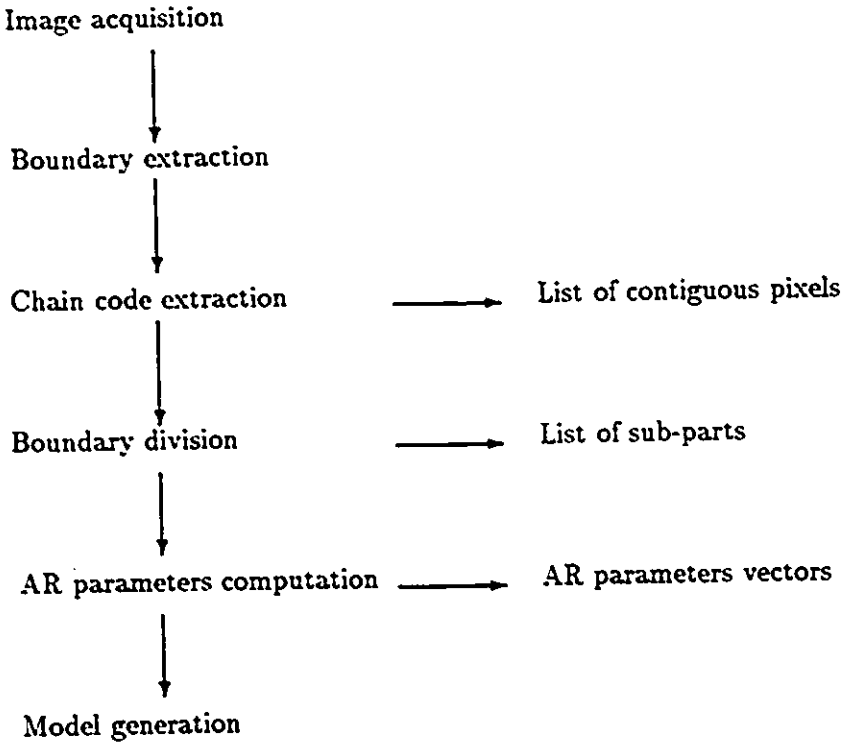

Figure 4.1: Processing steps in the learning phase.

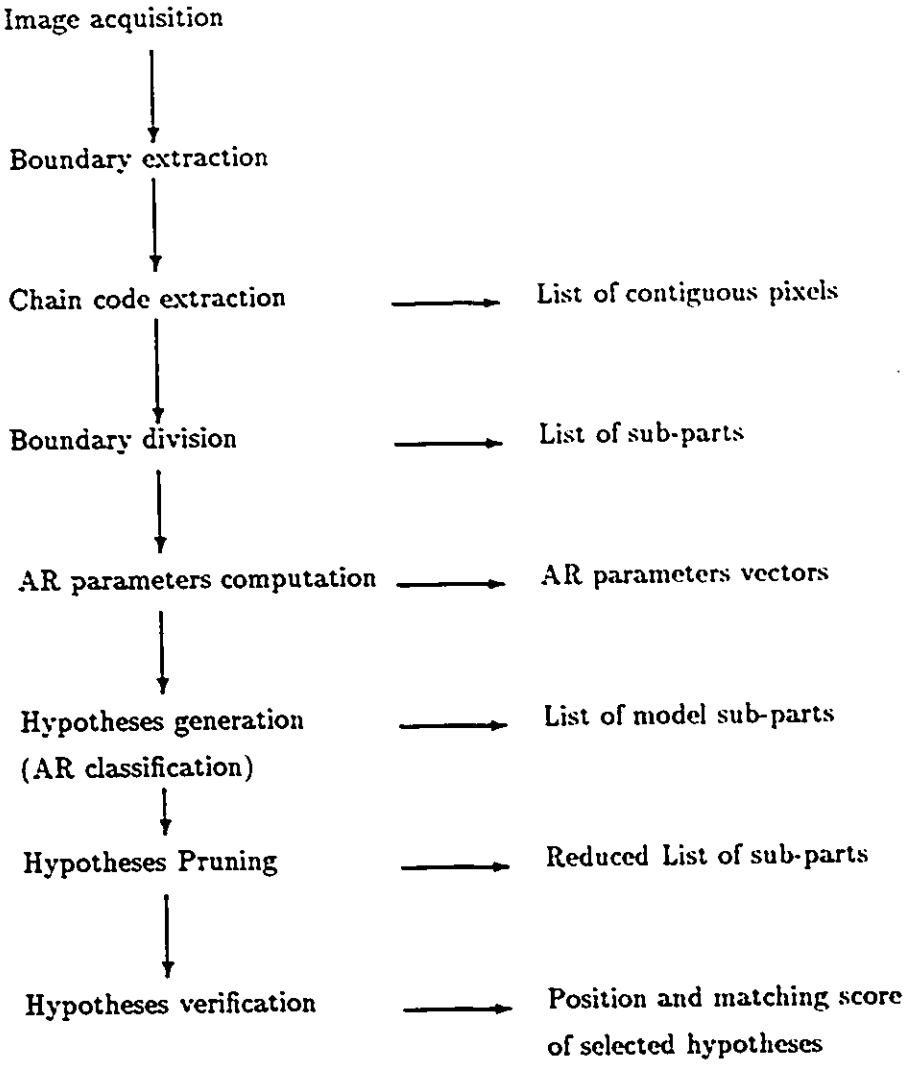

Figure 4.2: Processing steps in the recognition phase.

The system can go through an optional processing step. This step, occlusion detection, can sometimes help in the recognition of the objects and can use some characteristics of occlusion to guide the recognition process. The next section will describe the utility of this step and the results that it produces.

4.2 Occlusion Detection Algorithm

To detect occlusion a number of rules are used. The results of the different rules applied to the image are assigned weights and combined linearly into one occlusion confidence value expressing the likelihood that one region overlaps a given neighboring region. The weights are assigned based on heuristics. A side effect of this process is the determination of the region of intersection between the objects that form the scene. The rules are based on the following observations.

- Occlusion creates concavities in the scene that are not present in the model.
- Convex scene segments correspond to whole or partially occluded segments that belong to one object.
- Concave points are points where occlusion could start, and therefore are appropriate as limits of partially occluded segments.
- If a boundary has more than eight points of intersection with a circle whose center is the centroid of the scene, then occlusion could be present (see Figure 4.2). To find the intersection points a polar check is performed by scanning circles around the centroid of the object, and finding the intersections of the circles with the contour. The radius of the circle is the $\frac{2}{3}$ times the average of the N local maxima radii. Where a local maxima radius is the longest radius in

an extended neighborhood area consisting of a given number of pixels (usually 50).

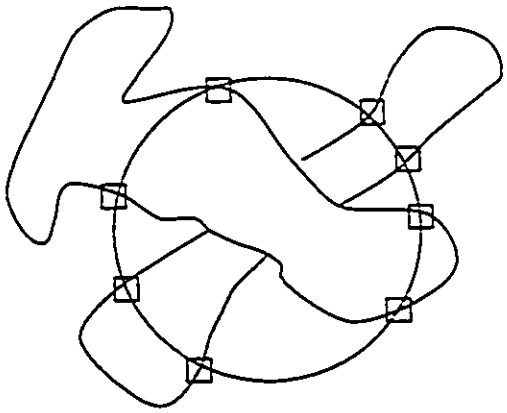

Figure 4.3: Intersection between a circle and two overlapping objects.

The above observations are true unless the objects have special positions in the scene.

- Occlusion creates a scene whose area is bigger that any single object in the scene. If the area of the scene is larger that of any object model times the upper limit of the scaling factor then we can be confident that the scene contains occluding objects.

Once the features of the scene are extracted, the recognition phase starts. The following section will explain the different steps of this phase and the techniques used to recognize the objects and to locate them in the scene.

4.3 Hypotheses Generation and Verification

The three main stages that make up the recognition phase are hypotheses generation, hypotheses pruning and elimination, and hypotheses verification. In the first stage hypotheses are generated according to the features of the scene. In the second stage the hypotheses are pruned to limit their number, the pruning proceeds by eliminating some hypotheses according to predefined criteria and a number of constraints. In the final stage the remaining hypotheses are tested and verified through a matching of the scene features and the models sub-parts features.

4.3.1 Hypothesis Generation

The hypothesis generation step is used to predict the identity of probable sub-parts and not the transformations of probable interpretations. The model transformations are not predicted since they consist of too many parameters to be predicted accurately: scaling besides translation and rotation.

The AR classifiers are used to obtain a list of hypotheses from the list of object models. This list is obtained by applying a classifier to the extracted AR parameters and looking for the sub-parts of the models in the database that have the closest AR parameters. The classification will result in a set of sub-parts and complete object models whose AR parameters fall in the same cluster as those of the sub-parts or objects in the scene under consideration. Two different classifiers were used to perform this step. The next section will explain the nature of the classifiers used and the results of their application.

4.3.2 AR classifiers

The AR model parameters are invariant under certain transformations of the boundary and can thus act as identifiers for the object's sub-parts. Two statistical recognition algorithms are used to investigate shape classification through AR parameters: the feature weighting method, and the hyperplane method. These techniques will map sub-part names to sets of AR parameters of scene sub-parts; each set of AR parameters represents one sub-part in a particular location and orientation.

By simply viewing the data of AR vectors we are unable to determine whether or not the AR model parameters for different shapes preserve enough information to reliably discriminate between classes of shapes. Therefore we need to employ mathematical techniques to measure inter-model differences.

The Feature Weighting Method

The feature weighting method emphasizes the common features of a set of samples representing one class, and de-emphasizes the distinct features.

Given a set of $(m+1)-$dimensional sample vectors of AR model order m parameters, where a sample s is represented as

$$s = \begin{pmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_m \\ \frac{\sigma}{\sqrt{\beta}} \end{pmatrix} \tag{4.1}$$

an unknown vector is recognized by finding the set of sample vectors with which it has the most common features. In other words, an unknown vector is assigned to the class identifying the closest cluster of sample vectors. The distance between an unlabeled pattern p and a set of class Q training samples $\{q_k\}$ is the sum of the weighted Euclidean distances between the unlabeled pattern and each training set pattern. The feature weights are selected to emphasize the common features of a set of samples representing one class and to de-emphasize the distinct features. Finding the weights that emphasize common class features is equivalent to scaling the axes of the pattern space coordinate system to cluster sample points more closely. Sebestyen [40] shows that the jth optimum feature weight for a particular class is inversely proportional to the jth feature of the standard deviation of the jth of the class training set. Sebestyen also shows that the Euclidean distance between p and $\{q_k\}$ is

$$S(p,\{q_k\}) = \left(\prod_{j=1}^{m+1}\sigma_j\right)^{2/(m+1)} \left(\sum_{i=1}^{m+1}\left(\frac{p_i-\overline{q}_i}{\sigma_i}\right)^2 + m + 1\right) \tag{4.2}$$

where $\overline{q}$ is the class Q sample mean vector and σ_i is the variance of the class Q training set in the ith original coordinate direction in vector pattern space. To recognize an unlabeled sample p, the distance S to each class is calculated, p is assigned to the class corresponding to the smallest S. The samples of that class form the cluster that is closest to p in the transformed or weighted pattern space.

Hyperplane Method

In the hyperplane method the least-mean squared error procedure uses differences between classes to form linear decision boundaries or hyperplanes that divide the

pattern space into unique convex polyhedral regions for each class. For each of c classes, a linear discriminator function is formed such that

$$g_i(p) > g_j(p) \quad for\ all\ j \neq i \quad if p \in class\ i$$

A least squares procedure uses all c training sets to find the discriminant functions which best satisfy the conditions

$$g_i(p) = 1 \quad for\ p \in class\ i,$$

$$g_j(p) = 0 \quad for\ all\ j \neq i$$

Devijver and Kittler [15] show that the ith class discriminant function, $g_i(x)$ which best fits the data is

$$g_i(p) = y^T \kappa^{-1} M_i,$$

where p is a pattern vector, y is the augmented pattern vector

$$y = \begin{pmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_m \\ \frac{\alpha}{\sqrt{\beta}} \\ 1 \end{pmatrix}$$

M_i is the class i mean vector, and

$$\kappa = \sum_{i=1}^{c} \kappa_i$$

where κ_i is the augmented correlation function for class i

$$\kappa_i = E_i(yy^T)$$

The decision rule is

$$p \in class\ i\ if\ y^T \kappa^{-1} M_i > y^T \kappa^{-1} M_j$$

$$for\ \ all\ \ j = 1, \cdots, c,\ \ j \neq i.$$

The Hyperplane method uses information about all classes in the vector pattern space to partition the space into unique decision regions for each class.

The two classification algorithms described previously are used to generate a list of hypotheses. The hypotheses are generated by keeping the N models sub-parts that have the highest H classification scores.

4.3.3 Hypothesis Pruning

In this phase the hypothesized objects are further examined to confirm or reject their presence in the scene. The obvious mismatches are eliminated from the list. This is done by checking and comparing the global features of the objects and the scene. The area of an object cannot be bigger than that of the scene, neither can the perimeter of the object. An object should in most cases have less concave vertices than the scene, since occlusion creates a number of concave vertices that depend on the number of objects in the scene and their geometric positions. Each scene segment is checked against each feature of each sub-part. If the model and scene segments are found to be very different the sub-part is removed from the hypotheses list. The compatibility criteria are segment length and orientation compared with that of the sub-part.

An effect of this step is to remove invalid model components from consideration; if no compatible scene segments are found for segments from a model sub-part, then there is no need to attempt to recognize that sub-part in the scene.

4.3.4 Hypothesis Verification

Given the ordered list of sub-parts, the system attempts to recognize them in the scene. A mapping of the object sub-parts onto the scene is conducted. This operation is equivalent to a string matching, where the transformed vertices of the object are compared to those of the scene, the strings are given by the segments formed by joining adjacent dominant points.

The transformation that maps each sub-part onto its closest match on the scene is calculated and is applied to all the segments forming the sub-part, the similarity of these segments to the corresponding segments in the scene is then evaluated. A good score suggests an accepted hypothesis. No further hypotheses need to be tested for this scene sub-part. Sub-parts from the same object are mapped on the scene using the same transformation and we check whether any of them match. The process continues with hypotheses generated by other scene sub-parts until all the objects forming the scene are identified.

The purpose of the verification step is to estimate the location of the model in the scene, and verify whether the hypothesis that this sub-part is in the scene is true. Location of the object in the scene is estimated by finding a coordinate transformation consisting of translation rotation, and scaling that maps the matched sub-parts in the model to the corresponding matched sub-parts in the scene in a least squared sense. A score based on the least squares error of the mapping is used to quantify the total goodness of match between the model and the scene.

Let k be the number of pairs of the model and scene sub-parts that match with each other,

$$\{(x_1, y_1), (x_2, y_2), ..., (x_k, y_k)\}$$

be the coordinates of the points in the matched model sub-part.

$$\{(u_1, v_1), (u_2, v_2), \ldots, (u_k, v_k)\}$$

be the coordinates of the points in the scene sub-part.

We want to find a coordinate transformation.

$$\begin{pmatrix} u' \\ v' \end{pmatrix} = \begin{pmatrix} e \\ f \end{pmatrix} + \begin{pmatrix} a & b \\ -b & a \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \tag{4.3}$$

with the scale factor $= \sqrt{(a^2 + b^2)}$,

the rotation angle $= tan^{-1}(\frac{a}{b})$, and the translation coefficients $\begin{pmatrix} e \\ f \end{pmatrix}$,

such that

$$\epsilon = \sum_{i=1}^{k} e_{u_i}^2 + e_{v_i}^2 \tag{4.4}$$

where

$$e_{u_i} = |u'_i - u_i| = |ax_i + by_i + e - u_i|$$

$$e_{v_i} = |v'_i - v_i| = |-bx_i + ay_i + f - v_i|$$

is minimized. By finding the partial derivatives of Equation 4.4 with respect to each coefficient of the coordinate transformation described by Equation 4.3, we can obtain the following coefficients of the least squares coordinate transformation:

$$\begin{pmatrix} a \\ b \\ e \\ f \end{pmatrix} = T^{-1} \begin{pmatrix} \sum_{i=1}^{k}(u_i x_i + v_i y_i) \\ \sum_{i=1}^{k}(u_i x_i - v_i y_i) \\ \sum_{i=1}^{k}(u_i) \\ \sum_{i=1}^{k}(v_i) \end{pmatrix}$$

where

$$T = \begin{pmatrix} \sum_{i=1}^{k}(x_i^2 + y_i^2) & 0 & \sum_{i=1}^{k}(x_i^2) & \sum_{i=1}^{k}(y_i^2) \\ 0 & \sum_{i=1}^{k}(x_i^2 + y_i^2) & \sum_{i=1}^{k}(y_i^2) & -\sum_{i=1}^{k}(x_i^2) \\ \sum_{i=1}^{k}(x_i) & \sum_{i=1}^{k}(y_i) & k & 0 \\ \sum_{i=1}^{k}(y_i) & -\sum_{i=1}^{k}(x_i) & 0 & k \end{pmatrix}.$$

If the scale factor of the object in the scene is available from the AR parameters, the scale factor derived from the least squared coordinate transformation can be used as an additional parameter for verifying the match. The hypothesis of the model is the scene is decided by the value of the match error; a small error verifies the hypothesis while a large one will cause it to be rejected. The error threshold is set empirically.

Once a sub-part has been identified and localized in the scene, that knowledge can be used to simplify the recognition of related sub-parts. All objects that contain the identified sub-part can be selected as possible candidate model objects to explore. The system can then predict the identity and transformation of related sub parts that will reduce the list of model object candidates. If enough components of a model object have been recognized, the resultant component shape is returned as a scene interpretation.

4.4 Conclusion

This chapter has described the main components of the recognition engine and how they operate together to accomplish the recognition of objects based on stored models. The implementation of the algorithms will be described in the next chapter. The verification part has been found the only one amenable to a parallel implementation. The rest deal mostly in quantities that have to be accessed in a random fashion and are thus difficult to implement efficiently in parallel.

