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.

MACHINE VISION TOOL MONITORING SYSTEM FOR AUTOMATED MANUFACTURING

A Thesis

Submitted to the Faculty of Graduate Studies and Research

In Partial Fulfillment of the Requirements

for the Degree of

Master of Science

in Industrial Engineering

Faculty of Engineering

University of Regina

By

Donatus Chimezie Dennis Oguamanam

Regina, Saskatchewan

1990

ISBN 0-315-59388-1

Canada

UNIVERSITY OF REGINA

FACULTY OF GRADUATE STUDIES AND RESEARCH

CERTIFICATION OF THESIS WORK

We, the undersigned, certify that Donatus C.D. Oguamanam, candidate for the Degree of Master of Science has presented a thesis on *Machine Vision Tool Monitoring System for Automated Manufacturing*, that the thesis is acceptable in form and content, and that the student demonstrated a satisfactory knowledge of the field covered by the thesis in an oral examination held on July 13, 1990.

External Examiner: Alan M. Law

Dr. Alan Law, Dept. of Computer Science, U of R.,

Internal Examiners: S.D. Bhole

Dr. S.D. Bhole, Co-Supervisor

Hazem Raafat

Dr. H. Raafat, Co-Supervisor

UNIVERSITY OF REGINA

FACULTY OF GRADUATE STUDIES AND RESEARCH

PERMISSION TO USE POSTGRADUATE THESES

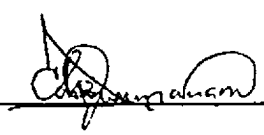
TITLE OF THESIS: *Machine Vision Tool Monitoring System for Automated Manufacturing*

NAME OF AUTHOR: Donatus C.D. Oguamanam

FACULTY: Faculty of Graduate Studies and Research

DEGREE: Master of Science

In presenting this thesis in partial fulfillment of the requirements for a postgraduate degree from the University of Regina, I agree that the Libraries of this University shall make it freely available for inspection. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by the professor or professors who supervised my thesis work, or in their absence, by the Head of the Department or the Dean of the Faculty in which my thesis work was done. It is understood that any copying, publication or use of this thesis or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to the University of Regina in any scholarly use which may be made of my material in my thesis.

SIGNATURE: 

DATE: July 13, 1990

To

The Glory of God
Mr. & Mrs. Eugene Chukwukere Oguamanam, my parents
Miss Ngozi Camela Ibekwe, my love

Abstract

The process of tool wear dulls a tool cutting edge, increases friction between the tool and the workpiece, and increases power consumption. Unexpected tool failure could destroy the workpiece and the tool. All these will increase scrap rate, reduce quality of the product(s), reduce machine utilization and therefore increase production costs. The different ambient conditions during cutting processes have made tool change at the appropriate time a stochastic and an inefficient process.

This thesis presents an effective machine vision system for monitoring wear and detecting breakage in single point cutting tools (CNC lathe tools). The newly developed algorithm is implemented in Microsoft C V5.1 and the system is composed of a microscope, an optic fiber illuminator, a Zenith IBM/XT compatible microcomputer with an Image Capture Board, a CCD RGB video camera and a colour monitor.

The image of a given tool is converted into a binary image accentuating the wear land. Then, five features are extracted and used in a classification scheme. A microcomputer reports on the state of the tool based on the results of the classification. The detection of tool breakage and the ease of integration with existing monitoring techniques are some of the advantages of the developed system.

Acknowledgement

I would like to express my sincere appreciation to my supervisors, Dr. H. Raafat and Dr. S. D. Bhole, for their advice, criticisms, patience and moral support during the course of this research. My regards go to Dr. S. M. Taboun for supervising the early stage of this research and for his moral support.

In a venture of this magnitude, it is not astonishing to note that I am greatly indebted to people, too numerous to mention. However, I wish to acknowledge the following for their moral support: Dr. and Mrs. A. Abiola, Dr. O. A. Olatunbosun, Dr. M. A. C. Chendo, Dr. R. Kasilingam, Dr. E. Hara, Mr. Z. B. Eshetu, Mrs. M. Jimale, Mr. S. M. Sawant, Dr. C. O. Oguamanam, Mr. A. E. Oguamanam, Mr. R. Jones, Mr. I. N. A. Oguocha, Mr. W. C. Nwaribe and Mr. M. S. C. Iberosi. The moral support and valuable contributions of Dr. X. D. Yang are highly appreciated.

I am grateful to the staff of Regina Water Research Institute for providing the microscope and fibre optic illuminator. I would also like to acknowledge Mr. H. Berwald and Mr. E. Lotocki, both of the Engineering Workshop, for their cooperation during the course of this research.

This research would not have been possible without the financial help from the Faculty of Graduate Studies through the provision of a Province of Saskatchewan Graduate Scholarship. For this, I am very grateful to the faculty and staff of

Graduate Studies, and to the government and people of Saskatchewan.

Lastly, I would like to thank the faculty, staff and fellow students of the Faculty of Engineering for being a great team.

Mr. Donatus C. D. Oguamanam

Contents

Abstract i

Acknowledgement ii

Table of Contents iv

List of Tables vi

List of Figures vii

Chapter 1. Introduction 1

Chapter 2. Background And Literature Review 5

2.1 Cutting Tools 5

2.1.1 Tool Failure and Life 11

2.1.2 Types of Wear 13

2.2 Machine Vision 15

2.3 On-Line Monitoring 24

2.4 Off-Line Monitoring 28

2.4.1 Radioactive Sensors 29

2.4.2 Electrical Resistance Sensors 30

2.4.3 Optical Sensors 31

Chapter 3. Development of the System **34**

3.1 System Setup and Operation 34

3.2 Feature Selection 35

3.3 Algorithm Development 38

Chapter 4. Experimental Results and Discussion **48**

4.1 Experiment 1 49

4.2 Experiment 2 49

4.3 Experiment 3 52

4.4 Experiment 4 62

Chapter 5. Summary And Future Research **76**

5.1 Summary 76

5.2 Future Research 78

References **80**

Appendix A. Neighbourhood and Connectivity **85**

Appendix B. Chain Codes **88**

Appendix C. Hough Transform **91**

Appendix D. Source Code **95**

List of Tables

4.1 Parameters evaluated from tool wear test 49

4.2 Parameters evaluated from the tool breakage test 52

4.3 Cutting parameters and tool angles used in tool wear monitoring test on stainless steel bar 53

4.4 Values of the parameters evaluated from tool wear monitoring test on stainless steel bar 58

4.5 Cutting parameters and tool angles used in monitoring tool wear . . 62

4.6 Values of the parameters evaluated from tool wear monitoring test on AISI C1045 steel 68

List of Figures

2.1 Schematic of a basic turning operation 7

2.2 Continuous chip 8

2.3 Discontinuous chip 8

2.4 Built-up edge 8

2.5 Nomenclature of a single point cutting tool 9

2.6 Schematic of flank and crater wear 14

2.7 Influence of cutting time on flank wear 15

2.8 A paradigm for image interpretation 18

3.1 Schematic of the system setup 35

3.2 Schematic of the measured parameters 37

3.3 Flowchart of the tool wear monitoring and breakage detection algorithm 39

4.1 Typical tool wear (image before processing) 50

4.2 Typical tool wear (image after processing) 50

4.3 Typical tool breakage (image before processing) 51

4.4 Typical tool breakage (image after processing) 51

4.5 Unprocessed image of the tool before machining the austenitic stainless steel 54

4.6 Processed image of the tool before machining the austenitic stainless steel 54
4.7 Unprocessed image of the tool after machining the stainless steel for 5.5 mins 55
4.8 Processed image of the tool after machining the austenitic stainless steel for 5.5 mins 55
4.9 Unprocessed image of the tool after machining the stainless steel for 30.0 mins 56
4.10 Processed image of the tool after machining the austenitic stainless steel for 30.0 mins 56
4.11 Unprocessed image of the tool before machining the austenitic stainless steel for 62.50mins 57
4.12 Processed image of the tool before machining the austenitic stainless steel for 62.50mins 57
4.13 The austenitic stainless steel wear width profile 59
4.14 The austenitic stainless steel wear land perimeter profile 60
4.15 The austenitic stainless wear land area profile 61
4.16 The austenitic stainless steel wear land compactness profile 63
4.17 Unprocessed image of the tool before machining the AISI C1045 steel 64
4.18 Processed image of the tool before machining the AISI C1045 steel . 64
4.19 Unprocessed image of the tool after machining the AISI C1045 steel for 2.0 mins 65
4.20 Processed image of the tool after machining the AISI C1045 steel for 2.0 mins 65

4.21 Unprocessed image of the tool after machining the AISI C1045 steel for 26.0 mins 66
4.22 Processed image of the tool after machining the AISI C1045 steel for 26.0 mins 66
4.23 Unprocessed image of the tool after machining the AISI C1045 steel for 43.0 mins 67
4.24 Processed image of the tool after machining the AISI C1045 steel for 43.0 mins 67
4.25 The AISI C1045 steel wear width profile 69
4.26 The AISI C1045 steel wear land perimeter profile 70
4.27 The AISI C1045 steel wear land area profile 71
4.28 The AISI C1045 steel wear land compactness profile 72
A.1 A scheme for identifying the relative locations of pixels with respect to the centre pixel 86
A.2 Schematic illustrating the role of mixed connectivity 87
B.1 4-directional chain code directions 90
B.2 8-directional chain code directions 90
B.3 Boundary representation using 4-directional chain code 90
C.1 Image space (x,y) 92
C.2 Parameter space (c,m) 92

Chapter One

Introduction

This thesis discusses the development of a machine vision tool wear monitoring and failure detection system for use in automated manufacturing. It also presents results obtained from tests on a laboratory CNC lathe tools.

The proliferation of manufacturing technology in the late 1970s has led to a more competitive global market place. Consequently, most manufacturers, in order to remain competitive, are investigating methods of reducing production/manufacturing costs without degrading the quality of their products. Among those being investigated is the automation of production systems. Although the automation of these systems could be the necessary panacea, it may be difficult to maintain production precision and reduce machine downtime. This is because of unexpected tool failure and early tool wear.

The process of tool wear dulls the tool cutting edge, increases friction between the tool and the workpiece, and increases power consumption. The dullness of the cutting edge leads to dimensional failure, thus affecting production precision because intended sizes on products are not achieved. Unexpected or catastrophic tool failure can destroy the workpiece or the tool, to cause work stoppage. All these

will increase scrap rate, reduce quality of the product(s), reduce machine utilization and thereby increase production costs. The different prevailing conditions during cutting operations have made tool sharpening or replacement at the appropriate time a stochastic and inefficient tool monitoring process.

In recent years, research efforts to automate tool monitoring processes have mushroomed. However, most of these are devoted to in-process (or indirect) monitoring by acquiring and analyzing one or more machining variables — force, torque, power, vibration and acoustic emission. The inflexibility and unreliability of these techniques have prompted the need to research into other methods such as off-line monitoring.

Machine vision technology can provide off-line (direct) monitoring. Direct monitoring can also be achieved using radioactive sensors, pneumatic sensors or electrical resistance sensors. The need to carry out research on machine vision monitoring is supported by the increased use of robotic systems in machine cells and the advanced development of integrated machine vision systems and robots.

Machine vision is the receiving of an image, using optical sensors, and interpreting this image with the aim of obtaining information about the objects in the image. The process of machine vision broadly entails acquisition of an image and its analysis. Image acquisition comprises image formation, sensing and digitization. Preprocessing, segmentation, feature extraction and interpretation fall into the category of image analysis.

The developed tool wear monitoring and tool failure detection system is composed of a microscope, a fibre optic illuminator, a Zenith IBM/XT compatible microcomputer with Image Capture Board, a CCD RGB video camera and a colour

monitor. The system algorithm is implemented in Microsoft C V5.1.

Chapter two is devoted to cutting tools and machine vision, concepts which are different in all aspects, each with its unique terminologies. A brief review of cutting tool history, cutting tool material, chip formation and various forms of wear is presented. In addition, factors affecting tool performance are discussed and the machine vision process and its application in various areas are introduced. This chapter also gives a general overview of the literature in the area of tool monitoring and failure detection. Techniques which monitor tool wear via an in-direct method of correlating machining variables with wear behaviour are presented under off-line monitoring. The techniques which give a direct measurement of tool wear are discussed under on-line monitoring.

The proposed system and the algorithm to implement it are presented in chapter three. This chapter contains a detailed description of the system components and setup. It also explains the thought process behind the development of the algorithm and uses a flowchart to present an overview of the system control logic. The system uses five features to determine if a tool is worn or broken. These features are: length of the wear land (W_l), area of the wear land (A_w), perimeter of the wear land (P_w), maximum width of the wear land (W_w) and the nearest tool edge point to the original tool tip (x_n, y_n).

Experiments were designed to test the developed algorithm capability to detect tool breakage and to monitor tool wear over a brief cutting period. The latter was implemented twice. The first used an austenitic stainless steel bar as the workpiece and the second used an AISI C1045 steel bar as the workpiece. In the second experiment, measurements of wear width were also made using a conventional method

(microscope). The results obtained using the vision system and the conventional method showed no significant difference. Details of the experiments and results are presented in chapter four. Finally chapter five contains conclusions and recommendations for further research.

Chapter Two

Background And Literature Review

This chapter discusses the basis of cutting tool and machine vision. A brief discussion of the history of cutting tools, chip formation, tool geometry and materials, and tool wear and failure is presented. It also discusses machine vision history and processes. Finally, it reviews the research efforts to develop fully automated tool monitoring techniques. These techniques are classified into on-line monitoring system and off-line monitoring system.

2.1 Cutting Tools

Cutting tool history dates back to the latter part of the 18th century [8]. The purpose of a cutting tool is to remove material under controlled conditions. The first cutting tool, a single-point cutting tool, was developed by John Wilkinson in 1775 and this was used in 1776 by James Watt to build the first successful steam engine [25].

Cutting tools are generally classified as single-point tools or multi-point tools. Single-point cutting tools are simply tools with one cutting edge and are often used in turning, shaping, and boring operations. Multi-point cutting tools are an

aggregation of two or more single-point tools which are securely held on a solid body. The tools on a multi-point tool are arranged such that each tool point contributes to the removal of the workpiece material. Examples of multi-point cutting tools are drills, milling cutters, and broaches.

In order for a cutting tool to achieve its objective, its material must be harder than the workpiece material. Furthermore, the workpiece should provide some resistance to the applied forces because cutting is made possible by this resistance. The material removal process is initiated with the compression of the material, followed by a flow of the material, and eventual separation from the base metal (workpiece).

The efficiency of material removal is dependent among other factors on the following:

- tool geometry
- relative hardness of the tool and workpiece materials
- cutting forces
- tool sharpness
- structural rigidity.

During a cutting process, the tool first deforms a layer of the workpiece material elastically and then separates the material near the cutting edge of the tool by plastic deformation. The type of chip produced depends on the material being machined, tool geometry, tool material, and the prevailing cutting conditions. Each of these factors or a combination affects tool life, tool wear, power consumption.

The prevailing cutting conditions - feed (f), depth of cut (d) and cutting speed (s) - are illustrated for a turning operation in Figure 2.1.

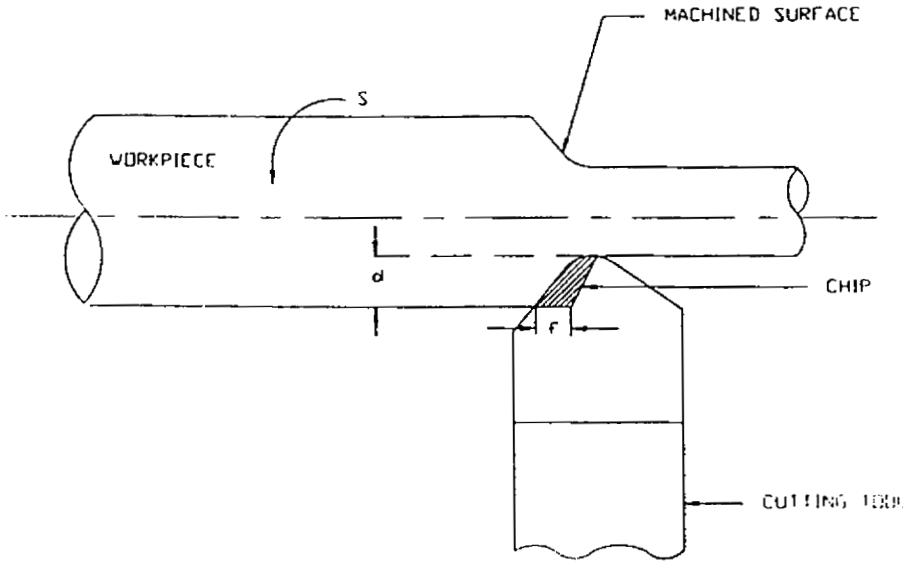


Figure 2.1: Schematic of a basic turning operation

Depth of cut, measured in mm, is defined as the depth at which the cutting edge engages the workpiece. The cutting speed is the relative speed between the tool and the workpiece and it is expressed in m/min. Feed is expressed differently for various machining operations - turning, shaping, planing, drilling, milling. In turning operations, it is defined as the axial advance of the tool along the workpiece during each revolution of the work and expressed in mm/rev.

The nature of the chip formed is classified as continuous, discontinuous, or built-up edge (see Figures 2.2, 2.3 and 2.4).

In previous paragraphs it was mentioned that the shape and relative position of a tool play an important role in machining process. The size of a tool is expressed in terms of its width, height and length. Figure 2.5 depicts the various parts and

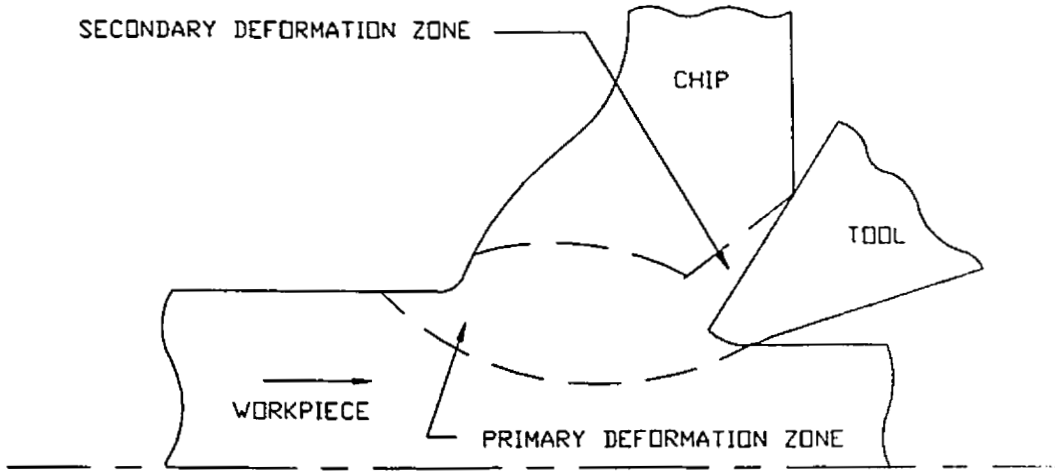


Figure 2.2: Continuous chip

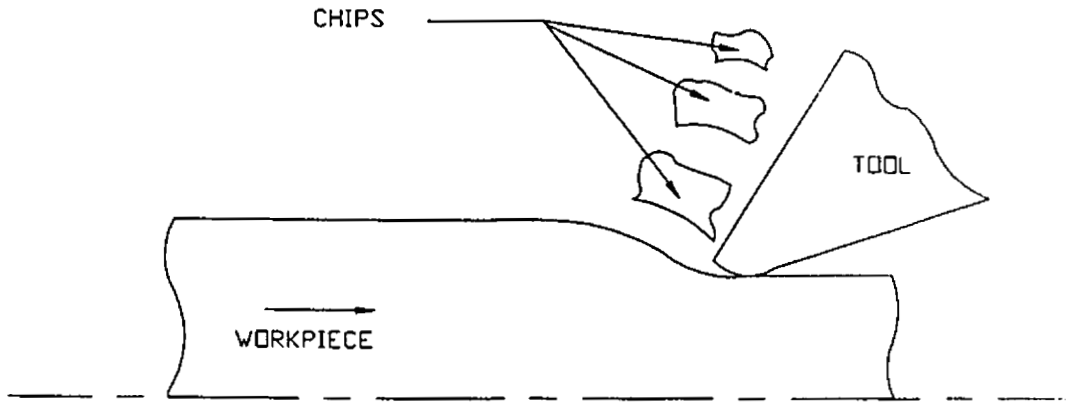


Figure 2.3: Discontinuous chip

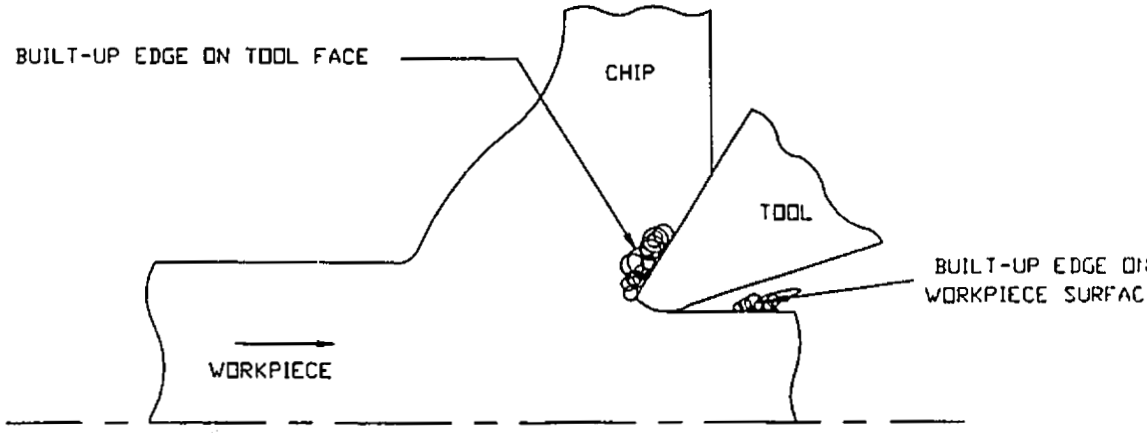


Figure 2.4: Built-up edge

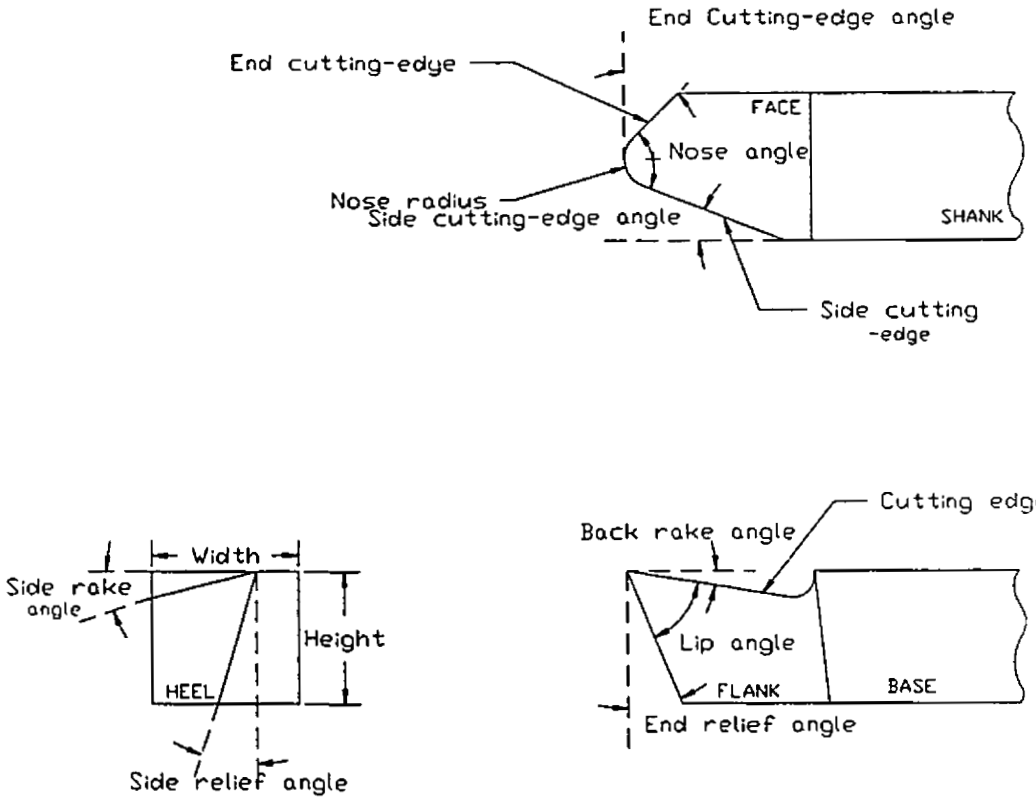


Figure 2.5: Nomenclature of a single point cutting tool

geometric parameters of a tool. The definition of these parts and the geometric parameters [7,8,25] are presented in the next three paragraphs.

The *shank* is the main body of the tool and it is the portion that is held in the tool holder. The bottom part of the tool is called the *base*. The *heel* is at the front of the tool and is adjacent to the base and flank intersection. As chips separate from the workpiece, they impinge on the surface of the tool. This surface is referred to as the *tool face*. The region of the tool shaped to produce the cutting edges and the face is called *tool point*. The *cutting edge* consists of side cutting edge, nose, and end cutting edge. It is the edge of the face which separates the chip from the workpiece. The side cutting edge and the end cutting edge are joined by a radius, chamfer or arc called the *nose* while *flank* is the surface below the side cutting edge.

Angles on cutting tools are standardized and depend on the type of cutting. *Relief angles*, sometimes called *clearance angles*, are primarily used to prevent rubbing between the non-cutting parts of a tool and the workpiece.

The *side rake angle* is defined in a plane perpendicular to the centerline of the shank and it is the angle between the face of the tool and the base. *Back rake angle* is defined on the plane parallel to the longitudinal centerline of the shank. It is the angle made by the face and the base of the tool and can be either positive or negative. The former slopes downwards and reduces the lip angle while the latter slopes upwards and increases the lip angle. The angle between a plane perpendicular to the base and the end flank is called *end relief angle*. *Side relief angle* is the angle between the flank and a plane perpendicular to the base just under the side cutting edge. The *end cutting angle* is defined as the angle between the face and a plane perpendicular to the side of the shank. The angle between the side of the shank and the line of intersection of the flank and the face is defined as the *side cutting edge angle*. And the radius which connects the side cutting edge with the end cutting edge is called the *nose radius*. It is worth mentioning that this connection could also be a chamfer.

Because machining processes involve high local stresses, abrasion, friction, and generation of heat, a cutting tool material must possess high strength, toughness, wear resistance, and hardness properties at either low or high temperatures. However, the retention of these properties at elevated temperature is more important because temperatures during cutting are very high. In a corrosive environment, a cutting tool must also be corrosion resistant.

Cutting tool materials include tool steel (low and medium alloy steel), high-speed steel, sintered carbide, ceramic and diamond. The use of a particular tool material is influenced by machine rigidity, cutting temperature, required surface finish, roughness of workpiece and workpiece material.

