

Portland State University

PDXScholar

Dissertations and Theses

Dissertations and Theses

11-2-1990

Decomposition of Measured Contours into Geometric Features for Dimensional Inspection

Devaraj Rajkumar
Portland State University

Follow this and additional works at: https://pdxscholar.library.pdx.edu/open_access_etds



Part of the [Mechanical Engineering Commons](#)

Let us know how access to this document benefits you.

Recommended Citation

Rajkumar, Devaraj, "Decomposition of Measured Contours into Geometric Features for Dimensional Inspection" (1990). *Dissertations and Theses*. Paper 4149.

<https://doi.org/10.15760/etd.6033>

This Thesis is brought to you for free and open access. It has been accepted for inclusion in Dissertations and Theses by an authorized administrator of PDXScholar. Please contact us if we can make this document more accessible: pdxscholar@pdx.edu.

AN ABSTRACT OF THE THESIS OF Devaraj Rajkumar for the Master of Science in Mechanical Engineering presented November 2, 1990.

Title: Decomposition of Measured Contours into Geometric Features for Dimensional Inspection.

APPROVED BY THE MEMBERS OF THE THESIS COMMITTEE:


Faryar Etesami, Chair


Hormoz Zareifar


Fu Li

Image processing systems used in Vision Assisted Dimensional Inspection usually output a set of boundary pixels representing the part edges. This boundary information must be divided into several subsets representing the various edges of the actual object, so that comparisons with the nominal part can be made.

The purpose of this project is to devise a method to divide the set of pixels obtained from the image processing system into subsets of pixels. Each of these subsets represent an edge in the contour of the actual object. This method must also detect transition points between the adjacent features. This project addresses only planar contours which are composed of straight and circular edges. Two new algorithms have been developed, the first algorithm detects the transition points involving straight edges and the second algorithm finds the transition points when circular features are involved. In addition, the measured features are also matched with their nominal

counterparts. The performance of these algorithms are demonstrated by simulated as well as images from the vision system.

DECOMPOSITION OF MEASURED CONTOURS INTO GEOMETRIC
FEATURES FOR DIMENSIONAL INSPECTION

by

DEVARAJ RAJKUMAR

A thesis submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE
in
MECHANICAL ENGINEERING

Portland State University
1990

TO THE OFFICE OF GRADUATE STUDIES:

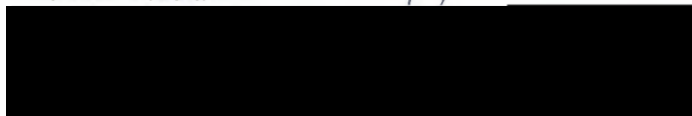
The members of the Committee approve the thesis of Devaraj Rajkumar presented November 2, 1990.



Faryar Etesami, Chair



Hormoz Zarefar



Fu Li

APPROVED:



Graig A. Spolek, Chair, Department of Mechanical Engineering



C. William Savery, Interim Vice Provost for Graduate Studies and Research

ACKNOWLEDGEMENTS

I wish to extend my heartfelt gratitude and appreciation:

To Professor Faryar Etesami for his encouragement and his ability to bring out the best in me. He has also been supportive and given me the freedom to be creative;

To the faculty and staff members of the Department of Mechanical Engineering especially Professor Hormoz Zarefar, Professor Herman Migliore, Debbie and Maxine for all they have done for me;

To Lorraine Duncan of Faculty Resource Center, who has helped me all the way in putting this presentation together with necessary presentation resources and encouragement;

To my parents, my brother and most of all Darcy for supporting my work and encouraging me during the difficult times;

And finally to Jim, Muthu, Selva, and all of my other friends for their valuable friendship.

Portland, Oregon

Devaraj Rajkumar

TABLE OF CONTENTS

	PAGE
ACKNOWLEDGEMENTS	iii
LIST OF TABLES	vi
LIST OF FIGURES	vii
CHAPTER	
I INTRODUCTION	1
1.1 Background	2
1.2 Objectives	6
II VISION SYSTEMS	7
2.1 Background	7
2.2 Equipment	7
2.3 Image processing in Dimensional Inspection	8
2.4 What is expected from a Vision System	12
III CONTOUR BREAKDOWN THEORY AND ALGORITHMS	13
3.1 Introduction	13
3.2 Existing Algorithms and the need for new algorithms	14
3.3 Algorithm-1	14
The Convex hull algorithm	16
The Width finding algorithm	19
3.4 Algorithm-2	19
3.5 Formal Definition of Edges	23
3.6 Algorithm-A	25
3.7 Algorithm-B	28
IV COMPARISON OF ACTUAL PART WITH THE NOMINAL PART	32
4.1 Introduction	32
4.2 Algorithm-C	33
4.3 Algorithm-D	37
4.4 Algorithm-E	38

	Theorem 1	42
V	PERFORMANCE AND ROBUSTNESS OF THE ALGORITHMS	45
	5.1 Performance on Simulated Contours	45
	5.2 Performance with Actual Contours	45
	5.3 Robustness of the Algorithms	47
	Type-1 error	50
	Type-2 error	50
	Type-3 error	52
VI	CONCLUSIONS AND FUTURE RESEARCH PROSPECTS	55
	REFERENCES	56

LIST OF TABLES

TABLE	PAGE
I Actual Part Feature Table (APFT)	36
II Nominal Part Feature Table (NPFT)	36

LIST OF FIGURES

FIGURE	PAGE
1. Dimensional Inspection System	3
2. Modules of Dimensional Inspection Software	4
3. Inspection System Components	9
4. Types of Noise	9
5. Decision Model	11
6. Simple Contour	15
7. Performance of Algorithm-1 with small W_S	17
8. Performance of Algorithm-1 with large W_S	18
9. Performance of Algorithm-2 with small W_S	21
10. Performance of Algorithm-2 with large W_S	22
11. Low Frequency Noise Edge	24
12. Step-A2 of Algorithm-A	27
13. Step-B1 of Algorithm-B	30
14. Step-B2 of Algorithm-B	31
15. Contours for Algorithm-C	34
16. Turn Angle between Vectors	39
17. Algorithm-E	41
18. Possible Turn Vectors	44
19. Unique Turn Vector Location	44
20. Actual Contour	46
21. After execution of Algorithm-A	48
22. After execution of Algorithm-B	49
23. Type-I error	51
24. Type-II error	51
25. Type-III (a) error	54
26. Type-III (b) error	54

CHAPTER I

INTRODUCTION

Portland State University is one of the leading universities involved in research on assembly verification of features by simulation and soft gaging or automated dimensional inspection. In order to perform inspection, the data corresponding to the features of the nominal part have to be compared to the data obtained from the actual part.

Recent technological advances have made it possible to transfer the dimensions and tolerances of a part from the CAD (Computer Aided Design) database to the CAM (Computer Aided Manufacturing) database with proper interfaces. In a CIM (Computer Integrated Manufacturing) environment, the product flow cannot be interrupted and stalled to perform inspection. Coordinate measurement machines are accurate but have a slow processing time. They are expensive and parts have to be stalled for the process of inspection. The above factors create "islands of automation"[10], thus making CIM less effective. One alternative that can be used in some situations is vision systems. Vision systems often eliminate stalling, reduce processing time and are also less expensive.

Presently several fields of engineering are using Vision Systems for quality control purposes. For example, in the electronic industry, most of the chip insertion machines are equipped with vision systems. A vision system serves two purposes. First, it enables a machine to set its "Home Position" or "0,0 position", by matching the crosswire on the PC board with the location specified by the database. Second, the vision system checks for defective chips which do not contain the right number of pins as in the database. Vision systems are also used to inspect automotive parts, guide welding torches in sheet metal fabrication, sealing industries and beverage

industries. New cameras come with higher resolutions, thus making vision systems more accurate. Vision systems can be used to inspect various features of an object to meet specified tolerances.

Figure 1 shows a part on a conveyor belt being transported below the camera. As soon as the part is right below the camera, the vision system is actuated to grab a frame. This image is processed by the image processor and the result is a contour of the actual object represented by a pixel set. This pixel set has to be processed based on the information from the CAD database, and then it must be compared with the nominal object of the same type. A prototype software to perform this operation is made up of several modules. These modules, which are shown in Figure 2, are image capturing, image processing, contour breakdown and feature extraction, feature comparison, and tolerance verification. This project addresses developing the contour breakdown and feature extraction, and the feature comparison modules. The work of tolerance comparison is a future research project.

1.1 BACKGROUND

Extensive research literature exist on computational geometry and image processing fields. For inspection purposes these fields go hand in hand. In order to deduce information from the pixel set obtained from image processing, techniques from computational geometry are used. The pixel set can be assumed to be a point set and by implementing suitable algorithms from computational geometry, the required information can be obtained. On-going research in computational geometry has led to the development of several methods to find convex hulls and to find minimum width rectangles that contain a point set. Similarly, image processing research has led to the development of many new methods to process images. Many books have been published with extensive discussions on these image processing methods and the theory. Recent results (one to two year old) are found in the journals. Most relevant to this project are the journals of Computer Vision, Graphics and Image Processing, and IEEE Transactions on Pattern Analysis and Machine

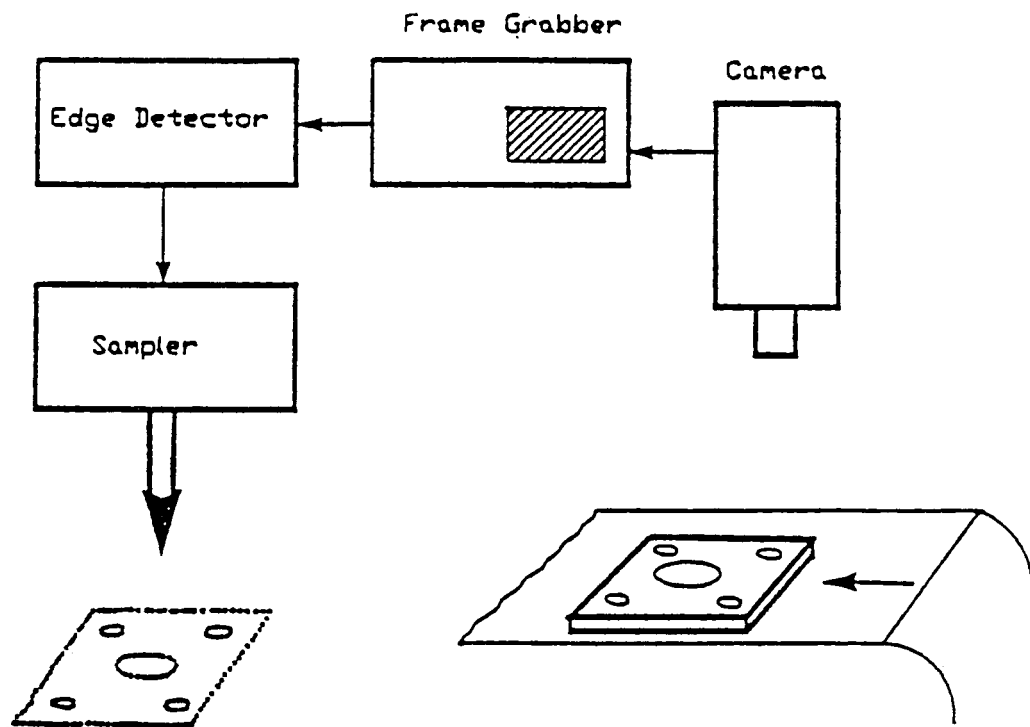


Figure 1. Dimensional inspection system. Source: [11].