Chapter 5

A Parallel Machine Implementation of the Overlapped Object Identification

In the previous chapters we have explained the methodology followed in the design of the vision system and the different algorithms that were used. In this chapter we will deal with the specifics of the implementation and the various techniques and methodologies that were used to develop the vision system. In Section 5.1 a brief overview of the **AIS-5000** Machine SIMD parallel machine architecture is presented. The architecture of the machine and its development environment have had a big effect on the implementation. The next sections will deal with the object oriented methodology and the Objective-C language used to implement some parts of the system.

This chapter contains some pseudo-code included only to give an idea on the implementation of the algorithms. For more details on the code please refer to [12].

5.1 The AIS-5000 Machine

The algorithms described in the previous two chapters have been implemented in the C and Objective C programming languages on an AIS-5000 parallel processor[38]. The AIS-5000 has a special architecture and most vision algorithms have to be written in a special way to fully use the power and capabilities of the machine.

5.1.1 Architecture of the AIS-5000

The AIS-5000 is a massively parallel processor with up to 1024 processing elements arranged in a single instruction multiple data (SIMD) architecture. The processing elements can be arranged in a one dimensional chain that can be as wide as the image. Each individual processing element is a programmable bit serial processor with its own memory. The AIS-5000 can be programmed with the C or Objective-C object oriented language. The system provides a software library that allows access to the parallel architecture, like feature extraction and shape manipulations.

The individual PE of the AIS-5000 is a general purpose bit serial processor that can perform three types of operations: Boolean, arithmetic, and neighborhood. Boolean operations are characterized by the number (two, three, or four) of input variables. the source inputs are bits read from parallel memory. The sources are transformed according to a truth table supplied as input. In the neighborhood operations a logical operation is performed upon a set of bits to produce one bit result. In arithmetic operations a sum bit and a carry bit are generated for two source bits and a carry bit input. The carry bit is stored in the carry register and used as input to the PE on the next cycle.

The architecture of the machine makes neighborhood and morphological operations applied to the whole image the most appropriate vision operations. This is useful in the preprocessing phase where the whole image is thresholded and filtered. but does not help in the symbolic processing of the shape. In this phase a pixel or a group of pixels on the boundary is operated upon and used as a basis for a particular decision: sub-part division, angular characteristics, etc. The parallel logical operators are useful in the object recognition phase where whole objects can be compared to each other by using (AND, OR and XOR, operators).

The PEs take the image column by column and process it in parallel. The programmer has to think in the same way.

In this section we have seen how the architecture of the machine influenced the implementation of the vision system. In the next section we will describe the programming language used and illustrate some of the particular techniques for programming the algorithms explained in the previous chapters. For a complete listing of the program see[12].

5.1.2 Development Environment on the AIS-5000

This section will give a brief overview of the **LAYERS Tools** development environment used for application development on the AISI parallel machines. **LAYERS** is as set of programming tools for creating image processing applications that run directly on AISI's vision computers. **LAYERS** supports application development in C and Objective-C[1]. **LAYERS** contains:

[1]the new **LAYERS 2.0** supports C and AISI's object oriented language Intelligent C.

- a library of C functions and Objective-C objects for system and hardware control, image processing and feature extraction, and user interfaces
- utilities for building image processing applications on the SUN workstation and down-loading them to run on the parallel machine. Figure 5.1.2 illustrates the process of developing image processing applications in **LAYERS** environment

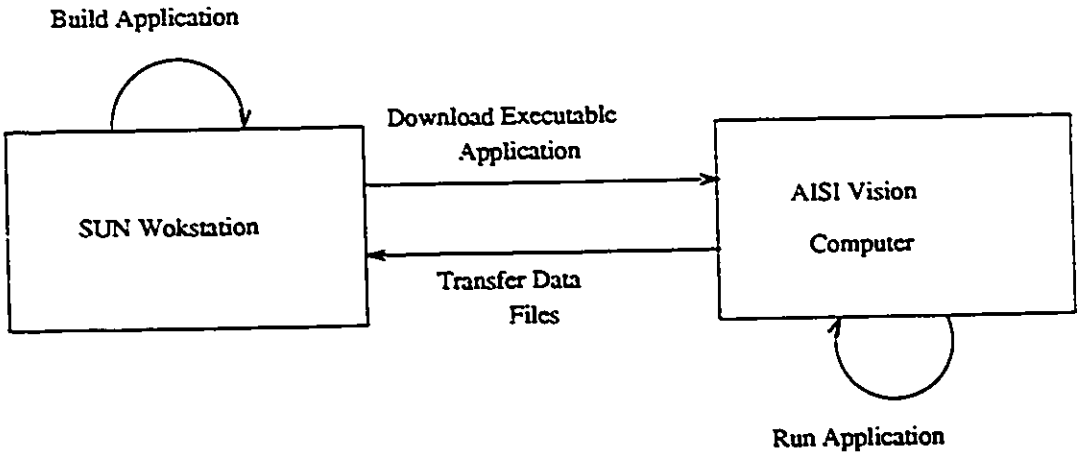

Figure 5.1: Application Development Process in **LAYERS**.

- a terminal emulation program to enable the workstation to attach to the parallel machine and up-load and down-load programs to the AISI vision computer and perform other functions. It acts as the programmer's interface to the vision computer, and can transfer data files between the two computers all via serial communications (RS-232).

5.2 Implementation Approach

A hybrid approach has been adopted for developing the vision system: the programs have been developed mostly in C and Objective-C. Objective-C has been used only

for parts of the program that could benefit from the object oriented approach. This choice was dictated by the nature of the vision algorithms where most of the algorithms in preprocessing and boundary processing stage are arithmetic in nature and there is little commonality between them and little scope for object oriented classes. The facilities and programming constructs needed to support object oriented programs here would slow the system and add to the size of the executable. The slowdown of the compilation and down-loading of applications increases as the number and size of the classes included in the application increases. Therefore we have restrained classes and objects to areas that would benefit greatly from them.

Clearly the object oriented approach is useful in the symbolic processing, feature extraction and the design and implementation of user interfaces aspects.

The use of object oriented analysis, design and implementation methods permits the reuse of third party code and eases the creation of reusable components for image processing and pattern recognition. The Object oriented methodology opens the way for reuse and tightly coupled cooperation between researchers in computer vision.

5.3 Implementation Steps

A number of pictures of industrial parts were used, they were taken with a CCD camera connected directly to the computer. The image is then thresholded and a binary image is obtained. A clockwise boundary follower is used to get a chain code representation of the boundary. The chain code is then analyzed and the points of locally minimum curvature are localized and stored separately. The pruning of this representation results in robust polygonal approximations. The next step is

the decomposition of the boundary into sub-parts. The processing of the images is performed according to the following steps:

5.3.1 Image Acquisition and Preprocessing

Acquire a 2-D grey level image using an RS-170 CCD camera. The image is stored on an image frame in the memory of the parallel processor. The two-dimensional image is then digitized, filtered, thresholded, and stored as a 512x512 binary image on the memory of the parallel processor. Three different methods can be used in this step:

1. Simple thresholding with a fixed threshold.
2. Adaptive thresholding with a variable threshold value computed through the histogram of the gray level image.
3. Application of an edge detection algorithm: A difference of gaussian operator has been found to offer the best results.

5.3.2 Aspect Ratio Correction

We correct the aspect ratio of the image by removing one column in each five columns band. This step is optional and aspect ratio correction can be performed later by multiplying the y coordinates by the value of the aspect ratio[2] in the y direction.

[2]The aspect ratio specifies the ratio of pixel y spacing to pixel x spacing. An RS-170 CCD camera has a total aspect ratio of 3/4. On the AIS-5000, in the configuration used for this system, 384 pixels are extracted in the x direction by 244 lines in the y direction, so the aspect ratio is = (3/244)/(4/384) = 1.1803.

To perform aspect correction in parallel a frame containing a 3 pixels wide vertical bar is *anded* with the original image, the result of the *and* operation is ored with the frame containing the result. The mask is shifted right by 4 pixels and the operation is repeated until the bar reaches the far right of the image.

The aspect correction helps improve the invariance of the image processing algorithms since the object would have the same size in any orientation. Performing the aspect correction in parallel is faster since it requires less operations and it is applied to the whole image instead of to each pixel as would be required in the sequential approach. But the parallel approach tends to add more quantization noise.

5.3.3 Contour Extraction and Chain Code Derivation

Apply morphological operations (erode, dilate, etc) to extract one pixel wide boundary of the object and other operations (shift, or, and, etc) to extract the chain code through a parallel algorithm. The detected boundary is represented by an array of points and an array of chain code values. To get the boundary of the image we extract the *North, South, East and West* edges of the image and then perform an *OR* operation to obtain the total boundary. The four edge frames are used in the computation of the pixels chain code values.

5.3.4 Boundary Points Extraction and Polygonal Representation

The boundary points are extracted by selecting an arbitrary point along the boundary and then retrieving its chain code value[3]. The chain code value of the pixel is used to specify which point is to be extracted next. The process is repeated until the current point in the boundary following process is the first point. Once the boundary points are obtained, the polygonal representation can be obtained in many ways:

1. *Scan-along:* in this method the dominant points that form the polygonal representation are obtained by eliminating the boundary points that do not represent a significant change in direction. The selection can be done in many ways. The following code fragment illustrates a selection based on testing whether the pixel represents a change of direction and whether it occurs at the end of a long segment.

   The scan along can also be implemented using the area deviation and the segment length deviation (see Section 3.3.1 and 3.3.2). Area and segment length deviation are computed using the incremental property and chain code values. A new polygon vertex is generated when the ratio area deviation per segment length exceeds an upper threshold.

2. *Split and Merge:* the split and merge technique has been explained in the Section 3.4. See the[12] for a complete listing of the code.

[3]This process is different from that used in traditional methods where the chain code is extracted through boundary following. The parallel computation of the chain code is mode robust and is less prone to quantization noise and spurious effects.

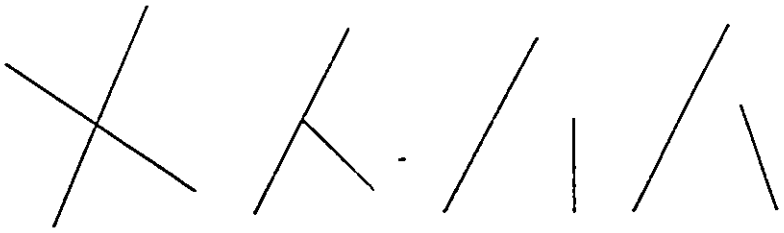

Figure 5.2: Testing whether line segments intersect: four cases.

5.3.5 Boundary Decomposition

To deal more effectively with occlusion and to have better models for the objects, each object is divided into sub-parts. The sub-parts are regions of the boundary that are delimited by concave points. The division of the boundary into subparts is a fast process that uses the concave regions of the boundary as a starting set the decomposition. We first compute the angular characteristics of the shape then we divide the shape into its sub-parts. For a complete listing of the code refer to [12].

The decomposition checks a number of conditions before deciding whether a line joining two concave segments is a valid dividing line or not. The conditions are:

1. the line joining the two concave segment is not allowed to intersect the boundary of the object. Figure 1 illustrates some of the situations that can arise. In the first case, the line segments intersect. In the second case, the endpoint of one segment is on the other segment: this is considered an intersection since the dividing line is not allowed to touch the boundary in more than two places; its starting point and its endpoint. In the last two cases the lines do not intersect.

   To check for this condition we use the following procedure based on a tool that

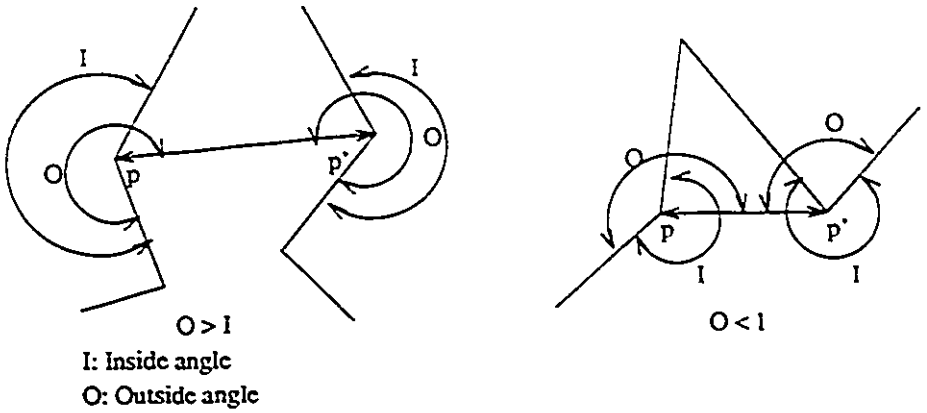

Figure 5.3: Comparison of the angles O and I shows whether the line lies inside.

checks the orientation of boundary traversal. Given three points A, B, and C we want to know whether in traveling from A to B to C we turn clockwise or counterclockwise.

The function compares the slopes of the three points p_1, p_2 and p_3. The slope of the line connecting p_1 to p_2 is $dy1/dx1$ and the slope of the line connecting p_2 to p_3 is $dy2/dx2$. If the slope of the second line is greater than the slope of the first a counterclockwise turn is required in the journey from p_1 to $p2$ to p_3 (the function returns a $+1$); if it is smaller than a clockwise turn is required (the function returns a -1). The function returns a 0 is p_3 is on the line between p_1 and p_2.

To decide whether two segments intersect, we check whether both endpoints of each line are on different "sides" of the other.

2. The second test checks whether the dividing line is inside the polygon. To check for this condition we compare the oriented angles between the original sides and the dividing line as illustrated in Figure 2. If the angle I between the original sides is greater than the angle O between the dividing line and either of the sides then the dividing line lies inside the polygon.

3. The third test checks whether the object area contained between the two endpoints is large enough. This test is necessary to remove very thin parts that

are not significant.

4. The fourth test checks whether the arc between the two end-points contains enough points. This test is necessary to remove small parts that are not significant and to create sub-parts that could contain enough points to derive the AR filter parameters.

The decomposition method is especially geared to deal with occlusion and differs from previous methods for object decomposition that were mainly devised for syntactic pattern recognition.

5.3.6 Object Modeling

We calculate the AR parameters for each generated sub-part and for the total object boundary. store the AR parameters and the polygonal representation of the subparts.

The models are stored in the following data structure and they are manipulated by functions in the file `model.c` see[12] for more details.

```
typedef struct point
{
        int x;
        int y;
} sPnt, *pPnt;

typedef struct sPrtInfo
Structure to store sub-part information
```

```
{       int sX, sY, eX, eY;
        int mConvX, mConvY;
        int nPnts;
        float *arPars;
        sPnt *coor;
        struct sPrtInfo *next;
} sPrtInf, *pPrtInf;

struct sTeach
Structure to store an object or a scene model
{       int numPrts;
        sPrtInf *pPrtInfo;
} ;
```

This is the last operation in the learning mode. In the recognition mode we add the following steps.

5.3.7 Hypotheses Generation

We compare the AR parameters of the scene sub-parts with the AR parameters of different models representing sub-parts of known objects. This is done by the classifiers described in Section 4.3.2. The result of this step is a list of hypothesized objects.

5.3.8 Pruning and Verification

We eliminate from the list of hypotheses the sub-parts that do not satisfy the constraints extracted from the scene. The constraints are based on area and perimeter comparisons and on the number of concave vertices. We then confirm or reject the presence of the next object from the list of hypothesized objects using a boundary match and a total object comparison. In this step a parallel algorithm is used to test the validity of the match by doing sub-part to sub-part comparisons. The matching tries to find a sufficient number of the object sub-parts in the scene before confirming its presence.

5.4 Conclusion

This chapter has described some implementation aspects of the vision system[4]. The special architecture of the machine has dictated a number of design and implementation choices for the preprocessing phase. Since most of the vision algorithms known today are serial in nature and were designed for serial machines and very few are suitable for a parallel SIMD machine, we decided to pursue a hybrid approach. In this approach the operations that are applied to the whole image like thresholding or erosion are carried on the parallel machine. Operations that are serial in nature and do not involve a large number of points, like sub-part division or the angular characteristics are carried on the SUN workstation. This approach would work best if there is a high speed real-time link between the visual computer and the Workstation. A dedicated Ethernet connection provides enough bandwidth and speed for the requirements of this project.

[4]The listings of the files are contained in[12]

The design of computer vision solutions on parallel machines demands a new set of algorithms and a different approach to problem solving. The algorithm can be dramatically different from those developed for sequential machines. Algorithms that have as a central data structure a point or some other special feature in the image tend to be less powerful and more time consuming that algorithms that treat the whole image like filtering, and some verification algorithms similar to template matching.

The implementation of this system is original since it has been accomplished on a SIMD parallel machine and a sequential machine and it had an object oriented design and implementation. Also, some of the algorithms are original and have been implemented in a particular fashion to take advantage of the architecture of the machine.

Chapter 6

Tests and Results

6.1 Introduction

In this chapter we discuss the testing methodology followed in the validation of the new algorithms described in chapter three and chapter four. We also evaluate the robustness, effectiveness, and determine the limitations of the algorithms and the complete recognition system. The tests seek to confirm that the capabilities of the algorithms are not severely affected by deformations in the objects such as: changes in the orientation (rotation), or the position (translation), or magnification (scaling), and finally the absence of one or more features of objects under consideration.

The experiments were conducted on a set of objects selected from the domain literature. The objects are taken from a list of generic objects used by a number of researchers to validate their approach to the occlusion problem [3],[16],[29]. Only the results and parameters of a subset of the objects used to validate the experiments are illustrated. Showing the results for a larger number of objects would take too

much space and will not contribute to a better understanding of the merits of the approach discussed in this thesis.

Each phase of the recognition process is tested and validated starting from the dominant point detection, and continuing with sub-part generation up to the last stage of object recognition. The tests will cover the following areas:

- First, we verify that the dominant points detected are always located in the same relative area of the object's boundary. That is the points are relatively stable under rotations and various scaling factors . A displacement in the location of dominant points, more specifically concave dominant points, would result in the deformation of the generated sub-parts. If a concave dominant point is not detected at all, some sub-parts will go undetected.