2.1.1 Tool Failure and Life

A tool is said to have failed if its use causes one or a combination of the following:

- poor surface finish
- dimensional failure
- increased power requirement
- decreased production
- increased scrap rate.

Tool failure could be caused by structural defect or failure, tool geometry or tool wear. Lack of rigidity of the tool holder, the machine or the workpiece can lead to structural failure, which is usually sudden, and leads to the premature end of a tool life. A tool must have the necessary geometry for a particular job if it is to meet its objective. Any discrepancy in the tool geometry can lead to one or a combination of the aforementioned indicators of a failed tool. Although tool geometry and structural failure lead to premature tool failure, the most important factor is tool wear [7,21]. This is because failure due to tool geometry and structural defects can be readily controlled.

Tool wear can be defined as the total loss of weight or mass caused by friction between the sliding parts. The mechanism of wear differs for different ambient

conditions. However, six basic mechanisms can be identified in cutting operations. These are:

- diffusion wear
- adhesion wear
- oxidation wear
- abrasion wear
- electro-chemical wear
- electrolytic and chemical wear.

The high temperature during cutting and the intimate contact between the tool and workpiece can cause diffusion wear which is a surface and interstitial diffusion of atoms from the tool to the workpiece. The amount of diffusion is dependent on the degree of atomic agitation, bonding affinity of the tool and workpiece, and the length of time the ambient temperature is high enough to cause diffusion.

The tool and workpiece material can be welded across the contact region by the high force and friction present in the region. The fracture of the weld tears out small fragments of the tool material which are carried away by the chips or deposited on the new workpiece surface. This is called adhesion wear and is dependent on the composition of the tool and workpiece. Furthermore, it is influenced by the forces applied between the tool and workpiece.

The high temperature existing in a cutting process could lead to the oxidation of carbide in the tool material. This will weaken the tool matrix and consequently reduce the strength of the tool thus causing oxidation wear.

It is not surprising to find hard-particles in any cutting operation. These particles may be hard constituents of the workpiece material, highly strain-hardened fragments of an unstable built-up edge or hard constituents of the tool material. Abrasion wear occurs when these particles plow into the tool surfaces as they sweep over the tool and remove the cutting tool material by mechanical action.

The high temperatures prevailing at the junctions between a cutting tool and the workpiece can generate a thermo-electric electromagnetic force. This leads to a circulation of large electric currents and consequent transfer of ions between the tool and workpiece. This results in electro-chemical wear which is a breakdown of the tool material in the region of the chip-tool interface.

Electrolytic and chemical wear occur only in chemically active environment. If a chemically active cutting fluid is used, there is the possibility of the tool and workpiece reacting chemically and resulting in a loss of tool material. This is called chemical wear. Electrolytic wear is caused by galvanic corrosion between the workpiece and the tool.

2.1.2 Types of Wear

Gradual wear of a cutting tool could be caused by chips flowing along the tool face, creating a crater, or by the rubbing action of the newly created workpiece surface on the flank. The former is called crater wear and the latter is called flank wear (Figure 2.6).

Crater wear occurs during the machining of ductile materials at low speeds. These conditions favour the creation of a large built-up edge which causes abrasion and adhesion wear as it flows over the tool face. Diffusion could also be a factor in

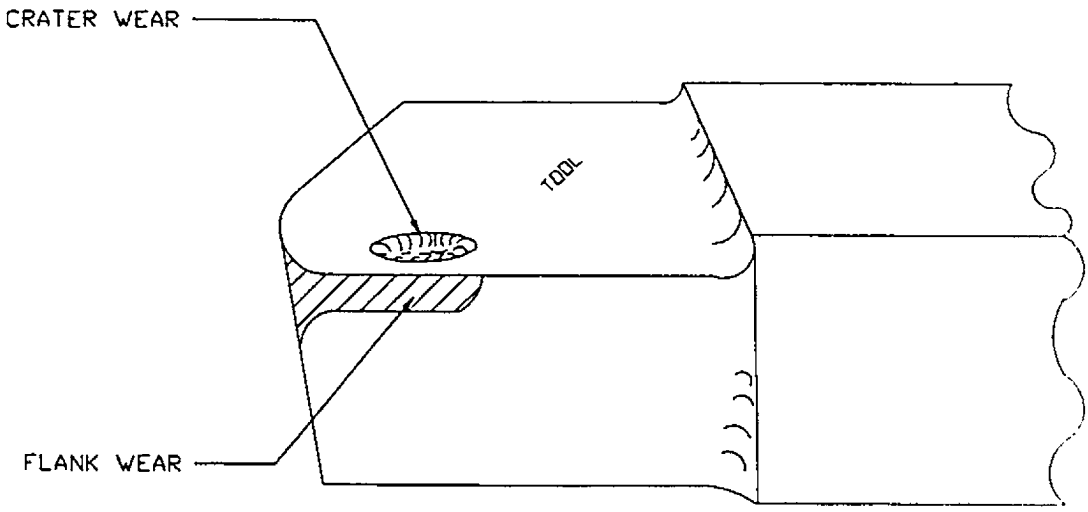


Figure 2.6: Schematic of flank and crater wear

the creation of crater wear because it occurs at a certain distance away from the tool tip where the maximum temperature is reached on the chip-tool interface [8].

From the above, it is seen that crater wear is prominent in high speed machining. The end result of crater wear is the weakening of the tool edge and eventual fracture.

Flank wear is the eroding of the tool flank and it is the factor used in the determination of tool life. Figure 2.7. illustrates the development of flank wear in cutting operations. From this figure, it is seen that flank wear can be grouped into three stages. The first stage, AB, is a period of rapid wear. This represents the initial blunting of an originally sharp tool. Region BC, the second stage, is a region of gradual wear. The third region, CD, is the catastrophic wear zone.

In machining processes, an attempt is made to avoid the development of the third stage because it will lead to increased temperature and consequent weakening

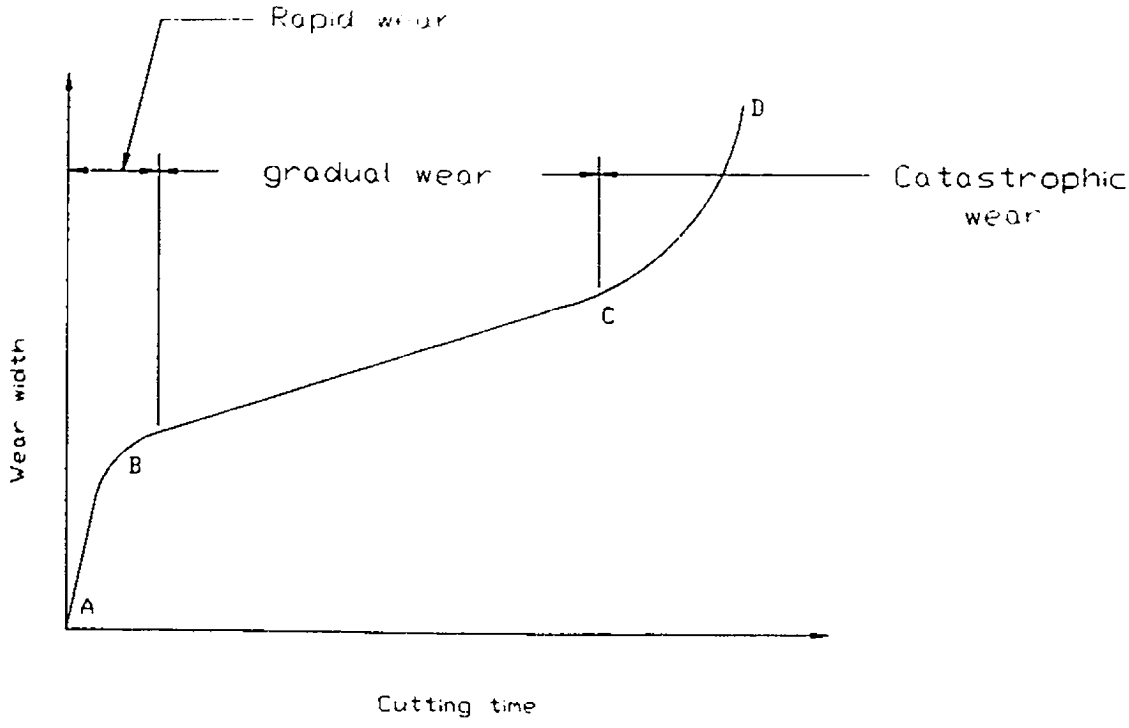


Figure 2.7: Influence of cutting time on flank wear

of the tool.

The parameters indicating tool failure are also used to determine tool life. Thus, it is seen that tool life is dependent on cutting conditions and the requirements of the operation. Tool life can be defined as the time elapsed between two successive repair grindings of a cutting tool, or time between replacement of a new tool or tip.

2.2 Machine Vision

Machine (Computer) vision, as defined by Machine Vision Association of the Society of Manufacturing Engineers (MVA/SME), is the use of devices of optical non-contact sensing to automatically receive and interpret an image of a scene, in order to obtain information and/or control machines or processes [36]. The first experiments in computer vision date back to the late 1950s [4]. However, the work of L. G. Roberts in 1963 is taken as the first serious work in the field of computer

vision [11].

Machine vision is an interdisciplinary field of study [17]. It uses the continuous and discrete subfields of mathematics. Machine vision attempts to understand the most valuable computational visual processes from biological and psychological field tests on humans. Since pictorial images lack depth, machine vision looks on numerical analysis, solid geometry and calculus of variations to infer depth from the flat images. Machine vision depends on graph theory, pattern recognition and statistical inference in order to develop algorithms that will match the pictorial image (input) to the most likely known parts and relations. The theories on vision are tested and explored using data structures, search techniques, software engineering tools and complexity analysis provided by computer science and artificial intelligence. Lastly, electrical engineering provides machine vision with signal processing techniques, control theory for its robotic systems, and VLSI and system design for its special purpose parallel hardware.

In a manufacturing system, the primary objectives of a machine vision system for monitoring tool condition are to improve production quality, increase productivity and decrease production costs.

The process of machine vision broadly entails the acquisition of an image and its analysis. Image acquisition is comprised of image formation, sensing and digitization. Preprocessing, segmentation, feature extraction and interpretation fall into the category of image analysis.

Image acquisition is a process to represent physical objects in a pictorial form and to transform this pictorial information into a form discernable to a computer. This eventual form is called a digital image. Rosenfeld [26] defined a digital image

as "a discrete array of numbers representing brightness or colour values at discrete grid points in the image plane." In more precise terms, the array values are averaged values in the neighbourhood of individual grid points. The elements of the array are called picture elements, or pixels, because each represents a discrete element of the digital image.

One or several values are measured at each grid (or sample) point and used to represent the point. When one value is measured, it is usually the brightness at the sample point or an averaged brightness of the sample point neighbours. This is called the gray level and it is used in most computer vision applications. When colour information is needed, several values are measured at each sample point and they represent brightness measurements in a set of K spectral bands.

A paradigm for interpreting images in machine vision can be represented as shown in Figure 2.8. This figure is used in the following paragraphs to explain the various steps involved in the receipt and interpretation of an image. For the purpose of this thesis, only the sections that are relevant to the research are described. Detailed explanation of the process of receiving and interpreting images can be found in [4,12,20,23,27].

Image formation consists of image sensing and digitization. Image sensing transforms optical information into electronic information by the use of cameras or scanners. Thereafter, a periodic sampling of the video data produces a digitized image.

The digitized image is not always noise free. Most of the noise is introduced by the camera or scanner and lighting arrangements. Thus, an early or preprocessing of the digitized image is usually implemented to filter out the noise and/or enhance the image. This is achieved by emphasizing needed features while reducing or

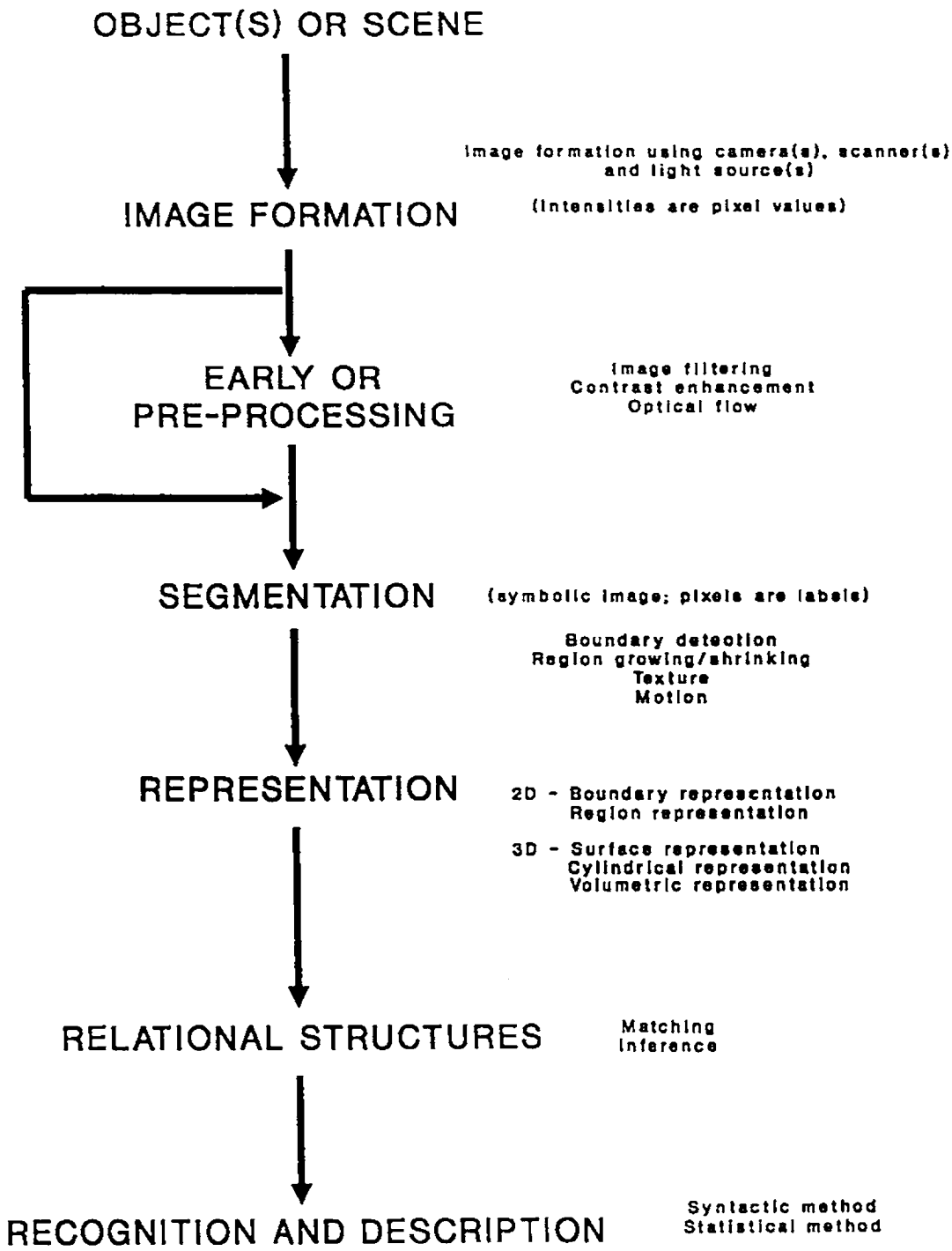


Figure 2.8: A paradigm for image interpretation

eliminating unwanted ones. Some of the common methods used are histogram manipulations (No. of pixels vs. Gray level) and noise filters.

Histogram manipulations can be divided into histogram sliding and histogram stretching. Histogram sliding is simply adding or subtracting a constant brightness to all pixels in the image. When this is done, the histogram of the output or modified image is shifted either to the left or to the right of the original or input image histogram. The end effect is a brightening or darkening of an image. It also has the advantage of sliding the gray level range into a new range that will be helpful in subsequent operations. Histogram stretching is accomplished by multiplying all pixel gray levels in an image by a constant factor. The histogram stretches when the factor is greater than one and shrinks when the factor is less than one. This results in an image with increased or decreased contrast depending on the increase or decrease of the dynamic range of the image.

Histogram manipulations are carefully done in order not to exceed the white gray level limit or create a loss in brightness resolution. It is worth mentioning that histogram stretching and sliding are not necessarily used separately as they can be combined for image enhancement.

Noise filters, often called smoothing operators, are masks used in performing spatial filtering. Spatial filtering is the separation of frequency components within a two-dimensional base of data. The frequency components relate to rate of gray level change over a certain spatial distance. These filters are normally classified as high pass filter or low pass filter [6,12]. Another form of the filter is called median filter.

A low pass spatial filter will accentuate low spatial frequency details and attenuate high spatial frequency details. The converse is true for high spatial filters. Low and high pass spatial filters, unlike median filters, are spatial convolution filters because they are based on the weighted average of nine pixels in a 3 x 3 kernel. With reference to an original image, images obtained from low pass spatial filters are blurred while those from high pass filters are sharpened. Median filters are very effective in highly spiked images but have the disadvantage of distorting information especially in images with a large percentage of high spatial frequency.

This stage, preprocessing or early processing, is ignored in situations where the effect of noise is insignificant.

The segmentation of an image into its constituent parts is usually carried out after preprocessing or image formation. This is an important step in computer vision because it is used in the extraction of entities or features from an image for analysis, i.e., representation, description and recognition. Segmentation is defined as a process which "identifies distinctive subpopulations of pixels, or partitions the image into the connected regions each of which is 'homogeneous' in some sense" [26].

Algorithms on segmentation are based either on discontinuity or similarity. They generally depend on absolute values of a gray level and differences of gray levels as indicators of partitions or segments. When discontinuity is used as a basis for segmenting an image, the partitions are obtained from sudden or abrupt changes in gray level. Detection of isolated points, line and edges in an image are some examples of segmentation algorithms based on discontinuity of gray levels. Of these algorithms, edge detection has generated the greatest interest. This can be attributed to two

main reasons:

1. Man can recognize objects through their boundaries which are composed of edge segments.

2. Image recognition based on texture properties to a large extent depends on edge detectors.

The boundary of an object is an aggregation of local edges which correspond to local gray level discontinuties. A local edge, which is a small region in an image where the local gray levels or intensities are rapidly changing, is determined by the use of edge operators. An edge operator can be defined as a mathematical operator or its computational equivalent designed to determine the presence of a local edge in an image (or image function).

Several researchers are involved in the development, implementation and use of edge detection (or operator) algorithms. It has been shown [1] that the performance of these algorithms is dependent on the prevailing testing conditions. The algorithms can be categorized as either sequential algorithms or parallel algorithms [31].

Sequential algorithms involve the use of *a priori* knowledge about the shapes of the image to be processed. Huckel operator and zero crossing techniques fall into this category. Sequential techniques are sensitive to noise which can be minimized by the use of the filters discussed earlier. However, the greatest disadvantage of these techniques is the dependence on *a priori* knowledge of the shape of the objects in the image.

Unlike the sequential algorithms, edge detectors using parallel algorithms are

independent of *a priori* knowledge about the shapes of the objects, but are sensitive to noise. Some of these are the differential based operators, viz. gradient and laplacian. Other examples of this method of edge detection include template matching and edge fitting operators. There are other methods of edge detection based on statistical inferences [14].

Thresholding and region-oriented segmentation are examples of segmentation methods based on gray level similarity. The differentiation of an object in an image from its background such as in printed or written document can be done using thresholding. This is more effective in images where the objects and backgrounds are homogeneous or uniform [27,35]. The general concept behind thresholding is to map image points whose gray levels are above a threshold to a certain gray level and others to another gray level. Thresholds could be described as global, local, or dynamic. A detailed analysis of thresholding techniques can be found in [15,33,34,35].

In region-oriented segmentation, segmentation is approached by finding regions directly. The essence is to partition a region, say R, into n subregions, say R_1, R_2, R_3, R_4, ..., R_n, based on some constraints. There are basically two types of region-oriented segmentation techniques: region growing by pixel aggregation, and region splitting and merging.

Region growing by pixel aggregation implies starting with a group of pixels and appending to each pixel in the group those neighbouring pixels that have similar properties. This property could be colour, connectivity (see Appendix A), gray level, or texture.

Unlike region growing by pixel aggregation, region splitting and merging initially

partitions an image into arbitrary, disjointed regions and then merges and/or splits the regions based on some constraints. A quadtree, a tree with each node having four children or descendants, database system is often used.

The output image of a segmentation process is "a 'symbolic image' in which the pixel 'values' are labels rather than gray levels" [26]. The next step after segmentation is that of representing and describing the segmented image in a form suitable for further computer processing. Representation is usually approached in one of two ways: representation by external characteristics and representation by internal characteristics. The former is used in situations where shape characteristics are paramount and the latter is used where reflectivity or textural properties are of interest. Describing the segmented image involves selecting features which are independent of variations in size, translation and rotation.

In three dimensional images, representation is usually achieved through the use of surface, cylindrical, or volumetric representation. Two dimensional images are generally represented using boundary or regional representation schemes. Boundary representation schemes include polylines and chain codes (see Appendix B). Regional representation includes quad trees and medial axis transformation.

At the end of representation, a further study of the available data is needed to understand the information they contain. One of the various techniques in use is shape analysis. Shape analysis is done using boundary, regional or relational descriptors. Boundary descriptors include area, length, curvature, diameter, shape number, Fourier descriptors and moments. Compactness - a ratio of square perimeter to area, Euler number, and topological descriptors fall into the category of regional descriptors. Relational descriptors assemble the components of an image

in such a way that any structural relationship between them are readily recognized. This is achieved using string grammars and languages. A detailed analysis of representation and shape analysis can be found in [4,12,20,23,27].

The final stage of image interpretation is the comparison of the information obtained from relational structures with those resident in the machine vision system. This comparison which is implemented by syntactic or statistical methods helps in the understanding and interpretation of the collated information.

What constitutes a machine vision application is still a subjective decision as it is often confused with image processing and pattern recognition [11]. According to [4,11,12,20,27], however, machine vision has many applications in areas such as robotics, remote sensing, aerial images, medicine, physics, chemistry, neuroanatomy, document processing, assembly line tasks and quality control.

2.3 On-Line Monitoring

On-line monitoring systems, also called indirect monitoring systems because they do not give a direct measurement of wear, are based on the acquisition using sensors, and analysis of one or more machining variables - force, temperature, torque, power, vibration amplitude and acoustic emission. These sensors are force, torque, vibration and acoustic emission detectors. The following paragraph briefly explains reasons for their use.

A force detector is used on the premise that unique force signals are transmitted by a worn tool or fractured tool. The heat produced due to an increase in frictional resistance provides the basis for using temperature as an indicator of tool breakage and wear. Vibrational signals on which the vibration detection technique is based

are generated by the fluctuation in force due to tool wear or breakage. The rapid release of energy by a material under strain generates high-frequency sound which forms the basis of the acoustic emission technique.

Subramanian and Cook [28] have developed relationships between drill flank wear on the one hand and torque and thrust force on the other. From these relationships, it can be inferred that under a constant workpiece hardness, the thrust force or torque will increase linearly with drill flank wear, irrespective of the cutting conditions. However, this assumption is an over-idealization because, in practice, the hardness of a workpiece is never constant and thus, random variations are created in the thrust force or torque under normal drilling conditions.

The use of the thrust force gradient rather than its magnitude in predicting tool failure has been suggested by Thangaraj and Wright [29]. This is due to the following reasons: the thrust force exerted by a drill in failure is dependent on the drill's geometry and the ambient cutting conditions; cutting conditions influence the magnitude of flank wear which leads to failure; and thrust force level may not show any significant change until excessive damage has occurred.

Experimental observations attributed the occurrence of sharp fluctuations in the thrust forces recorded for a drill under failure conditions to a microscopic stick-slip phenomenon. The stick-slip phenomenon is a cyclic process where the 'stick' is a macroscopic seizure caused by local welding and 'slip' is a macroscopic release caused by shear fracture. This phenomenon is mainly caused by the high temperature conditions at the outer corner of the drill which cause greater wear and which in turn leads to higher temperatures. However, the unsuitability of the technique for diverse cutting conditions limits its utilization in an unmanned manufacturing

environment.

Li and Wu [18], in an attempt to develop a system suitable for diverse cutting conditions, proposed a technique with learning ability and which is based on a fuzzy-C means algorithm. The algorithm consists of the following: feature selection; clustering; and classification. Clustering centres are found by an optimization procedure and a cost function is introduced in the classification process to grade the tool defect as 'initial', 'small', 'normal', or 'severe'.

The use of force signals in detecting tool breakage has also found application in milling operations. Altintas *et al.* [2] have developed, using a real-time algorithm, a scheme for processing force signal to detect cutter breakage. They modelled the difference in behaviour between adjacent teeth because in steady state cutting a perfect cutter repeats the same pattern for each tooth period. From the collated force data, a sharp change in distribution indicated a transient cutting condition (exit or entry of the cutter), tool breakage or tool run out.

Although the above technique showed that force averaging and normalized differencing can be used in designing tool breakage algorithms, the authors acknowledge the inherent difficulties in setting thresholds and in understanding the physics of very sharp transients in the force data.

In a machining process, the different changes in the machine tool and the workpiece has a significant influence on the vibrational behaviour of the active mechanical or machine components. Thus, it is seen that the vibration frequency spectrum could provide information concerning the actual condition of the machining process.

Bartal and Monostrori [5] have developed dedicated hardware-software for predicting tool conditions by monitoring vibration. They categorized this process into

four phases. The first phase, data acquisition, samples and converts selected analog signals to digital mode. In the second phase, preprocessing, the difference between the various measurements is determined via a transformation which is done in frequency mode. The third phase involved the use of pattern recognition techniques in the computation of cepstral and spectral features from the frequency spectrum. Drawing conclusions based on the output from the third phase is the last phase.

The hardware module has four functional blocks; HOST and DSP on one hand, and global RAM and Data acquisition on the other. The HOST has a 16-bit HOST CPU, 16kbytes EPROM and 16kbytes RAM as its components. The software module has two levels in order to exploit the parallel architecture of the hardware by distributing different operations between the processors.

Acoustic Emission (AE) signals, high frequency sound generated by the rapid release of energy by a material under strain, have been used in tool wear monitoring and tool breakage detection. During a machining process, acoustic emission signals are generated by the plastic deformation of the workpiece and by tool breakage. These signals can be analyzed using count-rate, RMS voltage, or spectral density technique.

Emel and Kannatey-Asibu [10] have developed a tool monitoring system based on the analysis of the acoustic emission signals from the machining process. Their method involved the collection of AE signals during turning of work materials on a CNC lathe. The work materials are heat treated to provide the necessary hardness to induce wear, and a threshold value was assumed for the flank wear.

Pattern recognition techniques are used to obtained two decision strategies: simultaneous decision strategy and binary tree decision strategy. This is further

amplified by the use of either a minimum cost or minimum error criterion. In simultaneous decision strategy, only the physical characteristics of the machining process, i.e., sharp tool, worn tool, chip noise, tool breakage, are considered. Binary tree decision strategy determines whether the signal is continuous or transient. Continuous signals are emitted by sharp or worn tools while tool fracture or chip breakage emits transient signals.