Prototype Dimensional Inspection System

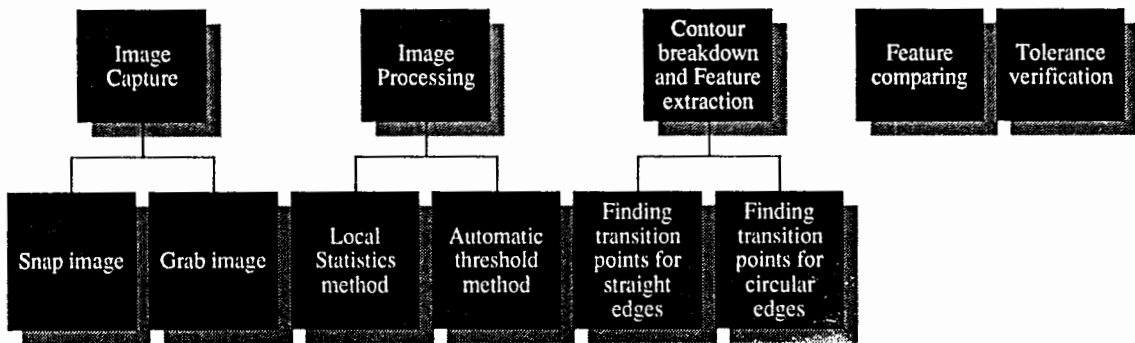


Figure 2. Modules of dimensional inspection software.

Intelligence.

The theory of image processing and the various techniques used have been dealt with in [4][6][7][8][9]. In the Master's Thesis presented by Ralf Koeppe[11] at Portland State University (predecessor of this work), the general behavior of the different image processing algorithms for dimensional inspection has been discussed. Koeppe's work focuses mainly on the accuracy of the image processing algorithms.

Recent research results in the field of computational geometry are found in Computer Vision, Graphics and Image Processing and in several other journals. Preparata and Shamos[1], in their book discuss the classic computational geometry theory and techniques. Imai Hiroshi and Iri Masao [5] present algorithms and theory for polygonal approximation of a curve. Algorithms in [5] are the forerunners for this research. In their publication, Imai and Iri discuss various techniques to breakdown a contour into a given number of edges or features.

This research project was undertaken to develop theory and methods to breakdown a set of points into acceptable features forming the contour. These algorithms also determine the transition points between the features. Furthermore, this research provides a method for comparing actual objects with nominal objects.

Some of the applications of vision systems involve detecting runways in complex airport scenes [13], classification of newspaper image blocks to enable the computer to understand a newspaper [14], extracting building in aerial images (could be used in defense to locate enemy headquarters) [15], representation, display and manipulation of 3D digital scenes and their medical applications [16], and establishing collision zones for obstacles moving with uncertainty [17]. Thus the application of vision systems has greatly increased in the recent years. One such application is dimensional inspection. The dream of automated dimensional inspection has come another step closer to reality through machine vision research.

1.2 OBJECTIVES

The objective of this project is to develop acceptable techniques to breakdown a contour into its constituent features for automated dimensional inspection. This project also focuses on developing methods for matching actual features with nominal features. The theories and algorithms developed apply only to point sets or pixel sets in 2D planes. The scope of this project includes the analysis of objects having straight and circular edges.

In order to achieve the objective, the following sub-objectives or sub-goals have been set forth.

1. Formalizing the definition of actual straight edges and circular edges.
2. Developing algorithms to determine the points of transition that exist between features.
3. Developing algorithms to classify the features into straight features and circular features.
4. Developing an algorithm to match the features of the actual part with the nominal part.
5. Developing a 2D prototype software system to implement all the above algorithms and to interface this system to a machine vision system.

The rationale behind using 2D planar features is two fold. First, the scope of this project can be narrowed down to implement a useful prototype system. Second, 2D systems are common in the industry. Extension of this work to 3D can be a future research project.

CHAPTER II

VISION SYSTEM USED IN THIS PROJECT

2.1 BACKGROUND

The camera used in this project is of a CCD type. CCD is the abbreviation of Charge Couple Device. A CCD is made up of an array of light sensitive elements. When these elements are exposed to light, an analog signal is generated. This analog signal is then converted to a digital signal. This information is stored in the frame memory or the frame buffer. Each element of the 2 Dimensional array is called a pixel, and is represented by an 8 bit value. The image used is of the gray scale type. Thus for any pixel, 256 possible gray values can be stored. A "0" value represents a perfect black pixel, while "255" represents a perfect white pixel. This information from the frame buffer is enough to perform image processing and several other manipulations on the image of the object. The information in the frame buffer can also be used to reconstruct and display the object on a monitor. By using a display monitor, the effect of various image processing operations can be seen. The image processing operations are performed by programs that use the information from the frame buffer.

2.2 EQUIPMENT

The camera used for this project is a Pulnix TS545. This camera has a 545 x 410 array of CCD elements and puts out a standard IEEE RS 170 signal with an aspect ratio of 3:4. The images are stored with a PC/AT frame grabber board [12]. The frame memory size is 1024 x 512 pixel of 8 bits per pixel[12]. Two images of 512 x 512 or one image of 640 x 480 pixel size can be

stored. The 640 x 480 pixel option is also called the "square pixel option". In the two image mode, the image grabber scales the object in the world coordinate system to fit the 512 x 512 image coordinate system. By using the 640 x 480 option to store the image, there is a one-to-one scaling between the RS 170 standard camera image and the image plane. This eliminates deformation of the object features due to aspect ratio difference. Due to this inherent advantage, all images used in this project are stored with the single image option. PCVISIONplus (imaging software) comes with sub-routines that interact with the frame grabber. These routines are written in "C" language. Some of the uses of these routines are to initialize, read and write pixels and processing of images. Ralf Koeppe[11] in his Master's Thesis has implemented several image processing algorithms into routines in "C" language, some of which are used in this project. These algorithms are discussed in the following section. Figure 3 shows the components of a digital image processing system.

2.3 IMAGE PROCESSING IN DIMENSIONAL INSPECTION

In order to use vision systems for dimensional inspection, the following questions must be answered.

1. How accurate are the vision systems?
2. What requirements must such a system meet? and
3. Which methods of image processing will provide these results?[11].

Ralf Koeppe[11] has addressed these questions in his Master's Thesis.

In his Master's Thesis, Koeppe has investigated the various methods of image processing with regards to dimensional inspection accuracy and determined the sources of noise that effect the accuracy. The sources of noise as shown in Figure 4 are

1. System noise
2. Texture edge noise and
3. Shadow

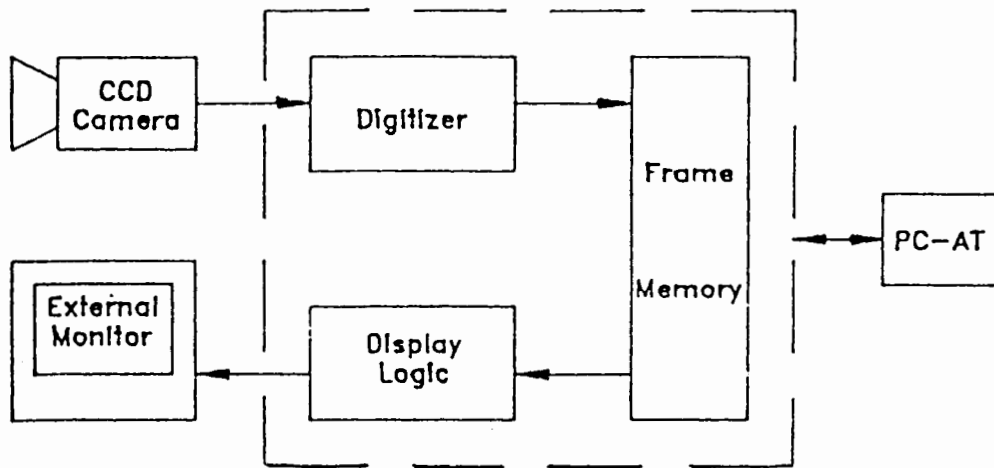


Figure 3. Inspection system components. Source: [11].

Noise	Origin	Type
System Noise	Hardware	additive random normally distributed with mean 0
Texture edge	Image	can be simulated by random normally distributed or by binary noise
Shadow	Lighting Conditions	not random noise

Figure 4. Types of noise. Source: [11].

Of the three, the first two sources of noise are randomly distributed, while the last one is not a random noise. On conducting several experiments, he suggests methods like image averaging and calculation of the mean of an array of pixels to eliminate the noise.

Koeppel also discusses several methods of image processing. He discusses various methods of point operations, local operations, and global operations with respect to dimensional inspection. He has also compared these techniques with one another on the basis of several important quantitative and qualitative parameters. Quantitative parameters include geometric stability, pixel stability, and repeatability of measurement. Qualitative parameters include noise occurrence and smoothness.

Figure 5 shows a decision model from [11] to select a particular method of image processing. Geometric stability (GS) is the relative percentage difference between the dimension of an object with noise and the dimension without noise found by imaging. The ideal value of GS ratio is zero, which implies that there is no noise. Pixel stability is the ratio of the number of edge pixels detected compared to the ideal edge. Pixel stability (PS) is also the inverted measure of the mean width of an edge. The ideal value of PS is one [11]. The standard deviation of the dimension of a feature on repeated detections is the measure of repeatability. The ideal value for repeatability is zero. By using the performance parameters discussed above and by defining a failure criteria for each of the parameters, an optimal method can be selected. The failure of a method is determined by the application and the accuracy demanded by the process of dimensional inspection. With a certain accuracy required, the failure criterion for pixel stability, geometric stability and repeatability can be established. By selecting the measurement conditions such as signal to noise ratio, additional selection criteria is found. From the available techniques, a subset of suitable methods can be found by testing the entire set against the failure criteria.

In this project, the image processing technique used is of the global operations type [11]. The rationale for choosing this technique is that the tested objects had sharp edges and that the global algorithms adequately work on such objects.

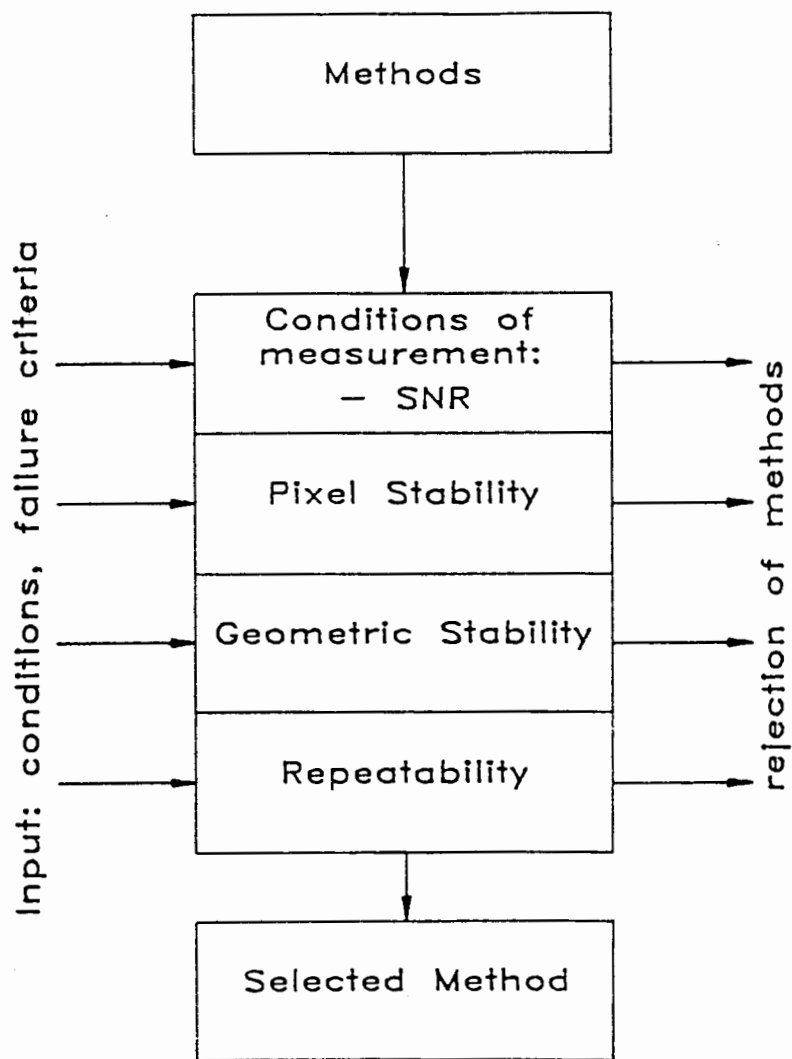


Figure 5. Decision model. Source: [11].