- Then in the test on the sub-part generation algorithms we try to check whether or not the detected sub-parts form significant and descriptive components that could be used to identify the original objects. We will also seek to establish that they are stable under rotation and scaling and examine the effects of the absence of some boundary points on the sub-part generation process. We will check whether the algorithm is capable of detecting sub-parts belonging to the unaffected area of the boundary and whether the sub-parts generated obey all the criteria outlined in Chapter 3. Finally we will check if the sub-parts are stable when the object is rotated, shrunk or magnified.

- We then check the stability of AR parameters under different changes in scaling and different degrees of rotations.

- Finally, the recognition algorithms are tested to see if they can identify objects in a number of settings including scenes where the objects occlude one another. We then evaluate their efficiency and robustness.

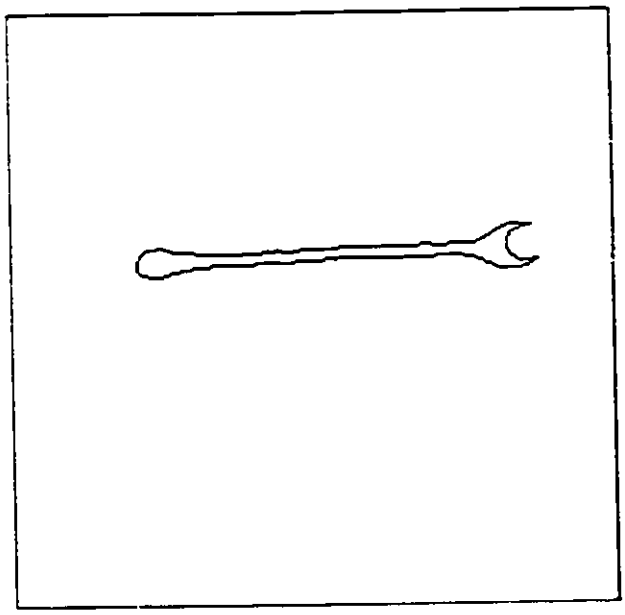

Figure 6.1: Object 1

The next section describes the results of the algorithms and operations used in the preprocessing phase. These algorithms have been applied to the objects shown in Figures 6.1 to 6.6.

6.1.1 Preprocessing Phase

The scenes containing the objects to be detected are first scanned in the system using a CCD camera connected to the computer through a Frame Grabber. The digital representation is an three dimensional array of 512×512 image elements (pixels) each pixel having an 8 bit gray scale value from 0 to 256. The gray scale image is then converted to a binary image where each pixel would either have a value of 1 or 0. This conversion is done by a thresholding process where each pixel whose gray scale value is less than the threshold is assigned 0 and each pixel with a gray value higher than the threshold is assigned 1. The binary image is then filtered

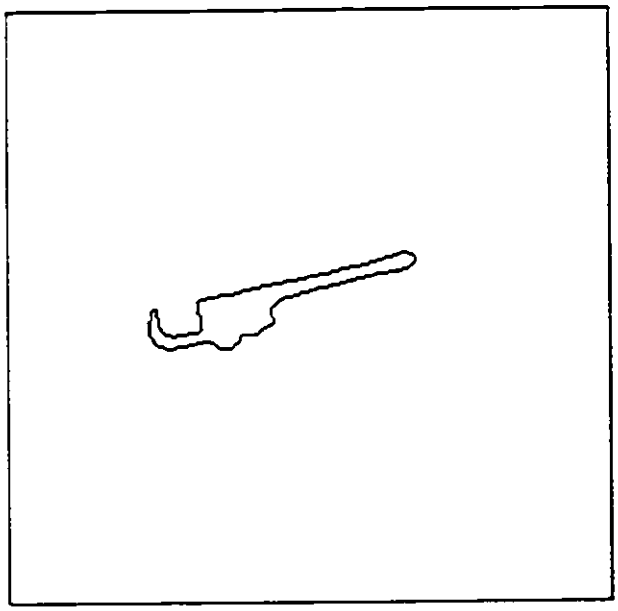

Figure 6.2: Object 2

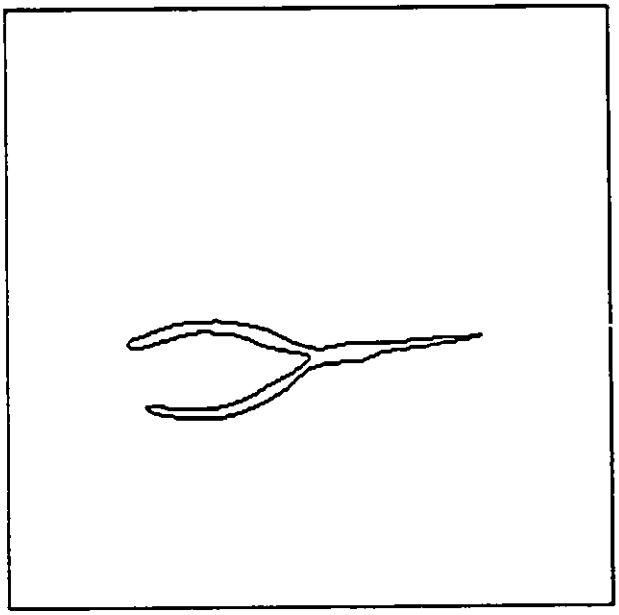

Figure 6.3: Object 3

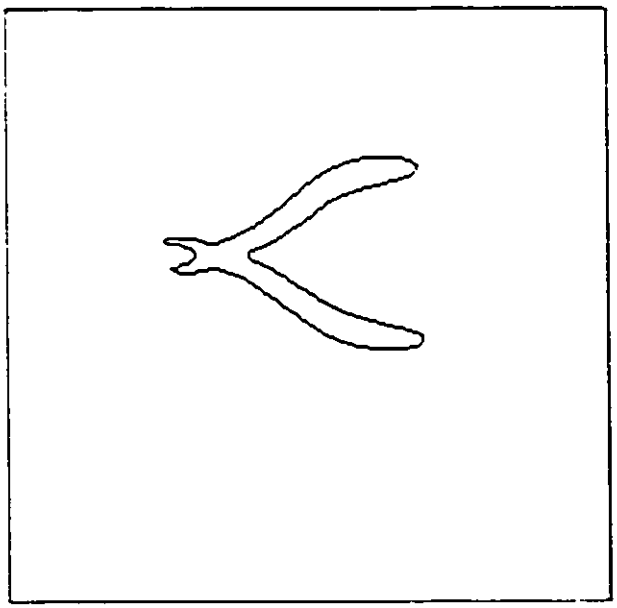

Figure 6.4: Object 4

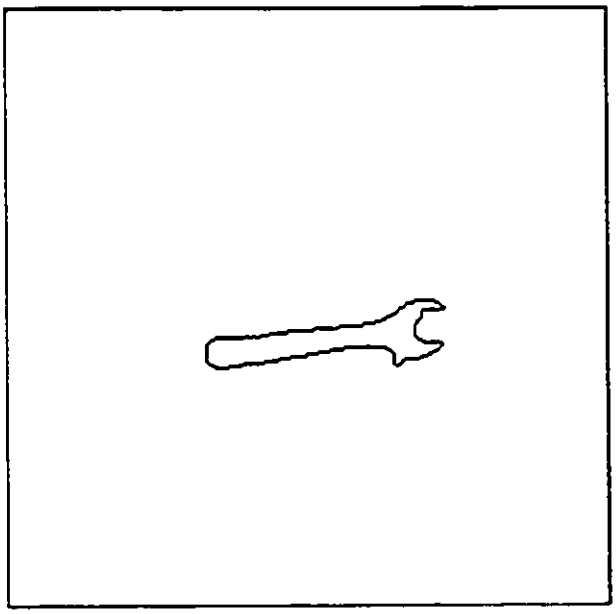

Figure 6.5: Object 5

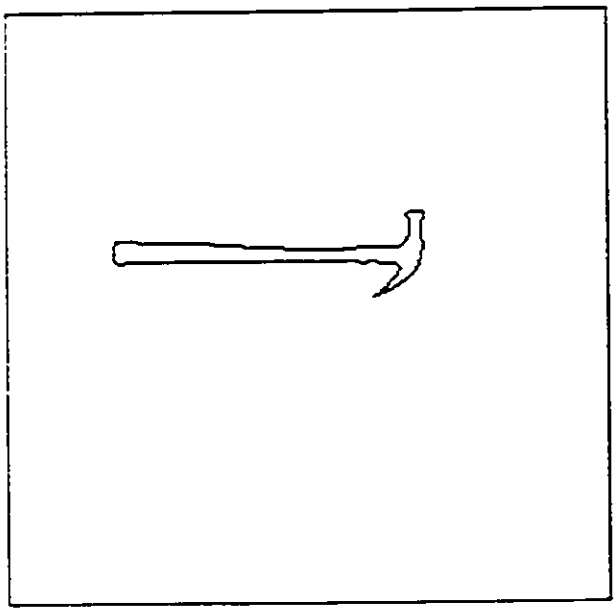

Figure 6.6: Object 6

and used to extract the boundary of the object or objects that compose it. The results of this phase are shown in Figure 6.1 to 6.6.

The boundary processing phase comprises all the preprocessing operations needed to extract a representation of the object that can lead to symbolic processing of the shape. All the algorithms in this phase are parallel and use morphological and logical operators. The edge of the object is extracted through morphological operations. The results are stored as the *North, South, East, and West* edges of the object. The detection of the north edge is performed by changing the value of a pixel from "1" to "0" if it has a north close neighbor with a value of "1". Otherwise the pixel value is unchanged. The same algorithm is carried out for each edge direction.

From Figure 6.1 we can see that quantization noise creates discontinuities and small "bumps" in the boundaries of the objects. These discontinuities pose special problems in subsequent processing steps and make it necessary to have a sophisti-

cated dominant point detection algorithm. The dominant point detection algorithm looks beyond the immediate neighborhood of a pixel to decide whether the pixel is a dominant point or not. Special precaution has to be taken to insure that the shape discontinuities do not provoke the detection of false dominant points and false con-cavities that would in turn result in the generation of false sub-parts. The next paragraph shows the tests on the dominant point location on the objects listed above.

6.1.2 Dominant Point Detection

This phase is the most critical in the system since the information conveyed by the dominant points, their relative position and angular characteristics are critical to the sub-part detection process and the model generation. If the object is to be decomposed in stable parts it is first necessary that the dominant points detected are always located in the same relative area of the object's boundary. This is similar to proving that the above points are relatively stable under translations, rotations and various scaling factors . A displacement in the location of dominant points, more specifically concave dominant points would result in the deformation of the generated sub-parts. If a concave dominant point is not detected at all, some sub-parts will go undetected.

The dominant points detection algorithm extracts the points that generate the best polygonal representation of the object's shape. Several methods have been tried for this phase. The very simple and fast methods like the *scan-along* seemed to generate an output that was comparable to that generated by more elaborate methods like the *split-and-merge* algorithm.

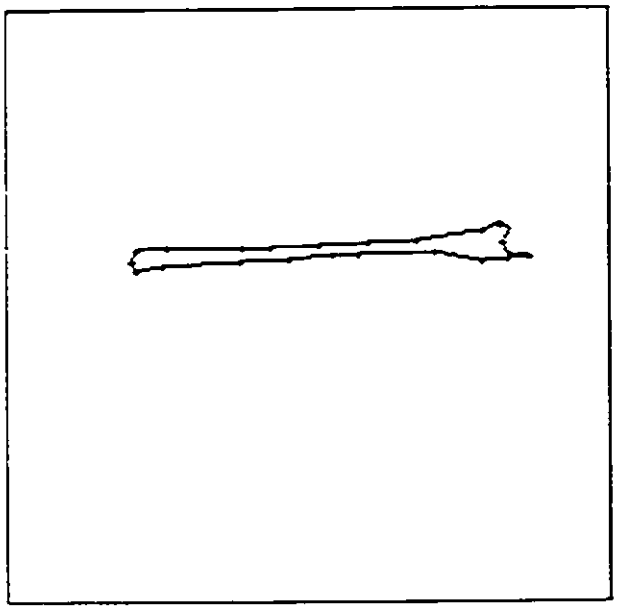

Figure 6.7: Dominant points detected by the scan along algorithm for the Object from Figure 6.1

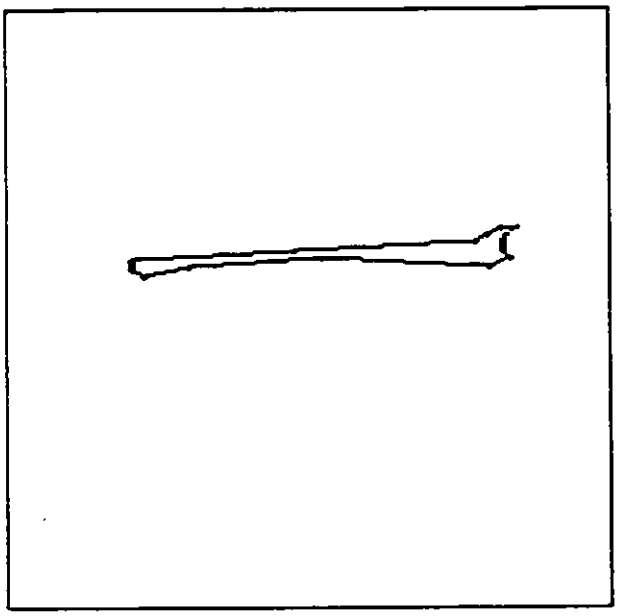

Figure 6.8: Dominant points detected by the split algorithm for the Object from Figure 6.1

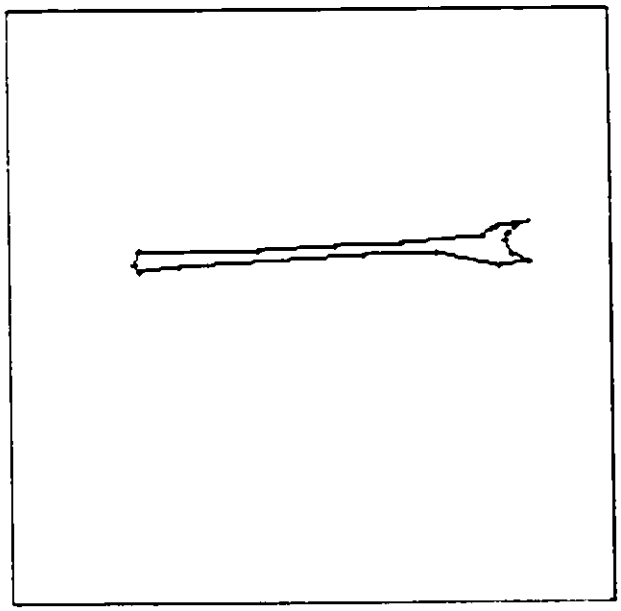

Figure 6.9: Dominant points detected by the split and merge algorithm for the Object from Figure 6.1

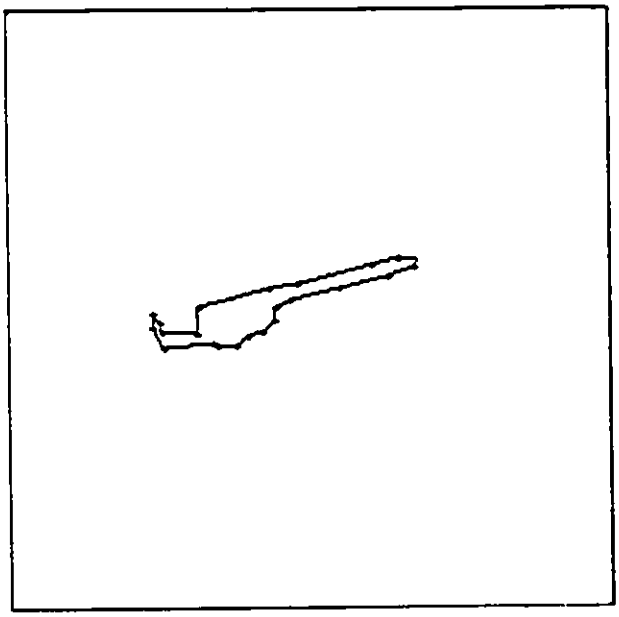

Figure 6.10: Dominant points detected by the scan along algorithm for the Object from Figure 6.2

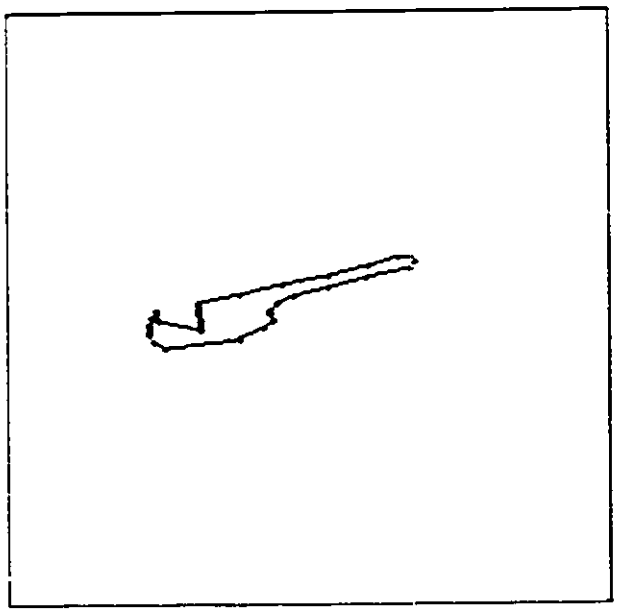

Figure 6.11: Dominant points detected by the split algorithm for the Object from Figure 6.1