Iwata and Moriwaki [13] have also tried to correlate acoustic emission signals with tool wear. They observed the occurrence of a burst type acoustic emission during a tool wear process and that the power spectrum increased with initial tool wear before saturation. The saturation value increased with cutting speed and three distinct regions were recognized during a count of the acoustic emission signals. These regions are said to be caused by the three phases of flank wear.

The above results notwithstanding, the authors acknowledged the need to carry out further research on the proper selection of threshold voltage and better understanding of the effect of friction and chips on acoustic emission. In addition, there is the need to experiment over a wider range of cutting conditions.

2.4 Off-Line Monitoring

In an automated tool monitoring system using on-line techniques, the decision to replace or resharpen a tool is governed by the signal(s) received from the sensors. In most cases, human beings are left to examine the state of the tool; thus, the system ceases to be fully automated. Off-line monitoring systems, sometimes called direct methods because they provide a direct measurement of the wear, are systems designed to replace this supervisory role. The sensors used in this technique include

radioactive, electrical resistance and optical sensors. The following sections briefly explain each of these sensors.

2.4.1 Radioactive Sensors

The general concept behind the use of radioactive sensors is to carry out a machining operation using activated (radioactively) cutting tools and to collect the produced chips for a radioactive count. This count, i.e., intensity of radioactivity, is correlated with the volume or mass of total material that adhered to the chips, thus giving an indication of tool wear. The cutting tools consist of materials with long half-life and are usually activated by neutron bombardment in a nuclear reactor. Half-life of a radioisotope is the time needed for the radioactivity of an element to decay to half its initial value. Early studies in this area were carried out by Merchant *et al.* [19] and by Opitz and Hake [22].

Merchant *et al.* [19] determined instantaneous rate of tool wear by measuring the amount of radioactivity of collected particles worn from a tool during a given period of machining. This was done by measuring the mass of chips produced. They established that the amount of radioactivity was directly proportional to the amount of material worn from a radioactive tool. By measuring the radioactivity of collected wear particles under different cutting conditions, they obtained relationships between rate of tool wear and tool life for these different conditions.

The graph of radioactivity against cutting time displayed a linear relationship between the two parameters. The relationship between accumulated radioactivity in cutting fluid and total cutting time was also found to be linear and it was observed that the use of different cutting fluids affected the rate of tool wear. Furthermore, the relationship between accumulated radioactivity and flank wear on one hand and

on total cutting time on the other, were established; they were found to be linear. The same was deduced from the graph of tool wear against cutting speed.

Opitz and Hake [22] in their work used counts per minute per grams of chip, which they called pulse frequency, to relate to the amount of tool wear. Their methodology is identical to that of Merchant *et al.* but they went a step further in isolating wear of the top rake face and the back relief angle. Their results showed that the volumetric wear for the front and for back flank increased linearly with cutting time. Relationships between tool life and cutting speed for back relief face wear and top rake face wear were also deduced.

The above experimental setups are unsafe because of the high likelihood of exposing workers to fatal levels of radiation. Moreover, the specialized nature of the experiments and the need for specialized equipment, all of which translate to high costs, is rather disadvantageous. All these explain the lack of research in the area of tool wear analysis using radioactive sensors and its practical application.

2.4.2 Electrical Resistance Sensors

The signals generated by electrical resistance sensors can be used in detecting wear. During a cutting process, the wear of a tool increases its contact area with the work, thus reducing the electrical resistance across the tool-workpiece junction.

A technique proposed by Uehara [30] uses a film resistor. The film resist or is made in two ways ; either a printed resistor with graphite ink (a mixture of graphite powder and phenol resin bond) or a metal film resistor which is formed by vacuum evaporation of chromium using a mask.

The resistor is printed on the clearance surface of the tool and is insulated by a thin layer of heat resisting paint. As machining progresses, the length of the resistor

is found to decrease with the progress of flank wear and it is measured in-process by the change in electrical resistance.

The results showed that the shape of the heat resisting paint rubbed by the machined surface workpiece is identical to the shape of the flank wear. Furthermore, the width of flank wear was found to be inversely proportional to the change in electrical resistance. Although this technique is quite effective, its specialized nature - film resistor and tool coating - means added machining cost.

2.4.3 Optical Sensors

Techniques involving the use of optical sensors apply the theories of computer vision which have been introduced earlier in this chapter.

Cuppini *et al.* [9] have developed a computer vision tool wear monitoring system. The system uses a TV camera for capturing a tool image. During the digitization phase, analog image signals are thresholded so that the eventual digital image will have the wear land represented by white pixels and the other parts of the tool are left black, thus creating a binary image. A value V_{Bmean}, the number of horizontal lines in the wear land, is determined and used in reporting the wear state of the tool. In observing tool failure, they suggest that there is a loss in material which can be interpreted as a change in the tool cutting edge shadow. Thus, a comparison of the tool templates before and after a cutting process will reveal the occurrence of breakage.

The above method has two main disadvantages. First, it fails to give a true idea of the wear land dimensions. This is because wear state could be better presented by determining such parameters as the length, width and area of the wear region. Furthermore, the feature V_{Bmean} does not give adequate information about the wear

because it does not provide real dimensions (i.e., standard units of measurements). Second, although tool breakage is usually associated with a change in the tool shadow, the use of template matching of the tool before and after cutting provides a trivial solution to detecting breakage. Also, its need for large computer memory space and high CPU time makes it computationally prohibitive.

An attempt at measuring other parameters of wear land has been made by Lee *et al.* [16]. They investigated flank and crater wear in a tool using a VICOM image analysis system. The tool to be investigated is positioned using a special fixture and the fibre optic lighting arrangement is adjusted to provide contrast between the tool wear land and the remainder of the tool body. The image is captured using a solid state camera with 480 * 380 pixel resolution which is connected to a microscope. The video data obtained from the camera are passed through a Hotronics time-base corrector to eliminate synchronization problems on the VICOM image analysis system during digitization. At the end of the digitization, a segmentation of the image is done interactively using menu utilities of the VICOM image analysis system. Parameters calculated from the image of flank wear include: average wear-land width, total flank wear area, and maximum wear land width. These same parameters are calculated for crater wear with the average nose radius being also determined.

The results of experiments on the system were different from those obtained using a toolmaker's microscope. This difference is explained by the fact that the machine vision system, unlike the toolmaker's microscope, includes dimensional changes caused by eroded tool surfaces in its computation of the wear dimensions. Furthermore, it was observed that there was a sudden increase in the chipped area

and nose radius during the occurrence of tool failure.

The above results are encouraging; however, the research failed to consider the possibility of a breakage. In addition, the use of an interactive segmentation is a drawback which is readily acknowledged by the authors. The accuracy of the system is dependent on both human control and the light source.

The basis of cutting tools and machine vision, and the different methods used in monitoring tool condition during cutting have been discussed. It is observed from this discussion that the present machine vision based methods for monitoring tool condition do not detect tool breakage. Furthermore, they fail to take advantage of the machine vision flexibility and capability by not providing more information on wear land.

The next chapter will discuss the configuration of the newly developed system to monitor tool wear and detect tool breakage. Furthermore, the chapter presents and discusses the development of the system algorithm.

Chapter Three

Development of the System

This chapter presents and discusses the development of a system which corrects the disadvantages mentioned in the previous chapter. The system reports on the wear state and breakage of cutting tools for a laboratory CNC lathe. The chapter also discusses the configuration and operation of the developed system.

3.1 System Setup and Operation

The proposed setup for monitoring tool wear and detecting breakage using machine vision technology is illustrated in Figure 3.1. The system components are: CCD RGB video camera, microscope, AT&T image capture board, fibre optic illuminator, Zenith IBM/XT compatible microcomputer and colour monitor. In an industrial setup, the microscope will have to be replaced by a magnifying camera lens and the tool will be mounted on a CNC lathe machine.

A fibre optic illuminator is used to obtain uniform light intensity and its illumination can be guided to provide desired contrast among the various regions of an image. The microscope which is set at a magnification of x40 magnifies the tool since the width of a wear land is less than $1mm$.

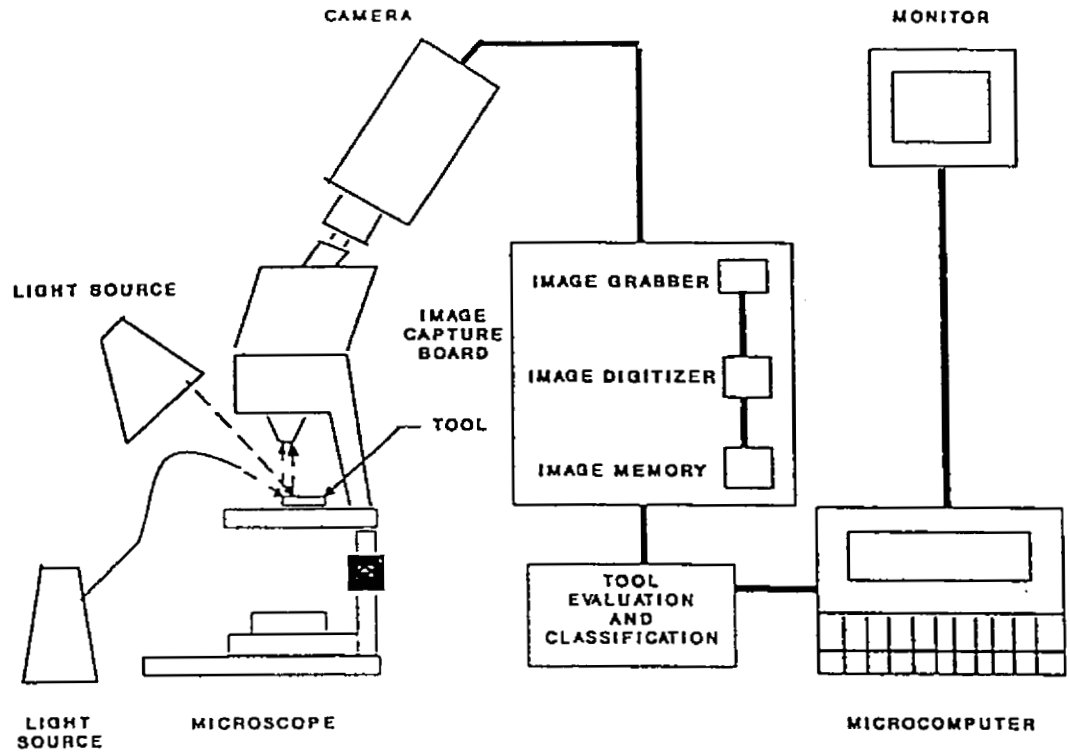


Figure 3.1: Schematic of the system setup

During a monitoring process, a tool is positioned on the microscope platform such that the camera focuses on its flank side. The lighting arrangement is adjusted to provide some contrast between the defective tool region and the remainder of the tool body. The image of this flank side is sensed by the CCD RGB video camera which then sends the electronic information (video data) to the AT&T image capture board for capture and digitization. The microcomputer analyzes the acquired image, extracting the necessary features. A comparison of the features is made with some threshold values and the microcomputer reports on the state of the tool.

3.2 Feature Selection

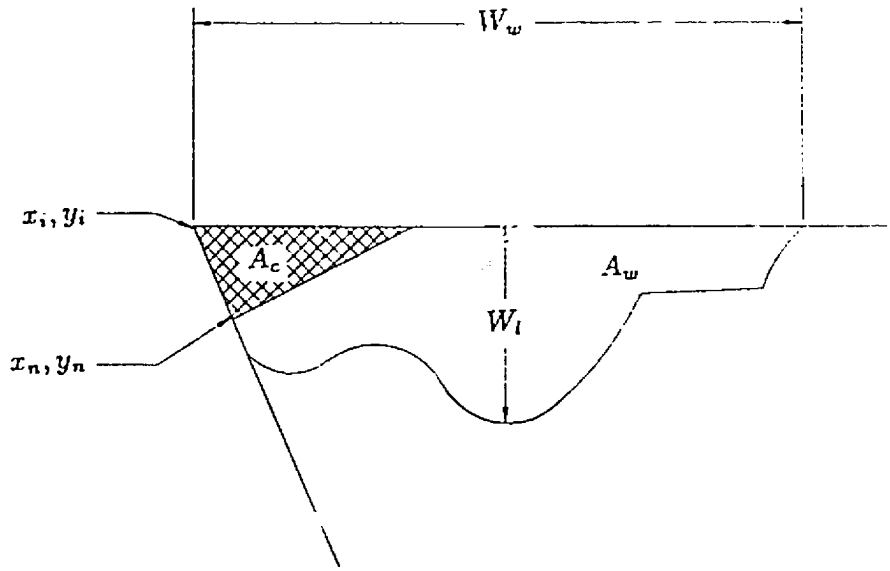
In the image of any given tool, three distinct regions are readily discernable: wear region, background and the rest of the tool body. Thus, the first problem is to

segment the image into these regions. At the end of segmentation, certain features are extracted. These features (Figure 3.2) are:

- length of the wear land (W_l)
- area of the wear land (A_w)
- perimeter of the wear land (P_w)
- maximum width of the wear land (W_w)
- nearest tool edge point to the original tool tip (x_n, y_n).

The first feature, length of the wear land, is the main feature used in the conventional tool monitoring method. Present tool monitoring techniques using machine vision technology use both the first and second features (i.e., length of wear land and area of wear land). The proposed system retains these two features and introduces three new features (viz. perimeter of the wear land, maximum width of the wear land and nearest tool edge point to the original tool tip) to provide additional information on the wear land.

The length of the wear land has been chosen as a measurement because it gives an idea of the extent of wear. Furthermore, it provides an insight into the value of the depth of cut and possible chip glide on the flank. It is measured from the tool tip along the cutting edge to the end of wear. Area of the wear land and the perimeter are parameters which will be needed in the development of a fully automated tool monitoring system. These two parameters will give a measure of the "compactness" of the wear land. Compactness, a nondimensional quantity, is the ratio of square of the perimeter to area. In addition, wear area could be useful when using weight



x_i, y_i	Tool tip coordinate
x_n, y_n	Nearest point to tool tip coordinate
W_w	Maximum wear width
W_l	Length of wear measured along the tool edge
A_w	Wear land area
A_c	Chipped region area
P_w	Perimeter of the wear land

Figure 3.2: Schematic of the measured parameters

loss as a monitoring measure. Width of the wear land is an important parameter measured by machinists for assessing the degree of wear and the chip flow across the tool. The distance between the nearest tool edge point and the original tool tip helps in predicting tool breakage. This is a deviation from the methods suggested by early researchers [9,16] and has been found to be effective. The details of feature extraction will be discussed in the next section.

3.3 Algorithm Development

A point form outline of the algorithm to extract the features discussed in the preceeding section is as follows:

1. capture and digitization of the image
2. segmentation of the image
3. determination of the tool tip using a Hough transform
4. check for breakage and wear by calculating parameters
5. report on the verdict,

and its flowchart is shown in Figure 3.3.

Capturing and digitizing the image are hardware implemented and require correct positioning of the tool. This stage has to be carefully implemented as it influences the quality of the image (in terms of sharpness, contrast and brightness), and thus the final verdict.

At the end of digitization, the program implements the segmentation process; this represents the various regions in the original image by "labels". Thus, a labelled image, an image in which the pixel intensities have been replaced by some

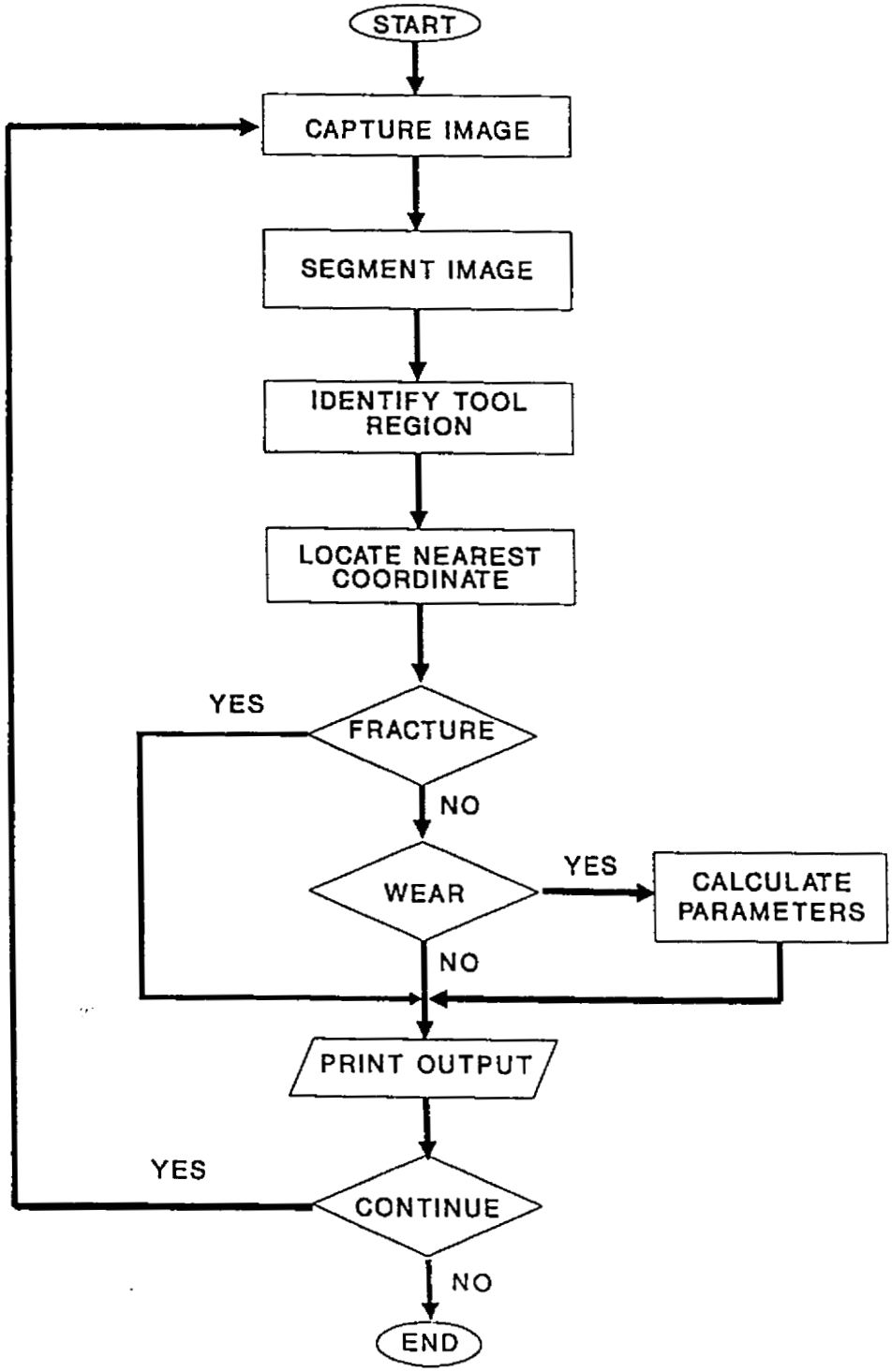


Figure 3.3: Flowchart of the tool wear monitoring and breakage detection algorithm

predetermined values, is obtained at the end of segmentation. There are several techniques for implementing this process and they have been treated in chapter two. The segmentation technique used here (thresholding) is solely based on direct pixel values. This approach saves both computer memory and CPU time while providing sufficient accuracy for the work. The use of a fibre optic illuminator has made it possible to create sufficient difference in intensities across the three regions of the image; wear region, rest of the tool body and the background.

In the previous chapter, thresholds were identified to be dynamic, local or global. As explained in [33], a threshold operator could be considered as a test against a function T of the form

$$T = T(x, y, N(x, y), g(x, y)) \tag{3.1}$$

where $g(x, y)$ is the gray level of the point (x, y), and $N(x, y)$ denotes some local property of the point - e.g., the average gray level over some neighbourhood centred at (x, y). The thresholded image can be obtained based on the following constraints:

$$g(x, y) = \begin{cases} k & \text{if } N(x, y) > T \\ l & \text{otherwise} \end{cases} \tag{3.2}$$

where k and l are arbitrary gray levels. Thus, in the thresholded image, pixels labelled k could be objects while those labelled l are background points.

When T is dependent on the image function, $g(x, y)$, only, the threshold is called global. If T is dependent on both the image function and local property, $N(x, y)$, the threshold is called local. In the event that T is dependent on the spatial coordinates (x, y), image function $g(x, y)$, and local property $N(x, y)$, it is called dynamic.

Thresholds (obtained from a learning period) are set to accommodate irregularities due to nonuniform illumination and tool surface finish. During the learning period, the histogram distibutions of several images of a tool, in different positions, were analyzed to obtain gray levels ranges for each of the three distinct regions of a defective tool. These thresholds are global because they are solely dependent on gray levels. Although this thresholding method successfully segments the images, its dependence on illumination is a disadvantage. This is because the system flexibility is hindered. Attempts at using local thresholding were unsuccessful because of computer memory limitations.

Based on the established thresholds, the pixels are labelled as yellow, blue or green. The yellow colour denotes the defective portion of the tool, blue colour represents the background and green colour represents the rest of the tool body. The labelling also shrinks the large number of intensities used to represent the image to three. The mathematical form of this process is

$$g(x,y) = \begin{cases} L_1 & \text{if } f(x,y) \leq T_1 \\ L_2 & \text{if } T_1 < f(x,y) \leq T_2 \\ L_3 & \text{if } T_2 < f(x,y) \leq T_3 \end{cases} \tag{3.3}$$

where $g(x,y)$ denotes the labelled image, $f(x,y)$ represents the input image, T_1, T_2 and T_3 are thresholds and L_1, L_2 and L_3 are arbitrary labels (i.e., yellow, blue, green). All the unlabelled pixels ($f(x,y) > T_3$) denote black coloured pixels and the portions of the image outside the range of the microscope. The occurrence of isolated black or dark spots on a tool body is caused by dirt, nonuniform illumination and surface finish.

At the end of the first segmentation operation, a second pass is made to locate and label the unlabelled pixels. During this process, the unlabelled pixels are compared with their neighbouring pixels to determine a maximum likelihood label. For example, in the case where an unlabelled pixel is found between pixels of the same label, say blue, it is logical to ascribe blue to the unlabelled pixel. However, on occasions where other possible labels are equal contenders, then the original pixel intensity value is compared to its 4-neighbour (see Appendix A) before a decision on the label to be assigned is made. The label with the maximum occurrence is assigned to the pixel. In the event that there is no label with the maximum occurrence, any of the labels of the four neighbours is selected, with preference given, in order, to yellow, blue and green.

With the completion of the segmentation exercise, the Hough transform (see Appendix C) is used to obtain two equations which denote the tool boundary when viewed from the flank side. The equation of the top edge of the tool is obtained by using edge points located on the good edge (i.e., unworn and unbroken) portion of the tool. This equation is then extended to cover the entire tool top region. The advantages of using this technique are: removal of glare introduced by the light intensity; and removal of built-up edge on the tool cutting edge. The determination of the other equation (i.e., the flank side) is more complicated because the possible range of lip angle cannot be covered by a single array without a great loss in accuracy. This is due to the restrictions in memory allocation placed by the programming language, Microsoft C V5.1. The accuracy of the Hough transform depends on the scale used on the parameter axes or on the size of the arrays (accumulator cells). Lip angle of a tool is dependent on several factors including the use

to which the tool is to be put and the tool material. For the laboratory CNC lathe used in the experimentation, the lip angle of a turning tool is usually in the range of 75° to 85°.

One way to circumvent the above is to subdivide the range of the lip angle and carry out a series of iterations to obtain the most likely equation at the end of each iteration. Thereafter, these equations are compared by examining the sizes of the accumulator cells to determine the best possible equation.

Since the labels assigned to the different regions have known values, the location of wear (or wear land) is achieved by searching for the pixels with this label. Because the intensities representing the wear land and fractured or chipped region (where part of the material is visible from the flank side) are identical, a check on breakage is made first and determination of other features is done only if breakage is absent.

The check for breakage is made by obtaining the coordinate of the nearest tool edge point (x_n, y_n) and that of the tool tip (x_i, y_i). The former is determined by carrying out a left-right top-down scan and the latter is found by determining the intersect of the tool top edge and side edge equations. The difference between these two coordinates is compared with a preset threshold. The tool is classified as broken if the threshold is exceeded, otherwise it is considered to be good and the wear parameters are then calculated.

The breakage threshold is obtained from a series of tests in which the tool tip coordinate of a sharp tool, directly observed from the image, is compared with that determined using the Hough transform. The allowable amount of difference, (i.e., the threshold) is set to accommodate the permissible amount of tool tip roundness and inaccuracies due to round-off errors. An allowance is made for tool tip roundness

because it is an accepted phenomenon for an originally sharp tool tip to round-off after a machining process. The round-off errors are caused by approximations and by the masking of float and double variables into integer variables. This is necessary because array counters can only be integers. Furthermore, the slope and intercept values obtained from the Hough transform were divided by constants so as to keep the accumulator cell array within the memory capabilities of the computer.

The perimeter of the wear land is determined by using a contour tracing algorithm developed by Pavlidis [23]. This algorithm, which is based on 8-neighbour connectivity of pixels, is simple and efficient. The contour tracing starts at an initial point and terminates when this initial point is reached. The initial point can be determined in several ways and the technique adopted in this thesis uses a left-right, top-down scan. Using a 3 x 3 kernel, the initial point is selected such that its 4-neighbour is not in the set. That is, all the four non-diagonal (i.e., vertical and horizontal) pixels are not possible edge points or of the same label.

Once the starting point is selected, it is labelled the current point, the search direction is set to six and a flag (say starter) is turned on. The search direction represents the new pixel of interest. The codes associated with pixels of a 3 x 3 kernel are shown in Figure A.1 (Appendix A). Provided flag starter is on or the current point is different from the initial point, another flag (say found) is turned off.

Then, while flag found is off the following steps are implemented at most thrice; a limit is placed on the number of times the steps can be repeated to avoid going into an endless loop. The pixel in the current search direction less one (i.e., the direction of search is reduced by one but the variable bearing the current direction is

unchanged) is checked to see if the 4-neighbour connectivity condition is satisfied. If the condition is satisfied, flag found is turned on and the search direction is decreased by two and taken as the current search direction. In the event that the 4-neighbourhood condition is not satisfied, the pixel in the current search direction is checked. If the condition is satisfied, the current point is taken to be this pixel and flag found is turned on. However, should the pixel in the current search direction fail to meet the 4-neighbour connectivity condition, the value of the current direction is increased by two (i.e., the direction of search is increased by two but the variable bearing the current direction is unchanged) and the pixel in the new direction is checked. If the condition is satisfied, the pixel is taken as the current point and flag found is turned on. When all the checked pixels fail to meet the 4-neighbour connectivity condition the search direction is increased by two and set as the current search direction.

At the end of the aforementioned testing of pixels, flag starter is then turned off and the current pixel point is compared with the initial point to determine if they are the same point. In addition, flag starter is checked to determine if it is on. The exercise is repeated if neither of the comparisons is true.

The area of the wear land could be deduced during the determination of the perimeter. However, the complex and unpredictable shape of what might constitute a wear land has made the method rather complicated and time consuming. A simpler but equally effective method is to count the number of pixels with the label used in denoting the wear land.