2.4 WHAT IS EXPECTED FROM A VISION SYSTEM

Henceforth the term vision system will be used to represent the image capturing system and the image processing system as a whole. In order to implement the theories and algorithms discussed in the forthcoming sections, the vision system is expected to give certain results. The results of the algorithms are very dependent on the results from the vision system.

The theories and algorithms developed in the course of this research project assumes that the contour obtained from the vision system is a closed contour. Therefore while performing image processing, care must be taken to see that the output is not a broken or an open contour.

In using Ralf Koeppe's subroutines, the input was manipulated to obtain a closed contour. The intensity threshold values were the key factors determining the output of the vision system.

CHAPTER III

CONTOUR BREAKDOWN THEORY AND ALGORITHMS

3.1 INTRODUCTION

When using a coordinate measurement machine, data obtained from the system is accurately divided into non-overlapping partitions each containing a specific nominal feature. Vision systems can be similarly treated by knowing where in the image plane each feature lies. But to use this technique with vision systems, the product should be loaded on a suitable fixture and stalled beneath the camera for imaging. Since this procedure is slow and tedious, this technique is inadequate when a vision system is used in an automated environment.

As a result of this research project, a solution to the problem mentioned above is presented. In principle, the task is done by finding straight edges through fitting subsets of the boundary into acceptable width rectangles, and for the circular edges, through fitting the boundary subsets into acceptable width rings.

Other possible methods to solve this problem may also be considered.

1. Spatial Positioning: A fixture can be used to hold the object in such a way that there is a fixed relationship (transformation) between the image points and actual points. This method is not used because it is slow and error prone.
2. Correlation Coefficients: This method is similar to the method used to find rectangle widths in this project. This technique however finds the points that match the correlation coefficients of straight edges and circular edges. This technique is not applied because it is not easy to define the correlation coefficients for a straight or a circular edge.
3. Contour Matching: The contour matching works on a principle that tries to match the

points of the actual part contour with the features of a nominal contour. Several operations like rotation and translation are performed on the nominal contour to find a good match. Since this method does not accurately locate the transition points, contour matching is not used.

4. Corner detection: This method relies on digital curvature to locate transition points. Whenever there is a large curvature between two consecutive points, a transition point can be detected. But when edges that have a gradual change in digital curvature are involved, this method fails to detect the transition point.

3.2 EXISTING ALGORITHMS AND THE NEED FOR NEW ALGORITHMS

Two algorithms for covering a sequence of points by rectangles are given in [5]. The first algorithm finds the minimum number of rectangles of width W_s covering a given sequence of points (point set), and the other algorithm finds a specified number of minimum width rectangles. The following is a brief description of the existing algorithms.

Consider a set of planar points representing the sampled boundary contour of a part as shown in Figure 6, that is composed of N straight edges. The objective is to devise an algorithm to decompose this point set into exactly N edges such that the transition points are "acceptable". For a perfect form contour the meaning of "acceptable" is easy to formalize, but imperfect contours are subjectively assessed.

3.3 ALGORITHM-1

This algorithm recursively divides the point set into halves until the width of the fitting rectangle gets smaller than a threshold value of W_s . This algorithm is commonly known as "divide-and-conquer". The worst case of computational complexity of the algorithm is computed as follows. The complexity of the convex hull finding is $O(hN)$ where h is the number of hull edges and is quite

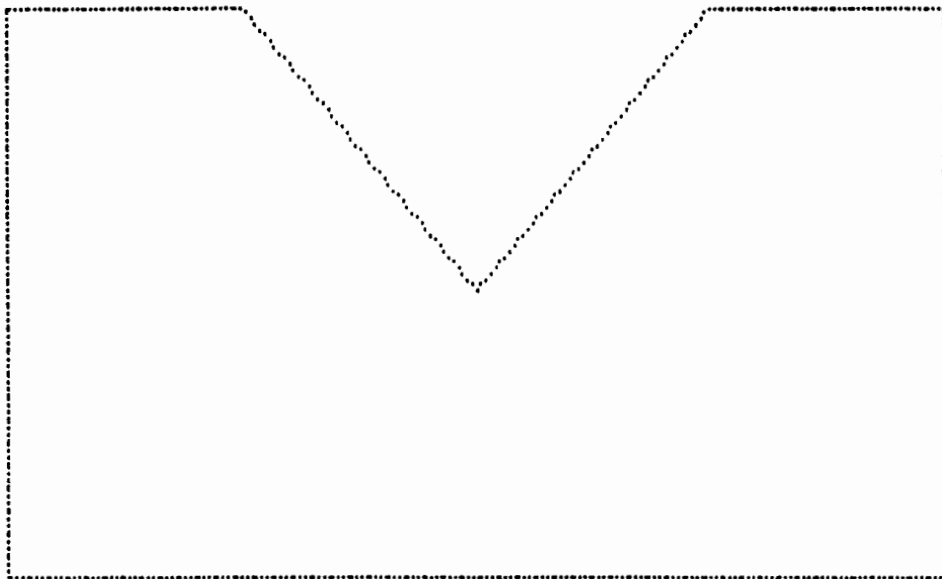


Figure 6. Simple contour.

small for the geometries under consideration. The value of h can be assumed to be a constant resulting in linear time complexity for the convex hull finding algorithm. The width finding algorithm works on the convex hull edges by testing each hull edge against all the hull vertices. This leads to a $O(h^2)$ computational complexity, for definition of computational complexity see [1]. Since h is small and assumed to be a constant, the whole width finding algorithm is an $O(N)$ operation. The worst case number of levels in the divide-and-conquer scheme is $\text{Log}N$, each can be performed in linear time. This leads to a computational complexity of $O(\text{Log}N)$ for the Algorithm-1.

Figure 7 shows a typical performance of this algorithm for a set of simulated points with some normal noise added and when W_s is relatively small. It can be seen, as highlighted by arrows, that the contour is excessively fragmented even along the straight regions.

Figure 8 shows the performance for a large W_s . Although the contour is less fragmented now, the margin by which the transition points are missed is quite large. The amount of normal random noise for both cases was normally and independently distributed with $(\mu=0.5, \sigma=0.2)$. The noise was applied independently to the X and the Y coordinates. For the two figures, W_s was set to 0.8 and 2.0 Figure 7 and Figure 8 respectively. The values 0.8 and 2.0 represent the dimensions measured in terms of the number of pixels. The execution time on a PC-AT 10 Mhz computer was approximately 2:15 minutes for $W_s=0.8$ and 1:30 for $W_s=2.0$.

The Convex hull algorithm

A convex hull of a set of points is the smallest region in 2D that contains every point such that if p and q are two arbitrarily selected points in the convex hull, then the entire line pq is within the convex hull. There are several techniques discussed in [1] for finding the convex hull vertices. Of these techniques, the Jarvis's march technique is used in this project due to its simplicity and low computational complexity. The algorithm starts at the lexicographically lowest point p_1 of the point set. This point is certainly a convex hull vertex, and the objective is to find the next vertex. This point p_2 is the one that has the least polar angle with respect to p_1 as origin. Similarly, the next point p_3 is found with respect to p_2 . The polar angle is measured with respect to the positive X axis

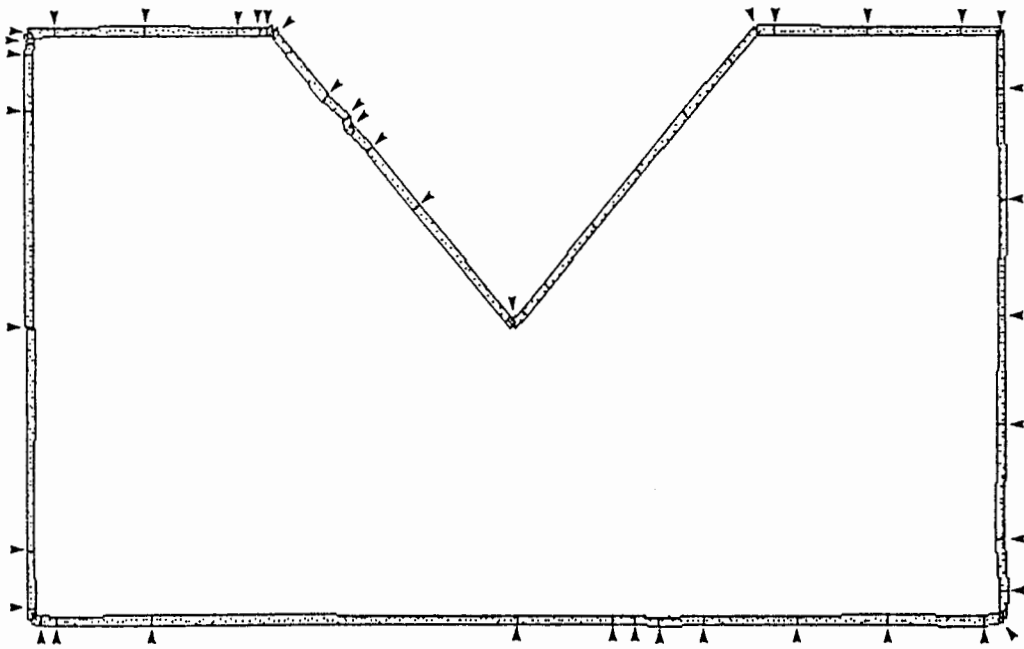


Figure 7. Performance of Algorithm-1 with small W_s .

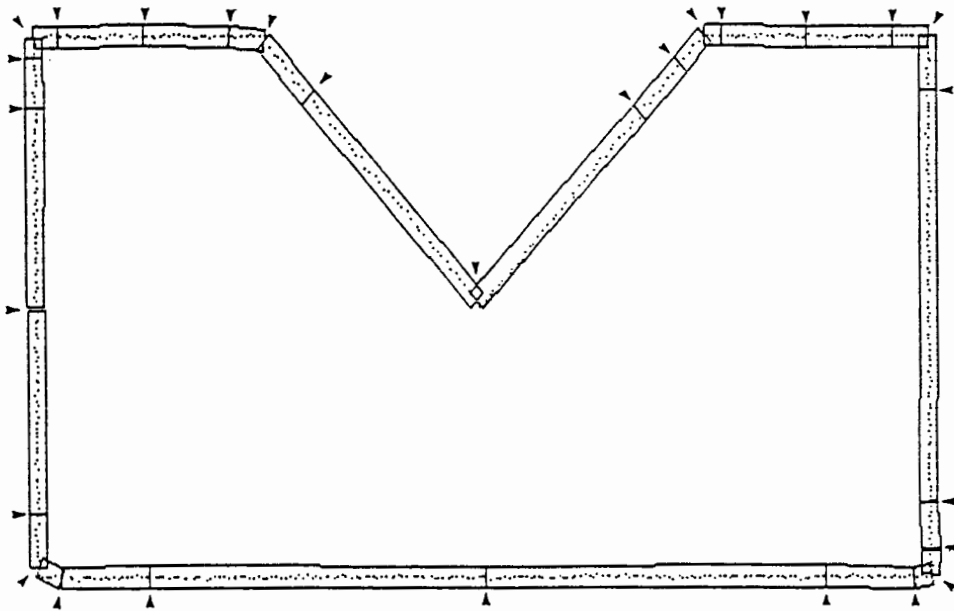


Figure 8. Performance of Algorithm-1 with large W_s .

until the lexicographically highest point is located. From this point onwards, the polar angles are measured with respect to the negative X axis due to the symmetry of the steps. Since the process of finding each vertex involves analyzing all the N points, the computational complexity of finding each vertex is $O(N)$. Therefore if there are h vertices in an object, the computational complexity of the algorithm is $O(hN)$.

The Width finding algorithm

The width finding algorithm finds the distance between a hull edge and every hull vertex. Since there are h edges, there are h vertices. Therefore to find the minimum distance for each edge, the computational complexity is $O(h)$. For h edges, the complexity amounts to $O(h \cdot h)$ which is $O(h^2)$.