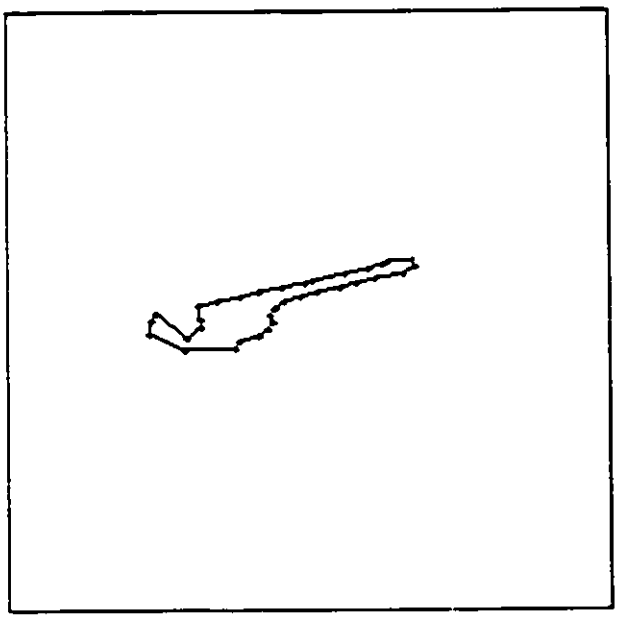

Figure 6.12: Dominant points detected by the split and merge algorithm for the Object from Figure 6.2

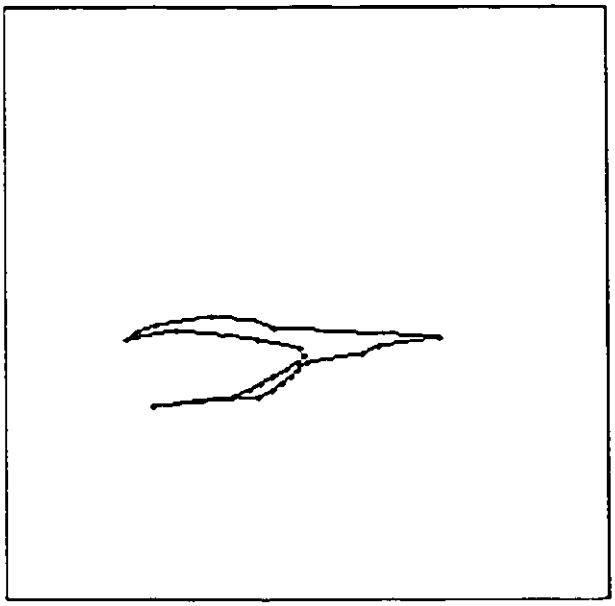

Figure 6.13: Dominant points detected by the scan along algorithm for the Object from Figure 6.3

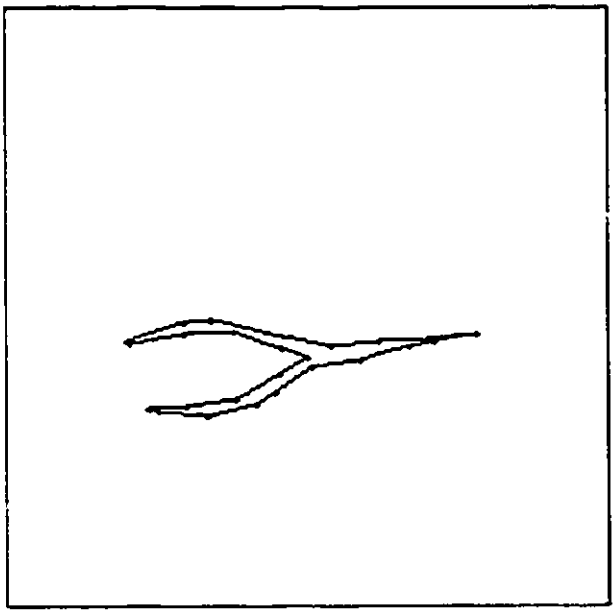

Figure 6.14: Dominant points detected by the split algorithm for the Object from Figure 6.3

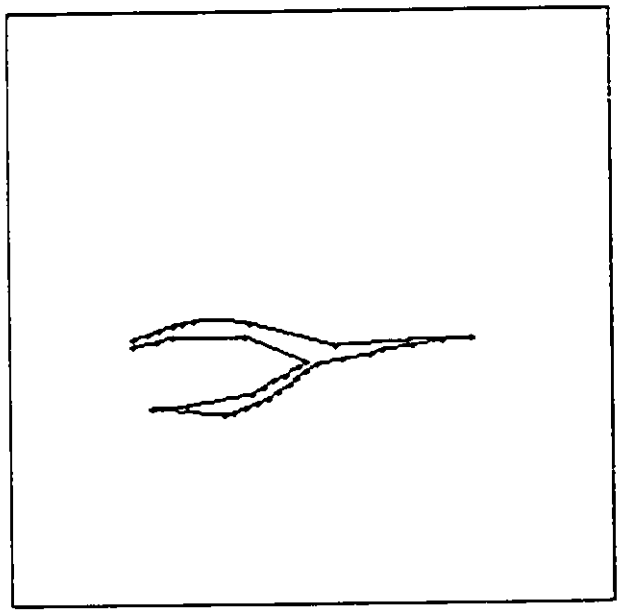

Figure 6.15: Dominant points detected by the split and merge algorithm for the Object from Figure 6.3

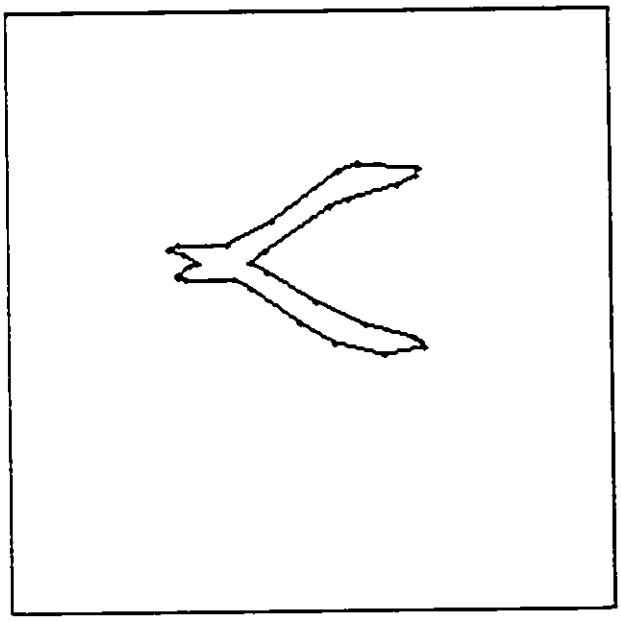

Figure 6.16: Dominant points detected by the scan along algorithm for the Object from Figure 6.4

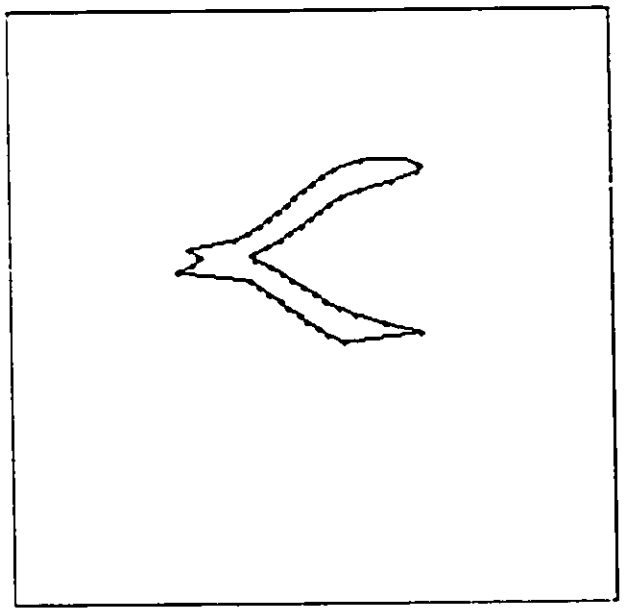

Figure 6.17: Dominant points detected by the split algorithm for the Object from Figure 6.4

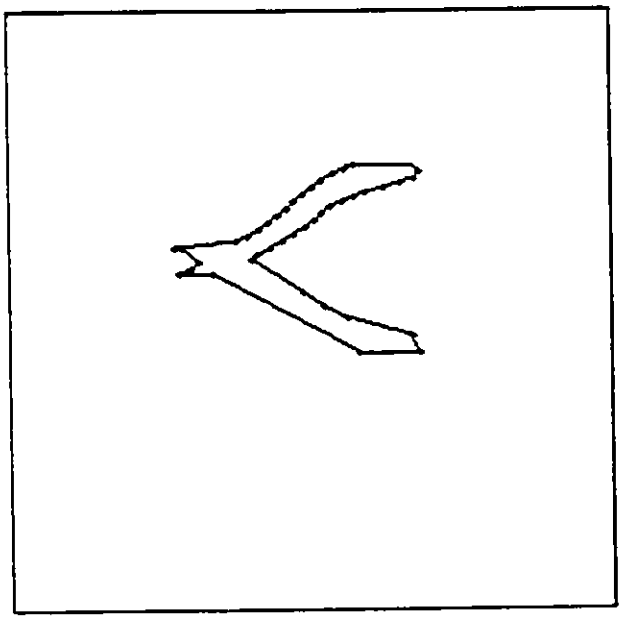

Figure 6.18: Dominant points detected by the split and merge algorithm for the Object from Figure 6.4

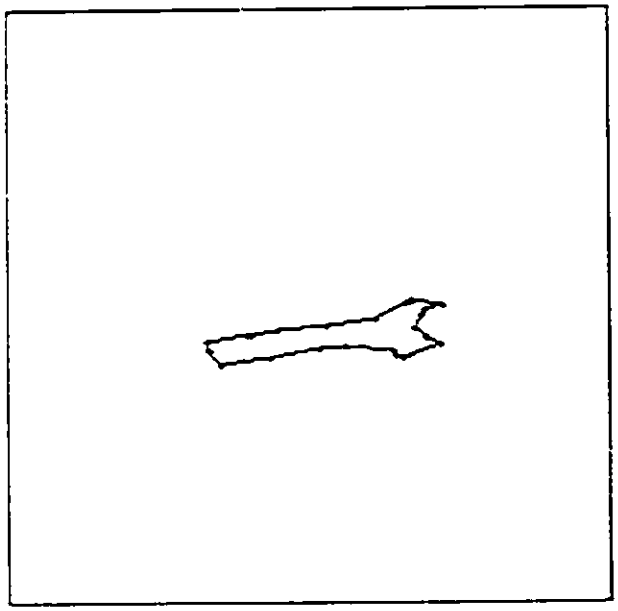

Figure 6.19: Dominant points detected by the scan along algorithm for the Object from Figure 6.5

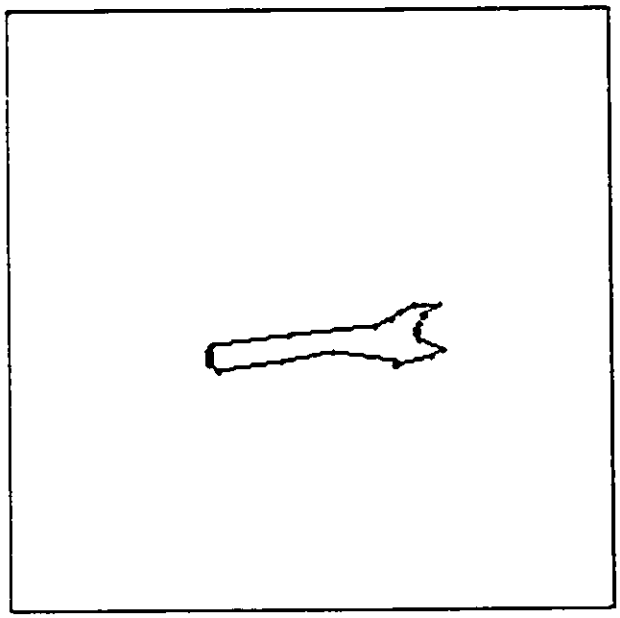

Figure 6.20: Dominant points detected by the split algorithm for the Object from Figure 6.5

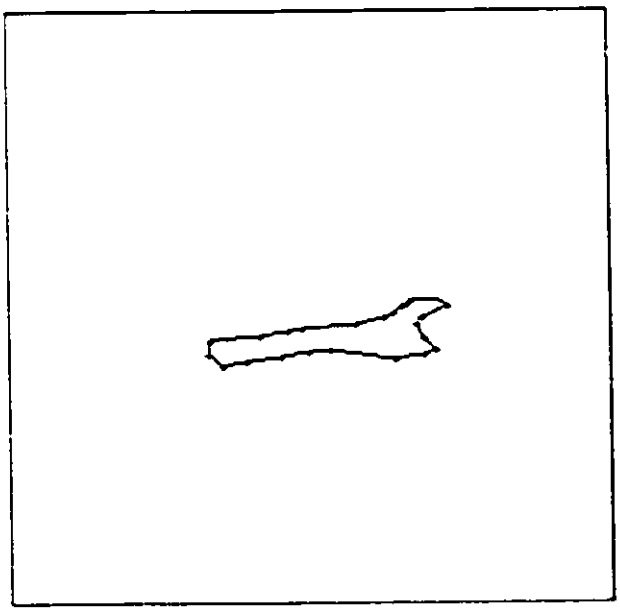

Figure 6.21: Dominant points detected by the split and merge algorithm for the Object from Figure 6.5

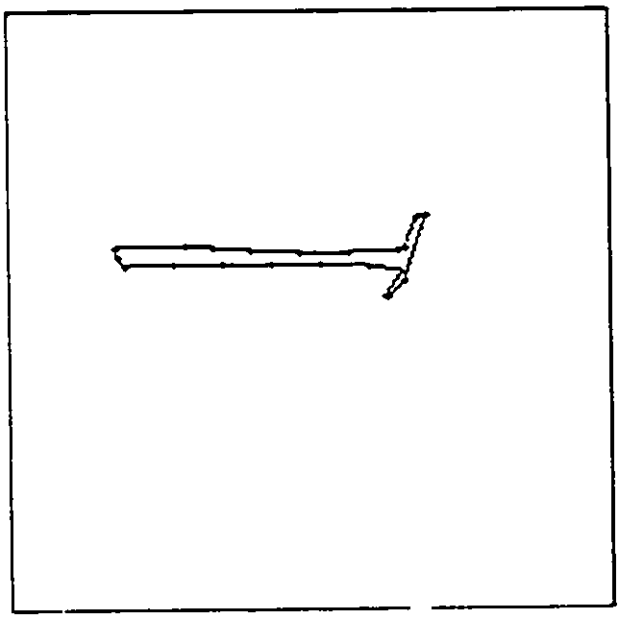

Figure 6.22: Dominant points detected by the scan along algorithm for the Object from Figure 6.6

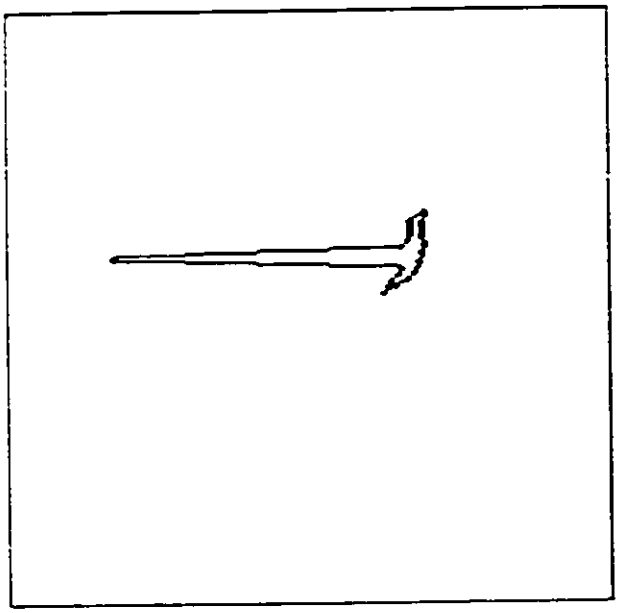

Figure 6.23: Dominant points detected by the split algorithm for the Object from Figure 6.6

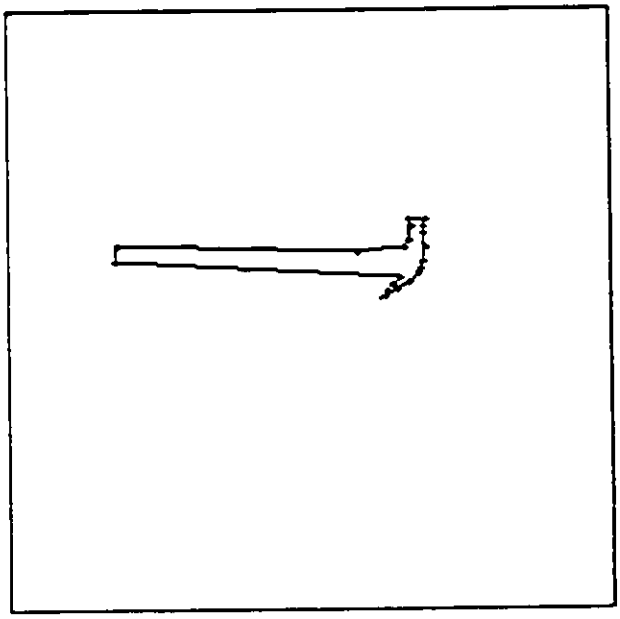

Figure 6.24: Dominant points detected by the split and merge algorithm for the Object from Figure 6.6

Figures 6.7 to 6.22 shows the results obtained through the scan along algorithm and Figures 6.8 to 6.23 the vertices generated by the split. Figures 6.7 to 6.22 are obtained using the split and merge algorithm.

The split-and-merge algorithm requires more memory and is slower since it is a two phase algorithm, while the scan along algorithm is faster and when used with the error measures described in Chapter 3 it tends to generate a reliable set of dominant points.

The dominant points generated in this phase are analyzed and an angular description of the boundary is computed, this leads to the determination of concave and convex points. The process of computing the angular characteristics of the boundary is done through the computation of the angle between successive dominant points. If the dominant points are too close (less that 3 pixels apart) the next neighboring vertex is used. The test to determine whether a vertex is concave or convex consists in checking whether the external angle made by this vertex with its two neighbors is less than 175 degrees and higher than 5 degrees. The angle threshold (175 and 5) is an empirical result that was determined after analyzing a number of objects, these values generated the best decompositions.