The length of the wear is obtained by finding the coordinates of the extreme right wear point subtracting its horizontal value from that of the tool tip. This

extreme point is found by using a top-down, right-left scan.

The maximum width of the wear land is calculated by finding the distance between the minimum position of wear and the top of the tool. The minimum position of wear can be obtained by examining the wear boundary.

In order to express the wear parameters in real dimensions (i.e., SI units of measurement) a calibration process to determine the number of pixels in $1mm^2$ is carried out. This is done by placing $1mm^2$ squares, drawn on a graph paper, on the microscope platform and capturing the image under different lighting conditions. A routine to determine the number of pixels in the various images is implemented and the average value is calculated. The results presented in this research are based on a calibration of 50 $pixels/mm$ or 2500 $pixels/mm^2$.

The flowchart as illustrated in Figure 3.3 has three underlining assumptions:

1. Adjudge a tool as good if neither breakage nor wear is evident.

2. First check for breakage and, if evident, adjudge the tool as bad. The nee d to obtain features such as area of chipped region and length of chipped region is irrelevant because a chipped tool needs to be replaced.

3. Check for wear only when breakage is absent. The features are then extracted and compared with predetermined thresholds. In the event that the thresholds are exceeded, a bad tool verdict is reported, otherwise the tool is classified as good. The steps involved in the determination and selection of these thresholds have been explained in the previous paragraphs.

A listing of the source code which is written in Microsoft C V5.1 is presented in Appendix D.

Having discussed the configuration of the developed system, the new breakage determination concept and the new features which provide additional information on wear land in this chapter, the next chapter will discuss experiments on the system and results obtained.

Chapter Four

Experimental Results and Discussion

Experiments were designed to test the algorithm capability to perform the following tasks:

1. Detect breakage.

2. Monitor tool wear over a brief cutting period.

The equipment involved in the above experiments is shown in Figure 3.2 and consists of a microscope, a fiber optic illuminator, a 60-watts light bulb, a CCD RGB camera, a monitor and a microcomputer. The other apparatus are austenitic stainless steel bar, AISI C1045 steel, stop clock, cutting fluid, vernier caliper, degreaser (Toluene) and Lathe machine.

Images of the tool at different stages of the experiment as observed on the monitor have been displayed. In the pictures taken before processing, the background is orange, the suspect defective region is white or yellow and the rest of the tool is brown. The pictures taken after processing have the background denoted by blue, the observed defective tool region by yellow and the rest of the tool body represented by green.

4.1 Experiment 1

This experiment is to test the system capability to detect wear and effectively measure width and other parameters or features.

A sharp CNC lathe turning tool was used in turning a hardened stainless steel shaft, so as to produce wear in a matter of seconds. The tool was then taken to the setup for capture and analysis. Figure 4.1 is a picture of the tool wear region and Figure 4.2 illustrates the image obtained at the end of processing.

Values of the computed parameters are shown in Table 4.1.

Table 4.1: Parameters evaluated from tool wear test

Parameter	Value
width (mm)	0.400
Perimeter (mm)	5.224
Area (mm^2)	0.430
Compactness	63.45

4.2 Experiment 2

This experiment is geared towards breakage detection. A keyholed workpiece was machined using a sharp CNC lathe turning tool. The workpiece and tool were set up such that the keyhole groove dropped on the tool resulting in breakage. The breakage detected by the system is illustrated in Figures 4.3 and 4.4. It is worth noting that the intensities obtained for a broken region correlate well with those obtained for the worn region.

Figure 4.1: Typical tool wear (image before processing)

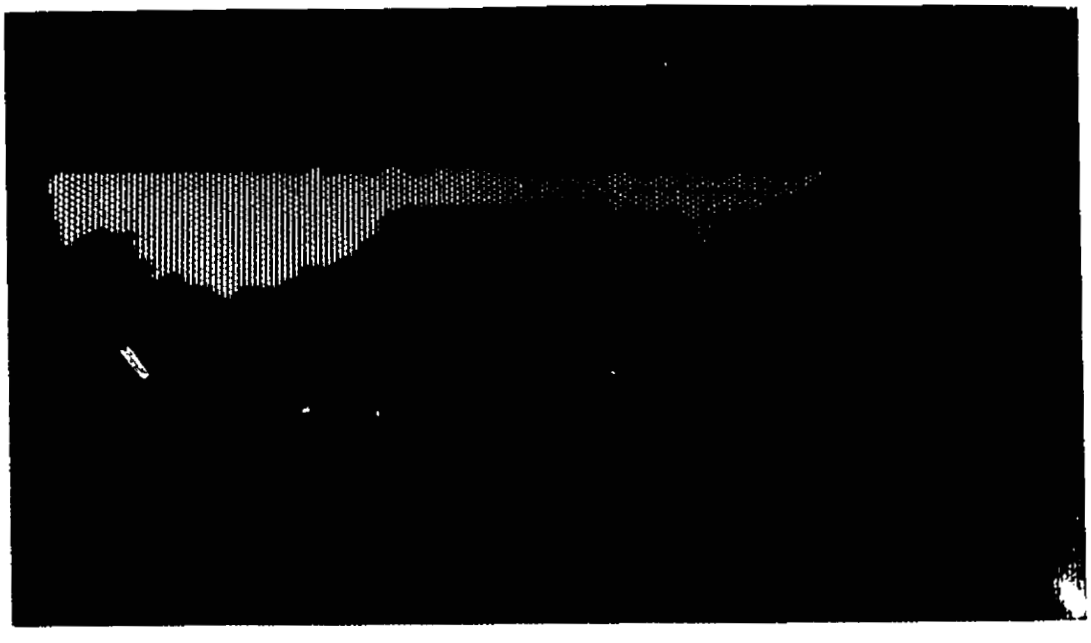


Figure 4.2: Typical tool wear (image after processing)

Original pictures are in colour. Reproduction in black and white may result in loss of quality.

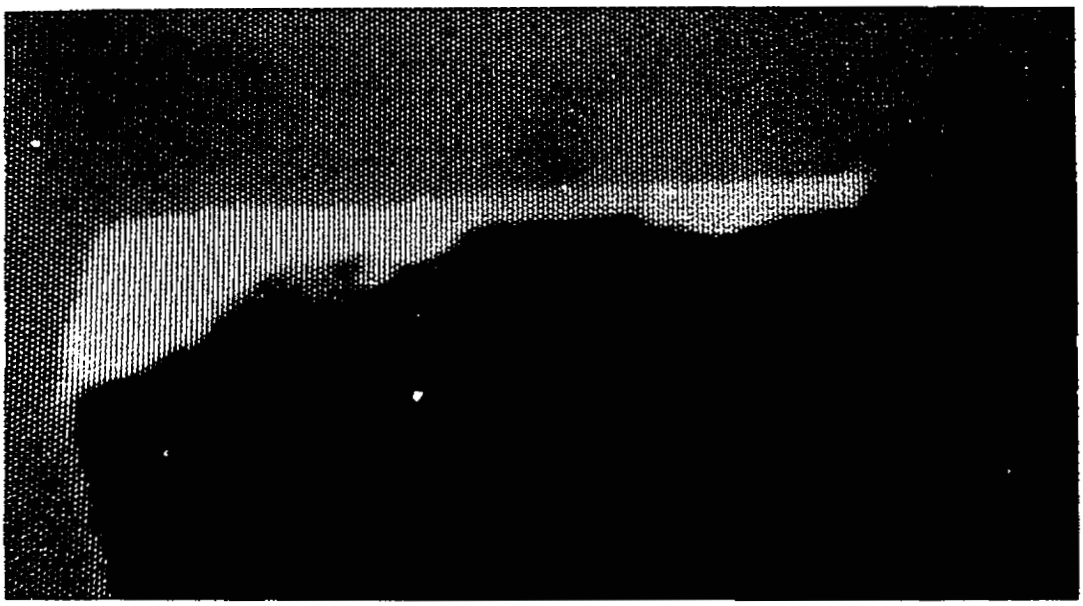


Figure 4.3: Typical tool breakage (image before processing)

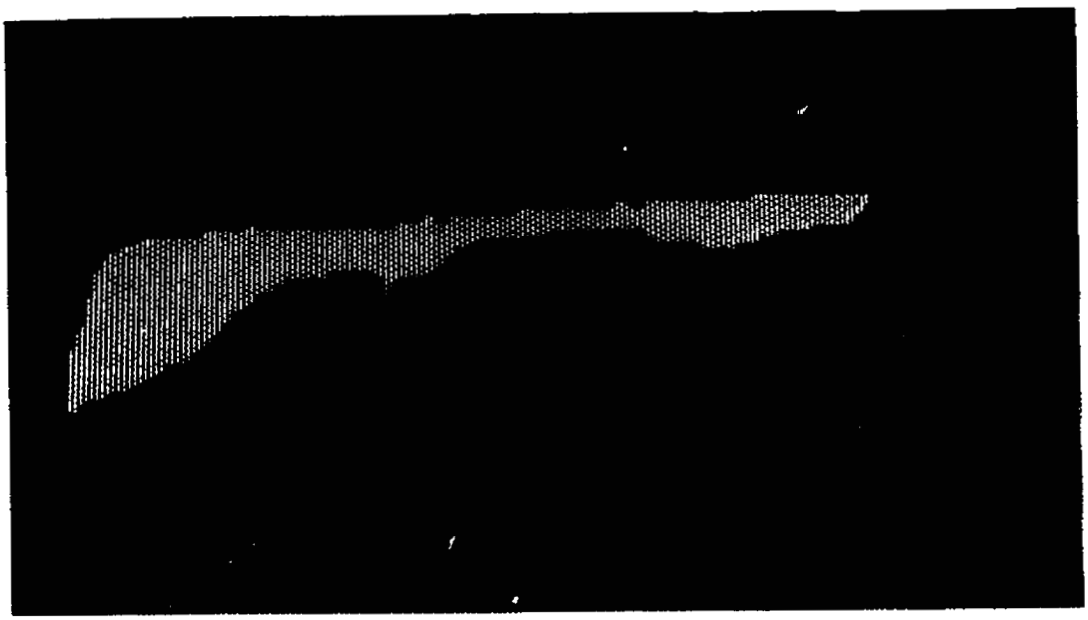


Figure 4.4: Typical tool breakage (image after processing)

Original pictures are in colour. Reproduction in black and white may result in loss of quality.

Table 4.2 shows the values of the parameters computed. The tool tip coordinate and nearest point coordinate are the only parameters presented because the program discovered that the tool was already broken and that it did not have to calculate the tool wear parameters. A tool is broken if its nearest point coordinates fall outside a $0.20mm$ square drawn with its upper left corner on the calculated tool tip coordinates. The $0.20mm$ or 10 pixel position is the threshold used in this experiment.

From the result, it is seen that the difference in the coordinates of the tool tip and the nearest tool point is 10 pixels in the x-direction and 23 pixels in the y-direction. Thus, the nearest tool point is outside the range.

Table 4.2: Parameters evaluated from the tool breakage test

parameter	value
Tool tip coordinates	(52, 133)
nearest point coordinates	(62, 110)

4.3 Experiment 3

This experiment is designed to monitor the state of a given tool over a brief cutting period. It involved machining a bar of austenitic stainless steel. The cutting parameters and tool angles used during this experiment are shown in Table 4.3. The angles and spindle speed have been chosen to be as close as possible to the optimum cutting angles specified for HSS tools [32].

First, the austenitic stainless steel bar was securely positioned between centres.

It was then turned or machined for about five minutes before the lathe was stopped and the tool removed and cleaned with the degreaser before being "air blown" to remove, as much as possible, any chip that might be on the flank. Then, the tool was taken to the vision system for analysis. There, the tool was placed so that its flank view was observed on the monitor. The fibre optic illuminator was then positioned to enhance the contrast between the worn region, if any, and the rest of the tool. The observed image was then captured and analyzed by the algorithm for wear or breakage. The parameters calculated are then recorded and the tool returned to the lathe for more turning. This process was repeated until it was clear that the tool could no longer cut "properly".

Pictures of the tool before the start of the experiment are shown in Figures 4.5 and 4.6, while Figures 4.7 through 4.12 depict the tool during some stages of the experiment. The results of the experiment are shown in Table 4.4.

Table 4.3: Cutting parameters and tool angles used in tool wear monitoring test on stainless steel bar

Entity	Austenitic Stainless steel
End cutting edge angle	14°
Side cutting edge angle	0°
Side rake angle	8°
Side relief angle	11°
Back rake angle	5°
End relief angle	8°
Depth of cut (mm)	1.375
Feed (mm)	0.178
Spindle speed (fpm)	96
Original diameter (mm)	37.846

Figure 4.5: Unprocessed image of the tool before machining the austenitic stainless steel

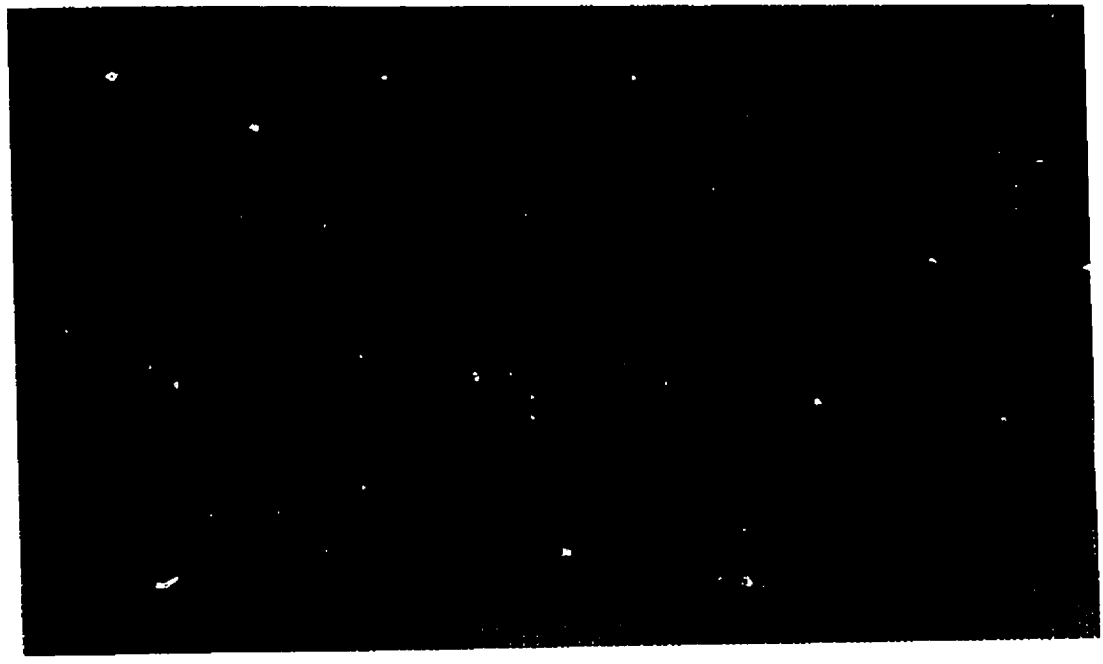


Figure 4.6: Processed image of the tool before machining the austenitic stainless steel

Original pictures are in colour. Reproduction in black and white may result in loss of quality.

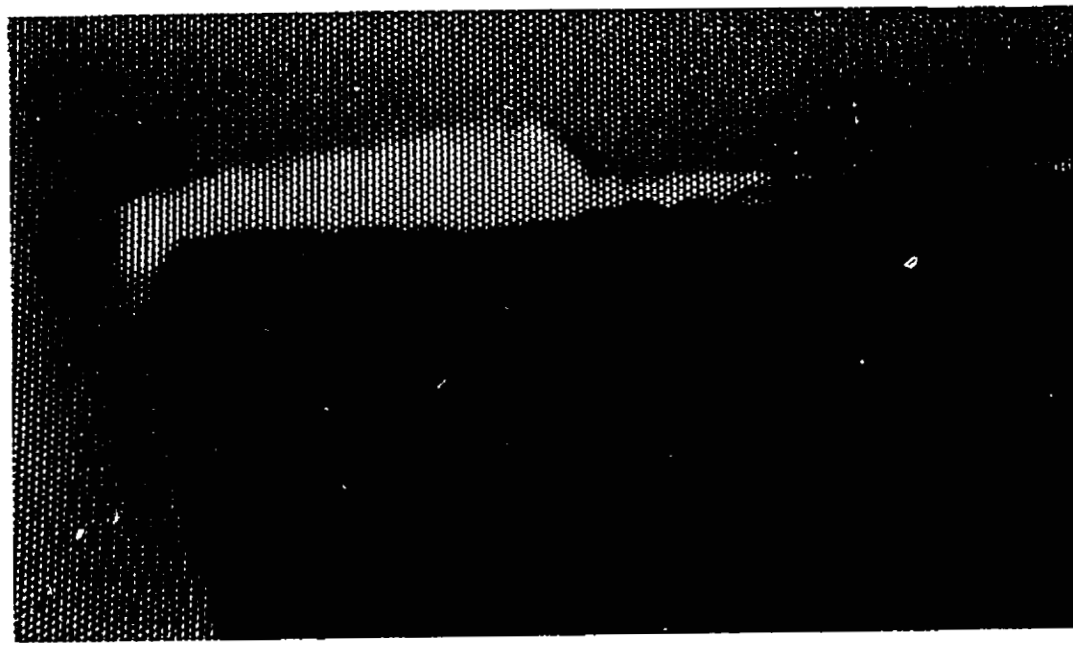


Figure 4.7: Unprocessed image of the tool after machining the stainless steel for 5.5 mins

Figure 4.8: Processed image of the tool after machining the austenitic stainless steel for 5.5 mins

Original figures are in colour. Reproduction in black and white may result in loss of quality.

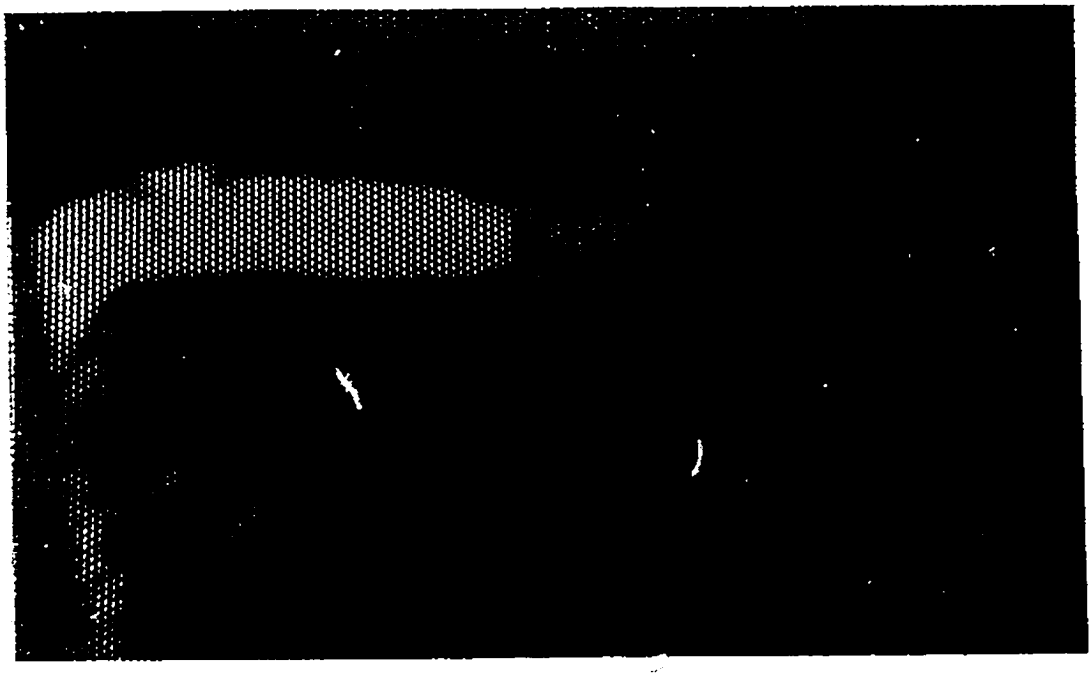


Figure 4.9: Unprocessed image of the tool after machining the stainless steel for 30.0 mins

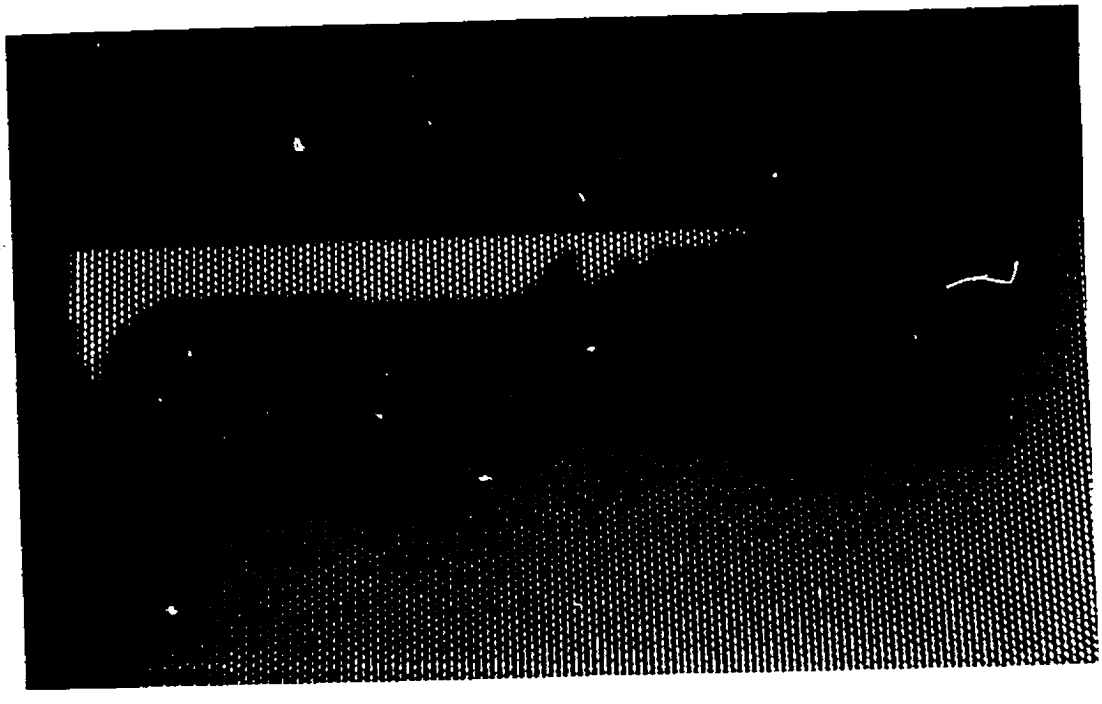


Figure 4.10: Processed image of the tool after machining the austenitic stainless steel for 30.0 mins

Original pictures are in colour. Reproduction in black and white may result in loss of quality.

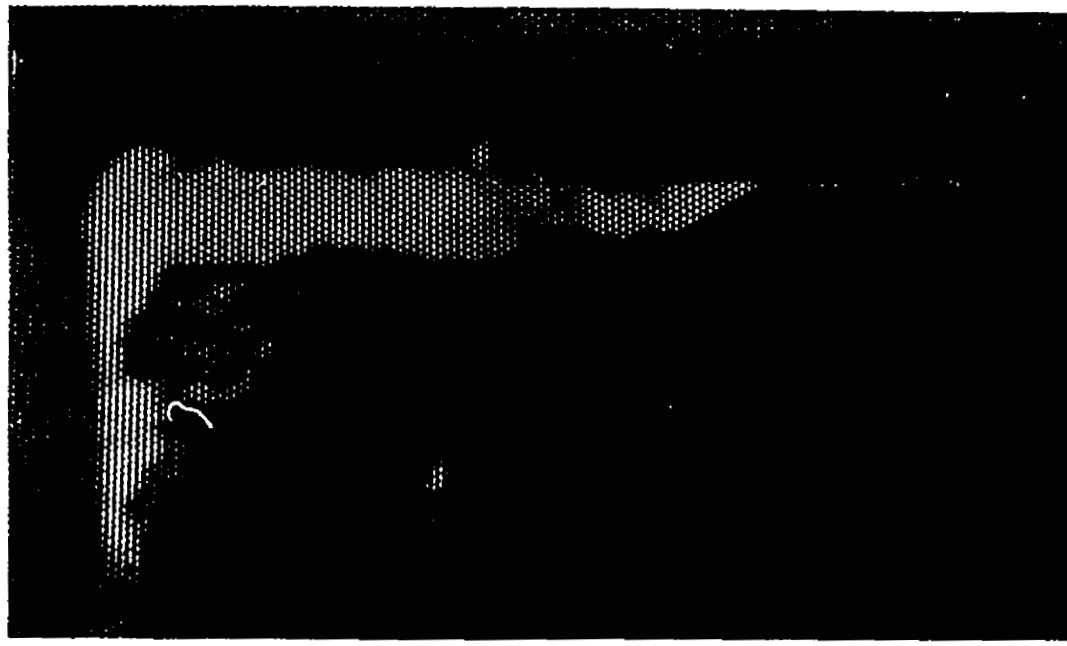


Figure 4.11: Unprocessed image of the tool before machining the austenitic stainless steel for 62.50mins

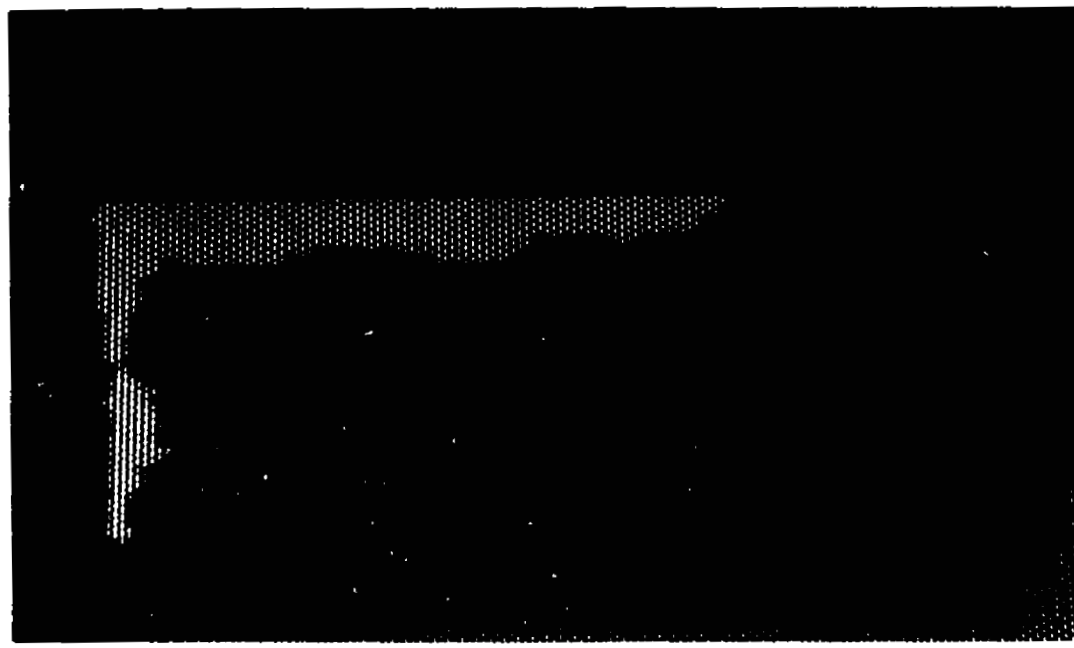


Figure 4.12: Processed image of the tool before machining the austenitic stainless steel for 62.50mins

Original pictures are in colour. Reproduction in black and white may result in loss of quality.