3.4 ALGORITHM-2

This algorithm finds the minimum number of rectangles of size W_s that cover a point set and is adopted from a paper by Imai and Iri [5]. This algorithm starts from a point and attempts to find the largest set of points that can be fit within a rectangle of width W_s . A combination of Greedy algorithm and divide-and conquer is used. The Greedy method is one that "never undoes what it did earlier"[1]. Consider Figure 9. If there are N points that form the contour from A through T, the algorithm finds the minimum width rectangle for the entire set of points. Since the width is greater than W_s which is 0.8, the algorithm breaks down the contour into halves. In other words, the points under consideration are 1 through N/2 i.e A through L. The minimum width to contain these points is found. If this width is greater than W_s , the points are divided again into halves. Let us assume that the algorithm has divided the contour such that the points under consideration are from A through C. At this point the minimum width of the rectangle is still greater than W_s . So the algorithm divides the point set again into halves. Now the point set under consideration is from A through X. This time when the width of the rectangle is calculated, it is found that the width is less than W_s . This implies that there might be a transition point between X and C which contributes the entire

point set from A to this point, a rectangle of width less than W_s . This transition point, if it exists, must be found. So the algorithm divides the points starting from X+1 to the set A through X and determines the width of the containing rectangle. If this width is less than W_s , the next point set is included to the previous point set. This goes on until the point B is attained. When B+1 is included to the point set, the width is greater than W_s . Thus point B becomes the transition point for this width and this point becomes the starting point for the next point set. The next point set starts from B through T and the entire process of finding the transition point is repeated until point T becomes a transition point.

Assuming a linear convex hull finding time, it has been shown [5], that the overall worst case complexity of the Algorithm-2 is $O[KN]$, where K is the average number of points per rectangle in the final decomposition. The value of K critically depends on W_s . If W_s is small, the whole boundary gets fragmented into pairs of points and in such a case $K=N/2$.

Figure 9 shows such a situation, where the contour is excessively fragmented. But the transition points between the edges are accurately located. The data set is the same as the one used in Algorithm-1. If the value of the width W_s is set to a larger value, the boundary is decomposed into the correct number of edges but the transition points are badly missed by neighboring contours as shown in Figure 10. The execution time for the two previous cases increase to 3:05 and 1:40 minutes respectively.

Although not implemented in the course of this project, there is another relevant algorithm, which we call Algorithm-3. In Algorithm-3, the contour is decomposed such that the edges can be fit within at most "m" rectangles of minimum width. The solution given by this algorithm automatically forces the number of rectangles to be equal to the number of polygon edges, and finds the transition points such that minimum width rectangles are formed. This algorithm works well on edges that are straight even when some random noise is included.

The algorithm however does not work well on edges that have a low frequency distortion which is frequently present in manufactured parts. For example, Figure 11 contains points with the

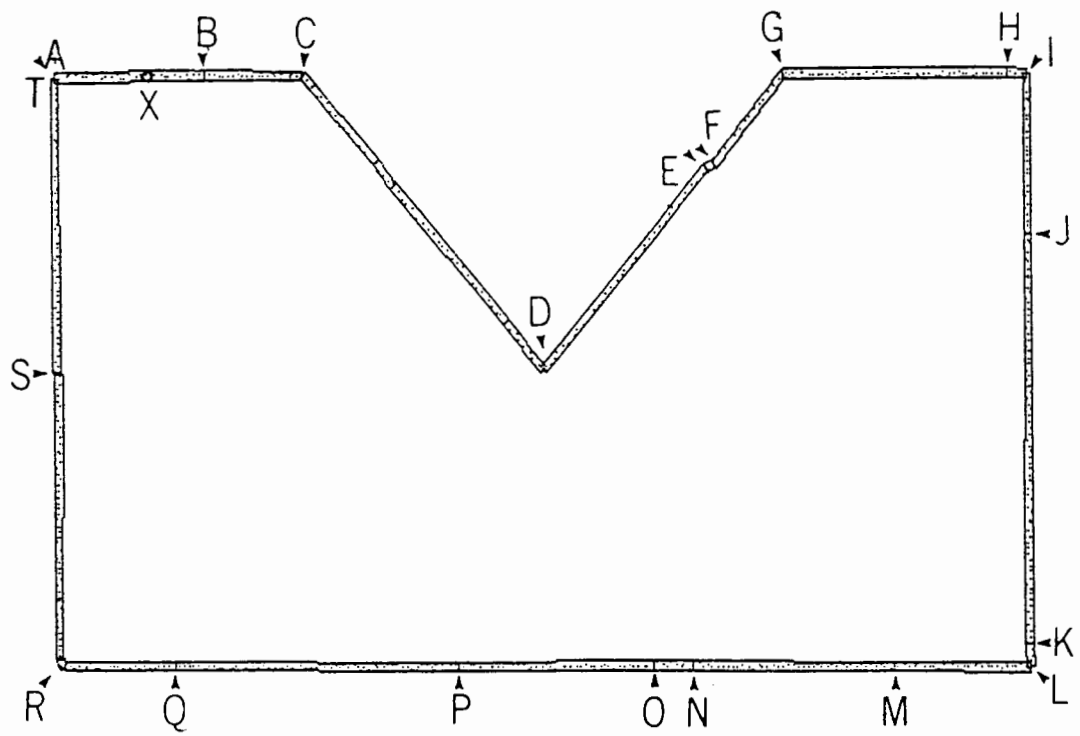


Figure 9. Performance of Algorithm-2 with small W_s .

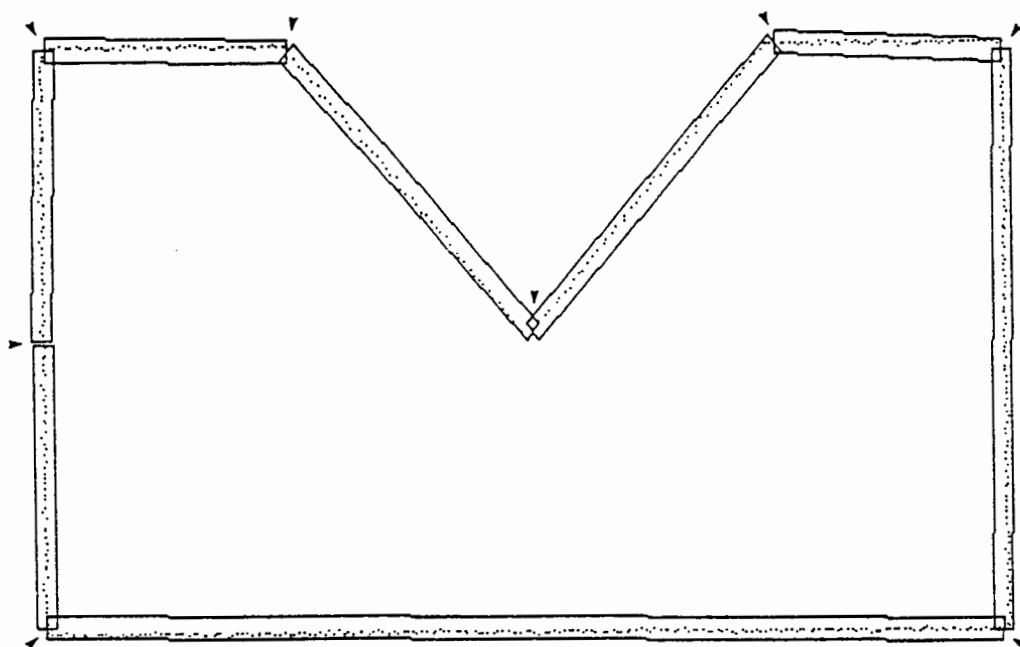


Figure 10. Performance of Algorithm-2 with large W_s .

low frequency noise exaggerated for clarity. The performance of Algorithm-3 leads to a decomposition shown in Figure 11 as rectangles, that clearly misses the true transition points. Another difficulty of using this algorithm, is the need to specify the maximum number of rectangles desired. Although this works well with polygons, when the contour includes circular features, it becomes very difficult to know what the best value of "m" might be.

Therefore it is clear that edges should not be defined as a set of points that can be fit within rectangles of a specified size due to low frequency distortion that are often present in edges. This leads to the formal definition of straight and circular edges which are presented in the next section.

3.5 FORMAL DEFINITION OF EDGES

Definition-1: A nominal contour, C_N , is a simple closed curve which can be decomposed into a set of straight and circular curves. These curves, or subsets of them, can be designated as nominal features F_{Ni} .

Definition-2: An actual contour, C_A , is a set of points with a corresponding nominal contour.

Definition-3: An actual straight edge, F_{Ai} is a subset of a C_A such that:

- 1- The subset F_{Ai} can be fit within a rectangle of infinite length and a width of W_g .
- 2- If the entire set C_A is fitted with rectangles of size $W_1 < W_g$, no member of F_{Ai} can belong to a rectangle of width W_1 unless every member of that rectangle is member of F_{Ai} .

Definition-4: An actual circular edge, F_{Ai} is a subset of a C_A such that:

- 1- F_{Ai} belongs to a subset $G_A \in C_A$ such that G_A does not include subsets that can be classified as straight edges.
- 2- The subset F_{Ai} can be fit within a ring of width of W_{gc} .
- 3- If the entire set G_A is fitted with rings of size $W_{1c} < W_{gc}$, no

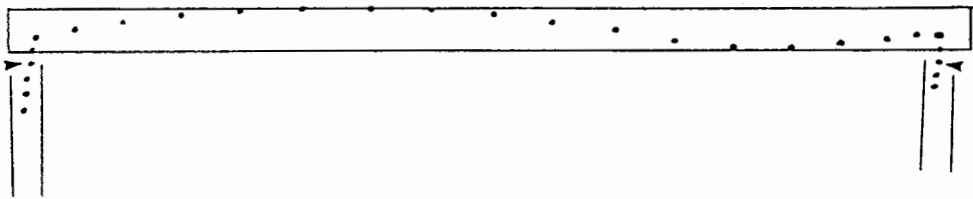


Figure 11. Low frequency noise edge.

member of F_{Ai} can belong to a ring of width W_{1c} unless every member of that ring is a member of F_{Ai} .

The main modification to the edge definition presented informally before, is that an edge is locally more accurate than globally due to the low frequency distortion of edges. This fact is true both for straight as well as circular edges or any other curve type. The algorithms in the following section are based on these definitions and will solve the edge finding problem.

3.6 ALGORITHM-A

The following algorithm is used to properly detect the transition points between straight edges, and transition point between straight and circular curves. This algorithm will not detect the transition point between two consecutive circular curves.

The algorithm can be sub-divided into two steps, Step-A1 and Step-A2.

Step-A1: The boundary contour is decomposed into subsets fitting within small-width rectangles of size W_1 as in the case of Algorithm-2.

Step-A2: The neighboring rectangles are merged into large width rectangles of size W_g .

By breaking down the boundary into small rectangles and then merging them with a larger width rectangle makes the process more localized. Therefore the transition points can be easily located by controlling W_g . Figure 9 shows the performance of this algorithm on Part-I after the execution of Step-A1. One can see that the boundary is broken up into small rectangles. Figure 12 shows the result after merging the small rectangles.

The principle behind merging is as follows. The algorithm starts with the first rectangle and adds the points of the second rectangle and finds the width of the containing rectangle. If this width

is less than W_g , then the points from the second rectangle are merged with the first rectangle point set to form a new point set. If width is greater than W_g , the algorithm concludes that the transition point must remain the same and repeats the same process starting from the second point set. In the case of Figure 9, point sets from A through B and B through C merged because the containing rectangle width was less than W_g . In the same example, when the algorithm tries to merge point sets Q through R and R through S, the algorithm fails because the containing rectangle width is greater than W_g . This makes the point R a transition point. It is seen that the algorithm performs well on finding the transition points as well as finding the correct number of edges. The performance of the next algorithm depends very much on the performance of this algorithm. If Algorithm-A makes a mistake in merging all the small rectangles making up a straight edge, the circular edge algorithm (Algorithm-B) fails.