The dominant points and angular characteristics detected by these algorithms are used for determining the sub-parts of the object or the scene. They also play a major role in the model generation, the recognition and verification operations. In Figures 6.25 and 6.26 the dominant point detection method is tested against rotation and translations respectively. An arbitrary rotation of 15^0 was considered. The same results were obtained for most other rotations, the quantization noise and other factors affect the angular characteristics of the boundary objects.

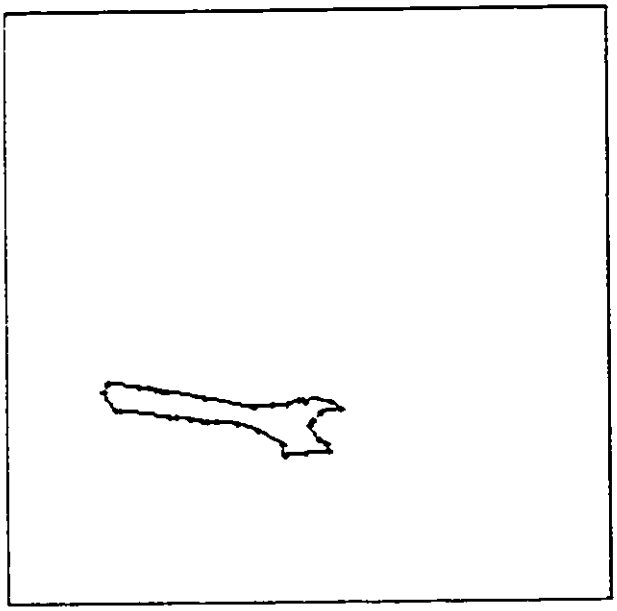

Figure 6.25: Dominant points detected by the scan-along algorithm for the Object from Figure 6.5 with a rotation of 15 degrees

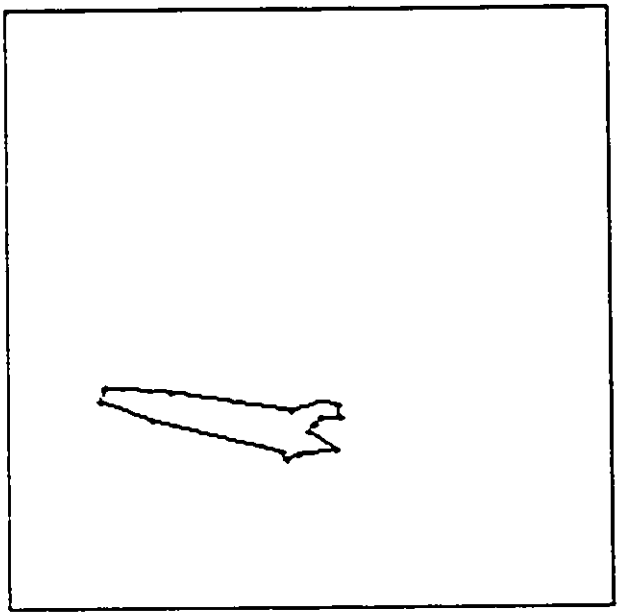

Figure 6.26: Dominant points detected by the split and merge algorithm for the Object from Figure 6.5 with a rotation of 15 degrees

6.1.3 Sub-part Generation

In the test on the sub-part generation processes we try to confirm and establish that the detected sub-parts are stable under translation, rotation and scaling. We will also examine the effects of the mis-detection of some concave dominant points on the sub-part generation process. We will try to check whether the algorithm is capable of detecting sub-parts belonging to the non-affected area of the boundary.

The object is divided into sub-parts that link certain concave dominant points of the boundary together. The sub-parts of the object are generated according to the position of the most concave boundary segments. The dividing lines have to obey the rules specified in section 3.6. Figures 6.27 to 6.32 illustrate the results of sub-part generation for six of the test objects. All the sub-parts detected are displayed regardless of whether they are used to generate AR parameters or not. Some sub-parts that have less than a certain number of points (64) or have a very small area are not used to generate AR parameters. The AR process requires a minimum number of points (64) to produce a meaningful vector of AR parameters.

From Figures 6.27 to 6.32 to we can see that the sub-part generation process is robust and is able to withstand the quantization noise.

Figure 6.33 shows the results of the decomposition when an object is rotated. It is clear that the decomposition algorithm generates the same decompositions when the object is rotated.

From Figures 6.34 and 6.35 we can see that in the case of a missing boundary part we are able to recover two of the three original sub-parts. The third sub-part contains the missing boundary segments and is not complete. Figures 6.36 to 6.40 illustrate the usefulness of the sub-part generation process in the presence of

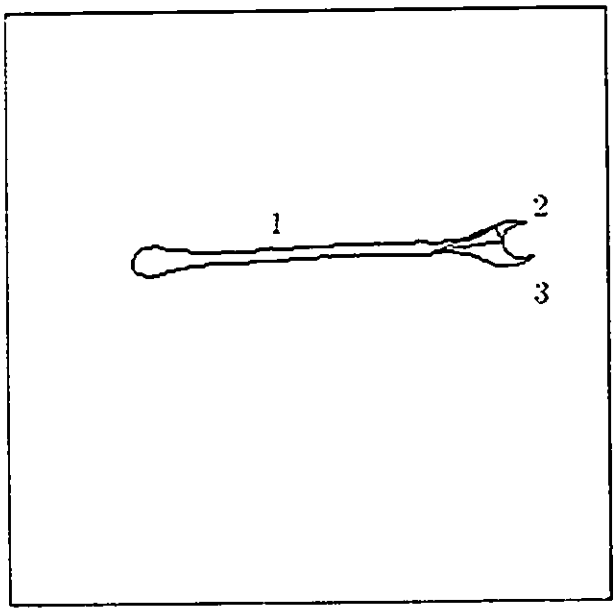

Figure 6.27: Sub-parts of the object shown in Figure 6.1. The sub-parts are delimited by the dividing lines. The first sub-part is always the one that contains the leftmost vertex (i.e. the vertex that has the lowest x coordinate). The next sub-part is the one that contains the vertices to the left of the first sub-part.

occlusion. In each of these cases we are able to detect sub-parts that belong to the original objects. The sub-parts are in some cases slightly modified but these small variations can be handled by the classification algorithms used with the AR parameters of the detected sub-parts.

In this section we have shown the robustness of the sub-part decomposition algorithm and its behavior in the presence of boundary quantization, rotation, missing object parts, and occlusion. In the next section we use the points of the generated sub-parts to create AR vectors for each sub-part.

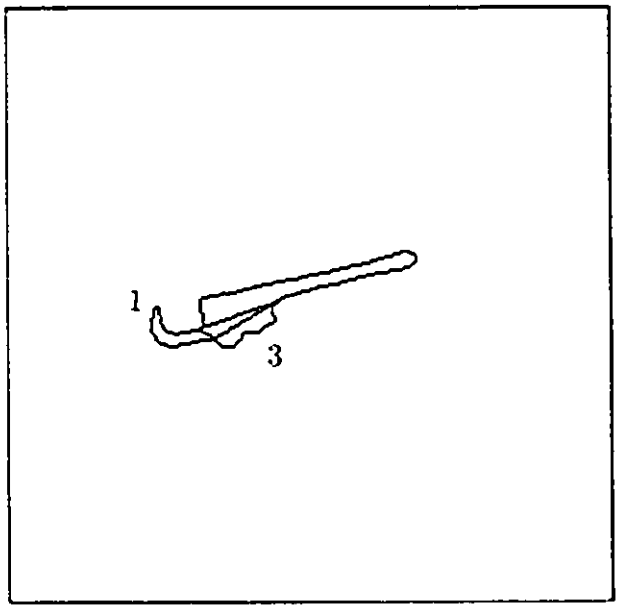

Figure 6.28: Sub-parts of the object shown in Figure 6.2. The sub-parts are delimited by the dividing lines. The first sub-part is always the one that contains the leftmost vertex (i.e. the vertex that has the lowest x coordinate). The next sub-part is the one that contains the vertices to the left of the first sub-part.

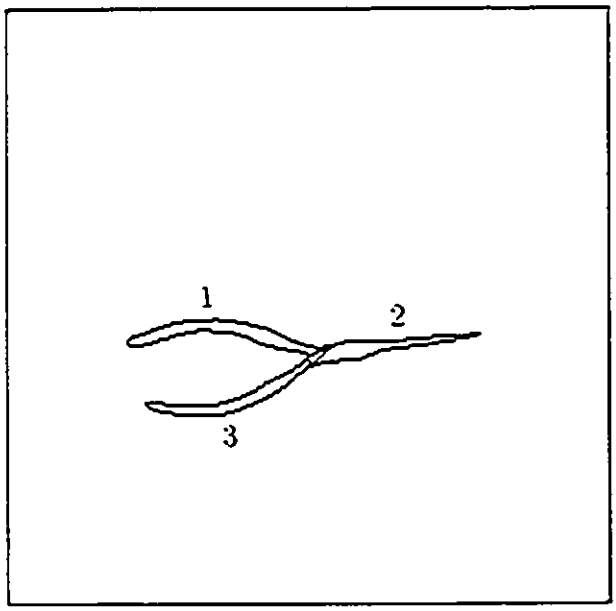

Figure 6.29: Sub-parts of the object shown in Figure 6.3. The sub-parts are delimited by the dividing lines. The first sub-part is always the one that contains the leftmost vertex (i.e. the vertex that has the lowest x coordinate). The next sub-part is the one that contains the vertices to the left of the first sub-part.

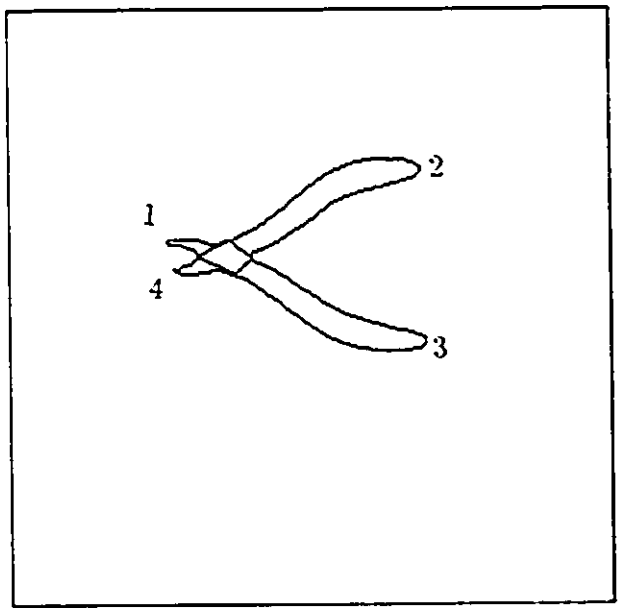

Figure 6.30: Sub-parts of the object shown in Figure 6.4. The sub-parts are delimited by the dividing lines. The first sub-part is always the one that contains the leftmost vertex (i.e. the vertex that has the lowest x coordinate). The next sub-part is the one that contains the vertices to the left of the first sub-part.

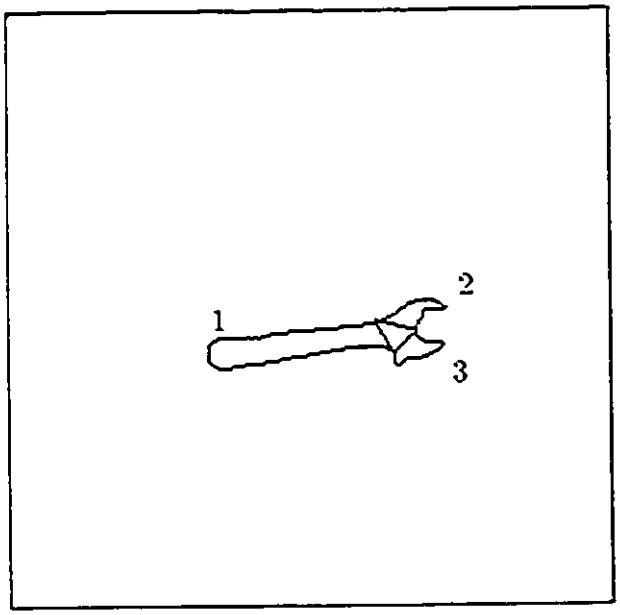

Figure 6.31: Sub-parts of the object shown in Figure 6.5. The sub-parts are delimited by the dividing lines. The first sub-part is always the one that contains the leftmost vertex (i.e. the vertex that has the lowest x coordinate). The next sub-part is the one that contains the vertices to the left of the first sub-part.

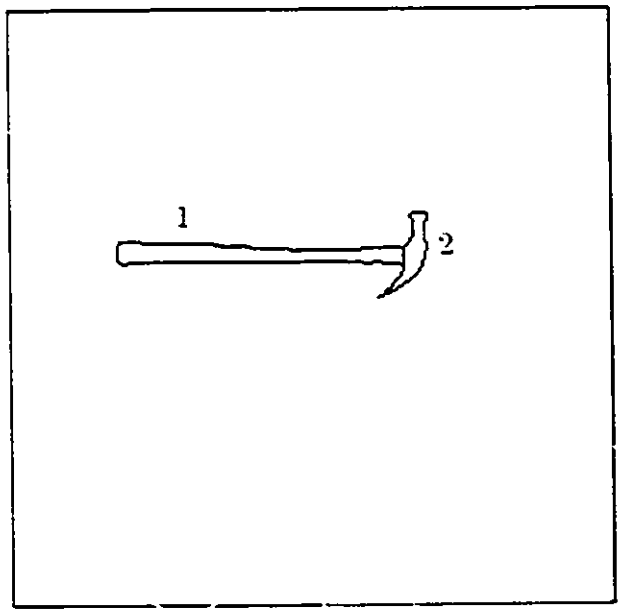

Figure 6.32: Sub-parts of the object shown in Figure 6.6. The sub-parts are delimited by the dividing lines. The first sub-part is always the one that contains the leftmost vertex (i.e. the vertex that has the lowest x coordinate). The next sub-part is the one that contains the vertices to the left of the first sub-part.

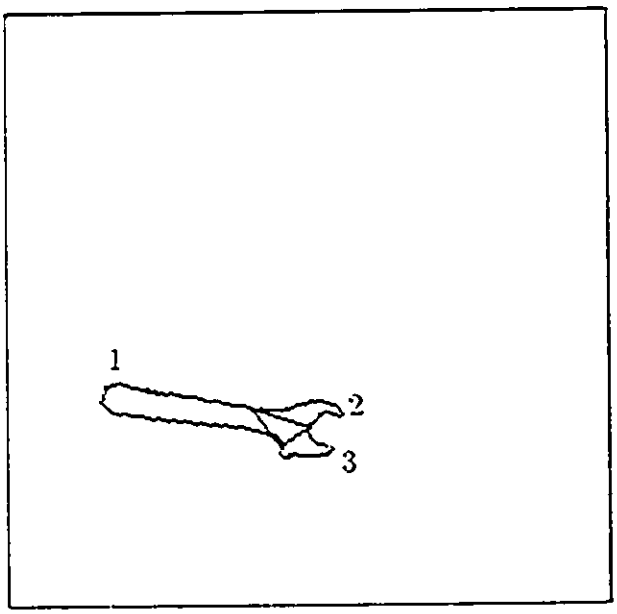

Figure 6.33: Sub-parts of the object shown in Figure 6.5 with a rotation of 15 degrees

6.2 Tests on Object Modeling through AR Parameters Computation

Once the sub-parts of the object are obtained we proceed with the calculation of the AR parameters. The AR parameters consist of a vector $(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \frac{\sigma}{\sqrt{\beta}})$, we stopped at order $p = 5$ since there is only insignificant improvements for higher filter orders [16].

This section lists the above introduced AR parameters for the objects shown in Figures 6.1 to 6.6. We then test the invariance of the AR parameters under changes of orientation and scaling.

The AR parameters of the test objects of Figures 6.1 to 6.6 are listed in Table 6.1 to Table 6.6. For Object 1 the AR parameters are listed in Table 6.1 we see that

there are two sub-parts that have been used to generate the AR parameters. The first row of the table lists the AR parameters for the first sub-part and the second row the AR parameters for the second sub-part. The AR parameters for object 2 are given in Table 6.2, etc.

Table 6.1: AR Parameters for Object 1. Each row lists the parameters of a given sub-part. Sub-parts with less than 64 points do not have an AR vector.

Sub-part	θ_1	θ_2	θ_3	θ_4	θ_5	$\frac{\sigma}{\sqrt{\beta}}$
1	-0.9424	0.0073	0.0884	0.0212	0.0473	2.4328
2	-0.9110	0.0960	0.1221	0.1835	0.3157	3.6045

Table 6.2: AR Parameters for Object 2. Each row lists the parameters of a given sub-part. Sub-parts with less than 64 points do not have an AR vector.

Sub-part	θ_1	θ_2	θ_3	θ_4	θ_5	$\frac{\sigma}{\sqrt{\beta}}$
1	-0.8187	0.2914	1.2210	2.8745	-1.7632	-0.6519
2	-0.9706	0.0707	0.0362	0.0447	0.0530	2.1814

Table 6.3: AR Parameters for Object 3. Each row lists the parameters of a given sub-part. Sub-parts with less than 64 points do not have an AR vector.

Sub-part	θ_1	θ_2	θ_3	θ_4	θ_5	$\frac{\sigma}{\sqrt{\beta}}$
1	-0.9518	0.0566	0.0366	0.0606	0.0382	1.9511
2	-0.8837	0.1171	0.1581	0.2611	1.3434	0.0056
3	-0.9934	0.0066	0.0066	0.0068	0.0067	2.0017

Table 6.4: AR Parameters for Object 4. Each row lists the parameters of a given sub-part. Sub-parts with less than 64 points do not have an AR vector.