Table 4.4: Values of the parameters evaluated from tool wear monitoring test on stainless steel bar

Plate	Time (mins)	Width (mm)	Perimeter (mm)	Area (mm^2)	Compactness
1	0	0	0	0	*
2	5.50	0.18	2.979	0.108	82.50
3	11.00	0.20	3.126	0.128	82.00
4	16.50	0.24	3.264	0.152	70.10
5	22.00	0.26	3.331	0.159	69.65
6	30.00	0.30	3.417	0.165	70.86
7	40.00	0.34	3.486	0.173	70.00
8	45.50	0.38	3.564	0.181	70.20
9	53.00	0.44	3.693	0.202	67.40
10	55.00	0.46	3.809	0.201	72.09
11	60.00	0.70	4.213	0.219	81.14
12	61.00	0.80	4.588	0.231	91.04
13	62.50	1.20	6.180	0.342	111.80

**undefined*

From the graph of wear width against cutting time (Figure 4.13), the three distinct regions shown in Figure 2.7 (rapid wear, gradual wear and catastrophic wear) could be recognized from the wear profiles. It is seen that the intermediate period, i.e., second phase, can be approximated by a line. The point of transition from the gradual wear stage to catastrophic wear stage is the permissible tool wear point. A permissible wear of about $0.448mm$ occurred after 53 mins of machining.

The graph of wear land perimeter against cutting time, Figure 4.14, depicts a profile similar to that obtained for wear width. Furthermore, the critical time obtained from the graph corresponds to that obtained from its wear width curve. The same observation can be made from graphs of wear land area, Figure 4.15.

Compactness, $perimeter^2/area$, is used in machine vision as an indicator of

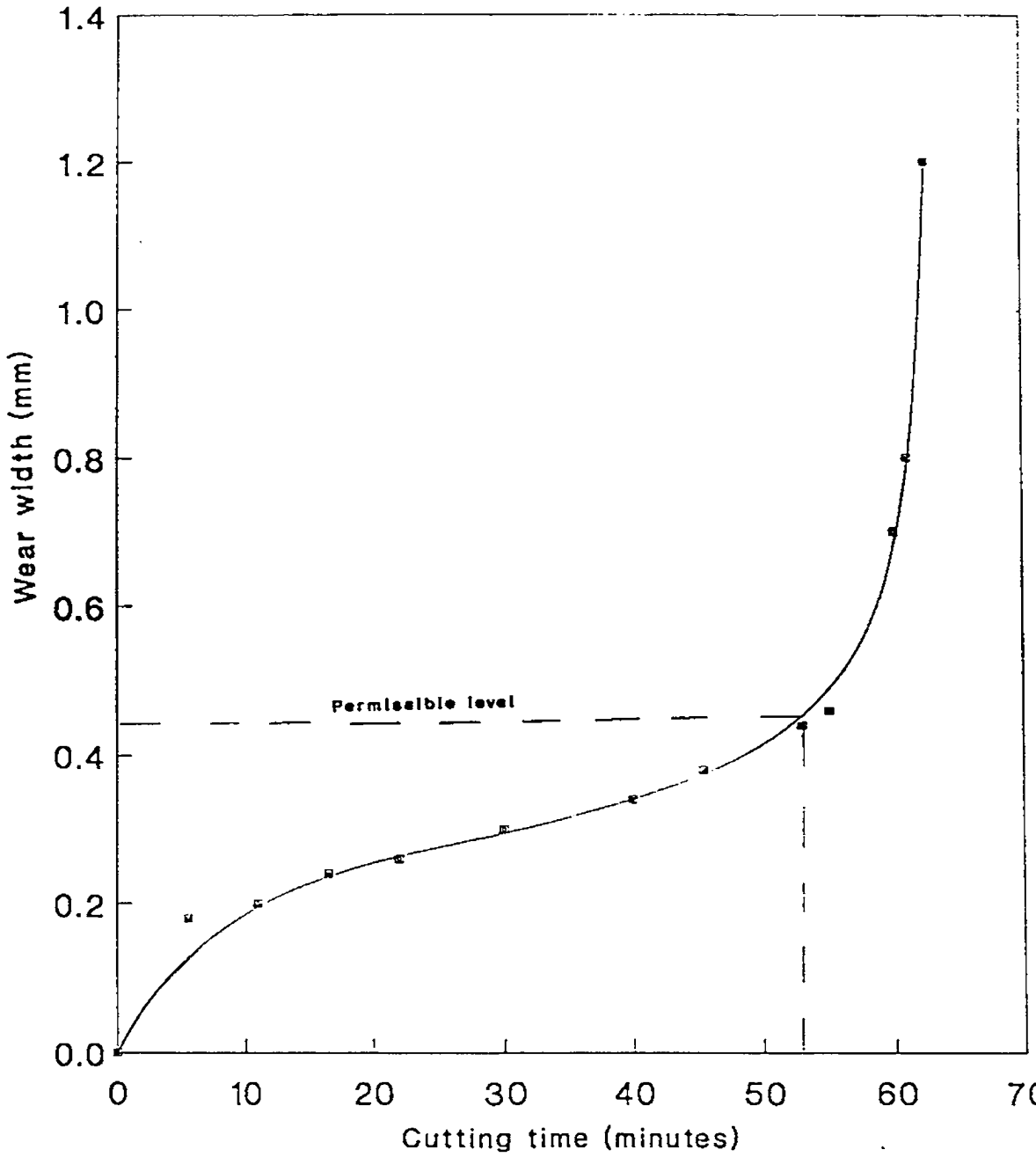


Figure 4.13: The austenitic stainless steel wear width profile

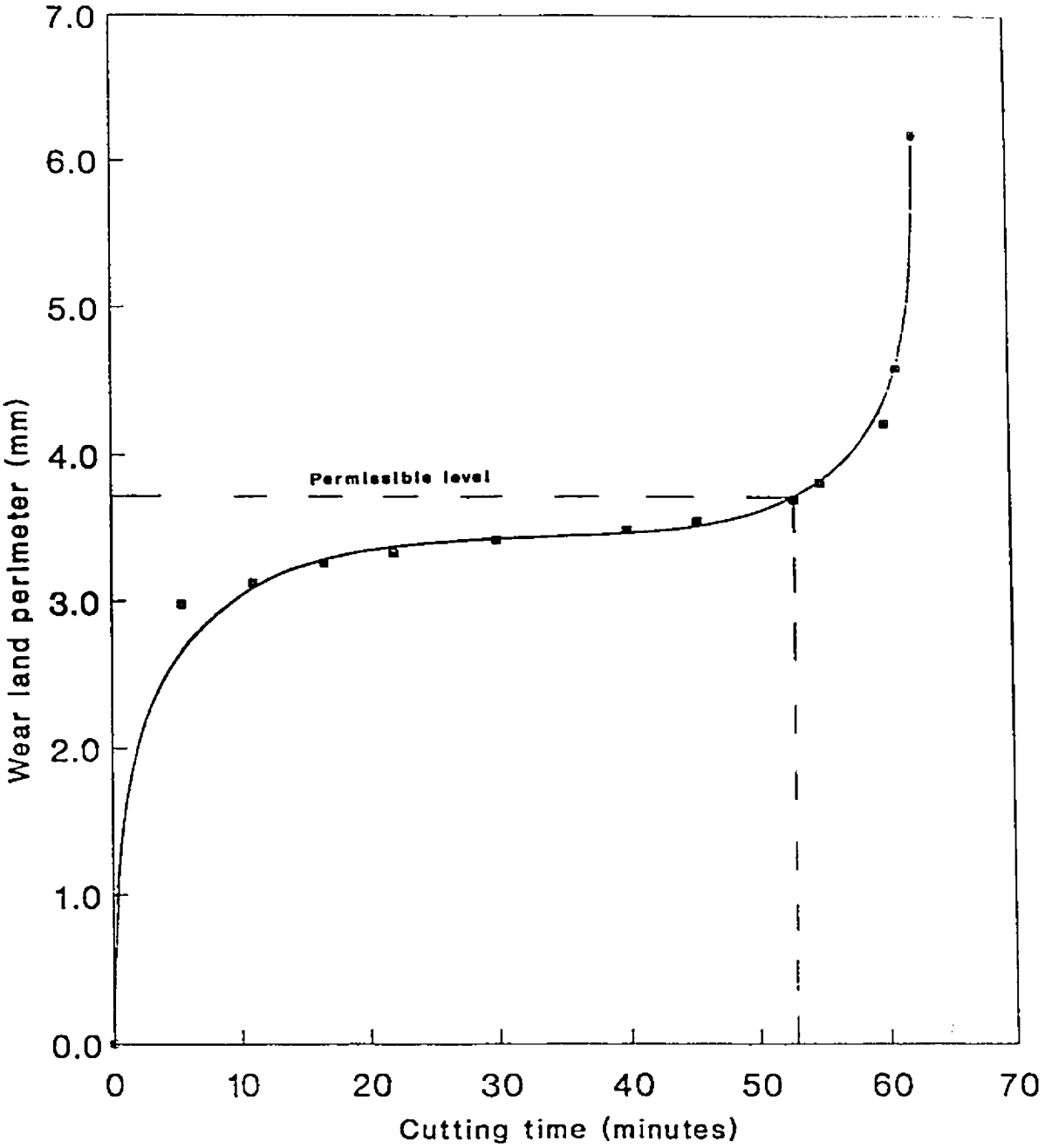


Figure 4.14: The austenitic stainless steel wear land perimeter profile

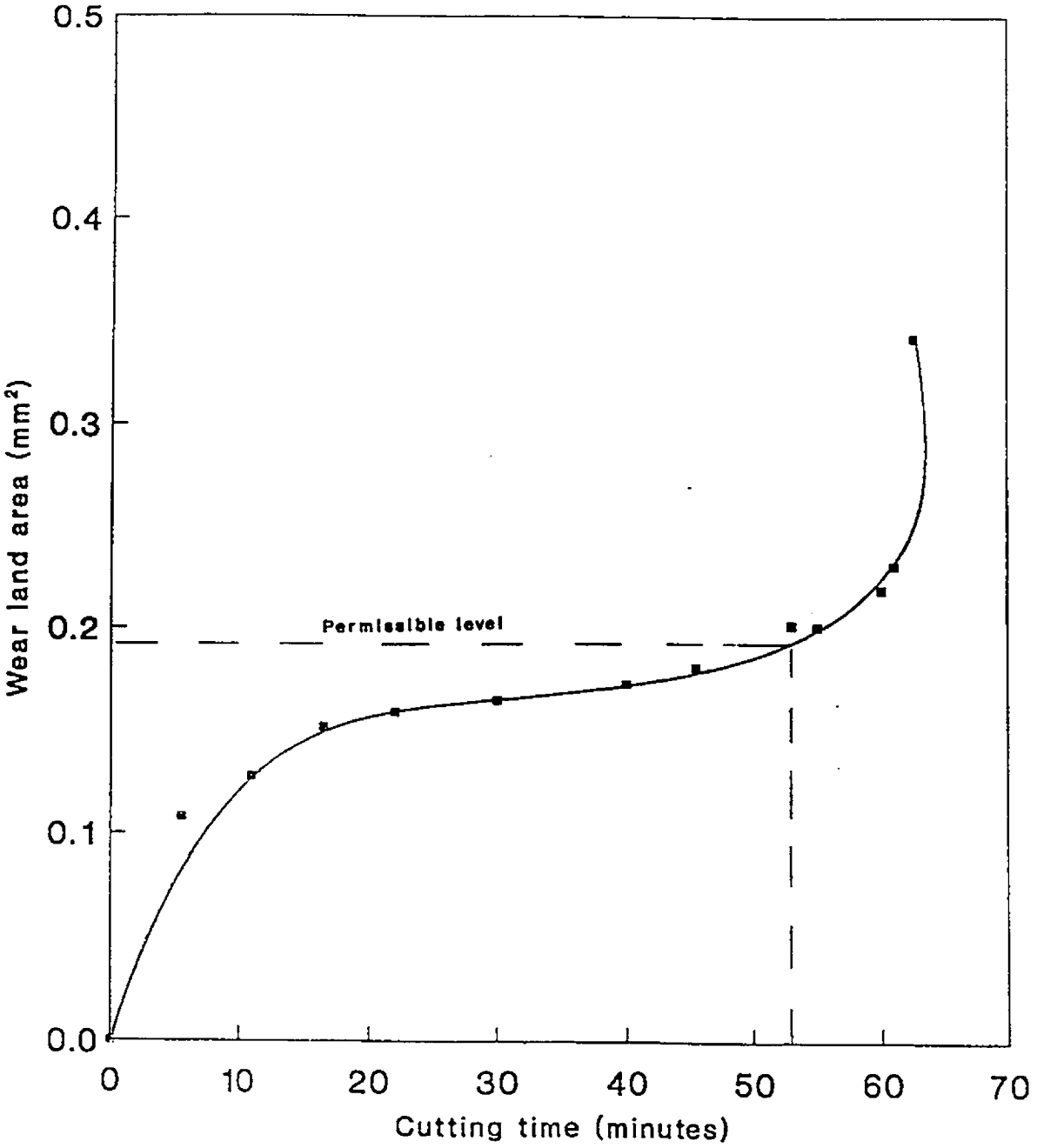


Figure 4.15: The austenitic stainless wear land area profile

shape. The graph for compactness, plotted against cutting time, Figure 4.16, shows that there is an initial drop in compactness as cutting progresses until the critical time is reached and thereafter it starts increasing.

4.4 Experiment 4

This experiment is identical to experiment three, the difference being in the material used — AISI C1045 steel in this case. However, in the experiment, conventional physical measurements of the wear width were made using a calibrated microscope. The cutting parameters and tool angles used during this experiment are shown in Table 4.5. They have been chosen to be as close as possible to the optimum cutting values specified for HSS tools [32]. Pictures of the tool at the start of the experiment are depicted in Figures 4.17 and 4.18, while those at subsequent stages of the experiment are depicted in Figures 4.19 through 4.24.

Table 4.5: Cutting parameters and tool angles used in monitoring tool wear

Entity	AISI C1045 steel
End cutting edge angle	5°
Side cutting edge angle	0°
Side rake angle	3°
Side relief angle	5°
Back rake angle	4°
End relief angle	5°
Depth of cut (mm)	1.375
Feed (mm)	0.165
Spindle speed (fpm)	96
Original diameter (mm)	38.10

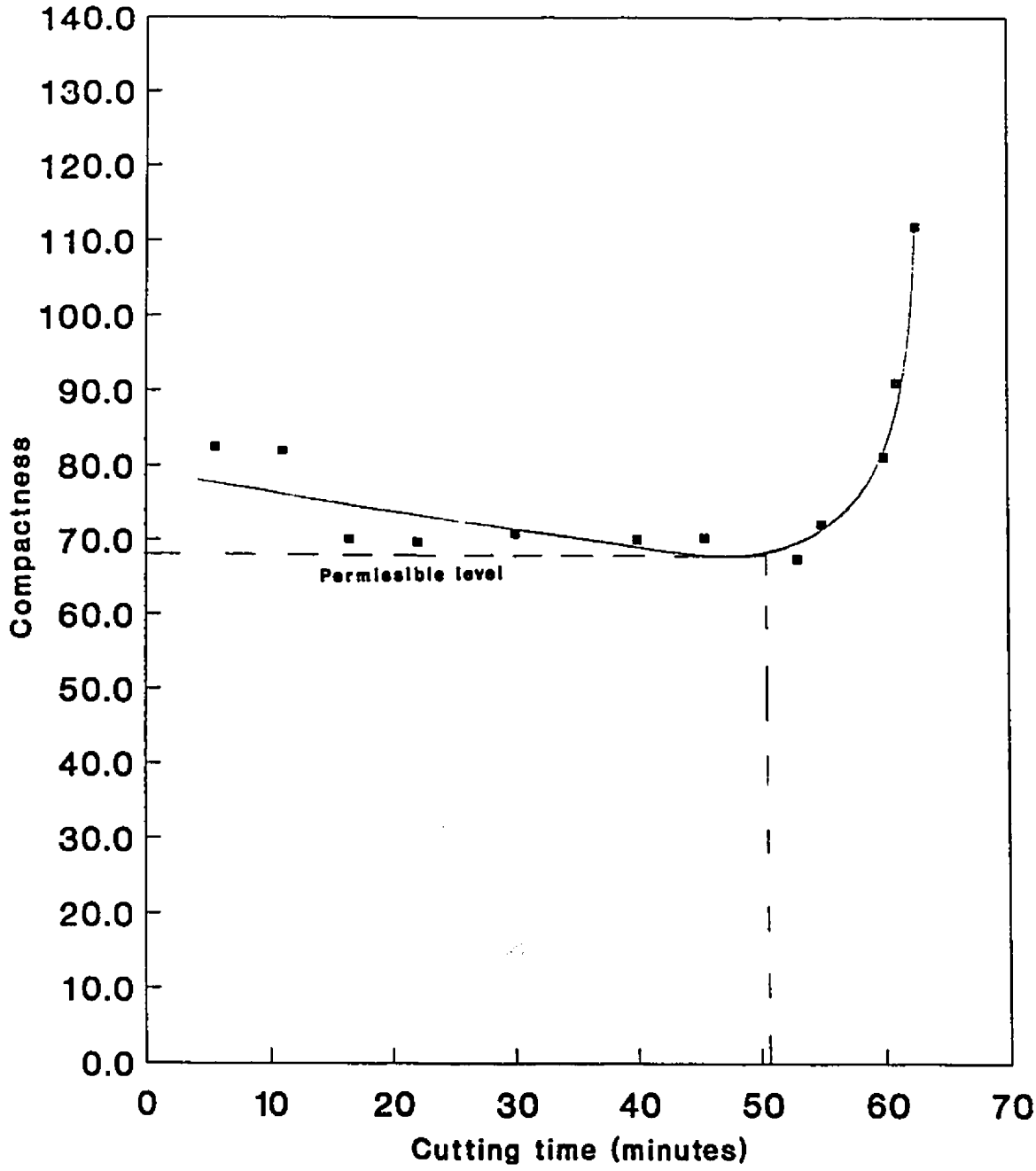


Figure 4.16: The austenitic stainless steel wear land compactness profile

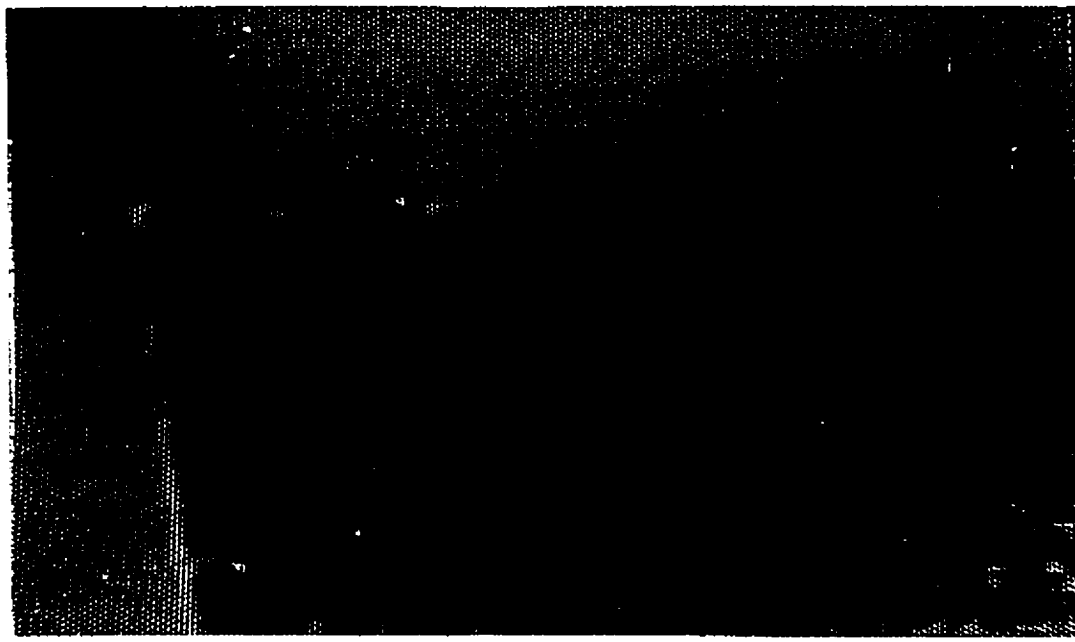


Figure 4.17: Unprocessed image of the tool before machining the AISI C1045 steel

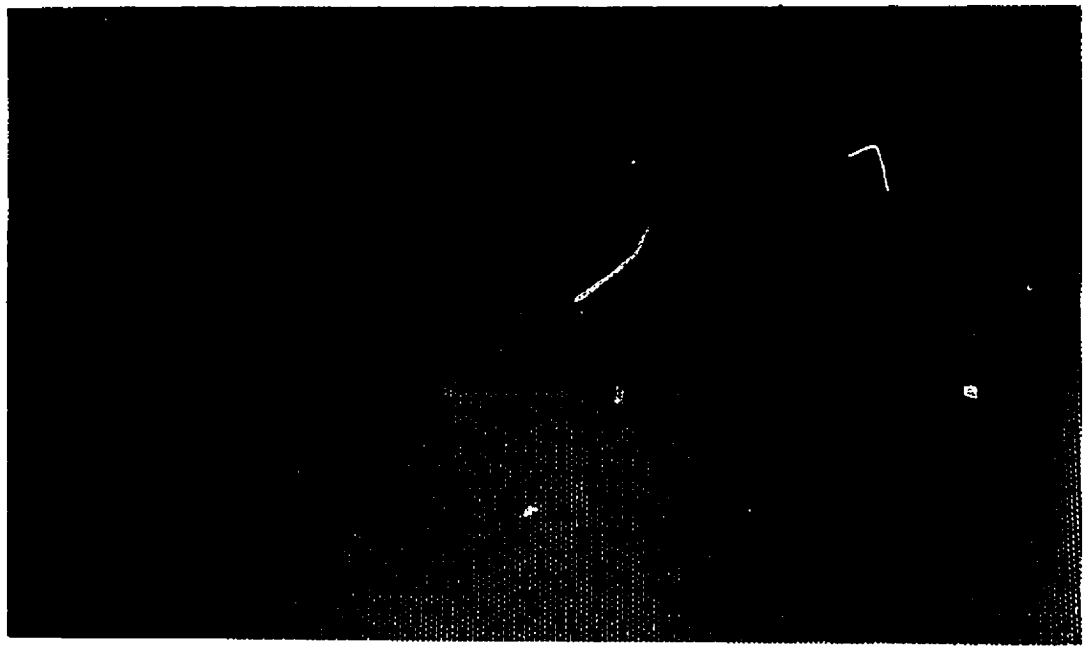


Figure 4.18: Processed image of the tool before machining the AISI C1045 steel

Original pictures are in colour. Reproduction in black and white may result in loss of quality.

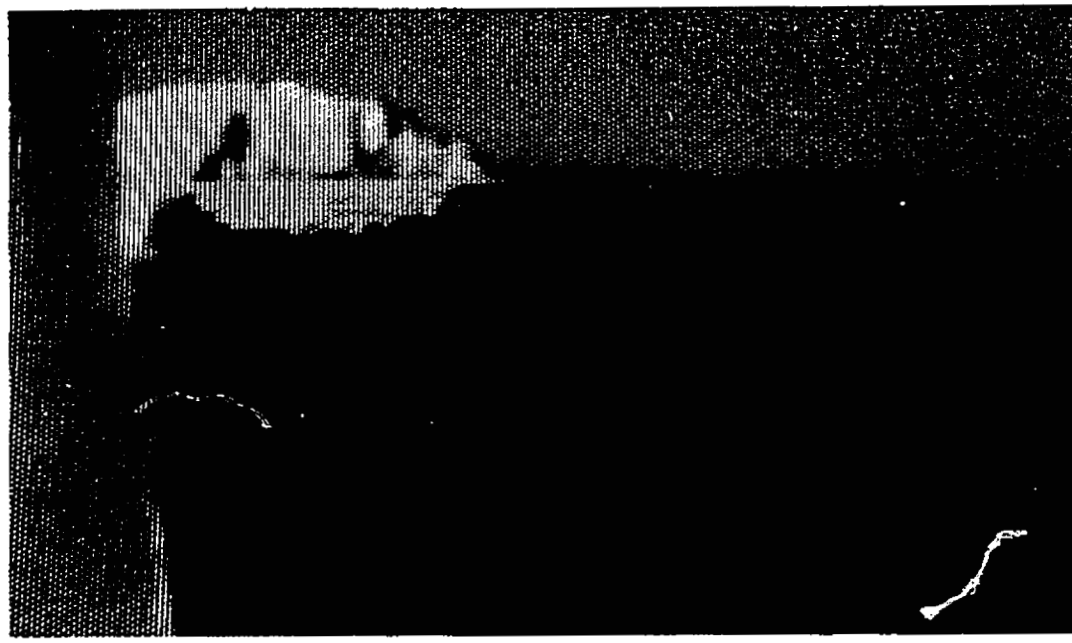


Figure 4.19: Unprocessed image of the tool after machining the AISI C1045 steel for 2.0 mins

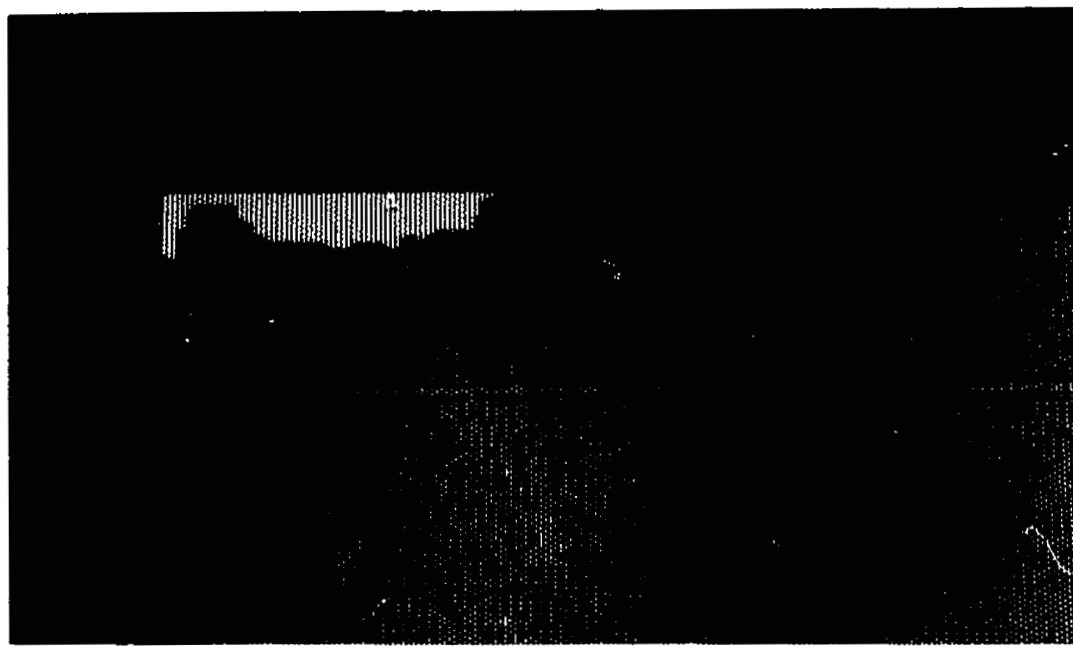


Figure 4.20: Processed image of the tool after machining the AISI C1045 steel for 2.0 mins

Original pictures are in colour. Reproduction in black and white may result in loss of quality.