The computational complexity of the algorithm is computed as follows. Let N_1 be the number of edges and m be the average number of points per edge. Two extreme cases can be identified. The first case is the "best case", and occurs when no edges can be merged. The second case can be termed the worst case and occurs when all edges can be merged. The computational complexity of the best case is of the order $\Omega(N_1m)$, which is equivalent to $\Omega(N)$. The worst case complexity is computed as follows:

complexity of merging the edges is= $O(m)+2*O(m)+.....+N_1*O(m)$

$$= (1+2+.....N_1) * O(m)$$

$$= N_1(N_1+1)/2 * O(m)$$

$$= O(m*N_1^2)$$

$$\text{Upper bound complexity} = O(N_1N)$$

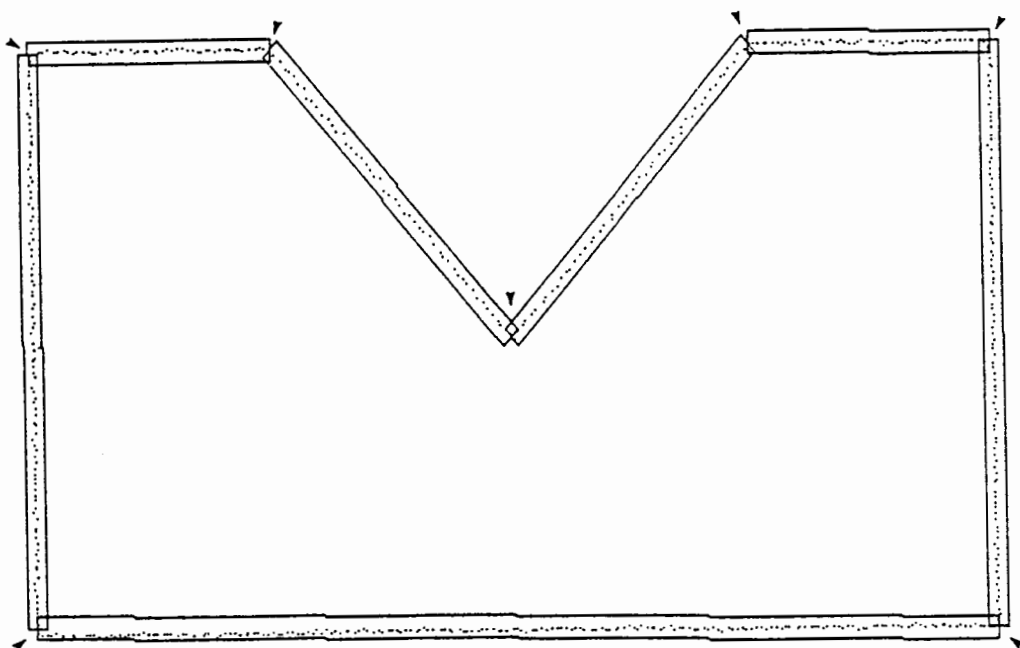


Figure 12. Step-A2 of Algorithm-A.

3.7 ALGORITHM-B

This algorithm detects the transition points between circular edges. Before this algorithm is executed, the merged rectangles from Algorithm-A are sorted according to their length. A nominal file which is usually obtained from the CAD database, supplies more information about the part. The information required now, is the number of straight edges "n" in the nominal part. The first "n" long rectangles are the straight edges and so these rectangles are removed from the list. The information about the remaining rectangles is retained. If the nominal part does not have any circular edges, no further action is needed since there are only straight edges. If the nominal part information suggests that there are circular edges, but there are no transition points between the circular edges then no further action is needed. The existence of no transition points between circles imply that the circles are not next to each other. If none of the above branches were taken, the following algorithm is executed:

- Step-B1 Decomposition of the curve with circular transition points into subsets that fit within rings of width W_1 .

- Step-B2 Merging the neighboring rings into larger-width rings of size W_g .

This algorithm is similar to Algorithm-A except that rectangle finding is replaced by ring finding. In this project, the minimum width ring located at the center of some "best" circular arc fit to the point set is used. This method is an approximation to the optimal position suggested in [2], but is simpler to implement. The best arc fit used in this project is an implementation of the algorithm by Thomas and Chan[3]. This arc fit technique uses an error norm based on areas. This gives two linear equations that can be solved in $O(M)$, where M is the number of points to be fitted. Thus the computational complexity of Algorithm-B is similar to that of Algorithm-A, but N is replaced with M .

Figure 13 shows a part fragmented into small rectangles after the execution of Algorithm-A. Figure 14 shows the approximate point of transition between the two circles after the execution of Algorithm-B. The detection of transition points by Algorithm-B depends on the type of object under consideration. If there are two circles with a large difference in radii, then the algorithm can easily find the transition point. But in the case of an object such as shown in Figure 14, there is a portion of the curve where the radius is within acceptable range from the centers of both the circles. This presents a complicated problem in finding the exact transition point.

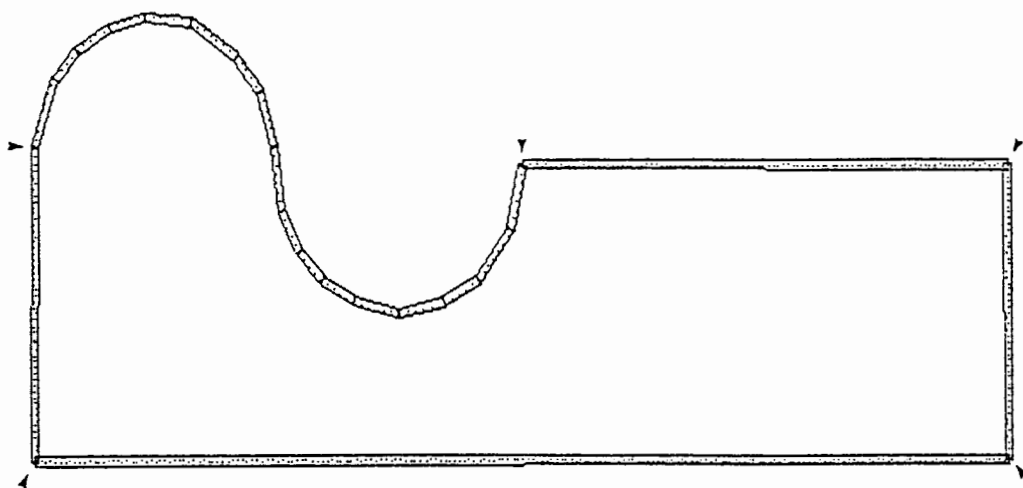


Figure 13. Step-B1 of Algorithm-B.

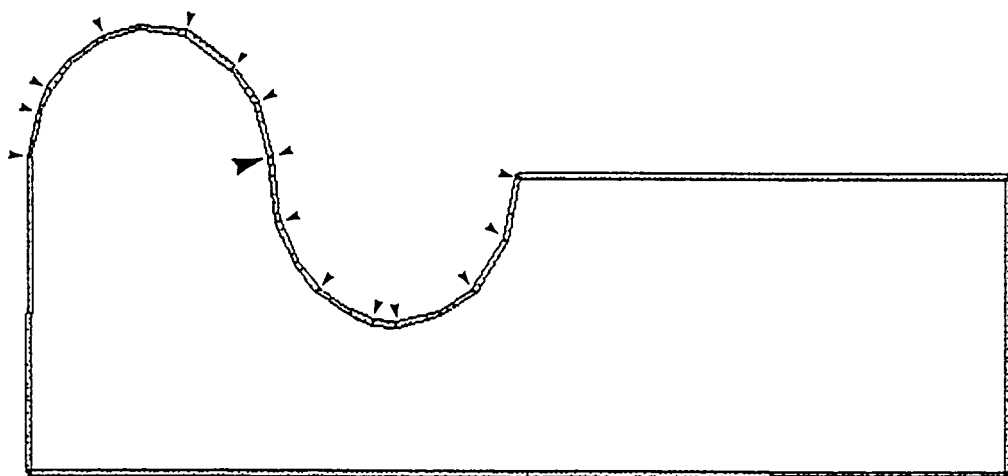


Figure 14. Step-B2 of Algorithm-B.

CHAPTER IV

COMPARISON OF ACTUAL PART WITH THE NOMINAL PART

4.1 INTRODUCTION

On the execution of Algorithm-B, the resultant rectangles and rings represent the features or the edges of the actual object. The next part of the project is to compare the features of the actual part to the features of the nominal part to examine if they are identical. This comparison is necessary before the actual process of inspection is performed.

At this point a feature table must be created. The actual part has an Actual Part Feature Table (APFT) and the nominal part has a Nominal Part Feature Table (NPFT). The feature table lists each feature with an index, the type of each feature, the start point and the end point, and the departure and arrival vectors. From this information, the relative length for a straight edge and the relative radius in the case of a circular feature are computed. The relative lengths and relative radii are determined as follows. The longest rectangle in the contour is given a value "1", and the other rectangles are given relative length values based on their length with respect to the longest rectangle. In the case of relative radius, the largest radius is given a relative radius value of "1", and the remaining radii are given proportionate values based on their radii in comparison to the largest radius. It is possible to have two features with relative lengths or radii of "1" in a contour. In the feature table, a "1" represents a straight feature while a "2" represents a circular feature. The feature table also keeps track of the start point and the end point of each feature. APFT also contains four more fields to accommodate the arrival and departure vectors. These vectors are non-zero only for circular edges and are set to zero for straight edges. The NPFT must contain all of the above information except the start point, end point, the arrival and departure vectors. The

NPFT has a field called the Turn Angle. Turn angles represent the angle that each edge or feature makes with the previous edge. An elaborate discussion on Turn Angles and about the arrival and departure vectors can be found in the Algorithm-E description. The nominal part file also provides a sequence in which the features are present in the nominal part file. Table I and Table II show a typical APFT and NPFT. Table I shows the various fields of the APFT. They are as follows :

1. Feature Type
2. Start point X coordinate
3. Start point Y coordinate
4. Departure vector X coordinate
5. Departure vector Y coordinate
6. Arrival vector X coordinate
7. Arrival vector Y coordinate
8. End point X coordinate
9. End point Y coordinate

Table II shows the different fields in NPFT. They are as follows:

1. Feature number
2. Feature type
3. Relative length for type "1" or radii for type "2"
4. Turn angle

The following discussion introduces three algorithms for the comparison of the parts.

4.2 ALGORITHM-C

This algorithm serves to find the match between the nominal and actual features based on the feature types. The start point and end point values in APFT are based on the pixel detection and hence the sequence of features in the APFT may be completely different from the NPFT sequence. The other objective of this algorithm is to find the starting feature in the APFT such that