Sub-part	θ_1	θ_2	θ_3	θ_4	θ_5	$\frac{\sigma}{\sqrt{\beta}}$
1	-0.8665	0.1760	0.2859	1.4298	1.3909	-2.7795
2	-0.8943	0.1267	0.1765	0.3025	1.6268	-1.8071
3	-0.9530	0.0495	0.0571	0.0627	0.0723	2.4373
4	-0.9903	0.0380	0.0367	0.0312	0.0491	1.9038

Table 6.5: AR Parameters for Object 5. Each row lists the parameters of a given sub-part. Sub-parts with less than 64 points do not have an AR vector.

Sub-part	θ_1	θ_2	θ_3	θ_4	θ_5	$\frac{\sigma}{\sqrt{\beta}}$
1	-0.9426	0.0626	0.0720	0.0852	0.1096	2.6698
2	-0.9256	0.0823	0.1019	0.1339	0.1882	3.0953

Table 6.6: AR Parameters for Object 6. Each row lists the parameters of a given sub-part. Sub-parts with less than 64 points do not have an AR vector.

Sub-part	θ_1	θ_2	θ_3	θ_4	θ_5	$\frac{\sigma}{\sqrt{\beta}}$
1	-0.9579	0.0128	0.0013	0.0120	0.0416	2.4102

6.2.1 Invariance to Scaling

To verify the invariance to scaling the AR parameters of the test objects were computed and compared for scaling of the object from 0.97 to 0.86. The results of this experiments are given in Table 6.7 to Table 6.12.

In Table 6.7 we can see that the AR parameters for the first object stay almost unchanged under various scaling ratios. The small changes of the AR parameters for the same sub-part can be tolerated by the classification algorithms.

Tables 6.8 to Table 6.12 show that AR parameters for Object 2 to Object 6 are very stable under the scaling ratios mentioned above.

Table 6.7: AR Parameters with Scaling for Object 1. Each row corresponds to the parameters of a sub-part scaled by a given factor. The object is scaled by a given factor ranging from 0.97 to 0.86.

Sub-part	Scaling	θ_1	θ_2	θ_3	θ_4	θ_5	$\frac{\sigma}{\sqrt{\beta}}$
1	1.00	-0.9424	0.0073	0.0884	0.0212	0.0473	2.4328
1	0.97	-0.9535	0.0501	0.0560	0.0635	0.0738	2.4763
1	0.94	-0.9535	0.0501	0.0560	0.0635	0.0738	2.4763
1	0.91	-0.9535	0.0501	0.0560	0.0635	0.0738	2.4763
1	0.89	-0.9535	0.0501	0.0560	0.0635	0.0738	2.4763
1	0.86	-0.9535	0.0501	0.0560	0.0635	0.0738	2.4763
2	1.00	-0.9110	0.0960	0.1221	0.1835	0.3157	3.6045
2	0.97	-0.9129	0.1010	0.1293	0.1832	0.3229	3.5577
2	0.94	-0.9129	0.1010	0.1293	0.1832	0.3229	3.5577
2	0.91	-0.9129	0.1010	0.1293	0.1832	0.3229	3.5577
2	0.89	-0.9129	0.1010	0.1293	0.1832	0.3229	3.5577
2	0.86	-0.9129	0.1010	0.1293	0.1832	0.3229	3.5577

Table 6.8: AR Parameters with Scaling for Object 2. Each row corresponds to the parameters of a sub-part scaled by a given factor. The object is scaled by a given factor ranging from 0.97 to 0.86.

Sub-part	Scaling	θ_1	θ_2	θ_3	θ_4	θ_5	$\frac{\alpha}{\sqrt{\beta}}$
1	1.00	-0.8187	0.2914	1.2210	2.8745	-1.7632	-0.6519
1	0.97	-0.8165	0.2679	1.2247	2.7321	-1.8225	-0.4268
1	0.94	-0.8165	0.2679	1.2247	2.7321	-1.8225	-0.4268
1	0.91	-0.8165	0.2679	1.2247	2.7321	-1.8225	-0.4268
1	0.89	-0.8165	0.2679	1.2247	2.7321	-1.8225	-0.4268
1	0.86	-0.8165	0.2679	1.2247	2.7321	-1.8225	-0.4268
2	1.00	-0.9706	0.0707	0.0362	0.0447	0.0530	2.1814
2	0.97	-0.9682	0.0334	0.0358	0.0387	0.0422	2.2840
2	0.94	-0.9682	0.0334	0.0358	0.0387	0.0422	2.2840
2	0.91	-0.9682	0.0334	0.0358	0.0387	0.0422	2.2840
2	0.89	-0.9682	0.0334	0.0358	0.0387	0.0422	2.2840
2	0.86	-0.9682	0.0334	0.0358	0.0387	0.0422	2.2840

Table 6.9: AR Parameters with Scaling for Object 3. Each row corresponds to the parameters of a sub-part scaled by a given factor. The object is scaled by a given factor ranging from 0.97 to 0.86.

Sub-part	Scaling	θ_1	θ_2	θ_3	θ_4	θ_5	$\frac{\sigma}{\sqrt{\beta}}$
1	1.00	-0.9518	0.0566	0.0366	0.0606	0.0382	1.9511
1	0.97	-0.9661	0.0358	0.0386	0.0420	0.0461	2.3089
1	0.94	-0.9661	0.0358	0.0386	0.0420	0.0461	2.3089
1	0.91	-0.9661	0.0358	0.0386	0.0420	0.0461	2.3089
1	0.89	-0.9661	0.0358	0.0386	0.0420	0.0461	2.3089
1	0.86	-0.9661	0.0358	0.0386	0.0420	0.0461	2.3089
2	1.00	-0.8837	0.1171	0.1581	0.2611	1.3434	0.0056
2	0.97	-0.8944	0.1270	0.1766	0.3026	1.6310	-1.3983
2	0.94	-0.8944	0.1270	0.1766	0.3026	1.6310	-1.3983
2	0.91	-0.8944	0.1270	0.1766	0.3026	1.6310	-1.3983
2	0.89	-0.8944	0.1270	0.1766	0.3026	1.6310	-1.3983
2	0.86	-0.8944	0.1270	0.1766	0.3026	1.6310	-1.3983
3	1.00	-0.9934	0.0066	0.0066	0.0068	0.0067	2.0017
3	0.97	-0.9949	0.0051	0.0052	0.0052	0.0053	2.0368
3	0.94	-0.9949	0.0051	0.0052	0.0052	0.0053	2.0368
3	0.91	-0.9949	0.0051	0.0052	0.0052	0.0053	2.0368
3	0.89	-0.9949	0.0051	0.0052	0.0052	0.0053	2.0368
3	0.86	-0.9949	0.0051	0.0052	0.0052	0.0053	2.0368

Table 6.10: AR Parameters with Scaling for Object 4. Each row corresponds to the parameters of a sub-part scaled by a given factor. The object is scaled by a given factor ranging from 0.97 to 0.86.

Sub-part	Scaling	θ_1	θ_2	θ_3	θ_4	θ_5	$\frac{\sigma}{\sqrt{\beta}}$
1	1.00	-0.8665	0.1760	0.2859	1.4298	1.3909	-2.7795
1	0.97	-0.8660	0.1716	0.2838	1.4320	1.3706	-2.3757
1	0.94	-0.8660	0.1716	0.2838	1.4320	1.3706	-2.3757
1	0.91	-0.8660	0.1716	0.2838	1.4320	1.3706	-2.3757
1	0.89	-0.8660	0.1716	0.2838	1.4320	1.3706	-2.3757
1	0.86	-0.8660	0.1716	0.2838	1.4320	1.3706	-2.3757
2	1.00	-0.8943	0.1267	0.1765	0.3025	1.6268	-1.8071
2	0.97	-0.8944	0.1270	0.1766	0.3026	1.6310	-1.3983
2	0.94	-0.8944	0.1270	0.1766	0.3026	1.6310	-1.3983
2	0.91	-0.8944	0.1270	0.1766	0.3026	1.6310	-1.3983
2	0.89	-0.8944	0.1270	0.1766	0.3026	1.6310	-1.3983
2	0.86	-0.8944	0.1270	0.1766	0.3026	1.6310	-1.3983
3	1.00	-0.9530	0.0495	0.0571	0.0627	0.0723	2.4373
3	0.97	-0.9535	0.0501	0.0560	0.0635	0.0738	2.4763
3	0.94	-0.9535	0.0501	0.0560	0.0635	0.0738	2.4763
3	0.91	-0.9535	0.0501	0.0560	0.0635	0.0738	2.4763
3	0.89	-0.9535	0.0501	0.0560	0.0635	0.0738	2.4763
3	0.86	-0.9535	0.0501	0.0560	0.0635	0.0738	2.4763
4	1.00	-0.9903	0.0380	0.0367	0.0312	0.0491	1.9038
4	0.97	-0.9949	0.0051	0.0052	0.0052	0.0053	2.0368
4	0.94	-0.9949	0.0051	0.0052	0.0052	0.0053	2.0368
4	0.91	-0.9949	0.0051	0.0052	0.0052	0.0053	2.0368
4	0.89	-0.9949	0.0051	0.0052	0.0052	0.0053	2.0368
4	0.86	-0.9949	0.0051	0.0052	0.0052	0.0053	2.0368

Table 6.11: AR Parameters with Scaling for Object 5. Each row corresponds to the parameters of a sub-part scaled by a given factor. The object is scaled by a given factor ranging from 0.97 to 0.86.

Sub-part	Scaling	θ_1	θ_2	θ_3	θ_4	θ_5	$\frac{\sigma}{\sqrt{\beta}}$
1	1.00	-0.9426	0.0626	0.0720	0.0852	0.1096	2.6698
1	0.97	-0.9428	0.0627	0.0723	0.0857	0.1060	2.6544
1	0.94	-0.9428	0.0627	0.0723	0.0857	0.1060	2.6544
1	0.91	-0.9428	0.0627	0.0723	0.0857	0.1060	2.6544
1	0.89	-0.9428	0.0627	0.0723	0.0857	0.1060	2.6544
1	0.86	-0.9428	0.0627	0.0723	0.0857	0.1060	2.6544
2	1.00	-0.9256	0.0823	0.1019	0.1339	0.1882	3.0953
2	0.97	-0.9258	0.0839	0.1022	0.1324	0.1907	3.0573
2	0.94	-0.9258	0.0839	0.1022	0.1324	0.1907	3.0573
2	0.91	-0.9258	0.0839	0.1022	0.1324	0.1907	3.0573
2	0.89	-0.9258	0.0839	0.1022	0.1324	0.1907	3.0573
2	0.86	-0.9258	0.0839	0.1022	0.1324	0.1907	3.0573

Table 6.12: AR Parameters with Scaling for Object 6. Each row corresponds to the parameters of a sub-part scaled by a given factor. The object is scaled by a given factor ranging from 0.97 to 0.86.

Sub-part	Scaling	θ_1	θ_2	θ_3	θ_4	θ_5	$\frac{\sigma}{\sqrt{\beta}}$
1	1.00	-0.9579	0.0128	0.0013	0.0120	0.0416	2.4102
1	0.97	-0.9661	0.0358	0.0386	0.0420	0.0461	2.3089
1	0.94	-0.9661	0.0358	0.0386	0.0420	0.0461	2.3089
1	0.91	-0.9661	0.0358	0.0386	0.0420	0.0461	2.3089
1	0.89	-0.9661	0.0358	0.0386	0.0420	0.0461	2.3089
1	0.86	-0.9661	0.0358	0.0386	0.0420	0.0461	2.3089

6.2.2 Invariance to Rotation

To verify the invariance to rotation the AR parameters of the test objects were computed and compared for rotation of the object from 10° to 30°. The results of this experiments are given in Table 6.13 to Table 6.18.

In Table 6.13 we can see that the AR parameters for the first object stay almost unchanged under various degrees of rotation. The small changes of the AR parameters for the same sub-part can be tolerated by the classification algorithms.

Tables 6.14 to Table 6.18 show that AR parameters for Object 2 to Object 6 are very stable under the degrees of rotation mentioned above.

The sub-parts and AR parameters have been found to be stable under rotations varying from 0° to 180°.

Table 6.13: AR Parameters with Rotation for Object 1. Each row corresponds to the parameters of a sub-part at a given rotation. The object is rotated from 0 to 30 degrees in increments of 5 degrees.

Sub-part	Rotation	θ_1	θ_2	θ_3	θ_4	θ_5	$\frac{\sigma}{\sqrt{\beta}}$
1	0	-0.9424	0.0073	0.0884	0.0212	0.0473	2.4328
1	10	-0.9535	0.0501	0.0560	0.0635	0.0738	2.4763
1	15	-0.9535	0.0501	0.0560	0.0635	0.0738	2.4763
1	20	-0.9535	0.0501	0.0560	0.0635	0.0738	2.4763
1	25	-0.9535	0.0501	0.0560	0.0635	0.0738	2.4763
1	30	-0.9535	0.0501	0.0560	0.0635	0.0738	2.4763
2	0	-0.9110	0.0960	0.1221	0.1835	0.3157	3.6045
2	10	-0.9129	0.1010	0.1293	0.1832	0.3229	3.5577
2	15	-0.9129	0.1010	0.1293	0.1832	0.3229	3.5577
2	20	-0.9129	0.1010	0.1293	0.1832	0.3229	3.5577
2	25	-0.9129	0.1010	0.1293	0.1832	0.3229	3.5577
2	30	-0.9129	0.1010	0.1293	0.1832	0.3229	3.5577

Table 6.14: AR Parameters with Rotation for Object 2. Each row corresponds to the parameters of a sub-part at a given rotation. The object is rotated from 0 to 30 degrees in increments of 5 degrees.

Sub-part	Rotation	θ_1	θ_2	θ_3	θ_4	θ_5	$\frac{\hat{\alpha}}{\sqrt{\hat{\beta}}}$
1	0	-0.8187	0.2914	1.2210	2.8745	-1.7632	-0.6519
1	10	-0.8165	0.2679	1.2247	2.7321	-1.8225	-0.4268
1	15	-0.8165	0.2679	1.2247	2.7321	-1.8225	-0.4268
1	20	-0.8165	0.2679	1.2247	2.7321	-1.8225	-0.4268
1	25	-0.8165	0.2679	1.2247	2.7321	-1.8225	-0.4268
1	30	-0.8165	0.2679	1.2247	2.7321	-1.8225	-0.4268
2	0	-0.9706	0.0707	0.0362	0.0447	0.0530	2.1814
2	10	-0.9682	0.0334	0.0358	0.0387	0.0422	2.2840
2	15	-0.9682	0.0334	0.0358	0.0387	0.0422	2.2840
2	20	-0.9682	0.0334	0.0358	0.0387	0.0422	2.2840
2	25	-0.9682	0.0334	0.0358	0.0387	0.0422	2.2840
2	30	-0.9682	0.0334	0.0358	0.0387	0.0422	2.2840

Table 6.15: AR Parameters with Rotation for Object 3. Each row corresponds to the parameters of a sub-part at a given rotation. The object is rotated from 0 to 30 degrees in increments of 5 degrees.

Sub-part	Rotation	θ_1	θ_2	θ_3	θ_4	θ_5	$\frac{\sigma}{\sqrt{\beta}}$
1	0	-0.9518	0.0566	0.0366	0.0606	0.0382	1.9511
1	10	-0.9661	0.0358	0.0386	0.0420	0.0461	2.3089
1	15	-0.9661	0.0358	0.0386	0.0420	0.0461	2.3089
1	20	-0.9661	0.0358	0.0386	0.0420	0.0461	2.3089
1	25	-0.9661	0.0358	0.0386	0.0420	0.0461	2.3089
1	30	-0.9661	0.0358	0.0386	0.0420	0.0461	2.3089
2	0	-0.8837	0.1171	0.1581	0.2611	1.3434	0.0056
2	10	-0.8944	0.1270	0.1766	0.3026	1.6310	-1.3983
2	15	-0.8944	0.1270	0.1766	0.3026	1.6310	-1.3983
2	20	-0.8944	0.1270	0.1766	0.3026	1.6310	-1.3983
2	25	-0.8944	0.1270	0.1766	0.3026	1.6310	-1.3983
2	30	-0.8944	0.1270	0.1766	0.3026	1.6310	-1.3983
3	0	-0.9934	0.0066	0.0066	0.0068	0.0067	2.0017
3	10	-0.9949	0.0051	0.0052	0.0052	0.0053	2.0368
3	15	-0.9949	0.0051	0.0052	0.0052	0.0053	2.0368
3	20	-0.9949	0.0051	0.0052	0.0052	0.0053	2.0368
3	25	-0.9949	0.0051	0.0052	0.0052	0.0053	2.0368
3	30	-0.9949	0.0051	0.0052	0.0052	0.0053	2.0368

Table 6.16: AR Parameters with Rotation for Object 4. Each row corresponds to the parameters of a sub-part at a given rotation. The object is rotated from 0 to 30 degrees in increments of 5 degrees.