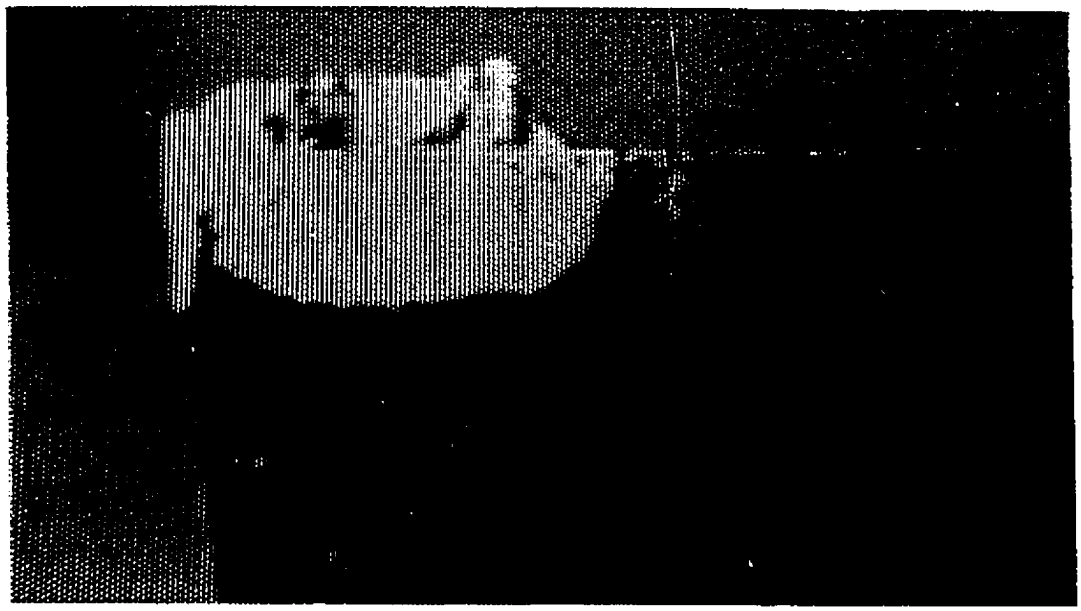


Figure 4.21: Unprocessed image of the tool after machining the AISI C1045 steel for 26.0 mins

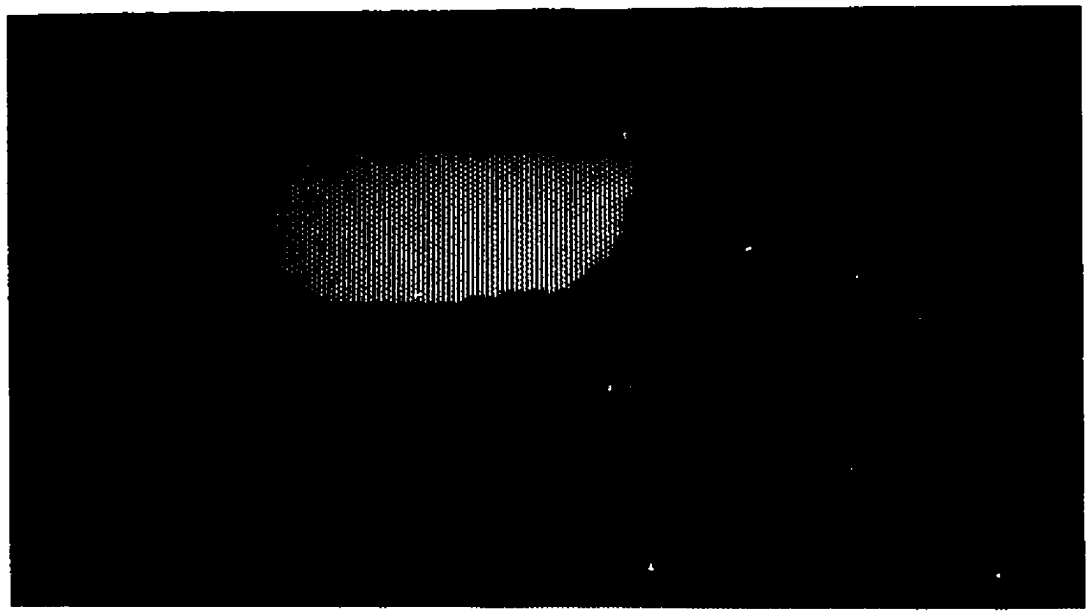


Figure 4.22: Processed image of the tool after machining the AISI C1045 steel for 26.0 mins

Original pictures are in colour. Reproduction in black and white may result in loss of quality.

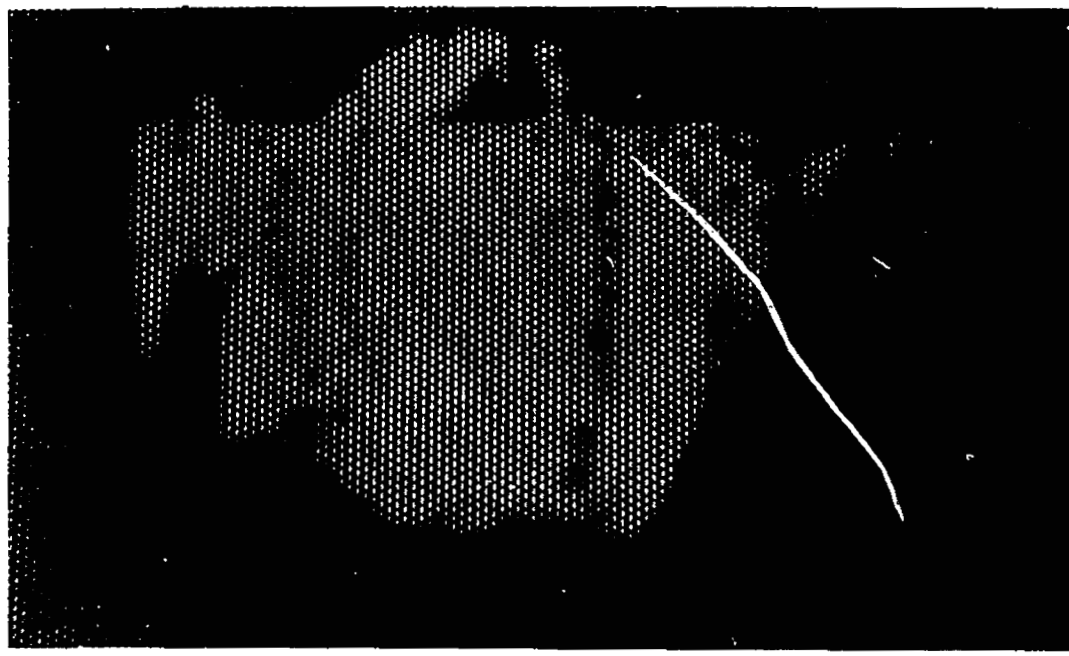


Figure 4.23: Unprocessed image of the tool after machining the AISI C1045 steel for 43.0 mins

Figure 4.24: Processed image of the tool after machining the AISI C1045 steel for 43.0 mins

Original pictures are in colour. Reproduction in black and white may result in loss of quality.

Results from the experiments are shown in Table 4.6.

Table 4.6: Values of the parameters evaluated from tool wear monitoring test on AISI C1045 steel

Plate	Time (mins)	Width (vision) (mm)	Width (conventional) (mm)	Perimeter (mm)	Area (mm^2)	Compactness
1	0	0	0	0	0	*
2	2.00	0.22	0.236	2.521	0.130	48.88
3	6.00	0.32	0.336	3.152	0.291	34.12
4	14.00	0.40	0.410	3.693	0.451	30.26
5	22.00	0.48	0.526	4.053	0.514	31.99
6	26.00	0.54	0.557	4.200	0.590	29.91
7	30.00	0.56	0.587	4.315	0.622	29.93
8	34.00	0.72	0.713	4.814	0.728	31.82
9	37.80	0.72	0.764	4.916	0.750	32.22
10	41.00	0.86	0.868	4.932	0.859	28.31
11	43.00	0.88	0.902	5.248	0.864	31.89

**undefined*

Graphs of Figures 4.25 through 4.28 are identical to the corresponding graphs obtained from test on austenitic stainless steel. The graph of Figure 4.25 indicates that the permissible wear occurred after 32 mins of machining. This was found to be $0.626mm$ from the conventional measurement scheme and $0.589mm$ from the machine vision measurement.

These values (i.e., permissible wear determined in experiments three and four) are not presented as standard permissible wear values for the materials tested because machining conditions which influence this threshold vary. However, it is estimated that for rough turning, the permissible wear for a HSS tool should be between $0.75mm$ and $1.50mm$ [21]. The values obtained in the present experiments

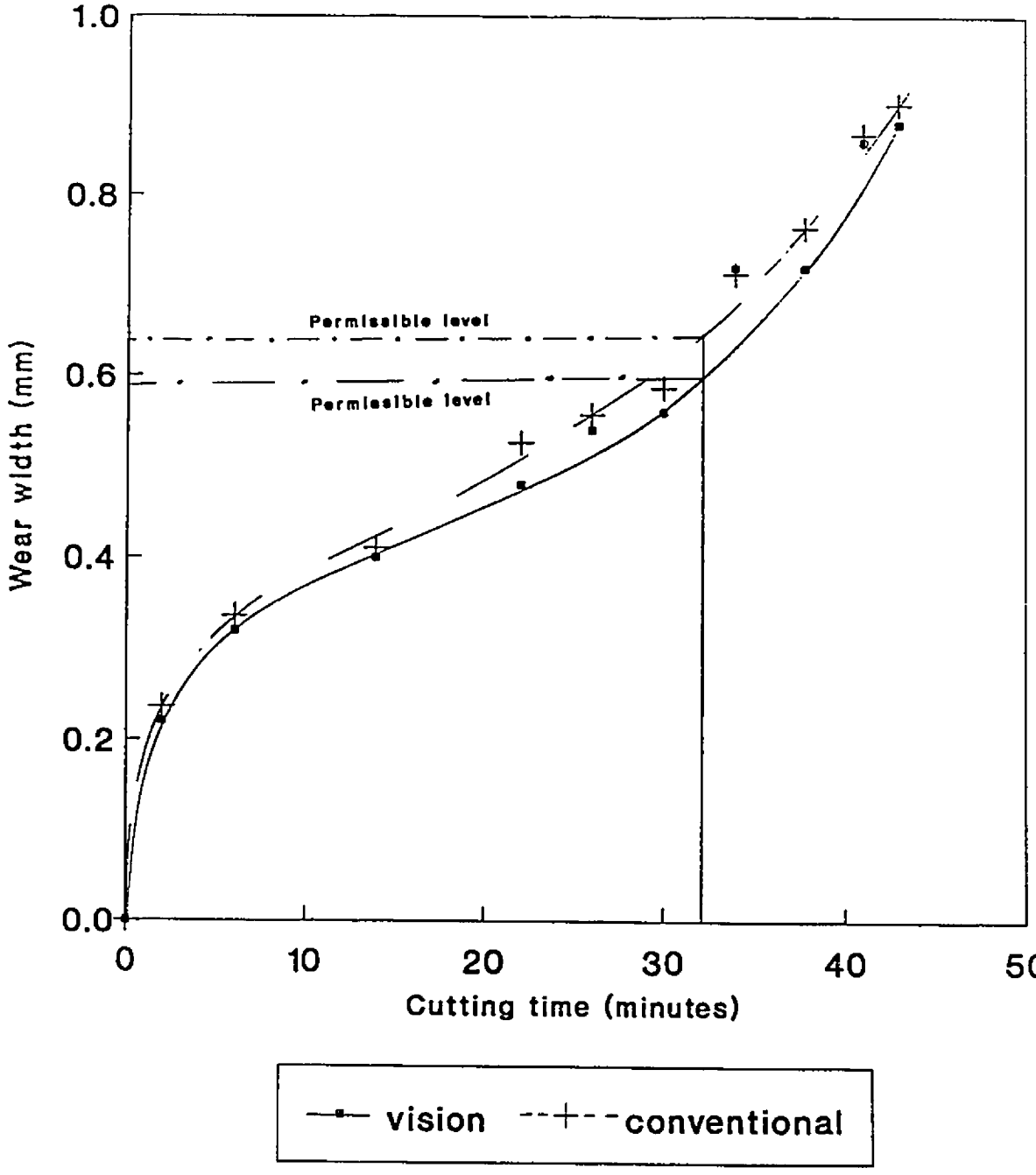


Figure 4.25: The AISI C1045 steel wear width profile

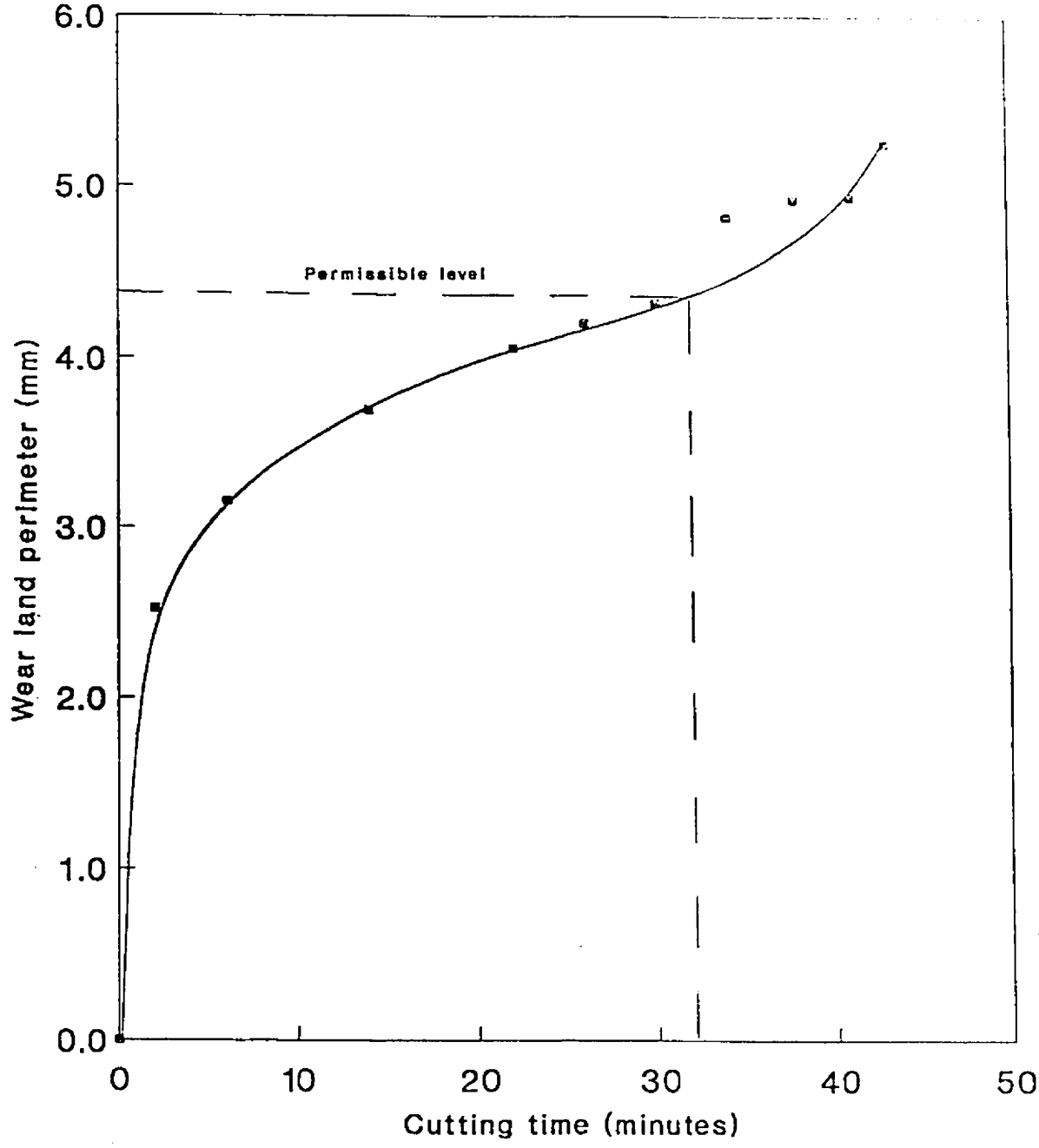


Figure 4.26: The AISI C1045 steel wear land perimeter profile

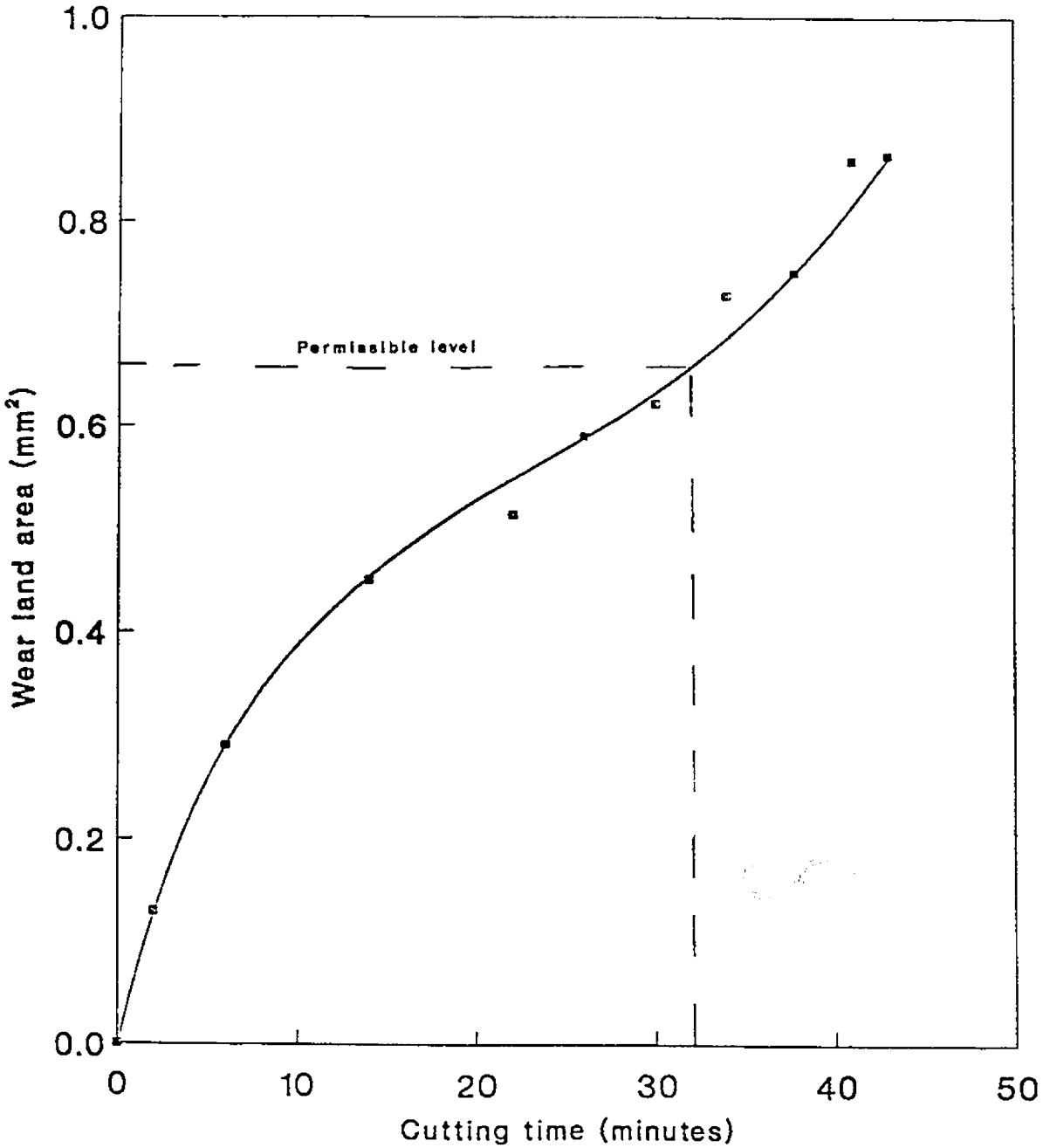


Figure 4.27: The AISI C1045 steel wear land area profile

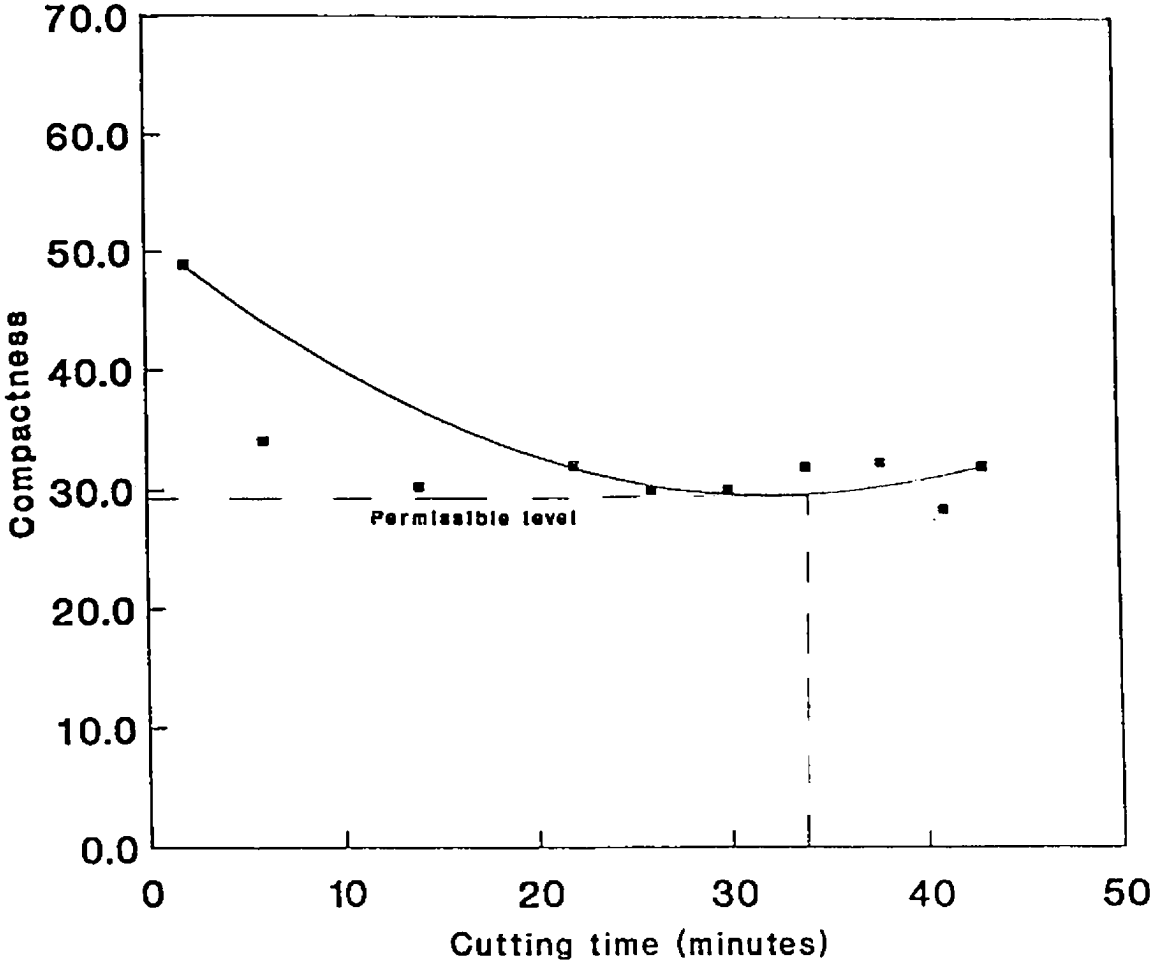


Figure 4.28: The AISI C1045 steel wear land compactness profile

are lower, possibly because no coolant was used. The cutting fluid used simply aided turning but did not cool the tool. This would make the tool wear faster.

Figure 4.25 also shows that the values obtained from the machine vision measurement are quite close to those obtained from conventional method. The degree of closeness can be inferred from a paired t-test of the results (Table 4.6) which showed that no significant deviation exists at a confidence level of 0.002. This implies that there is a 0.996 probability that measurements obtained using the machine vision system will be equal to those obtained using the conventional method.

The graph of wear land perimeter against cutting time, Figure 4.26, depicts a profile similar to the graph of the wear width against cutting time (Figure 2.7) in which there are three distinct regions (viz. rapid wear, gradual wear and catastrophic wear). And the critical time obtained from the graph corresponds to that obtained in its wear width curve. The same observation can be made from the graph of wear land area, Figure 4.27. The graph for compactness, Figure 4.28, shows that there is an initial drop in compactness as cutting progresses until the critical time is reached and thereafter it starts increasing; a similar profile was observed in Figure 4.16.

An indication of the suitability of this algorithm or technique to monitor wear using wear land area can be inferred from the fact that the critical times obtained from wear area graphs correspond to those obtained from the wear width curve. Weight loss is another conventional method of monitoring tool wear. The mass of any geometrical object is a product of its density and volume, where the volume is calculated from a product of its cross-sectional area and perpendicular height. If, in the present context, it is assumed that the shift in the surface plane (flank

side) caused by wear is uniform over the entire worn region and that this shift is analogous to a perpendicular height, then, given that density is constant, the weight loss could be represented by area. This hypothesis will be true if only it is assumed that the shift between wear test periods is infinitesimal.

Any error in the results presented can be attributed to both human and technical factors. The former is mainly from parallax error and the latter is caused by round-off errors, poor resolution of the Image Capture Board, limitation of the programming language, and poor lighting and fixture arrangements.

The results obtained from the above experiments on the system indicate that the developed system is effective and provides extra information about the wear land. The proposed breakage detection concept (i.e., comparing the coordinates of the nearest tool point with those of the tool tip) has been used in detecting tool breakage. The two new features of the wear land (i.e., perimeter and maximum width) are extra information which can be used to analyze wear. Furthermore, the system introduces the parameter of compactness to describe the morphology of the wear land. The compactness of a shape is the ratio of the square of its perimeter to its area.

The values obtained for compactness show that the wear shapes in experiment four are more compact than those of experiment three. For example, Figure 4.24 is more compact than Figure 2.12. This confirms, as stated in chapter two, that tool wear is dependent on the workpiece material.

The experiments designed to test the developed system and results obtained have been discussed in this chapter. The chapter also explained the information that can be derived from the new features and the possible use of compactness as a

vital feature in the analysis of the wear land. The next chapter presents a summary of the thesis and a list of possible areas requiring further research.