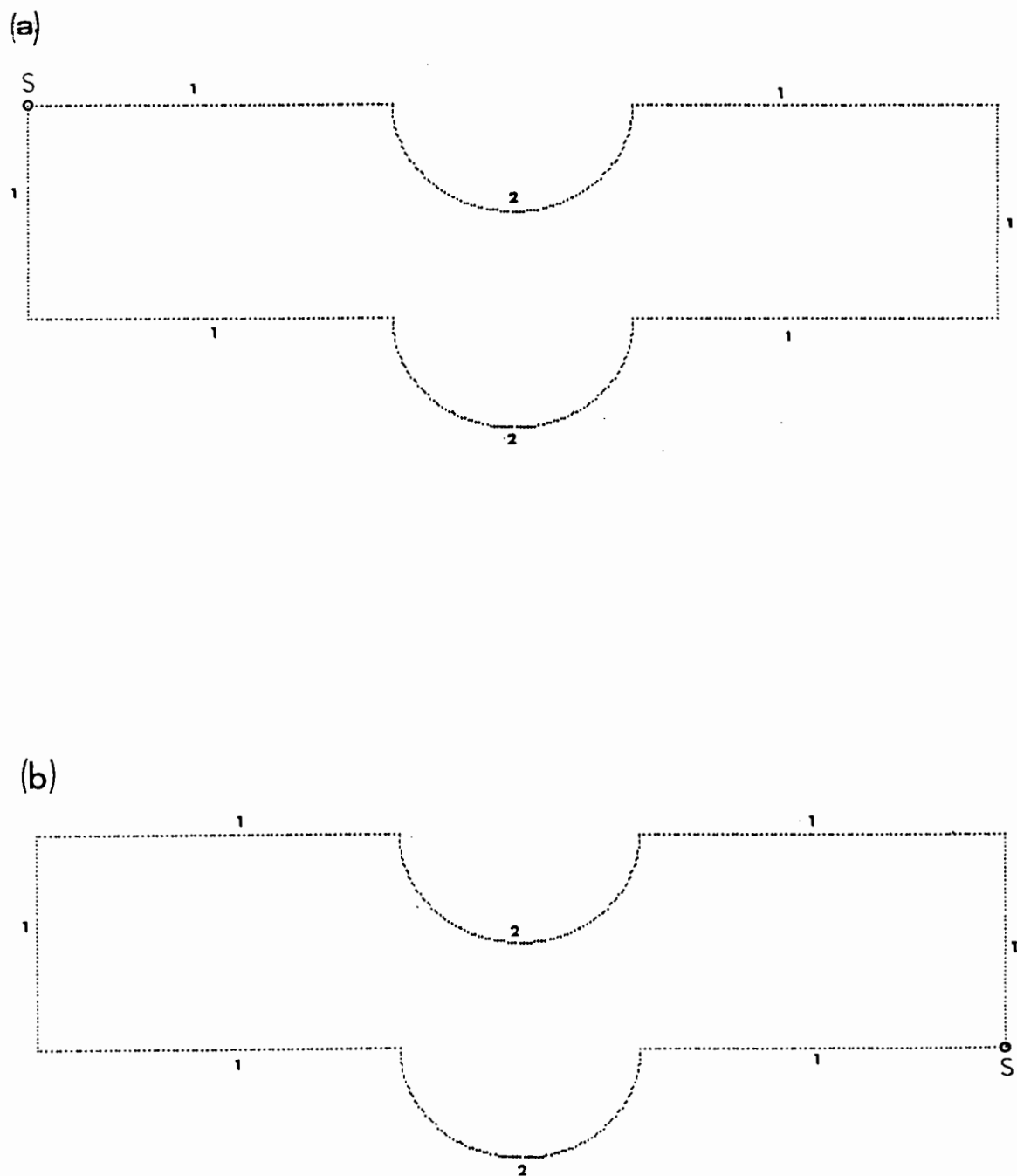


Figure 15. Contours for Algorithm-C.

(a) Actual part.

(b) Nominal part.

the sequence of features in APFT is identical to NPFT. The technique used is as follows:

The algorithm assumes that both APFT and NPFT have a pointer to point to each item in the table. To start with, these pointers are initialized to the first item in their respective tables. Then these pointer values corresponding to the feature type are compared. If they are the same then both pointers are moved down to the next item and a counter is incremented. If this item matches also, the pointer is moved down on item. But if the comparison fails, the pointers are initialized to a new starting point and the counter is set to zero. In the case of the APFT it would be the second item in the table but for NPFT, the pointer still starts at the first item. This process continues until every item in the APFT has become a starting point. Also every starting point that produced a counter value equal to the number of items in the table is recorded. If there are more than one counter reading recorded, it implies that there is a possibility of two or more acceptable sequences of features existing in this object.

Figure 15 shows a contour of the actual part and the nominal part along with the starting points for their feature sequence. The actual part has a feature sequence of { 1 2 1 1 1 2 1 1 }. While the nominal part has a feature sequence of { 1 2 1 1 1 2 1 1 }. The algorithm sets the APFT pointer to the first feature, i.e "1". The NPFT pointer is also set to the first feature in the sequence i.e "1". The procedure mentioned in the above discussion is followed. The starting point on the APFT produces a counter reading of "8". This suggests that this point, point "a", could be a starting point for the feature sequence and is recorded. Next the APFT pointer starting point is set to the second feature and the counter is set to zero. As mentioned in the previous paragraph, the NPFT pointer is still set to the first feature. On comparing the feature types of the starting point, the algorithm finds that the features are dissimilar. Therefore this point can not be a starting point and is not recorded. This procedure continues until all the features in APFT have been tested for starting points. Now the algorithm checks the recorded starting points. In this case, the algorithm finds two possible starting points, point "a" and point "b".

If the result was only one recording, then the task is complete. But if there is more than

TABLE I
ACTUAL PART FEATURE TABLE (APFT)

1	2	3	4	5	6	7	8	9
1	100.00	100.00	000.00	000.00	000.00	000.00	100.00	150.00
1	100.00	150.00	000.00	000.00	000.00	000.00	200.00	150.00
2	200.00	150.00	200.00	149.00	250.00	149.00	250.00	150.00
1	250.00	150.00	000.00	000.00	000.00	000.00	350.00	150.00
1	350.00	150.00	000.00	000.00	000.00	000.00	349.00	100.00
1	349.00	100.00	000.00	000.00	000.00	000.00	250.00	100.00
2	250.00	100.00	250.00	099.00	200.00	099.00	200.00	100.00
1	200.00	100.00	000.00	000.00	000.00	000.00	101.00	100.00

TABLE II
NOMINAL PART FEATURE TABLE (NPFT)

1	2	3	4
1	1	0.619048	4.712384
2	1	0.952381	4.712384
3	2	1.000000	4.712384
4	1	1.000000	4.692386
5	1	0.619048	4.732381
6	1	0.952381	1.570796
7	2	0.999971	1.570796
8	1	0.952381	4.732381

one acceptable sequence as in the above example, both Algorithm-D and Algorithm-E are used to establish a possible unique sequence of features. If there was no recording, Algorithm-C denies any similarity between the actual part and the nominal part. In such a case, an error message is printed stating that the objects under comparison are not the same.

4.3 ALGORITHM-D

Algorithm-D is used only if there is no unique solution from Algorithm-C. The objective of Algorithm-D is to find a unique sequence of features if one exists, starting from the starting points obtained from Algorithm-C and comparing the relative lengths of the features.

The methodology of comparison is the same as in the case of Algorithm-C except that the pointer in APFT is initialized to the starting points determined from Algorithm-C. The pointer in NPFT is initialized to the first item in the table. The difference in magnitude is always checked to be less than a prescribed percentage value. This maximum value is about five percent or .05 in relative length. In other words, the acceptable error in magnitude is five percent of the magnitude of the largest length edge.

When using this algorithm for parts with circular edges, the comparison is made between the relative radii of the nominal part edge and the actual part edge. As in the straight edge, a prescribed value of error in percentage is accepted. The maximum percentage error is five percent of the nominal part radii. Values more than this can be used, but may not lead to the right solution.

If the difference in both cases is less than the prescribed value, a counter is incremented. If the counter reaches the number equal to the number of items in the file, the starting point is recorded. The pointer of APFT is then reset to the next starting point obtained from Algorithm-C. If the difference is more than the given value and the counter is less than the number of items in the feature table, then the starting obtained from Algorithm-C is not an actual starting point for the sequence and thus can be discarded. This process is repeated until all the starting points obtained from Algorithm-C are explored. If this search leads to more than one starting point then Algorithm-E

is used. On the other hand, if only one starting point was recorded, the sequence is unique and is displayed.

4.4 ALGORITHM-E

Algorithm-E is the final algorithm on the comparison mode. This algorithm is used only if Algorithm-D fails to give a unique sequence of features. Here the Turn Angles for each feature are computed and compared with the nominal turn angles. Turn angle is the angle that a feature makes with its previous edge. This angle may take any value between zero to three hundred and sixty degrees.

A graphical definition of turn angle is as follows, consider Figure 16. In this figure, E1 and E2 represents two edges between points P1 to P2, and P2 to P3, respectively. Let V1 and V2 be two unit vectors along E1 and E2 respectively and let C.S represent a right handed coordinate system at point P2 whose "x" axis along the direction of V1. The turn angle is defined as the angle made by vector V2 with respect to the coordinate system "C.S" in the conventional direction of angle measurement (counterclockwise). For the edges in the figure, V2 makes a turn angle of 270° .

The turn angle computation varies with the type of edge. Straight edges have only one vector. But circular edges have two vectors, departure and arrival. The departure vector is the unit vector which is normal to an imaginary line passing through the center and the first point of the circular edge. The arrival vector is the unit vector which is normal to an imaginary line passing through the center of the circle and the last point on the circular edge.

The turn angle calculation can be sub-divided into four cases based on the following sequence of consecutive features:

1. Straight edge followed by Straight edge.
2. Straight edge followed by Circular edge.
3. Circular edge followed by Straight edge.
4. Circular edge followed by Circular edge.

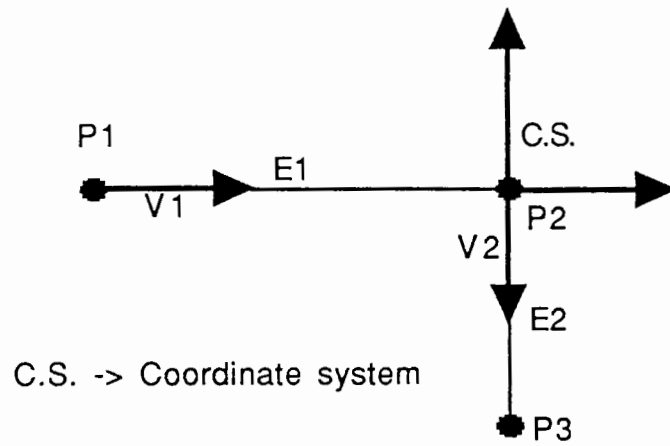


Figure 16. Turn angle between vectors.

The first case is the simplest of all. This involves calculating the angle made by the unit vector on the second straight edge with respect to an unit vector on the second straight edge. In the second case, the turn angle is the angle between the unit vector on the straight edge and the departing vector of the circular edge. For the third case, the turn angle is computed by finding the angle made by the arrival vector with respect to an unit vector on the following straight edge. In the last case, the turn angle is the angle made by the departure vector of the second circular edge and the arrival vector of the first circular edge.

Figure 17 shows the same contour used in Algorithm-C. In this figure, the names V1 through V8 denote the unit vectors. Suffixes "a" and "d" denote the arrival and departure vectors and are found only for circular edges. The magnitude and direction of the vectors are computed from the information about the circular feature such as start point, end point, and radius. E1 to E8 represent the edges in the contour. Edges E1, E3, E4, E5, E7, E8 are straight and E2 and E6 are circular edges. The following table is an example of the turn angle list for the contour in Figure.14. The edge symbol E_s represents the edge for which the turn angle is listed and the edge symbol E_t represents the edge to which the edge under E_s makes the turn angle. V_i and V_s represent the vectors of the first edge and the second edge respectively that were used for the angle computation.

E_s	E_t	V_s	V_t	Turn angle(Degrees)
E1	E8	V1	V8	270
E2	E1	V2d	V1	270
E3	E2	V3	V2a	270
E4	E3	V4	V3	270
E5	E4	V5	V4	270
E6	E5	V6d	V5	90
E7	E6	V7	V6a	90
E8	E7	V8	V7	270

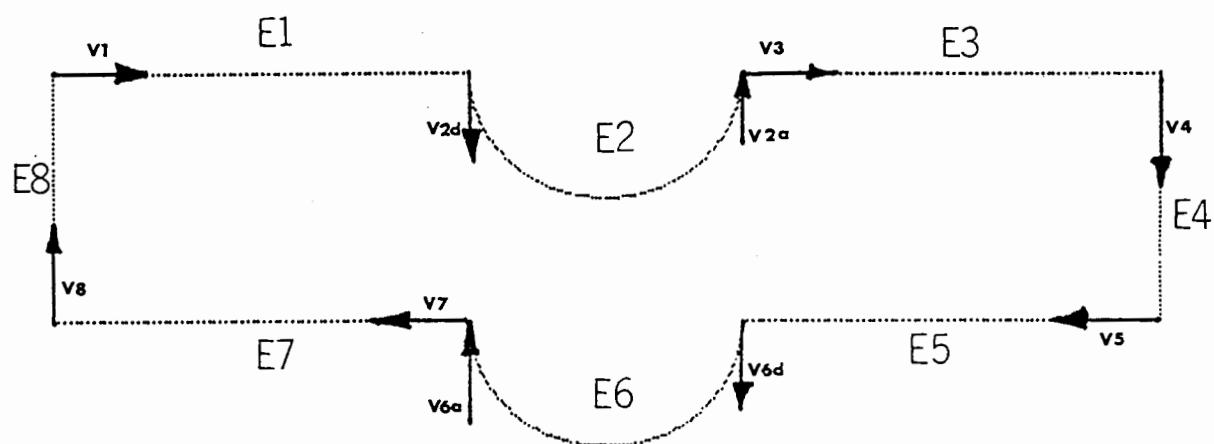


Figure 17. Algorithm-E.