Sub-part	Rotation	θ_1	θ_2	θ_3	θ_4	θ_5	$\frac{\sigma}{\sqrt{\beta}}$
1	0	-0.8665	0.1760	0.2859	1.4298	1.3909	-2.7795
1	10	-0.8660	0.1716	0.2838	1.4320	1.3706	-2.3757
1	15	-0.8660	0.1716	0.2838	1.4320	1.3706	-2.3757
1	20	-0.8660	0.1716	0.2838	1.4320	1.3706	-2.3757
1	25	-0.8660	0.1716	0.2838	1.4320	1.3706	-2.3757
1	30	-0.8660	0.1716	0.2838	1.4320	1.3706	-2.3757
2	0	-0.8943	0.1267	0.1765	0.3025	1.6268	-1.8071
2	10	-0.8944	0.1270	0.1766	0.3026	1.6310	-1.3983
2	15	-0.8944	0.1270	0.1766	0.3026	1.6310	-1.3983
2	20	-0.8944	0.1270	0.1766	0.3026	1.6310	-1.3983
2	25	-0.8944	0.1270	0.1766	0.3026	1.6310	-1.3983
2	30	-0.8944	0.1270	0.1766	0.3026	1.6310	-1.3983
3	0	-0.9530	0.0495	0.0571	0.0627	0.0723	2.4373
3	10	-0.9535	0.0501	0.0560	0.0635	0.0738	2.4763
3	15	-0.9535	0.0501	0.0560	0.0635	0.0738	2.4763
3	20	-0.9535	0.0501	0.0560	0.0635	0.0738	2.4763
3	25	-0.9535	0.0501	0.0560	0.0635	0.0738	2.4763
3	30	-0.9535	0.0501	0.0560	0.0635	0.0738	2.4763
4	0	-0.9903	0.0380	0.0367	0.0312	0.0491	1.9038
4	10	-0.9949	0.0051	0.0052	0.0052	0.0053	2.0368
4	15	-0.9949	0.0051	0.0052	0.0052	0.0053	2.0368
4	20	-0.9949	0.0051	0.0052	0.0052	0.0053	2.0368
4	25	-0.9949	0.0051	0.0052	0.0052	0.0053	2.0368
4	30	-0.9949	0.0051	0.0052	0.0052	0.0053	2.0368

Table 6.17: AR Parameters with Rotation for Object 5. Each row corresponds to the parameters of a sub-part at a given rotation. The object is rotated from 0 to 30 degrees in increments of 5 degrees.

Sub-part	Rotation	θ_1	θ_2	θ_3	θ_4	θ_5	$\frac{\sigma}{\sqrt{\beta}}$
1	0	-0.9426	0.0626	0.0720	0.0852	0.1096	2.6698
1	10	-0.9428	0.0627	0.0723	0.0857	0.1060	2.6544
1	15	-0.9428	0.0627	0.0723	0.0857	0.1060	2.6544
1	20	-0.9428	0.0627	0.0723	0.0857	0.1060	2.6544
1	25	-0.9428	0.0627	0.0723	0.0857	0.1060	2.6544
1	30	-0.9428	0.0627	0.0723	0.0857	0.1060	2.6544
2	0	-0.9256	0.0823	0.1019	0.1339	0.1882	3.0953
2	10	-0.9258	0.0839	0.1022	0.1324	0.1907	3.0573
2	15	-0.9258	0.0839	0.1022	0.1324	0.1907	3.0573
2	20	-0.9258	0.0839	0.1022	0.1324	0.1907	3.0573
2	25	-0.9258	0.0839	0.1022	0.1324	0.1907	3.0573
2	30	-0.9258	0.0839	0.1022	0.1324	0.1907	3.0573

Table 6.18: AR Parameters with Rotation for Object 6. Each row corresponds to the parameters of a sub-part at a given rotation. The object is rotated from 0 to 30 degrees in increments of 5 degrees.

Sub-part	Rotation	θ_1	θ_2	θ_3	θ_4	θ_5	$\frac{\sigma}{\sqrt{\beta}}$
1	0	-0.9579	0.0128	0.0013	0.0120	0.0416	2.4102
1	10	-0.9647	0.0535	0.0306	0.0327	0.0666	2.2035
1	15	-0.9661	0.0427	0.0335	0.0360	0.0635	2.2681
1	20	-0.9664	0.0423	0.0383	0.0382	0.0500	2.2905
1	25	-0.9664	0.0414	0.0403	0.0346	0.0507	2.2975
1	30	-0.9664	0.0403	0.0385	0.0376	0.0490	2.3019

The experiments suggest that the differential capabilities of an AR filter of order 5 are adequate for the purpose of hypotheses generation. The AR parameters for the various scenes shown in Figure 6.36 to Figure 6.40 are listed in Table 6.19 to Table 6.23. The AR parameters of the scene sub-parts can be correlated to the corresponding sub-parts in the original objects.

Table 6.19: AR Parameters for Scene 1. Each row corresponds to the AR parameters from a given sub-part.

Sub-part	θ_1	θ_2	θ_3	θ_4	θ_5	$\frac{\sigma}{\sqrt{\beta}}$
1	-0.9124	0.1014	0.1206	0.1801	0.3303	3.6109
2	-0.8937	0.1215	0.1753	0.2990	1.6298	-1.2355
3	-0.9356	0.0746	0.0846	0.1040	0.1361	2.8133

Table 6.20: AR Parameters for Scene 2. Each row corresponds to the AR parameters from a given sub-part.

Sub-part	θ_1	θ_2	θ_3	θ_4	θ_5	$\frac{\sigma}{\sqrt{\beta}}$
1	-0.9133	0.0990	0.1415	0.1870	0.3100	3.4860
2	-0.9432	0.0659	0.0716	0.0935	0.1117	2.5994
3	-0.9353	0.0716	0.0870	0.1015	0.1353	2.7801
4	-0.8159	0.2671	1.2172	2.8211	-1.7791	-0.4497
5	-0.9951	0.0146	0.0347	0.0164	0.0322	1.9516

Table 6.21: AR Parameters for Scene 3. Each row corresponds to the AR parameters from a given sub-part.

Sub-part	θ_1	θ_2	θ_3	θ_4	θ_5	$\frac{\sigma}{\sqrt{\beta}}$
1	-0.9569	0.0363	0.0484	0.0538	0.0713	2.4272
2	-0.8660	0.1715	0.2836	1.4304	1.3760	-2.7436
3	-0.9122	0.0966	0.1286	0.1817	0.3184	3.4930
4	-0.9486	0.0549	0.0619	0.0827	0.0874	2.5011
5	-0.8938	0.1261	0.1609	0.3138	1.6328	-1.4450
6	-0.8657	0.1687	0.2828	1.4316	1.3636	-2.4572

Table 6.22: AR Parameters for Scene 4. Each row corresponds to the AR parameters from a given sub-part.

Sub-part	θ_1	θ_2	θ_3	θ_4	θ_5	$\frac{\sigma}{\sqrt{\beta}}$
1	-0.8657	0.1700	0.2827	1.4298	1.3736	-2.5413
2	-0.9506	0.0617	0.0813	0.0841	0.0900	2.4806
3	-0.9128	0.1010	0.1292	0.1830	0.3225	3.5005
4	-0.9256	0.0840	0.0986	0.1315	0.1890	3.0506
5	-0.9128	0.1009	0.1292	0.1830	0.3226	3.5141
6	-0.9495	0.0645	0.0624	0.0842	0.0884	2.5338
7	-0.8944	0.1270	0.1766	0.3024	1.6296	-1.2238
8	-0.9258	0.0862	0.0999	0.1291	0.1927	3.0827
9	-0.9128	0.1010	0.1292	0.1831	0.3226	3.5952

Table 6.23: AR Parameters for Scene 5. Each row corresponds to the AR parameters from a given sub-part.

Sub-part	θ_1	θ_2	θ_3	θ_4	θ_5	$\frac{\sigma}{\sqrt{\beta}}$
1	-0.8672	0.1813	0.2886	1.4309	1.4026	-2.5363
2	-0.9464	0.0460	0.0513	0.0557	0.0824	2.6145
3	-0.9124	0.0991	0.1265	0.1812	0.3233	3.5178
4	-0.9346	0.0638	0.0837	0.1029	0.1346	2.8172
5	-0.8937	0.1255	0.1749	0.2999	1.6131	-0.8414
6	-0.8659	0.1715	0.2833	1.4281	1.3838	-1.9778

6.2.3 Stability of the AR scheme

The AR parameters generated show a very good stability of the autoregressive method as applied to object modeling through subpart decomposition; the sub-part parameters stay almost unchanged under rotation and scaling. This stability can only be maintained within a given threshold: if the scaling factor is increased to more than 0.7 the object starts loosing some of its features and the boundary shape is changed. We are unable in most cases to extract the same sub-parts and hence the AR parameters for these sub-parts cannot be generated. The scaling factor also affects the number of points for a given sub-part and if the number is less than 64 the generated AR parameters do not have the same stability or validity.

The stability of the AR parameters can also be affected by the stability of the sub-part detection process which is in turn totally dictated by the dominant points of the boundary. So the modeling process can be affected by a number of factors including the quantization noise, the aspect ratio of the camera, the lighting, the dominant point detection, and the sub-part detection.

6.3 Identification of Overlapping Objects

The final phase in the process is the recognition of objects in scenes where they are overlapping other objects. This phase, as it was described in the previous chapter, consists of three operations: hypotheses generation where the AR parameters of the object and the AR parameters of the stored sub-parts are compared to generate a list of sub-parts that could be in the scene. Then there is the hypotheses pruning phase where some tests are conducted to eliminate sub-parts that are considered as obvious mis-matches, due to area and perimeter limitations or other constraints.

Then there is the hypotheses verification where the selected sub-parts are mapped onto their hypothesized counter-parts in the scene and the quality of the match between the model sub-part and the scene sub-part is assessed.

Figure 6.34 shows a scene of the decomposition of an object with a missing part. We can see that the decomposition method enables us to find most of the original sub-parts for an object with a missing part. Figure 6.36 and Figure 6.37 show a scene of two overlapping objects after sub-part decomposition and Figures 6.36 to 6.40 illustrate the sub-part decomposition for scenes containing two to three overlapping objects. As can be seen from these images the sub-part detection process is capable of recovering the original sub-part or at least a good approximation of the original sub-part.

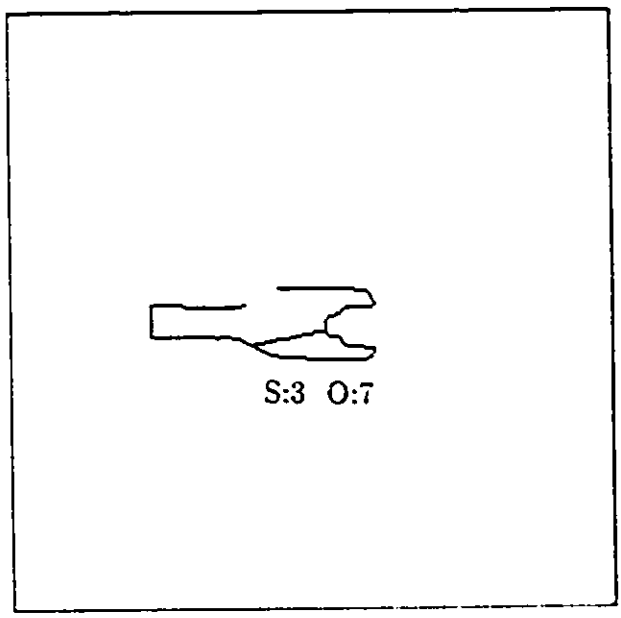

Figure 6.34: Decomposition of a scene containing an object with a missing part.

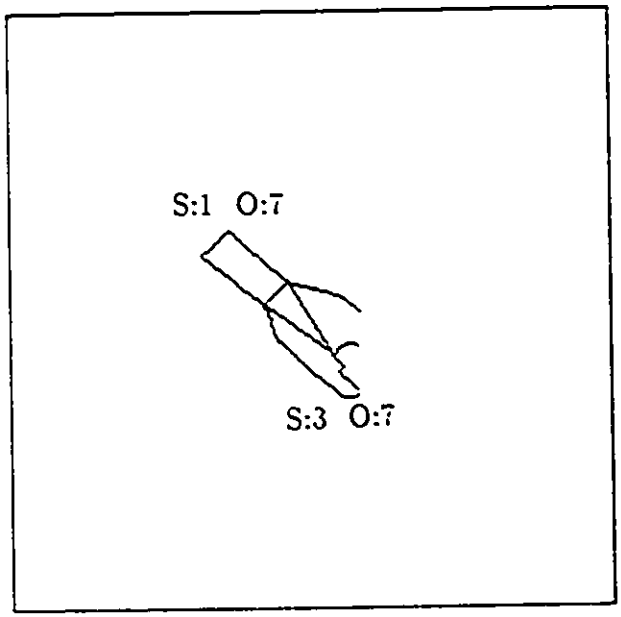

Figure 6.35: Decomposition of a scene containing an object with a missing part.

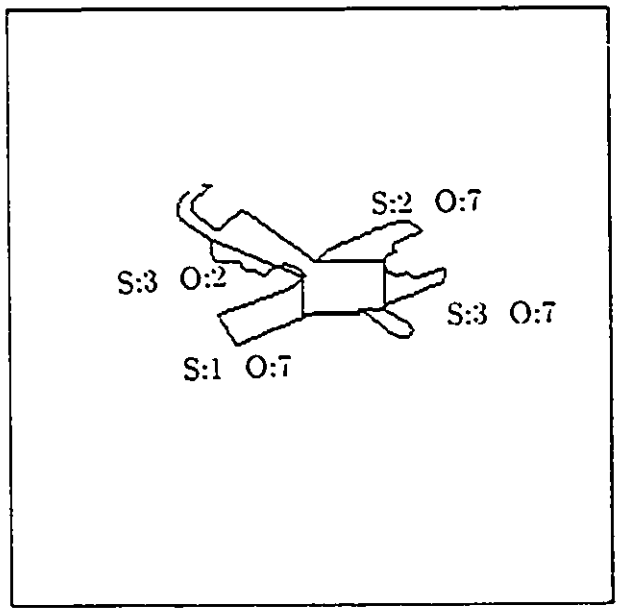

Figure 6.36: Decomposition of a scene with two overlapping objects. (Scene 1)

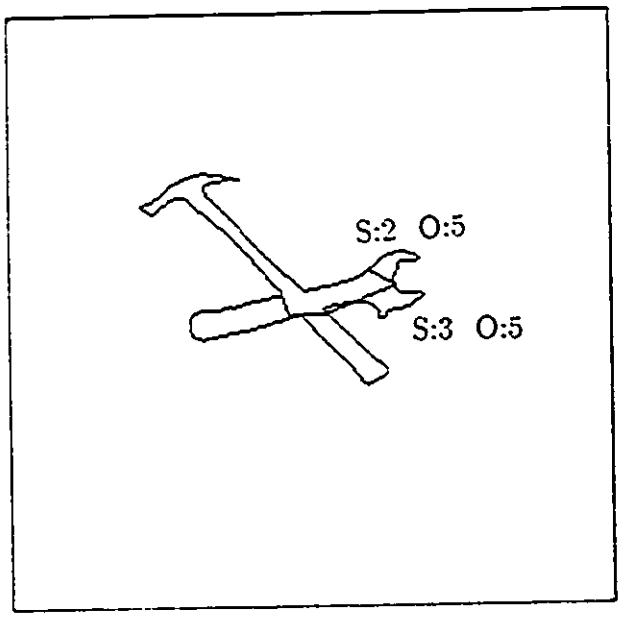

Figure 6.37: Decomposition of a scene with two overlapping objects. (Scene 2)

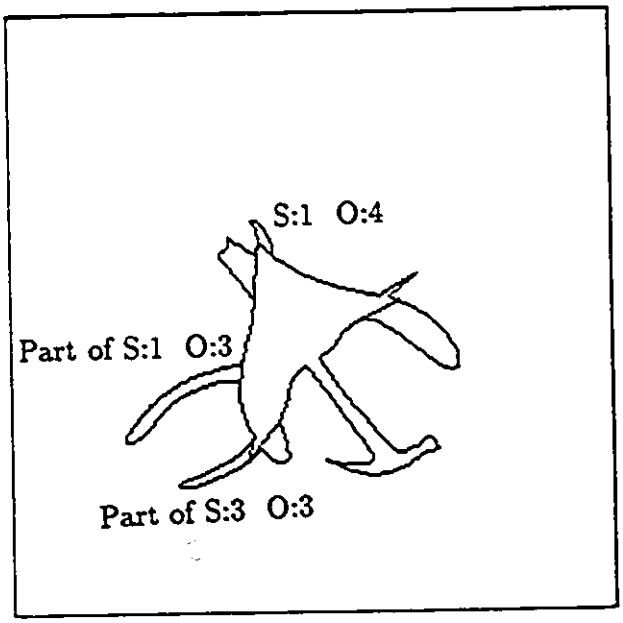

Figure 6.38: Decomposition of a scene with three overlapping objects. (Scene 3)

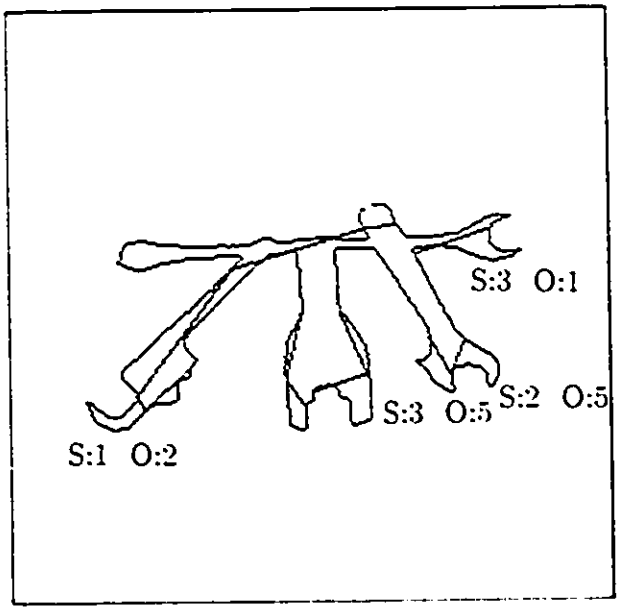

Figure 6.39: Decomposition of a scene with four overlapping objects. (Scene 4)

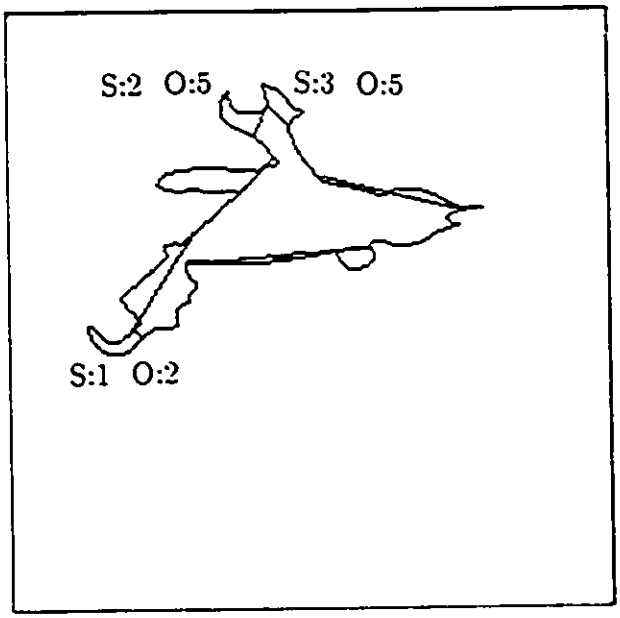

Figure 6.40: Decomposition of a scene with three overlapping objects. (Scene 5)

The above sub-parts are then used to generate AR vectors, which in turn serve to search for models' sub-parts with similar AR vectors from the database. A transformation is then applied to the points of the hypothesized models' sub-parts and a distance function is used to compute the degree of matching between the hypothesized sub-part and the sub-part of the scene. The transformation is updated using a Kalman filter like the one used by [4], the hypothesis with the highest score is kept and the recognition moves to the next sub-part. The code than implements this operation is listed in the file hypos.c in [12]. The results of the recognition process as applied to the objects in Figure 6.36 and Figure 6.37 are shown in the

The Hyper [4] like recognition engine implemented in the hypos.c takes combinations of edges and angles of the sub-parts from the object present in the scene and looks for the same combination of edges in the model. The results of the recognition process are given in Figure 6.41 and Figure 6.42. The system starts by a sub-part that matches one of the models' sub-parts (has the highest score: the handle), in this case it is be from the object present in Figure 6.34, then the system proceeds with the remaining sub-parts. In a similar way the recognition process works on the objects displayed in Figure 6.42. This experiment proves that the AR parameters together with the sub-part decomposition method for building object models is useful for object recognition in machine vision. However, we have to stress here that the Hyper like method was not the object of this thesis; once we can prove that the method decomposes, in a stable way, an object into its sub-parts, even when overlapped by other objects (as shown in Figure 6.36 to Figure 6.40) the validity of the method is demonstrated. The Hyper like technique was added to the test results for the sake of system completeness.