Chapter Five

Summary And Future Research

Chapters one through four have discussed the basics of tool life and wear, and machine vision. Furthermore, the thought processes involved in the development of the tool wear monitoring and breakage detection system and the discussion of the experimental results were also presented. Here, a summary of these chapters, conclusions inferred from the research and suggestions of areas that need further research are discussed.

5.1 Summary

Automated tool wear monitoring and tool breakage detection is an area of tremendous importance in the progress towards fully automating machining centres. This thesis has addressed the problem using machine vision approach.

Chapter one discussed the reasons for an automated tool monitoring and breakage detection process. The chapter also provided a general introduction to the thesis. In chapter two, a background information on cutting tools and machine vision and the review of research efforts in developing automated tool monitoring system were presented. This chapter highlighted the shortcomings of the present

machine vision based tool monitoring techniques.

The configuration and operation of the developed system were discussed in chapter three. The developed system, a two-dimensional tool wear monitoring and breakage detection system, uses an AT & T Image Capture Board to capture and digitize images sensed by a CCD RGB camera. A computer program written in Microsoft C V5.1 and run on an XT compatible Zenith Z150 series computer extracts the following five features which are then used to classify a tool:

1. length of the wear land
2. area of the wear land
3. perimeter of the wear land
4. maximum width of the wear land
5. nearest tool edge point to the original tool tip.

Chapter four discussed the experiments to test the system capability to detect breakage and to monitor tool wear over a brief cutting period. In addition, the chapter presented and discussed the results obtained. The following conclusions can be inferred from observations made during the course of the research:

1. A new algorithm to monitor tool wear and detect tool breakage has been developed.
2. Comparison between conventional measurements and those obtained using the developed algorithm showed that there exists no significant change between both methods at a confidence level of 0.002.

3. The system eliminates built-up edges by using a tool region identification scheme.

4. The vision system provides additional information on wear which was not possible from the conventional technique.

5. The system can be readily integrated with existing in-process techniques and a robot to produce a fully automated monitoring system.

6. Compactness is introduced as a source of information on wear land shape.

7. The system is dependent on lighting, background and tool surface conditions.

5.2 Future Research

The need for further study is supported by the flexibility a machine vision tool management system can provide and by the increased use of robotics and integrated machine vision systems in industries. The following are possible areas for further research.

1. Development of a better segmentation technique. This, however, will require the use of a more powerful image preprocessor and computer.

2. Categorizing flank wear into different classes by studying the relative position of the maximum wear width with respect to the length of the wear land measured along the tool edge.

3. Further experiments to understand the profile of compactness with respect to cutting parameters such as cutting time. Different workpiece materials give different flank wear shapes on the cutting tool. There is the potential for using

compactness ratio to relate the tool flank wear shape to different workpiece material.

4. Development of an expert tool management system.

5. The use of regression analysis to investigate the relationship between the features and to develop mathematical relationships between these and flank wear.

6. Cost analysis of the various tool monitoring and breakage detection systems. These systems can also be graded based on their effectiveness and efficiency.

References

[1] Abdou, I. E., and Pratt, W. K., "Quantitative Design and Evaluation of Enchancement/Thresholding Edge Detectors," *Proceedings of IEEE*, Vol. 67, No. 5, pp. 753-763, May 1979.

[2] Altintas, Y., Yellowley, I., and Tlusty, J., "The Detection of Tool Breakage in Milling Operations," *Journal of Engineering for Industry*, Vol. 110, pp. 271-277, August 1988.

[3] Ballard, D. H., "Generalizing the Hough Transform to Detect Arbitrary Shapes," Readings in Computer Vision; issues, problems, principles, and paradigms, Fischler, M. A., and Firchein, O., pp. 714-725, Morgan Kaufmann, San Mateo, 1987.

[4] Ballard, D. H., and Brown, C. M., Computer Vision, Prentice-Hall, New Jersey, 1982.

[5] Bartal, P., and Monostrori, L., "A Pattern Recognition Based Vibration Monitoring Module for Machine Tools," *Robotics and Computer Manufacturing*, Vol. 4, No. 3/4, pp. 465-469, 1988.

[6] Baxes, G. A., Digital Image Processing - A Practical Primer, Prentice-Hall, New Jersey, 1984.

[7] Bhattacharyya, A., and Ham, I., Design of Cutting Tools, ASTME, Dearborn, 1969.

[8] Boothryoyd, G., Fundamentals of Metal Machining, Edward Arnold Publishers Ltd., 1965.

[9] Cuppini, D., D'Errico, G., and Rutelli, G., "Tool Image Processing with Application to Unmanned Metal Cutting: A Computer Vision System for Wear Sensing and Failure Detection," *Proceedings of SPIE* , Vol. 701, pp. 416-422, 1986.

[10] Emel, E, and Kannatey-Asibu, E, "Tool Failure Monitoring in Turning by Pattern Recognition Analysis of AE Signals," *Journal of Engineering for Industry*, Vol. 110, pp. 137-145, May 1988.

[11] Ganapathy S., "Applications of Computer Vision," *SPIE Advances in Image Transmission*, Vol. 87, pp. 111-116, 1976.

[12] Gonzalez, R. C., and Wintz, P., Digital Image Processing, 2nd edition, Addison-Wesley, 1987.

[13] Iwata, K., and Moriwaki, T., "An Application of Acoustic Emission Measurement to In-Process Sensing of Tool Wear," *CIRP Annals*, Vol. 26, No. 1, pp. 21-26, 1977.

[14] Kundu, A., and Mitra, S. K., "A New Algorithm for Image Edge Extraction Using Statistical Classifier Approach," *IEEE Transactions on PAMI*, Vol. PAMI-9, No. 4, pp. 569-577, July 1987.

[15] Kittler, J., Illingworth, J., Foglein, J., and Paler, K., "An Automatic Thresholding Algorithm and Its Performance," *Proceedings of the Seventh International Conference on Pattern Recognition"*, Vol. 1, pp. 287-289, 1984.

[16] Lee, Y. H., Bandyopadhyay, P., and Kaminski, B., "Cutting Tool Wear Measurement Using Computer Vision," *SME Technical Paper MR86-934*, 1986.

[17] Li, H., and Kender, J. R., "Scanning the Issue: Computer Vision," *Proceedings of the IEEE*, Vol. 76, No. 8, pp. 859-861, August 1988.

[18] Li, P.G., and Wu, S.M., "Monitoring Drilling Wear States by Fuzzy Pattern Recognition Technique," *Journal of Engineering for Industry*, Vol. 110, pp. 297-300, August 1988.

[19] Merchant, M. E., Ernst, H., and Krabacher, E. J., "Radioactive Cutting Tools for Rapid Tool-Life Testing," *ASME Transactions*, Vol. 75, No. 4, pp. 549-559, May 1953.

[20] Nevatia, R., Machine Perception, Prentice-Hall Inc., New Jersey, 1982.

[21] Oberg, E., Jones, F. D., and Horton, H. L., Machinery's Handbook, Industrial Press Inc., New York, 22nd ed., 1985.

[22] Opitz, H., and Hake, O., "Wear Analysis of Hard Metal Turning Tools by Means of Radioisotopes," *Microtecnic*, Vol. 10, No. 1, pp. 5-9, 1956.

[23] Pavlidis, T., "A Review of Algorithms for Shape Analysis," *Computer Graphics and Image Processing*, Vol. 7, pp. 243-258, 1978.

[24] Pavlidis, T., Algorithms for Computer Graphics and Image Processing, Computer Science Press, Inc., Rockville, 1982.

[25] Pollack, H.W., Tool Design, Reston Publishing Company, Inc., Virginia, 1976.

[26] Rosenfeld, A., "Computer Vision: Basic Principles," *Proceedings of the IEEE*, Vol. 76, No. 8, pp. 863-868, August 1988.

[27] Rosenfeld, A., and Kak, A. C., Digital Picture Processing, 2nd edition, Vol. 2, Academic Press, New York, 1982.

[28] Subramanian, K, and Cook, N.H., "Sensing of Drill Wear and Prediction of Drill life, " *Journal of Engineering for Industry Transaction ASME*, Vol. 103, pp. 295-301, May 1977.

[29] Thangaraj, A, and Wright, P.K., "Computer-assisted Prediction of Drill-failure Using In-process Measurements of Thrust Force,"*Journal of Engineering for Industry*, Vol. 110, page 192-143, May 1988.

[30] Uehara, K., "New Attempts for Short Time Tool-Life Testing,"*CIRP Annals*, Vol. 22, No. 1, 1973.

[31] Van Vliet, L. J., and Yong, I. T., "A Nonlinear Laplacian Operator as Edge Detector in Noisy Images," *Computer Vision, Graphics and Image Processing*, Vol. 45, No. 2, pp. 167-195, February 1989.

[32] Weingartner, C., Machinists' Ready Reference, Prakken Publications, Ann Arbor, 6th ed., 1981.

[33] Weszka, J. S., "A Survey of Threshold Selection Techniques," *Computer Graphics and Image Processing*, Vol. 7, pp. 259-265, 1978.

[34] Weszka, J. S., Nagel, R. N., and Rosenfeld, A., "A Threshold Selection Technique,"*IEEE Transactions on Computers*, Vol. 23 , pp. 1322-1326, 1974.

[35] Weszka, J. S., and Rosenfeld, A., "Threshold Evaluation Techniques," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC-8, No. 8, pp. 622-629, August 1978.

[36] White, K. W., and Reynolds, R. J., The Fundamentals of Machine Vision Tutorial, Robots 12/Vision '88 Conference, June 6, 1988.

Appendix A

Neighbourhood and Connectivity

Digital images lose some geometric concepts because they are represented by discrete grids called pixels [24]. One way of correcting some of these defects is by developing some relationships among the pixels. Neighbourhood and Connectivity are examples of these relationships.

Neighbourhood could be understood from Figure A.1. A pixel, P, with (x, y) as coordinate has four vertical and horizontal neighbours, and four diagonal neighbours. The four vertical and horizontal neighbours are pixel numbers 0, 2, 4, 6 (even numbers) and have $(x+1, y), (x, y+1), (x-1, y), (x, y-1)$ as coordinates. The diagonal neighbours are pixel numbers 1, 3, 5, 7 (odd numbers) and have $(x+1, y+1), (x-1, y+1), (x-1, y-1), (x+1, y-1)$ as coordinates.

Thus, it is seen that the four vertical and horizontal neighbours are pixels which share a common side with the reference pixel, P. These sets of pixels are often referred to as d-neighbours or direct neighbours. Diagonal neighbours are pixels which touch the reference pixel only at a corner and are often called i-neighbours or indirect neighbours.

A combination of the d-neighbours and i-neighbours is called 8-neighbours. It

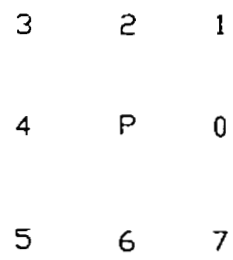


Figure A.1: A scheme for identifying the relative locations of pixels with respect to the centre pixel

is worth mentioning that some points in the 8-neighbours will be outside the image if the reference pixel is at the border point. However, there are various ways of accounting for this defect, one of which is the assumption that the missing rows or columns are a direct reflection of the previous row or column.

Connectivity is a concept mainly used in determining components of regions or boundary pixels in an image. It is based on the fulfillment of certain constraints (e.g., pixel intensities might not differ by more than a certain threshold value) and the establishment of adjacency (e.g., d-neighbours or i-neighbours). Thus, if we assume a binary image scenario, two pixels A and B can be connected if and only if they are adjacent in some sense and that there gray levels are both say 1 (if 0 is taken to be the background).

There are three different types of connectivity: 4-neighbour, 8-neighbour and mixed connectivity (m-connectivity). Given that two pixels A and B have satisfied the gray-level requirement, they are said to be 4-neighbour connected if pixel B is a d-neighbour of pixel A. The two pixels are said to be 8-neighbour connected if pixel B is an i-neighbour of pixel A. Mixed connectivity is mainly used in situations where the use of an 8-neighbour connectivity will be ambiguous (multiple paths).

In mixed connectivity, the two pixels are connected based on two criteria. The first is that pixel B is a d-neighbour or i-neighbour of pixel A. The second is that no pixel shall be d-neighbours of both pixels i.e., A and B. Figure A.2 is a schematic illustrating the effect of mixed connectivity.

Figure A.2: Schematic illustrating the role of mixed connectivity

From the above discussion, it is seen that a pixel A is adjacent to B if they are connected. A path is defined after the connectivity between the two points or pixels is established. In defining a path, it is assumed that a pixel has a nondimensionalized unit length and a unit square area. A path of length n from pixel A with coordinates (x, y) to pixel B with coordinates (a, b) is a sequence of points with coordinates $(x, y) = (x_0, y_0), (x_1, y_1), ..., (x_n, y_n) = (a, b)$ such that (x_i, y_i) is adjacent to, or a neighbour of, (x_{i-1}, y_{i-1}), $1 \leq i \leq n$ where n is the length of the path. Because a path length is dependent on whether it is obtained using 4-neighbour or 8-neighbour, it is always referred to as being a 4-path or an 8-path [27]. A detailed explanation of neighbourhood and connectivity is presented in [12,24,27].

Appendix B

Chain Codes

Representing or describing boundaries by using all the boundary coordinate points requires a large computer memory space. One of the several methods used in circumventing this is to save only the coordinates of the starting point and incremental changes. This method is referred to as chain coding and the code deduced or obtained is often called chain code.

Chain code is a concatenation of integers which is obtained by using numbers (codes) to denote the direction of travel along the boundary. Chain coding is therefore a representation scheme employed to develop compact data useful in computation of descriptors. The coding is based on d-neighbour or 8-neighbour connectivity (Figures B.1 and B.2, respectively) and the direction of travel is either clockwise or counterclockwise; though clockwise direction seems to be the accepted convention.

From Figure B.1, it is seen that movements are strictly restricted to four directions, i.e., east, west, north and south. A movement in the eastern direction is coded as 0, 1 and 2 in the northern and western directions, and 3 along the southern direction. In Figure B.2 an allowance is provided for movements along $\pm 45^{\circ}$ and $\pm 135^{\circ}$. Thus, unlike Figure B.1, there are eight directions (0, 1, 2, 3, 4, 5, 6, 7) in

Figure B.2 instead of four.

An example of how a chain code is obtained is presented in Figure B.3. The circles (i.e., nodes of the grid) represent boundary pixels, arrows denote direction of travel and bullets denote starting points.

The chain code is 0030003332222222110101 and is obtained by traversing the boundary, starting from the bullets, in clockwise direction and using the code in Figure B.1 to label directions relative to other pixels.

The boundaries represented by chain codes cannot be described if their starting coordinates are absent. This, however, can be overcome by adopting a "start point normalization" scheme. This is done [12] by treating the code as a circular sequence of numbers and redefining the starting point so that the resulting sequence of numbers forms an integer of minimum magnitude.

A difference or derivative chain code is a first difference of the original chain code obtained from traversing a boundary in clockwise direction. The integers which make up a derivative chain code are of mod 4 or mod 8 depending on whether a 4-neighbour or 8-neighour connectivity is used. The derivative or difference chain code is independent of rotation and represents the number of left hand or counter-clockwise turns of 45° or 90° needed by the chain segment in question to achieve the direction of the next chain element. From Figure B.3, the derivative chain code is 03100300300000303131. Detailed explanations of chain codes can be found in [4,12,27].

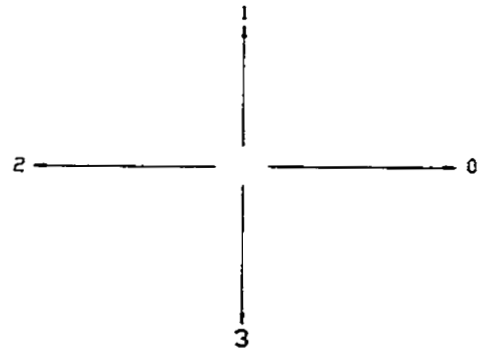


Figure B.1: 4-directional chain code directions

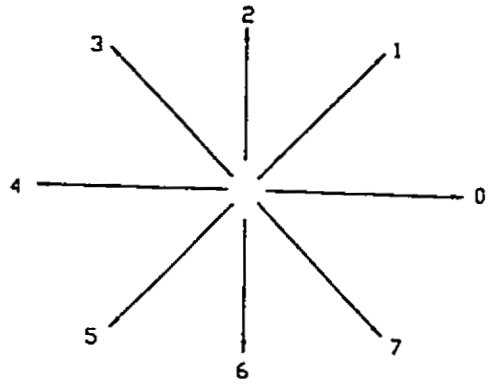


Figure B.2: 8-directional chain code directions

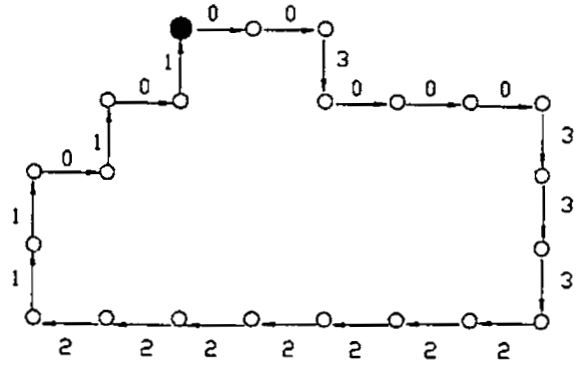


Figure B.3: Boundary representation using 4-directional chain code

Appendix C

Hough Transform

The Hough transform provides a technique used in the detection of specific or unique structural relationships between pixels in an image. The transform has been successfully applied in the detection of lines and curves. It is worth mentioning that this transformation has been generalized to detect curves with no simple analytic representation [3]. However, for the purpose of this research, the application of the technique to detect lines shall be discussed.

Assuming that there are, say, n points in an image which have been identified to be possible edge points, the problem is to find subsets of these points that lie on a line. This can be achieved using several methods, but one that might readily be adopted is to find, initially, all lines formed by every pair of points and then find all subsets of points that are close to particular lines. This method is, however, computationally expensive because the order of operation is approximately $n^3 + n^2$ [12].

Hough attempts to approach the above problem by redefining the image space (x, y) into a parameter space (c, m) (see Figures C.1 and C.2). Based on the general

equation for a line in slope-intercept form,

$$y = mx + c, \tag{C.1}$$

a point (x_i, y_i) will have the associated equation,

$$y_i = mx_i + c. \tag{C.2}$$

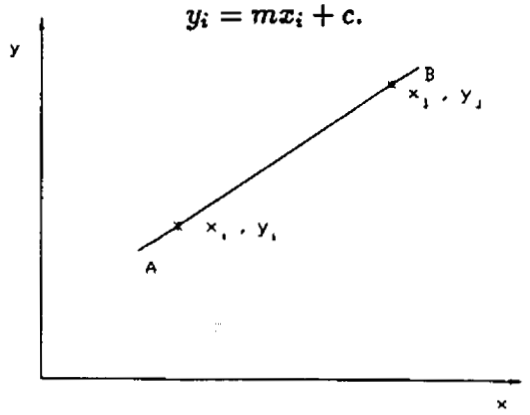


Figure C.1: Image space (x,y)

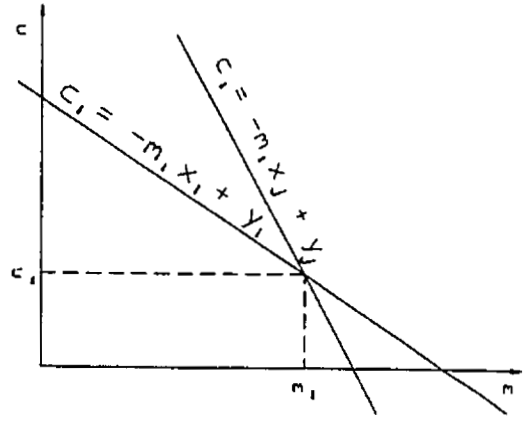


Figure C.2: Parameter space (c,m)

For this same point, (x_i, y_i), there exist infinitely many lines, with varying values of m and c, passing through it. Equation C.2 expressed in the parameter space will give

$$c = -x_i m + y_i. \tag{C.3}$$

This is the equation of a unique line for a fixed value pair (x_i, y_i). If another point (x_j, y_j) is selected such that a line through it on the parameter space intercepts the line obtained using (x_i, y_i) at, say (m_1, c_1), it will be seen that both points lie on a line in the image space. Thus, it is inferred that all points on the same straight line on the image space will have their lines on the parameter space intersecting at a point.

It is the above intersection concept that gives rise to the idea of using "accumulator cells" when writing the Hough transform software. Accumulator cells are arrays used in subdividing the parameter space. In practice, a bound is set for m values (m_{min}, m_{max}) and c values (c_{min}, c_{max}), and a two-dimensional array is created such that the coordinate (i, j), with accumulator array $A(i, j)$, denotes the pixel point associated with the parameter space coordinates (m_i, c_i).

All the accumulator cells are initialized to zero at the start of the program. Then, for every coordinate, say (x_k, y_k), in the image plane, m is allowed to increase over its range and for each value, a corresponding value of c is obtained using equation C.3. These values of c are rounded off to the nearest integer based on the scale used on the c-axis. In explaining how the accumulator cells are built, we assume that the use of a m value, say m_p, resulted in a c value, say c_q. Then, the accumulator cell $A(p, q)$ is incremented by one. At the end of the iteration, if the highest number of counts (say T) is found in the accumulator cell $A(i, j)$ then it is implied that on the image plane (i.e., xy plane), there are T points on the line

$$y = m_i x + c_i. \tag{C.4}$$

The use of the slope-intercept equation will be impossible when detecting or representing vertical lines [12,27]. This is because the slope and intercept tend to

inifinity as the line tends to a vertical position. A way of circumventing this problem is to represent a line in its polar form, i.e.,

$$xcos\theta + ysin\theta = \rho. \tag{C.5}$$

The techniques applied to slope-intercept representation are applicable here, the difference being that loci which are sinusoidal curves are generated in the parameter space, $\rho\theta$. A detailed discussion of the Hough transform can be found in [3,4,12,27].