The turn angles thus calculated are stored in the APFT. As in the case of Algorithm-C and Algorithm-D, Algorithm-E also maintains a pointer in APFT and NPFT. The APFT pointer is initialized to the first starting point from Algorithm-D, while the NPFT pointer is initialized to the first item. The angles are now compared. The magnitude of the difference is added up for each item and stored. The process is continued for each of the starting points obtained from Algorithm-D.

The best sequence is the one which has the least value stored in the magnitude item. But this value must be checked against a threshold value which is normally a few degrees. If there are more than one recording, then there is more than one starting point that provides the same result. Mirror-image-like objects can have more than one starting point for their feature sequence.

Matching of type, relative lengths and radii are necessary conditions for a matching sequence. The following theorem shows that matching turn angles provide the sufficient conditions for matching.

Theorem 1

If two contours match in type sequence, relative lengths and radii (i.e. pass the necessary conditions), then they are identical if and only if they match in turn angles. Circles must also agree on the center-side with respect to the departure vector.

Proof: Arbitrarily match an edge AB with its corresponding A'B' having the same length. If the edge AB is followed by a straight edge, say BC, then the edge BC is uniquely defined through

- (1) its length and
- (2) its angle of turn with respect to AB.

Since these two are identical by the theorem premise, point C must match point C'.

If edge AB is followed by a circular edge BC, then the circle is uniquely defined by:

- (1) its radius
- (2) its point of departure B
- (3) its starting and departing vectors

(4) the side where the center is located with respect to the departure vector.

The proof is a graphical one as shown in Figure 18 and Figure 19. Consider point B (starting point) and radius R . Then, there are only two possibilities for the circle tangent to the departure vector and of radius R .

Since the center must be on a particular side of the departure vector, only one of the circles are acceptable. The arrival vector uniquely identifies the end of the circular arc. Noting that both contours must be traversed in the same direction, point C is unique. This way, the entire contour can be matched and the contours are identical. The "only if" part of the theorem is obvious. If the contours are identical, so are the turn angles.

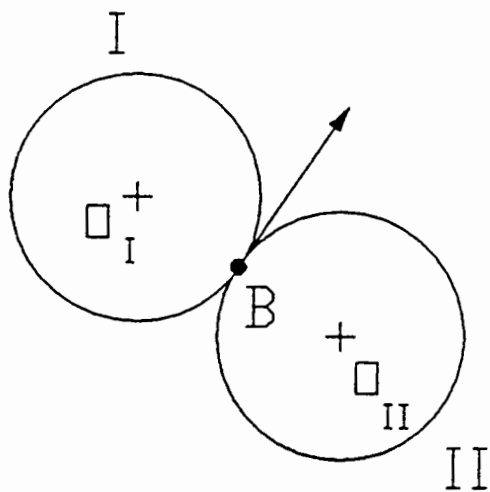


Figure 18. Possible turn vectors.

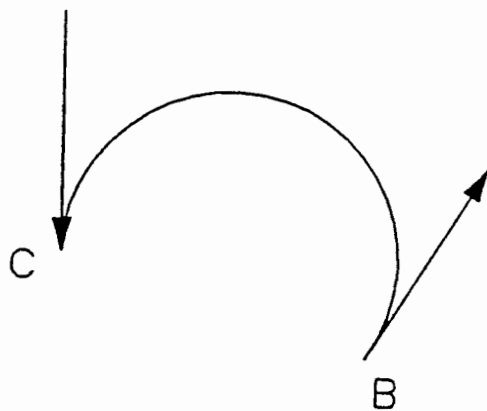


Figure 19. Unique turn vector location.

CHAPTER V

PERFORMANCE AND ROBUSTNESS OF THE ALGORITHMS

5.1 PERFORMANCE ON SIMULATED CONTOURS

All the contours used in the description of the algorithms were simulated contours. These contours are obtained from the simulator program. This simulator program takes as input, the number of edges and the type of each edge. Based on the type of edge, the query is formulated to define the object. Here again, "1" represents a straight edge while "2" represents a circular edge.

If the input to the type of edge decision is "1", then the program asks for the start point and the end point of the straight line. On the other hand if the type of edge is "2", then the program asks for the start point, the end point, the radius, and an arbitrary third point. Using this information, the simulator determines the center coordinates and the appropriate arc segment of circle. It is also possible to simulate a straight edge with low frequency distortion with this simulator. In such a case, the edge is defined to be of type "2" and the inputs are the start point, the end point, a large radius, and an arbitrary point on the desired segment of the circle.

A random normal noise can be added to the points generated by the simulator and applied independently to the "X" and "Y" axes. Using this option, it is possible to generate various amounts of noise distortion on the contours and study their effect on the algorithm performance.

5.2 PERFORMANCE WITH ACTUAL CONTOURS

The boundary contour of a part was carefully drawn on a paper and processed using the vision system. This is shown in Figure 20. The part has a semi-circle followed by a quarter circle. The contour was obtained by processing the part's image obtained using the image capturing

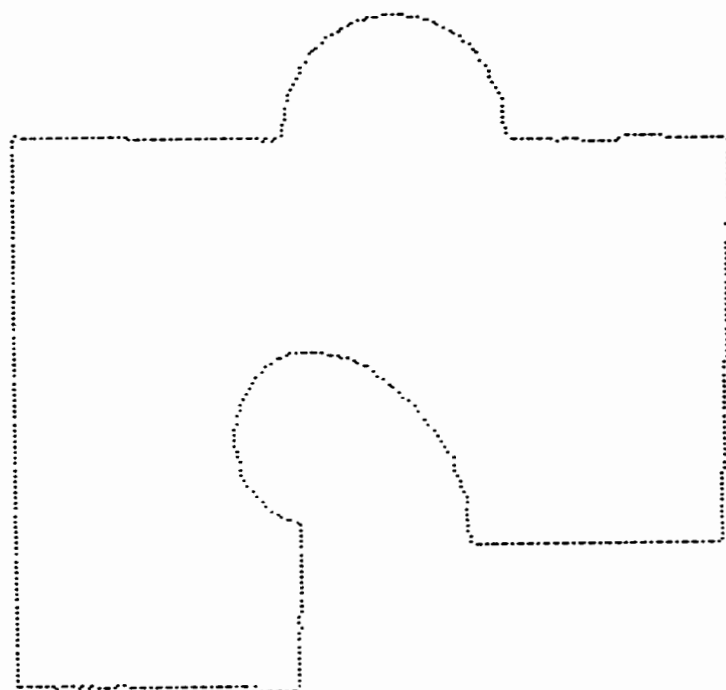


Figure 20. Actual contour.

system listed under the section titled Equipment. A simple thresholding followed by a contour tracing technique was used to detect the boundary pixels [11]. For dimensional inspection, these boundary pixels must be transformed to actual coordinates through camera calibration. This calibration also compensates for the lens distortion. Since this transformation will make no difference in the location of the transition points, the pixel coordinates are used directly.

Figure 21 is obtained as a result of executing Step-A1 followed by Step-A2 of Algorithm-A. The remaining non-straight edges are then processed by Algorithm-B. Algorithm-B detects the transition points shown in Figure 22.

The program output closely matches the desired output both in the number of edges and the points of transition even for a crude and noisy measurement.

5.3 ROBUSTNESS OF THE ALGORITHMS

Any algorithm developed for a production line application must be multi-tasking and must be capable of handling a wide variety of situations. In an automated environment, any failure prone or highly sensitive programs are not used. Therefore it is important to list the cases where the suggested algorithms can lead to erroneous results along with their consequences. The following is a list of possible errors that may result:

- Type-1: The number of detected edges is different from the number of nominal edges. This is a fatal error because there is no point in comparing two completely different objects.

- Type-2: The detected edges are incorrectly classified. This is a fatal error because there is no point in comparing two dissimilar objects.

- Type-3: Transition points are inaccurately identified. This is not a fatal error, but may affect the inspection results.

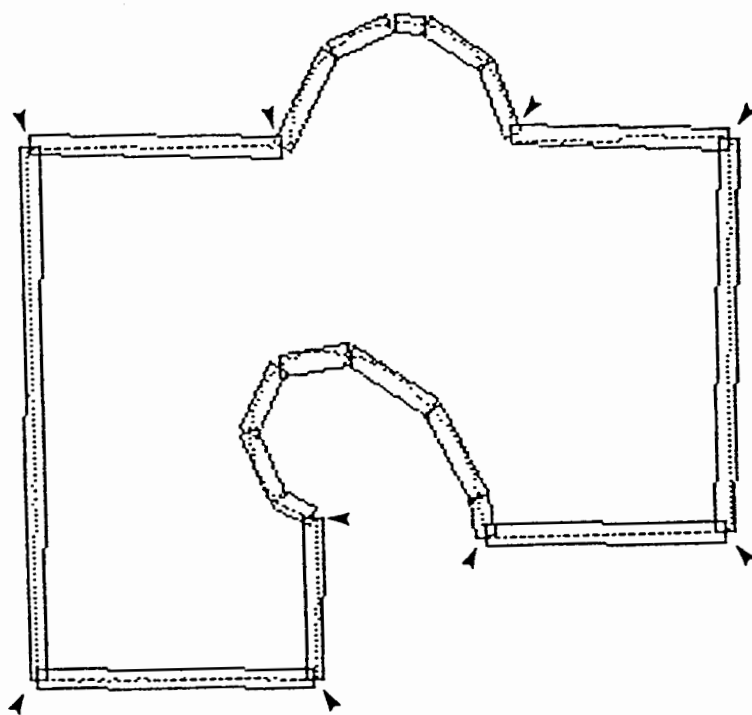


Figure 21. After execution of Algorithm-A.

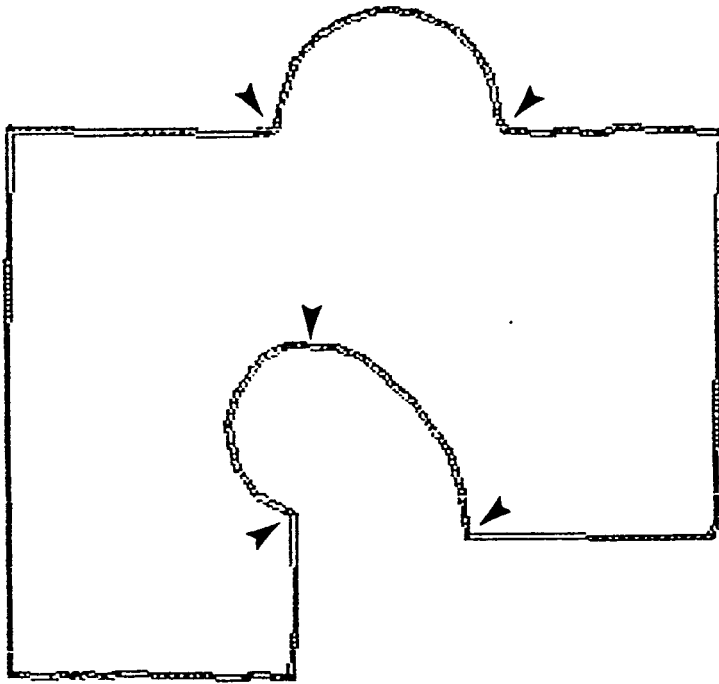


Figure 22. After execution of Algorithm-B.

Type-1 error

Type-1 error can happen after merging, when single edges are incorrectly fragmented or separate edges are coalesced by mistake. In the first case, the entire edge can not be fit within one rectangle or ring of the specified size W_g size. In the second case, W_g is so large that it covers more than one edge. If the maximum global distortion is W_{gmax} for a typical machined edge is 0.005 inch; this implies that the machined edge will not have a distortion of more than 0.005 inch from beginning to the end. If the theoretical part contains two or more consecutive edges that can fit within one rectangle or ring of size $W_g=0.005$ inch, then there is possibility of type-1 error.