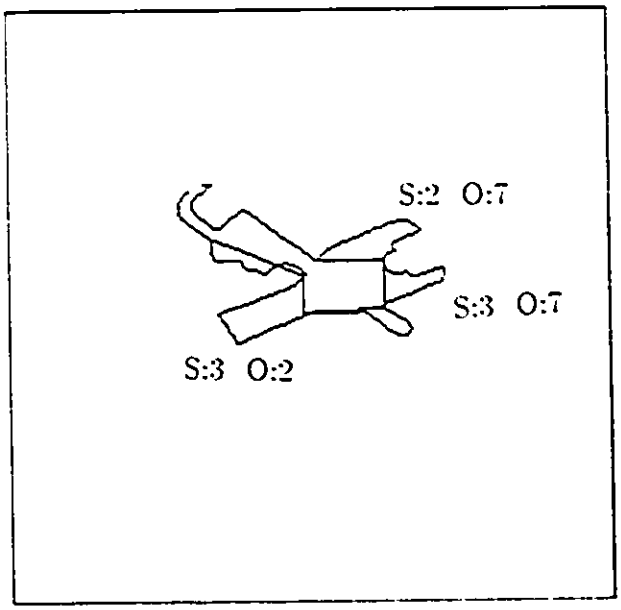

Figure 6.41: Results of the recognition process of the scene from Figure 6.36. The Recognition process is able to use three sub-parts that correspond to sub-parts in the object shown in Figure 6.34. After the recognition of this object most of the remaining edges in the scene belong to object 2.

6.4 Conclusion

This chapter showed the results of the different phases of the recognition process and the invariance of the algorithms employed under some degrees of rotation and scaling ratios. The sub-part decomposition algorithms were shown to work properly under various changes in the orientation and scaling of the object. The AR parameters were also stable and an AR model of order 5 was found to be adequate for the representation of the sub-parts for the purposes of this system. In order to avoid the inclusion of too many tables and figures only the results obtained with an AR filter of order 5 were included. The usefulness of the methods described in the previous chapters was illustrated through an example of recognition of a scene that

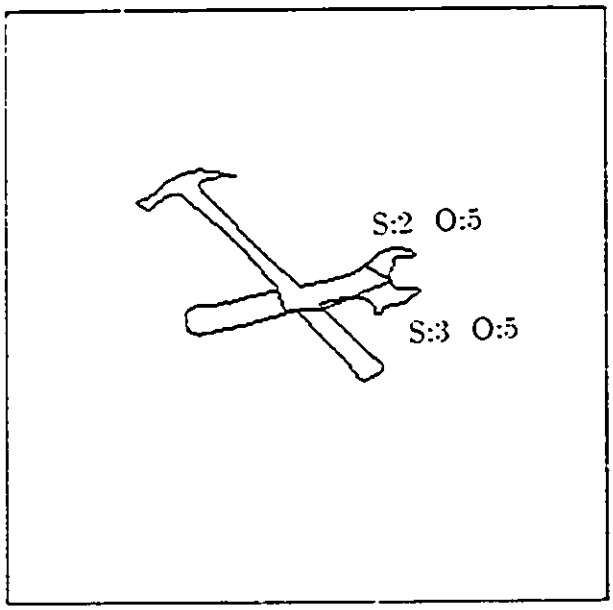

Figure 6.42: Results of the recognition process of the scene from Figure 6.37. The Recognition process is able to use two sub-parts that correspond to sub-parts in the object shown in Figure 6.5. After the recognition of this object most of the remaining edges in the scene belong to object 6.

contained a number of overlapping objects.

Chapter 7

Conclusion

Object recognition is considered one of the most challenging problems in computer engineering. It has received a tremendous interest from a large number of researchers. The algorithms that were proposed in the past were mostly designed and implemented for sequential machines and did not deal with applications in real time vision. This thesis dealt with conception and design of a hybrid object recognition system that was implemented on a parallel SIMD machine the AIS 5000 as well as on a SUN Sparcstation. The system was designed in an object based approach to ease its extension and the addition of new methods and algorithms in the future. The design focused on pursuing efficient methods for feature extraction, representation and recognition.

The work discussed in this thesis is a first implementation of the generic object recognition system. The first objective of the project was to design and implement a system that can learn and recognize objects in a number of situations determined by the combination of possible overlapping positions. A second objective was to have the system recognize objects in a short and predictable period of time so that

it can be applied in real time vision problems.

We have managed to fulfill the objectives of the project, since the vision system can deal with occlusion, the recognition time is short and predictable. It was shown to work efficiently for a number of objects under different orientations and scaling ratios. The main original features of the system that helped in the fulfillment of the goals were, as well as the contributions of the author to the solution of this problem are:

- The sub-part generation algorithms which simplifies the recognition process and makes sure that the parts fed to the recognition engine came from the same model. They also enable the system to deal with non-rigid objects and objects with moving parts (like scissors).
- The AR model used to describe sub-parts of the boundary and to build a model for each object sub-part is another original contribution of the author. It was proved of getting a concise and stable sub-part model. The AR parameters provide a number of advantages: an accurate description of shape using a small number of parameters, limited storage requirements, size and orientation invariance, and a short recognition time.
- A set of rules to detect occlusion and guide the recognition process.
- The object oriented approach used in the design and implementation helped in the maintenance and extension of the system, since it was relatively easy to make changes and add algorithms to the system. The system was first implemented on an SIMD machine, then it was implemented on the SUN Sparctstation. A number of algorithms were experimented with for the various stages of the recognition process: we tried three chain code generation algorithms, four dominant point detection algorithms, six sub-part generation

algorithms, three object recognition algorithms, and two object verification algorithms. The structure of the model was also changed a number of times to match the algorithms being used.

- A parallel implementation of the system. The system can also be configured to run on both a SIMD machine that will execute global operations that are parallel in nature like thresholding, filtering, chaincoding, polygonal approximation, etc. and a sequential machine that will execute operations that need random access to the objects boundary like sub-part decomposition and object recognition and verification. A parallel match was also implemented and tested.

- The chain code extraction algorithm was also enhanced and modified to run in parallel and to separate between the inner object contours and outer object contours.

For future research the following subjects which can improve the system to make it more generic, and might extend it to support more recognition algorithms:

- Symbolic processing of the shape to obtain a more intelligent sub-part decomposition. The symbolic processing of the shape will also enable the system to deal more effectively with 3-D objects and to make the pruning step in the recognition phase more effective.

- Adding rules for the coupling of scene sub-parts which will lead to a speed up in the object recognition and verification process.

- Improvement of the object modelling to make the system aware of the particular characteristics of non-rigid objects and varying shape (as in the problem of hand-writing recognition).

- Support of more robust and generic recognition algorithms that can deal with arbitrary representations and 3-D shapes like the iterative closest point algorithm [7].

- Introduce powerful indexing methods in object modelling so that the performance of the system does not degrade when the number of object models in the object database increases. One of the methods considered is structural indexing described in [42].

Bibliography

[1] H. Akaike, "Power spectrum estimation through autoregressive model fitting," *Ann. Inst. Stat. Math.*, vol. 21, pp. 407-419, 1969.

[2] H. Akaike, "A new look at statistical model identification," *IEEE Trans. Automatic Control*, vol. AC-19,pp. 716-723, Dec. 1974.

[3] N. Ansari and E.J. Delp, "Partial shape recognition: A landmark-based approach," *IEEE Trans. Pattern Anal. Machine Intell.* , vol. PAMI-12, no. 5, pp. 470-483, 1990.

[4] N. Ayache and O. D. Faugeras, " HYPER - A new approach for the recognition and positioning of two dimensional objects," *IEEE Trans. Pattern Anal. Machine Intell.* , vol. PAMI-8, no. 1, pp. 44 -54, 1986.

[5] H. Baird, *Model-Based Image Matching Using Location*. Cambridge, MA: MIT Press, 1986.

[6] S. Berman, P. Parikh and C. S. G. Lee, "Computer recognition of overlapping parts using a single camera," *Computer*, vol. 18, no. 3, pp. 70-80, 1985.

[7] P. J. Besl and N. McKay,"A method for registration of 3-D shapes," *IEEE Trans. Pattern Anal. Machine intell.* , vol. PAMI-14, no. 2, pp. 239-256, 1992.

[8] B. Bhanu and O. D. Faugeras, " Shape matching of two-dimensional objects," *IEEE Trans. Pattern Anal. Machine intell.* , vol. PAMI-6, no. 2, pp. 137-155, 1984.

[9] B. Bhanu and J. C. Ming, " Recognition of occluded objects: A cluster-structure algorithm," *Pattern Recognition* , vol. 20, no. 2, pp. 199-211, 1987.

[10] R. C. Bolles and R. A. Cain, " Recognizing and locating partially visible objects: The local-feature-focus method," *Int. J. Robotics Res.*, vol. 1, no. 3, pp. 57-82, 1982.

[11] R. T. Chin and C. R. Dyer, "Model based recognition in robot vision," *ACM Comput. Surv.* , vol. 18, no. 1, pp. 67-108, 1986.

[12] T. Damerji and D. Ionescu, *Recognition of Overlapping Object.* Technical Report No. 123, Dept. of Electrical Engineering, University of Ottawa, Ottawa, March 1993.

[13] Damerji, T., Karoui, B., Ionescu, D.: " Recognizing Partially Overlapped Objects" Proc. of the IEEE 1990 Canadian Conference on Electrical and Computer Engineering, Ottawa, 1990, pp 37.5.1-37.5.4.

[14] T. Damerji, D. Ionescu:."A Method for Subpart Decomposition for Overlapped Object Identification" in Proceedings of CCECE 93, Vancouver, September 14-17, 1993, pp.991-994.

[15] P. A. Devijver and J. Kittler, *Pattern Recognition: A Statistical Approach.* London: Prentice-Hall, 1982.

[16] S. R. Dubois and F. L. Glanz, "An autoregressive model approach to two-dimensional shape classification," *IEEE Trans. Pattern Anal. Machine Intell.* , vol. PAMI-8, no. 1, pp. 55-65, 1986.

[17] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*. New York: Wiley, 1973.

[18] H. F. Feng and and T. Pavlidis, "Decomposition of polygons into simpler components: Feature generation for syntactic pattern recognition," *IEEE Trans. Comput.* , vol. C-24, no. 6, pp. 636-650, 1975.

[19] K. B. Fung, C Andronic, T. Damerji, and D. Ionescu: "An Interactive System for Matching Optical Images" in Proceedings of IGARSS'93, Tokyo, August 18-21, 1993, pp.1342-1344.

[20] W. E. L. Grimson and T. Lozano-Pérez, "Localizing overlapping parts by searching the interpretation tree," *IEEE Trans. Pattern Anal. Machine Intell.* , vol. PAMI-9, no. 4, pp. 469-482, 1987.

[21] W. E. L. Grimson , "On the recognition of curved objects," *IEEE Trans. Pattern Anal. Machine Intell.* , vol. PAMI-11, no. 6, pp. 632-643, 1989.

[22] Grimson2 W. E. L. Grimson, *Object Recognition by Computer– The Role of Geometric Constraints*. Cambridge, MA: MIT Press, 1990.

[23] C. Guerra and G.G. Peroni, "A graph-theoretic method for decomposing two-dimensional polygonal shapes into meaningful parts," *IEEE Trans. Pattern Anal. Machine Intell.* , vol. PAMI-4, no. 4, pp. 405-408, 1982.

[24] A. Kalvin, E. Schonnberg, J. T. Schwartz, and M. Sharir, " Two-dimensional, model-based, boundary matching using footprints," *Int. J. Robotics Res.*, vol. 5, no. 4, pp. 38-55, 1986.

[25] R. L. Kashyap and R. Chellapa, "Stochastic models for closed boundary analysis: Representation and reconstruction," *IEEE Trans. Info. Theory*, vol. IT-27, no. 5, pp. 627-637, 1981.

[26] T. F. Knoll and R. C. Jain , " Recognizing partially visible parts using feature indexed hypotheses," *IEEE J. Robotics Automat.*, vol. RA-2, no. 1, pp. 3-13, 1986.

[27] M. W. Koch and R. L. Kashyap, " Using polygons to recognize and locate partially occluded objects," *IEEE Trans. Pattern Anal. Machine Intell.* , vol. PAMI-9, no. 4, pp. 483-494, 1987.

[28] J. Makhoul, "Linear Prediction: A tutorial review," *Proc. IEEE*, vol. 63, pp. 561-580, April 1975.

[29] R. Mehrotra and W. Grosky, " Shape matching utilizing indexed hypothesis generation and testing," *IEEE Trans. Robotics Automation*, vol. 5, no. 1, pp. 70-77, 1989.

[30] A. J. Nevins, "Region extraction from complex shapes," *IEEE Trans. Pattern Anal. Machine Intell.* , vol. PAMI-4, no. 5, pp. 500-511, 1982.

[31] E. Parzen, "Some recent advances in time series modeling," *IEEE Trans. Automatic Control*, vol. AC-21, pp. 723-730, Dec. 1974.

[32] T. Pavlidis and S. L. Horowitz, "Segmentation of plane curves," *IEEE Trans. Comp.* , vol. C-23, no. , pp. 860-870, 1974.

[33] T. Pavlidis, *Structural Pattern Recognition*. New York ,N.Y.: Springer Veralg, 1977.

[34] T. Pavlidis, "Algorithms for shape analysis of contours and waveforms," *IEEE Trans. Pattern Anal. Machine Intell.* , vol. PAMI-2, no. 4, pp. 301-312, 1980.

[35] K. E. Price, "Matching closed contours," in *Seventh International Conference on Pattern Recognition*, pp. 990-992, Montreal, Canada, July 30- Aug 2, 1984.

[36] J. Rissanen, "A universal prior for the integers and estimation by minimum description length," *Ann. Stat.*, vol. 11, pp. 417-431, 1983.

[37] W. S. Rutkowski, "Recognition of occluded shapes using relaxation," *Comput. Graphics Image Processing*, vol. 19, pp. 111-128, 1982.

[38] L. A. Schmitt and S. S. Wilson, "The AIS-5000 parallel processor," *IEEE Trans. Pattern Anal. Machine Intell.* , vol. PAMI-10, no. 10, pp. 320-330, 1988.

[39] J. T. Schwartz and M. Sharir, " Identification of obscured objects in two and three dimensions by matching noisy characteristic curves," *Int. J. Robotics Res.*, vol. 6, no. 2, pp. 29-44, 1987.

[40] G. S. Sebestyen, *Decision Making Processes in Pattern Recognition*. New York: Macmillan, 1962.

[41] L. G. Shapiro and R. M. Haralick, "Decomposition of two-dimensional shapes by graph-theoretic clustering," *IEEE Trans. Pattern Anal. Machine Intell.* , vol. PAMI-1, no. 1, pp. 10-20, 1979.

[42] F. Stein and G. Medioni, "Structural indexing: efficient 2-D object recognition," *IEEE Trans. Pattern Anal. Machine Intell.* , vol. PAMI-14, no. 12, pp. 1198-1204, 1992.

[43] C. Teh and R. T. Chin, "On the detection of dominant points on digital curves," *IEEE Trans. Pattern Anal. Machine Intell.* , vol. PAMI-11, no. 8, pp. 859-872, 1989.

[44] V. Torre and T. A. Poggio,, "On edge detection," *IEEE Trans. Pattern Anal. Machine Intell.* , vol. PAMI-8, no. 2, pp. 147-163, 1986.

[45] J. L. Turney, T.N. Mudge, and R.A. Volz. " Recognizing partially occluded parts," *IEEE Trans. Pattern Anal. Machine Intell.* , vol. PAMI-7, no. 4. pp. 420-421, 1985.

[46] K. Wall and P-E. Danielsson, "A fast method for polygonal approximation of digital curves," *Computer Graphics and Image Processing.* vol. 28, pp. 220-227, 1984.