Appendix D

Source Code

```
/*************************************************************************
 * NAME: MR. DONATUS C. D. OGUAMANAM              DATE: MAY 7, 1990 *
 *                                                                       *
 * THIS PROGRAM MONITORS THE CONDITION (WEAR/BREAKAGE) OF A GIVEN        *
 * SINGLE POINT CUTTING TOOL. IT ACCEPTS IMAGES WHICH ARE ALREADY        *
 * STORED IN THE HARD DRIVE.                                             *
 *                                                                       *
 * THE PROGRAM IS WRITTEN IN MICROSOFT C V5.1 AND RUNS ON A ZENITH       *
 * XT COMPATIBLE MICROCOMPUTER                                           *
 *                                                                       *
 * ALGORITHM:                                                            *
 *     1. capture and digitization of the image                          *
 *     2. segmentation of the image                                      *
 *     3. determination of the tool tip by using Hough transform         *
 *     4. check for breakage and wear by calculating parameters          *
 *     5. report on verdict                                              *
 *************************************************************************/

/*************************************************************************
 * VARIABLES USED IN MAIN PROGRAM:                                       *
 *                                                                       *
 * tool_image_name[] ------------ stores the image to be processed       *
 * refx ------------------------- x coordinate of a reference point      *
 * refy ------------------------- y coordinate of a reference point      *
 * pointx ----------------------- x coordinate of the tool tip           *
 * pointy ----------------------- y coordinate of the tool tip           *
 * w_perimeter ------------------ perimeter of the wear land             *
 * w_width ---------------------- width of the wear land                 *
 * w_length --------------------- length of the wear land               *
```

```
 * w_area ------------------------ area of the wear land            *
 * break_thres ------------------- breakage threshold               *
 * wear_thres -------------------- wear threshold to predict wear   *
 ************************************************************************/

# include       <stdio.h>
# include       <stdlib.h>
# include       <float.h>
# include       <math.h>
# include       <ctype.h>
# include       <icbdev.h>

# define        OVERBIT 0x8000
# define        BLACK   0x0000
# define        WHITE   0x7fff
# define        YELLOW  0x7fe0
# define        GREEN   0x03E0
# define        BLUE    0x1cff
# define        RED     0x7c00
# define        TRUE    1
# define        FALSE   0
# define        CONV    0.0174532

extern struct ICBStruct icb;

main()
{
        char            tool_image_name[20];
        int             refx, refy, pointx, pointy, breakage_finder(),
                        continue_monitor();
        double          w_perimeter, w_width, w_length, w_area, break_thres,
                        wear_thres;
        void            wear_region(), fill_out(), tracer(), width_finder(),
                        length_finder(), area_finder(), hough_transform(),
                        starting_point_finder();

     do {
        /* Hardware Initialization */

        GraphInit();
        SetOverMode();
        GenLock(1);
        SetInterLace(0);
```

```
ERASE(OVERBIT);
SetLiveMode();                  /* Go live */
GrabFrame();                    /* Freeze the image */
SetDispMode();                  /* Go to display mode */

/* Display the tool */

printf("Enter the name of the file containing your ICB image\n");
gets(tool_image_name);
GetPic(tool_image_name);

/* Enter the thresholds for breakge and wear */

printf("Enter breakage and wear thresholds respectively\n");
scanf("%lf %lf",&break_thres,&wear_thres);

/* Identify wear region by changing its color to yellow */

wear_region();


/* Fill out spots in the wear region */

fill_out();

/* Identify tool region */

hough_transform(&pointx,&pointy);

if(breakage_finder(pointx,pointy,break_thres))
        printf("Tool is chipped: REPLACE\n");
else {
        starting_point_finder(&refx,&refy);
        tracer(refx,refy,&w_perimeter);
        area_finder(&w_area);
        if(w_area >= 0.016){
                width_finder(pointy,&w_width);
                length_finder(pointx,&w_length);
                printf("\n\n");
                printf("The wear land length = %5.3f\n",w_length);
                printf("The wear land width  = %5.3f\n",w_width);
                printf("The  perimeter       = %5.3f\n",w_perimeter);
                printf("The area is          = %5.3f\n",w_area);
```

```
                        if(wear_thres <= w_width)
                                printf("\nTHE TOOL NEEDS TO BE CHANGED\n");
                        else
                                printf("\n   THE TOOL IS OKAY\n");
                }
                else
                                printf("\n  THE TOOL IS OKAY\n");
        }
        GraphEnd(); }while(continue_monitor());
}


/*************************************************************************
 * This function identifies the wear region pixel by colouring it yellow*
 *                                                                       *
 * VARIABLES                                                             *
 *                                                                       *
 *      x, y ------------------- counters                                *
 *************************************************************************/

void    wear_region()
{
        int     x, y;

        for(y = 0; y <= 180; y++)
                for(x = 0; x <= 250; x++){
                        if((GetPix(x,y) <= -500) && (GetPix(x,y) >= -2000)){
                                if(abs(GetPix(x,y) + 1025) > 20)
                                        PutPix(x,y,BLUE);
                                else
                                        PutPix(x,y,YELLOW);
                        }
                        else if((GetPix(x,y) >= -28000) &&
                                        (GetPix(x,y) <= -3000)){
                                if(GetPix(x,y) > -2000)
                                        PutPix(x,y,YELLOW);
                                else
                                        PutPix(x,y,GREEN);
                        }
                        else if(GetPix(x,y) > -500)
                                PutPix(x,y,YELLOW);
                }
}
```

```
/*************************************************************************
 * This function filters out spots in the wear region                    *
 *                                                                       *
 * VARIABLES                                                             *
 *                                                                       *
 *      x, y   -------------- counters                                   *
 *************************************************************************/


void    fill_out()
{
        int     x, y;
        void    filler();

        for(y = 0; y <= 180; y++)
                for(x = 0; x <= 250; x++)
                        if((GetPix(x,y) == BLUE) || (GetPix(x,y) == GREEN)){
                                if((GetPix(x-1,y) == YELLOW) &&
                                                (GetPix(x+1,y) == YELLOW))
                                        PutPix(x,y,YELLOW);
                                else if((GetPix(x,y+1) == YELLOW) &&
                                                (GetPix(x,y-1) == YELLOW))
                                        PutPix(x,y,YELLOW);
                        }
        filler();
}


/*************************************************************************
 * This function fills out spots in someother areas                      *
 *                                                                       *
 * VARIABLES                                                             *
 *                                                                       *
 *      x, y  ---------------------  counters                            *
 *************************************************************************/


void    filler()
{
        int     x, y;
        void    final_fill();
```

```
        for(y = 1; y <= 180; y++)
                for(x = 0; x < 250; x++)
                        if((GetPix(x,y) != GREEN) && (GetPix(x,y) != BLUE))
                                if((GetPix(x,y) != YELLOW) && (GetPix(x,y) >
-50
                                                                        00))
                                        PutPix(x,y,BLUE);
        final_fill();
}


/*************************************************************************
 * This function corrects any pixel that is wrongly filled or left out *
 *                                                                       *
 * VARIABLES                                                             *
 *                                                                       *
 *      x, y  ----------------------- counters                           *
 *************************************************************************/


void    final_fill()
{
        int x, y;

        for(y = 1; y <= 180; y++)
                for(x = 0; x < 250; x ++)
                        if(GetPix(x,y) == BLUE){
                                if((GetPix(x,y+1) == YELLOW) &&
                                                (GetPix(x,y-1) == GREEN))
                                        PutPix(x,y,YELLOW);
                                else if((GetPix(x,y-1) == GREEN) &&
                                                (GetPix(x,y+1) == YELLOW))
                                        PutPix(x,y,YELLOW);
                                else if((GetPix(x-1,y) == YELLOW) &&
                                                (GetPix(x+1,y) == GREEN))
                                        PutPix(x,y,YELLOW);
                                else if((GetPix(x-1,y) == GREEN) &&
                                                (GetPix(x+1,y) == YELLOW))
                                        PutPix(x,y,YELLOW);
                                else if((GetPix(x,y+1) == GREEN) &&
                                                (GetPix(x,y-1) == GREEN))
                                        PutPix(x,y,GREEN);
```

```
                                else if((GetPix(x+1,y) == GREEN) &&
                                        (GetPix(x-1,y) == GREEN))
                                    PutPix(x,y,GREEN);
                                else if((GetPix(x-1,y) == YELLOW) &&
                                        (GetPix(x+1,y) == YELLOW))
                                    PutPix(x,y,YELLOW);
                            }
}

/*************************************************************************
 * This function identifies the tool region: It implements the Hough    *
 * transform to identify the tool top edge and calls on a function to    *
 * determine the equation of the tool flank side edge.                   *
 *                                                                       *
 * VARIABLES                                                             *
 *                                                                       *
 * xcord  -------------------------  x coordinate of the tool tip       *
 * ycord  -------------------------  y coordinate of the tool tip       *
 * hval[] -------------------------  the number of points in the        *
 *                                   largest accumulator cell for each *
 *                                   iteration for determining the     *
 *                                   flank side equation               *
 * mval[] -------------------------  slope of the most likey equation   *
 *                                   obtained during each iteration    *
 *                                   for determining the flank side    *
 *                                   equation                          *
 * cval[] -------------------------  intercept of the most likely       *
 *                                   equation obtained during each     *
 *                                   iteration for determining the     *
 *                                   flank side equation               *
 * m2 -----------------------------  slope of the most likely equation *
 *                                   determine from a comparison of all*
 *                                   the likely equations obtained for *
 *                                   each iteration                    *
 * c2 -----------------------------  intercept of the most likely       *
 *                                   equation determined from a        *
 *                                   comparison of all the likely      *
 *                                   equations obtained for each       *
 *                                   iteration                         *
 * sval, cnt, knt, x, y, m, mm  ----  counters                         *
 * h[][]  -------------------------  accumulator cells                 *
 * m1, mhval  ---------------------  slope of the most likely equation *
```

```
 *                                   of the tool top edge              *
 * c1, chval  ---------------------  intercept of the most likely      *
 *                                   equation of the tool top edge     *
 * refx  --------------------------  x coordinate of a reference point *
 ************************************************************************/

void    hough_transform(xcord,ycord)
int     *xcord, *ycord;
{

        int     hval[3], mval[3], cval[3], sval, mhval, chval;
        int     h[60][30], cnt, knt, x, y, c, y1, refx;
        float   m;
        double  m1, m2, c1, c2, mm;
        void    side_hough_transform(), max_horz_hough(), rectify();

        /* Initializing the accumulator cells */

        for(cnt = 0; cnt < 30; cnt++)
                for(knt = 0; knt < 60; knt++)
                        h[knt][cnt] = 0;

        /* Locate starting position */

        y = 180;
        x = 130;
        while(GetPix(x,y) == BLUE) x++;

        refx = x - 3;
        x = refx - 19;
        while(x < refx){
                y = 170;
                while(y >= 100){
                        if((GetPix(x,y) == BLUE) && (y > 99)) y--;
                        if((y != 99) && (GetPix(x,y) != YELLOW)){
                                if(GetPix(x,y) == GREEN){
                                        y1 = y - 100;
                                        m = 0;
                                        while(m <= 0.1){
                                                mm = tan(m * CONV);
                                                c = (int)(y1 - mm * x);
                                                if(c >= 0)
                                                        h[c][(int)(m * 100)]++;
```

```
                                        m += 0.01;
                                }
                                y = 99;
                        }
                }
        }
        x++;
}

max_horz_hough(h,&mhval,&chval);
c1 = chval + 100;
m1 = mhval;
rectify((int)c1);
side_hough_transform(h,hval,mval,cval,&sval);
m2 = mval[sval] * 0.5;
m2 = ( 70 + sval * 5 + m2) * CONV;
m2 = -tan(m2);
if (sval == 2)
        c2 = cval[sval] * 10 + 500;
else
        c2 = cval[sval] * 10;
x = (int)((c1 - c2) / (m2 - m1));
y = (int)(m1 * x + c1);
*xcord = x;
*ycord = y;
}

/*************************************************************************
 * This function determines the accumulator cell that will most likely*
 * represent the tool top edge equation                                  *
 *                                                                       *
 * VARIABLES                                                             *
 *                                                                       *
 * cnt, knt  -------------------  counters                              *
 * hou[][]  --------------------  accumulator cells                      *
 * *maxi  ----------------------  intercept of the most likely          *
 *                                equation of the tool top edge          *
 * *maxj  ----------------------  slope of the most likely equation     *
 *                                of the tool top edge                   *
 * max  ------------------------  dummy variable                         *
 *************************************************************************/
```

```
void    max_horz_hough(hou,maxj,maxi)
int     hou[60][30], *maxi, *maxj;
{
        int     cnt, knt, max;

        max = 0;

        for(cnt = 0; cnt < 60; cnt++)
                for(knt = 0; knt < 30; knt++)
                        if(hou[cnt][knt] > max){
                                max = hou[cnt][knt];
                                (*maxi) = cnt;
                                (*maxj) = knt;
                        }


}


/*****************************************************************************
 * This function removes possible built_up edges and erroneous information*
 * introduced by the illumination                                            *
 *                                                                           *
 * VARIABLES                                                                 *
 *                                                                           *
 * val  ---------------------------  intercept of the most likely       *
 *                                   equation of the tool top edge       *
 * x, y  --------------------------  counters                             *
 *****************************************************************************/


void    rectify(val)
int     val;
{
        int     x, y;

        for(y = val + 1; y <= 180; y++)
                for(x = 10; x <= 250; x++)
                        if((GetPix(x,y) == GREEN) || (GetPix(x,y) == YELLOW))
                                PutPix(x,y,BLUE);
}
```

```
/*************************************************************************
 * This function determines the equation of the flank side              *
 *                                                                       *
 * VARIABLES                                                             *
 *                                                                       *
 * x, y, cnt, knt, x1, y1, refx, m  ------ counters                      *
 * c  ---------------------------------- intercept of any equation      *
 * flag  -------------------------------- an indicator                   *
 * m2  ---------------------------------- slope of any equation         *
 * m1  ---------------------------------- used in shortening the        *
 *                                       expression used in              *
 *                                       calculating m2                  *
 * hval[]  ------------------------------ the number of points in       *
 *                                       the largest accumulator         *
 *                                       cell for each iteration         *
 *                                       during the detemination of      *
 *                                       the flank side equation         *
 * mval[]  ------------------------------ slope of the most likely       *
 *                                       equation obtained during        *
 *                                       each iteration to               *
 *                                       determine the flank side        *
 *                                       equation                        *
 * cval[]  ------------------------------ intercept of the most          *
 *                                       likely equation obtained        *
 *                                       during each iteration to        *
 *                                       determine the flank side        *
 *                                       equation                        *
 * h[][]  ------------------------------- accumulator cells              *
 * sval1  ------------------------------- counter                        *
 *************************************************************************/


void    side_hough_transform(h,hval,mval,cval,sval1)
int     hval[3], mval[3], cval[3], h[60][30], *sval1;
{
        int     x, y, cnt, knt, refx, x1, y1, c, flag;

        float   m;
        double  m1, m2;
        void    maxima(), max_side_hough();
```

```
/* Initialize cells to contain the most likely cells obtained
   for each level of iteration */

for(cnt = 0; cnt < 3; cnt++){
        hval[cnt] = 0;
        mval[cnt] = 0;
        cval[cnt] = 0;
}


for(flag = 0; flag < 3; flag++){
        x = refx = 130;
        y = 40;

        /* Initialize accumulator cells */

        for(cnt = 0; cnt < 30; cnt++)
                for(knt = 0; knt < 60; knt++)
                        h[knt][cnt] = 0;

        while(y < 110){
                while(GetPix(x,y) == GREEN) x--;
                if((GetPix(x,y) != GREEN) &&
                                (GetPix(x,y) != YELLOW)){
                        x1 = x + 1;
                        m = 0 ;
                        while(m <= 0.1){
                                m1 = 70.0 + flag * 5 + m * 50;
                                m2 = m1 * CONV;
                                c = (int)(y + tan(m2) * x1);
                                if(flag != 2)
                                        h[(int)c / 10][(int)(m * 100)]++
                                                                ;
                                else
                                        h[(int)((c - 500) / 10)][(int)(m
                                                        * 100)]++;
                                m += 0.01;
                        }
                        x = refx;
                        y++;
                }
                if(GetPix(x,y) == YELLOW){
                        while(GetPix(x,y) == YELLOW) x--;
```

```
                                if(GetPix(x,y) == BLUE){
                                        x = refx;
                                        y++;
                                }
                        }
                }
                maxima(&hval[flag],&cval[flag],&mval[flag],h);
        }

        max_side_hough(hval,sval1);
}


/*************************************************************************
 * This function finds for a particular level of iteration, the most *
 * likely accumulator cell to be used in representing the equation of*
 * the flank side                                                    *
 *                                                                   *
 * VARIABLES                                                         *
 *                                                                   *
 * *max  ----------------------- the number of points in a given    *
 *                                accumulator cell                   *
 * *xvalue ---------------------- intercept of a given equation      *
 * *yvalue ---------------------- slope of a given equation         *
 * hou[][] ---------------------- accumulator cells                  *
 * cnt, knt --------------------- counters                          *
 *************************************************************************/

void    maxima(max,xvalue,yvalue,hou)
int     *max, *xvalue, *yvalue, hou[60][30];
{
        int     cnt, knt;

        for(cnt = 0; cnt < 60; cnt++)
                for(knt = 0; knt < 30; knt++)
                        if(hou[cnt][knt] > (*max)){
                                (*max) = hou[cnt][knt];
                                (*xvalue) = cnt;
                                (*yvalue) = knt;
                        }
}
```

```
/*****************************************************************************
 * This function compares the major cells already identified to determine *
 * the one that best descsribes the equation                                *
 *                                                                          *
 * VARIABLES                                                                *
 *                                                                          *
 * cnt, *knt  ------------------------  counters                            *
 * max[]  --------------------------  the number of points in the          *
 *                                    largest accumulator cell for each *
 *                                    iteration during the determination*
 *                                    of the flank side equation          *
 * maxim  --------------------------  dummmy variable                      *
 *****************************************************************************/


void    max_side_hough(max,knt)
int     max[3], *knt;
{
        int     cnt,  maxim;

        maxim = 0;
        for(cnt = 0; cnt < 3; cnt++)
                if(max[cnt] > maxim){
                        maxim = max[cnt];
                        (*knt) = cnt;
                 }

}


/*****************************************************************************
 * This function determines if a given tool is broken or not              *
 *                                                                          *
 * VARIABLES                                                                *
 *                                                                          *
 * xcord  -------------  x coordinate of the tool tip                       *
 * ycord  -------------  y coordinate of the tool tip                       *
 * threshold  ---------  breakage threshold                                 *
 * x1  ----------------  x coordinate of the nearest point to the tool *
 *                       tip                                                *
 * y1  ----------------  y coordinate of the nearest point to the tool *
 *                       tip                                                *
 * xxcord  ------------  difference in the x-axis values of the tool   *
```

```
 *                        tip and the nearest point                      *
 * yycord  ------------   difference in the y-axis values of the tool    *
 *                        tip and the nearest point                      *
 ************************************************************************/


int     breakage_finder(xcord,ycord,threshold)
int     xcord, ycord;
double  threshold;
{
        int     x1, y1;
        void    nearest_point_finder();
        double  xxcord, yycord;

        nearest_point_finder(&x1,&y1);
        xxcord = (abs(xcord - x1)) / 50.0 ;
        yycord = (abs(ycord - y1)) / 50.0;
        if((xxcord  > threshold) || (yycord  > threshold))
                return(TRUE);
        else
                return(FALSE);
}


/************************************************************************
 * This function determines the coordinates of the nearest tool body    *
 * to the tool tip                                                      *
 *                                                                      *
 * VARIABLES                                                            *
 *                                                                      *
 * *x1  ---------------  x coordinate of the nearest point to the tool *
 *                       tip                                            *
 * *y1  ---------------  y coordinate of the nearest point to the tool *
 *                       tip                                            *
 * x, y ---------------  counters                                       *
 * flag ---------------  indicator                                      *
 ************************************************************************/


void    nearest_point_finder(xl,yl)
int     *xl, *yl;
{
        int     x, y, flag;
```

```
        x = 40;
        flag = TRUE;
        while((flag) && (x < 230)){
                y = 160;
                while((GetPix(x,y) != YELLOW) && (y > 50)) y--;

                if((y >= 50) && (GetPix(x,y) == YELLOW)){
                        (*xl) = x;
                        (*yl) = y;
                        flag  = FALSE;
                }
                else if((y == 50) && (GetPix(x,y) != YELLOW))
                        x++;
        }
}


/*************************************************************************
 * This function is used in determining the point to start doing    *
 * the contour tracing                                              *
 *                                                                  *
 * VARIABLES                                                        *
 *                                                                  *
 * *xref  --------------  x coordinate of a reference point        *
 * *yref  --------------  y coordinate of a reference point        *
 * x, y  ---------------  counters                                  *
 * indicator  ----------  flag                                      *
 *************************************************************************/


void    starting_point_finder(xref,yref)
int     *xref, *yref:
{
        int     x, y, indicator, is_other_pix_in_set();

        y = 198;
        indicator = FALSE;
        while((y >= 0) && (indicator == FALSE)){
                x = 0;
                while((x <= 210) && (indicator == FALSE)){
                        if(GetPix(x,y) != YELLOW) x++;
```

```
                        else if(is_other_pix_in_set(x,y) >= 1){
                                indicator = TRUE;
                                (*xref) = x;
                                (*yref) = y;
                        }
                }
                if(!indicator)
                        y--;
        }
}


/*****************************************************************************
 * This function is used in selecting the reference point to start tracing *
 *                                                                           *
 * VARIABLES                                                                 *
 *                                                                           *
 * xval, yval, count  --------------------- counters                        *
 *****************************************************************************


int     is_other_pix_in_set(xval,yval)
int     xval, yval;
{
        int     count = 0;

        if(GetPix(xval - 1,yval + 1) == YELLOW) count++;
        if(GetPix(xval,yval + 1) == YELLOW) count++;
        if(GetPix(xval + 1,yval + 1) == YELLOW) count++;
        if(GetPix(xval + 1,yval) == YELLOW) count++;
        if(GetPix(xval - 1,yval - 1) == YELLOW) count++;
        if(GetPix(xval,yval - 1) == YELLOW) count++;
        if(GetPix(xval + 1,yval - 1) == YELLOW) count++;

        return(count);
}

/*****************************************************************************
 * This function traces a given contour                                      *
 *                                                                           *
 * VARIABLES                                                                 *
```

```
 *                                                                    *
 * rx    --------------------  x coordinate of a reference point     *
 * ry    --------------------  y coordinate of a reference point     *
 * *peri --------------------  perimeter of the wear land            *
 * dir   --------------------  direction of travel                   *
 * indic1, indicf, flag -----  indicators                             *
 * deltax -------------------  change in x-axis value                *
 * deltay -------------------  change in y-axis value                *
 * x, y, count  -------------  counters                              *
 * oldx  --------------------  x coordinate of an old pixel of       *
 *                             interest                               *
 * oldy  --------------------  y coordinate of an old pixel of       *
 *                             interest                               *
 * curx  --------------------  x coordinate of the current pixel of  *
 *                             interest                               *
 * cury  --------------------  y coordinate of the current pixel of  *
 *                             interest                               *
 **********************************************************************/


void    tracer(rx,ry,peri)
int     rx, ry;
double  *peri;
{
        int     dir, indic1, indicf, curx, cury, x, y,
                is_diff(), count, flag, oldx, oldy;
        double  deltax, deltay;
        void    is_in();

        curx = x = rx;
        cury = y = ry;
        (*peri) = 0.0;
        indic1 = TRUE;
        dir = 6;
        while(is_diff(rx,ry,curx,cury) || indic1){
                indicf = FALSE;
                oldx = curx;
                oldy = cury;
                count = 1;
                while(!indicf && (count <= 3)){
                        flag = FALSE;
                        is_in(dir - 1,&x,&y,&flag);
                        if(flag){
```

```
                        dir = dir - 2;
                        curx = x;
                        cury = y;
                        indicf = TRUE;
                }
                else{
                        is_in(dir,&x,&y,&flag);
                        if(flag){
                                curx = x;
                                cury = y;
                                indicf = TRUE;
                        }
                        else{
                                is_in(dir + 1,&x,&y,&flag);
                                if(flag){
                                        curx = x;
                                        cury = y;
                                        indicf = TRUE;
                                }
                                else
                                        dir += 2;
                        }
                        count++;
                }
        }
        indic1 = FALSE;
        deltax = curx - oldx;
        deltay = cury - oldy;
        (*peri) = (*peri) + sqrt(fabs(deltax) + fabs(deltay));
    }
    (*peri) = (*peri) / 50.0;
}


/*************************************************************************
 * This function assigns the right coordinates and flag for the next *
 * step                                                                  *
 *                                                                       *
 * VARIABLES                                                             *
 *                                                                       *
 * dir   --------------------  direction of travel                      *
 * *i, *j, x, y  ------------------  counters                           *
```

```
 * *flag  -------------------  indicator                                    *
 ***************************************************************************/


void    is_in(way,i,j,ind)
int     way, *i, *j, *ind;
{
        int     x, y,modi();
        void    indic();

        switch (modi(way)){
                case 0:
                        x = (*i) + 1;
                        y = (*j);
                        break;
                case 1:
                        x = (*i) + 1;
                        y = (*j) + 1;
                        break;
                case 2:
                        x = (*i);
                        y = (*j) + 1;
                        break;
                case 3:
                        x = (*i) - 1;
                        y = (*j) + 1;
                        break;
                case 4:
                        x = (*i) - 1;
                        y = (*j);
                        break;
                case 5:
                        x = (*i) - 1;
                        y = (*j) - 1;
                        break;
                case 6:
                        x = (*i);
                        y = (*j) - 1;
                        break;
                case 7:
                        x = (*i) + 1;
                        y = (*j) - 1;
                        break;
```

```
        }
        if(GetPix(x,y) == YELLOW){
                if((GetPix(x - 1,y) != YELLOW) ||
                                (GetPix(x + 1,y) != YELLOW)){
                        (*ind) = TRUE;
                        (*i) = x;
                        (*j) = y;
                }
                else if((GetPix(x,y + 1) != YELLOW) ||
                                (GetPix(x,y - 1) != YELLOW)){
                        (*ind) = TRUE;
                        (*i) = x;
                        (*j) = y;
                }
                else
                        (*ind) = FALSE;
        }
        else
                (*ind) = FALSE;
}


/*****************************************************************
 * This function returns a modulo of 8                           *
 *                                                               *
 * VARIABLES                                                     *
 *                                                               *
 * direction  ---------------  direction of travel             *
 *****************************************************************/


int     modi(direction)
int     direction;
{
        return(direction - 8 * (direction / 8));
}




/*********************************************************************
 * This function determines if the point is the same as that of      *
```

```
 * the reference point                                                *
 *                                                                    *
 * VARIABLES                                                          *
 *                                                                    *
 * rx    --------------------  x coordinate of a reference point     *
 * ry    --------------------  y coordinate of a reference point     *
 * cx    --------------------  x coordinate of the current pixel of  *
 *                             interest                               *
 * cy    --------------------  y coodinate of the current pixel of   *
 *                             interest                               *
 **********************************************************************/

int     is_diff(rx,ry,cx,cy)
int     cx, cy, rx, ry;
{
        if((rx == cx) && (ry == cy))
                return(FALSE);
        else
                return(TRUE);
}




/**********************************************************************
 * This function determines the area of the wear region               *
 *                                                                    *
 * VARIABLES                                                          *
 *                                                                    *
 * *area1  ----------------------- area of the wear land              *
 * xval, yval -------------------- counters                           *
 **********************************************************************/


void    area_finder(area1)
double  *area1;
{
        int     xval, yval;

        for((*area1) = 0, yval = 20; yval <= 180; yval++)
                for(xval = 20; xval <= 200; xval++)
```

```
                        if(GetPix(xval,yval) == YELLOW) (*area1)++;
        (*area1) = (*area1) / 2500;
}


/*********************************************************************
 * This function determines the minimum wear point                   *
 *                                                                   *
 * VARIABLES                                                         *
 *                                                                   *
 * ycord  ---------------------- y coordinate of the tool tip        *
 * *minwid  -------------------- width of the wear land              *
 * xcnt, ycnt, minx, miny ------ counters                            *
 *********************************************************************/


void    width_finder(ycord,minwid)
int     ycord;
double  *minwid;
{
        int     xcnt, ycnt, minx, miny;

        miny = 2000;

        for(xcnt = 0; xcnt <= 250; xcnt++){
                ycnt = 0;
                while((GetPix(xcnt,ycnt) != YELLOW) && (ycnt <= 180)) ycnt++;
                if((ycnt == 180) && (GetPix(xcnt,ycnt) != YELLOW)) ycnt = 0;
                else if(GetPix(xcnt,ycnt) == YELLOW)
                        if(miny > ycnt){
                                miny = ycnt;
                                minx = xcnt;
                        }
        }
        (*minwid) = (ycord - miny + 1) / 50.0;
}


/*********************************************************************
 * This function is used in determining the length of the wear land  *
 *                                                                   *
 * VARIABLES                                                         *
```

```
 *                                                                    *
 * xcord  ---------------------- x coordinate of the tool tip        *
 * *leng  ---------------------- length of the wear land             *
 * x2     ---------------------- x coordinate of the extreme-right   *
 *                               point                                *
 **********************************************************************/


void    Length_finder(xcord,leng)
int     xcord;
double  *leng;
{
        int     x2;
        void    extreme_right_finder();

        extreme_right_finder(&x2);
        (*leng) = (x2 - xcord+1) / 50.0;
}



/**********************************************************************
 * This function determines the coordinate of the extreme right pixel  *
 *                                                                     *
 * VARIABLES                                                           *
 *                                                                     *
 * *xr     ----------------- x coordinate of the extreme-right point   *
 * x, y ------------------- counters                                   *
 * flag ------------------- indicator                                  *
 **********************************************************************/


void    extreme_right_finder(xr)
int     *xr;
{
        int     x, y, flag;

        x = 220;
        flag = TRUE;
        while((flag) && (x > 40)){
                y = 160;
                while((GetPix(x,y) != YELLOW) && (y > 50)) y--;
                if((y >= 50) && (GetPix(x,y) == YELLOW)){
```

```
                        (*xr) = x;
                        flag  = FALSE;
                }
                else if((y == 50) && (GetPix(x,y) != YELLOW))
                        x--;
        }
}


/*****************************************************************
 * This function determines if monitoring is to be continued     *
 *                                                               *
 * VARIABLES                                                     *
 *                                                               *
 * choice  ----------------------  option                        *
 *****************************************************************/


int     continue_monitor()
{
        int     choice, flush_it();
        void    message(), is_alpha();

        choice = getchar();
        do {
                message();
                choice = getchar();
                if((choice != '\n') && flush_it()){
                        if(!isalpha(choice))  is_alpha(&choice);
                        if(!islower(choice)) choice = tolower(choice);
                }
                else
                        choice = 'z';
        }while((choice != 'c') && (choice != 'q'));
        if(choice == 'c')
                return(TRUE);
        else
                return(FALSE);

}


/* This function gives out the task required */
```

```
void    message()
{
        printf("Please enter 'c' to continue and 'q' to quit\n");
}



/*********************************************************************
 * This function determines if the key typed is an alphabet or not *
 *                                                                   *
 * VARIABLES                                                         *
 *                                                                   *
 * *letter   -------------------  option                            *
 *********************************************************************/


void    is_alpha(letter)
int     *letter;
{

        void    message();
        int     flush_it();

        do {
                message();
                (*letter) = getchar();
                if(!flush_it())
                        (*letter) = 3;
        }while(!isalpha(*letter));
}



/*********************************************************************
 * This function gets rid of other characters that might have been *
 * entered                                                           *
 *                                                                   *
 * VARIABLES                                                         *
 *                                                                   *
 * dummy  ---------------  dummy variable                           *
 * count  ---------------  counter                                  *
```

```
*********************************************************************/

int     flush_it()
{
        int     dummy, count;

        count = 0;
        do{
                dummy = getchar();
                count++;
        }while(dummy != '\n');

        if(count <= 1)
                return(TRUE);
        else
                return(FALSE);
}
```