Consider Figure 23, where L_1 and L_2 are the lengths of the first and second edges respectively and θ is the angle between the two edges. From this figure, the following relationship is established using simple geometric techniques.

$$\theta_1 = \cos^{-1}(W_g / L_1)$$

$$\theta_2 = \cos^{-1}(W_g / L_2)$$

$$\theta = \theta_1 + \theta_2$$

$$\theta > \cos^{-1}(W_g / L_1) + \cos^{-1}(W_g / L_2)$$

for Type-1 error to occur.

If $L_1=3$ inches, $L_2=2$ inches, and $W_g=0.005$ inch, the angle θ was found to be 179.7 degrees. In this case, for type-1 error to occur, the angle between the edges should be greater than 179.7 degrees. Since designs rarely have edges with this high an angle between them, type-1 error provoking situations are virtually none.

Type-2 error

Type-2 error happens when the nominal part includes circular arcs with such a large radius of curvature that the entire arc or a portion of it can be fit within a rectangle so long that its length exceeds the length of at least one straight edge. In such a case, the arc will be classified as a straight edge. Alternatively, a straight edge can be so small that its length ends up becoming smaller than a part of a circle fitting into a rectangle of size W_g . Considering Figure 24 and some



Figure 23. Type-I error.

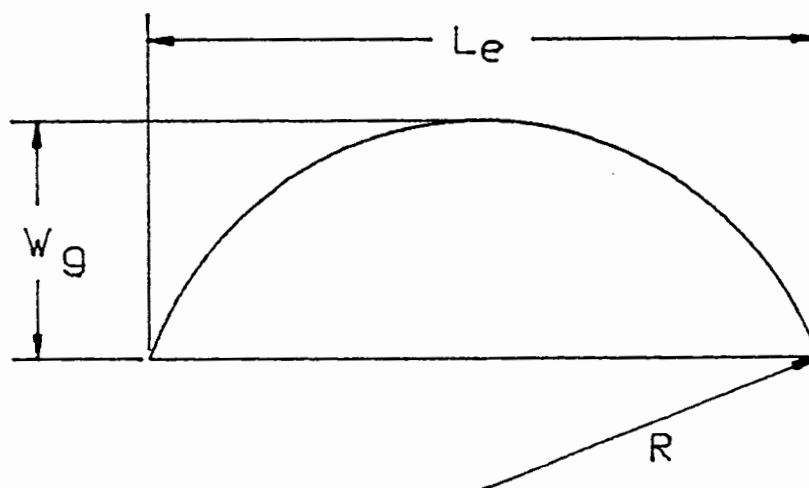


Figure 24. Type-II error.

elementary geometry, the relationship between the edge length L_e and the arc radius R is approximately quadratic as shown below:

From this figure it is seen that,

$$L_e = 2 \cdot R \cdot \sin \theta$$

$$W_g = R - R \cdot \cos \theta$$

$$\cos \theta = (W_g - R) / R$$

$$L_e = 2 \cdot R \cdot \sqrt{1 - \cos^2 \theta}$$

$$L_e = 2 \cdot \sqrt{W_g \cdot (2 \cdot R - W_g)}$$

Since $W_g \ll 2 \cdot R$

$$L_e \approx 2 \cdot \sqrt{2 \cdot R \cdot W_g}$$

$$\approx 2 \cdot \sqrt{2} \cdot \sqrt{R \cdot W_g}$$

$$L_e^2 = 8 \cdot R \cdot W_g$$

or

$$R = L_e^2 / W_g$$

This implies that for a type-2 error, the radius of the nominal circle must be greater than that given in the above equation. As an example, for a part with a 0.500 inch smallest edge length and a maximum distortion of $W_g = 0.005$, the radius of an arc of the part must be at least 6.250 inches. Although this is unlikely for most mechanical parts, the above expression must be checked to assure that type-2 error will not happen.

Type-3 error

Type-3 error is more likely to happen, but only in rare cases it leads to incorrect inspection. The reason is that dimensional inspection of features can be performed from the edge information far removed from its transition points. This type of error takes place when W_l is set too large such that the points belonging to two edges may be included in one rectangle, where W_l is the local rectangle width. This situation is most severe when the difference in slopes of the neighboring edges is small. Using elementary geometry, it is possible to find the relationship between the error

length L_{err} , W_l , and the angle θ between two straight edges as follows:

From Figure 25, the following relationship is derived.

$$W_l = L_{err} * \sin(\pi - \theta)$$

$$L_{err} = W_l / \sin(\pi - \theta)$$

For a typical $W_l=0.001$ and $\theta \leq 170$ we find the error length to be 0.0057 inch which is quite small. A similar analysis for transition between straight edges and circles in the worst case, i.e., when the edge and circle are tangent, gives the maximum error length L_{err} and can be derived as follows:

Consider Figure 26, where R denotes radius and W_l denotes the width of the rectangle.

From the figure it is seen that,

$$L_{err} = R * \theta$$

$$R = W_l + R * \cos\theta$$

$$\cos\theta = (R - W_l)/R$$

$$\theta = \cos^{-1}((R - W_l)/R)$$

or

$$L_{err} = R \cos^{-1}(R - W_l/R)$$

where R is the circle radius. For a typical W_l of 0.001, and R of 1.00 inch, the error length is 0.045 which is quite small. For $R=10$ inches, the error is 0.14 which is still tolerable.

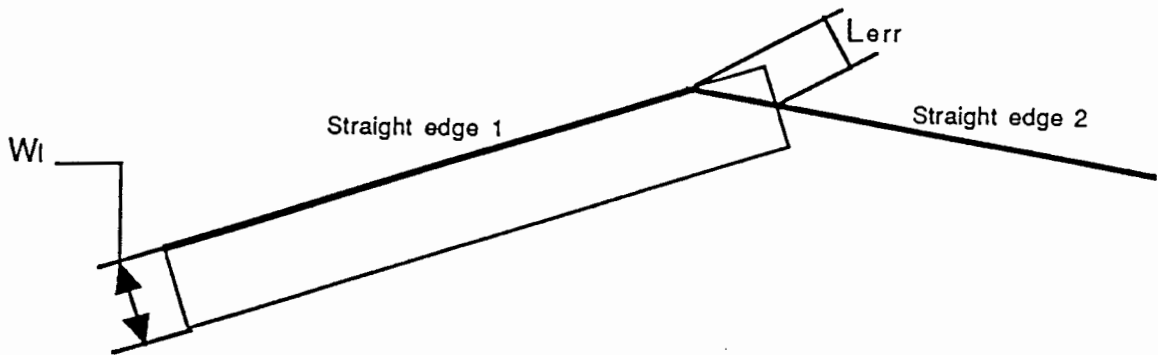


Figure 25. Type-III (a) error.

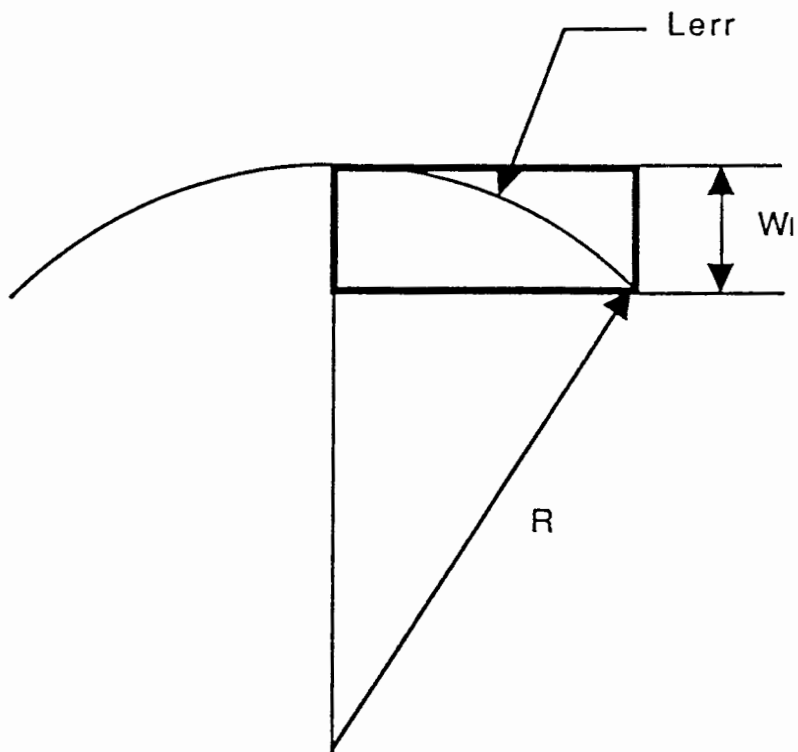


Figure 26. Type-III (b) error.

CHAPTER VI

CONCLUSIONS AND FUTURE RESEARCH PROSPECTS

In this research project, the problem of contour decomposition into straight and circular edges has been addressed. Two algorithms were presented to accomplish the task. The algorithms work well even when artificial high and low frequency noise is added to the data. On implementation of these algorithms on high performance workstations, the computational complexity of the algorithms pose no threat.

The next possible stage of research would be to develop modules that can extract the needed information from a CAD database and create a nominal part file and nominal part feature table. The other module that could pose an interesting ground of research is the tolerance verification and its data structure, and image processing trainer module. Each object needs a particular image processing technique for best results. By using an image processing trainer module, a suitable technique can be chosen based on the results from implementing the available techniques.

On completion of all these modules, it is possible to automate dimensional inspection.

REFERENCES

- [1] Preparata Franco P., and Shamos Michael I., "Computational Geometry", Springer-Verlag, 1985.
- [2] Etesami Faryar and Qiao Hong, "Analysis of 2D Dimensional Measurement Data for Automated Inspection and Modeling of Manufactured Parts", Journal of Manufacturing Systems, Vol. 30, March 1990.
- [3] Thomas Samuel M., and Chan Y.T., "A Simple Approach for the Estimation of Circular Arc Center and Its Radius", Computer Vision, Graphics, and Image Processing 45, pp. 362-370, 1989.
- [4] T.Y. Young, K.S. Fu, "Handbook of Pattern Recognition and Image Processing", Academic Press, 1986.
- [5] Imai Hiroshi, and Iri Masao, "Computational-Geometric Methods for Polygonal Approximations of a Curve", Computer Vision, Graphics, and Image Processing 36, pp. 31-41, 1986.
- [6] T.S. Huang, "Picture Processing and Digital Filtering", Springer Verlag, Berlin and New York, 1975.
- [7] A. Rosenfeld and A.C. Kak, "Digital Picture Processing", Academic Press, New York, 1982.
- [8] T. Peli and D. Malah, "A Study of Edge Detection Algorithms", Computer Graphics and Image Processing 20, pp. 1-21, 1982.
- [9] L.J. Van Vliet and I.T. Young, "A Non-Laplace Operator as Edge Detector in Noisy Images", Computer Vision, Graphics, and Image Processing 45, pp. 167-195, 1989.
- [10] Teicholz, Eric and Orr, Joel M., "Computer Integrated Manufacturing Handbook", McGraw-Hill, 1987.
- [11] Koeppe, Ralf H., "A Comparative Study of the Performance of Various Image Analysis Methods For Dimensional Inspection with Vision Systems", Master's Thesis, Portland State University, 1989.
- [12] "ITEX PCplus Programmer's Manual", Image Technology Incorporated, 1987.
- [13] A. Huertas, W. Cole and R. Nevatia, "Detecting Runways in Complex Airport Scenes", Computer Vision, Graphics, and Image Processing 51, 107-145, 1990.
- [14] Dacheng Wang, and Sargur N. Srihari, "Classification of Newspaper Image Blocks using Texture Analysis", Computer Vision, Graphics, and Image Processing 47, 327-352, 1989.

- [15] Yuh-Tay Liow, and Theo Pavlidis, "Use of Shadow for Extracting Building in Aerial Images", *Computer Vision, Graphics, and Image Processing* 49, 242-277, 1990.
- [16] Lih-Shyang Chen, and Marc R. Sontag, "Representation, Display, and Manipulation of 3D digital scenes and their Medical Applications", *Computer Vision, Graphics, and Image Processing* 48, 190-216, 1989.
- [17] N. Kehtarnavaz, and N. Griswold, "Establishing Collision Zones for Obstacles Moving with Uncertainty", *Computer Vision, Graphics, and Image Processing* 49, 95-103, 1990.