9390 9343

1492114

A DOMAIN INDEPENDENT ADAPTIVE IMAGING SYSTEM FOR VISUAL INSPECTION

STEPHEN PANAYIOTOU

A thesis submitted in partial fulfilment of the requirements of the University of Greenwich for the degree of Doctor of Philosophy

November 1995

the University of Greenwich London

Thesis
UNIVERSITY OF GREENWICH LIBRARY

the
UNIVERSITY
of
GREENWICH

Faculty of Technology

Acknowledgements

My grateful thanks to Dr. Alan Soper and Professor Brian Knight for their valuable advice and helpful criticism throughout the project. They have provided me with both moral and physical support through the years this project has lasted.

I would also like to thank Dr. Sati McKenzie and Dr. Geoff Allwood for helping me select the direction in which this project has progressed and developed. They have provided me with a good insight on how to plan for this project.

I would also like to acknowledge and thank Stone Foundries Ltd. and Keith Preddy for providing the data and knowledge needed to complete this project. Without their help it would have not been passable to test the system fully and gain the knowledge needed within an industrial environment.

Finally I would like to thank my family and friends, and the staff at Greenwich University, for their support and encouragement throughout the last few years. In particular Ellie Andricopoulou, Waseem Naqvi, Elizabeth Bacon, Alex Fedoric, Peter Smith, Philip Robins, and Efthymios Karanassos.

This thesis is dedicated to
Kostandina Simiykaki Panayiotou
and Kostandino Panayiotou.

Abstract

Computer vision is a rapidly growing area. The range of applications is increasing very quickly, robotics, inspection, medicine, physics and document processing are all computer vision applications still in their infancy. All these applications are written with a specific task in mind and do not perform well unless there under a controlled environment. They do not deploy any knowledge to produce a meaningful description of the scene, or indeed aid in the analysis of the image.

The construction of a symbolic description of a scene from a digitised image is a difficult problem. A symbolic interpretation of an image can be viewed as a mapping from the image pixels to an identification of the semantically relevant objects. Before symbolic reasoning can take place image processing and segmentation routines must produce the relevant information. This part of the imaging system inherently introduces many errors. The aim of this project is to reduce the error rate produced by such algorithms and make them adaptable to change in the manufacturing process. Thus a prior knowledge is needed about the image and the objects they contain as well as knowledge about how the image was acquired from the scene (image geometry, quality, object decomposition, lighting conditions etc,). Knowledge on algorithms must also be acquired. Such knowledge is collected by studying the algorithms and deciding in which areas of image analysis they work well in.

In most existing image analysis systems, knowledge of this kind is implicitly embedded into the algorithms employed in the system. Such an approach assumes that all these parameters are invariant. However, in complex applications this may not be the case, so that adjustment must be made from time to time to ensure a satisfactory performance of the system. A system that allows for such adjustments to be made, must comprise the explicit representation of the knowledge utilised in the image analysis procedure.

In addition to the use of a priori knowledge, rules are employed to improve the performance of the image processing and segmentation algorithms. These rules considerably enhance the correctness of the segmentation process.

The most frequently given goal, if not the only one in industrial image analysis is to detect and locate objects of a given type in the image. That is, an image may contain objects of different types, and the goal is to identify parts of the image. The system developed here is driven by these goals, and thus by teaching the system a new object or fault in an object the system may adapt the algorithms to detect these new objects as well compromise for changes in the environment such as a change in lighting conditions. We have called this system the Visual Planner, this is due to the fact that we use techniques based on planning to achieve a given goal.

As the Visual Planner learns the specific domain it is working in, appropriate algorithms are selected to segment the object. This makes the system domain independent, because different algorithms may be selected for different applications and objects under different environmental condition.

Contents

Chapter 1
Introduction

1. Introduction

Many industrial applications require some sort of computer interpreted vision system, for example quality control inspection, object selection from a bin of parts etc. For a computer added vision system to achieve these goals it must posses *image understanding* properties. Image understanding is concerned with the analysis of grey scale images to the effect of detecting and identifying given objects in the image. Objects are characterised in the image by given features of the two-dimensional (2D) grey level distribution. Therefore, to recognise an object, its features must first be extracted from the image.

This is a process that requires a number of transforming steps applied to the original image to enhance its features and *segment* it, the resulting image is split into sub-images. These features are now known as the *signal.* The signal is separated from the remainder of the image known as the *clutter.* Since these transformations are applied to the image, this part of the image analysis procedure is called the *iconic phase.* Iconic phase algorithms are also known as Image Processing and Segmentation algorithms. Such routines are explained well in the books by Gonzalez and Wintz and in C.A. Lindly (Gonzalez 87, Lindly 91).

Any image processing or segmentation procedure has a given error rate which can manifest itself in two directions in an industrial application. These are *false acceptance* or *false rejection.* False acceptance means for example, that a part is falsely accepted as flawless while in reality it has some flaws. False rejection means the reverse, the rejection of a correct item. Both these problems can be costly both in time and money.

Another problem encountered in industrial inspection systems, is their inflexibility. Once the system has been programmed for a specific domain it is unable to adapt to changes easily. For instance if the manufacturing process changes or a new object is added to the production line, the system may not be able to cope with it. Yet another problem encountered by such a system is the available set of image processing and segmentation algorithms itself. The algorithms selected by the programmers may not be the best for the task at hand, or the algorithms' own thresholds may be set incorrectly, dependant on environmental change or another factor not considered.

1.1 Aims of the Project

To overcome these problems within an industrial environment an *Adaptive Image Analysis System* has been proposed and built known as the Visual Planner. The system will be able work within many domains, be adaptable to change in the manufacturing process and within the industrial environment it is working in.

An industrial vision system should at least have a number of desirable properties in order for it to meet the requirements for the inspection task at hand, these are as follows:

1) The system should be flexible so that it can easily adapt to changes in the inspection task or changes in the manufacturing process.

2) The inspection system should produce a low false alarm rate, i.e. the algorithms must be robust. The few false positives the system may produce could, if required, be checked manually.

3) It should be able to process simple images. A simple image is one that has been captured in an industrial environment from a fixed view point or from a number of sensing devices such that three-dimensional (3D) information may be extracted if needed. By process we mean that it should be able to apply the appropriate image analysis procedures to the image.

To achieve these requirements each task will be taken in turn and analysed to obtain an understanding of the problems encountered at each stage. Clearly a number of aims arise from the statements above. These are as follows:

1) The system should be able to select and change image processing and segmentation algorithms dependent on the domain knowledge and scene description. This would allow the system to update itself if the manufacturing process, inspection task, or environmental conditions change.

2) The system should be able to cope with segmentation irregularities, such as cracks in edges, region splitting or joining etc.

3) The system must be flexible and easily updated, i.e.., new algorithms, rules, or scene information can be added as the need for them arises.

1.2 Objectives

The main design and implementation objectives of this project are as follows:

1) To design and implement an image processing system for the acquisition and subsequent image processing of data such as noise removal and edge detection (Lindly 91).

2) To incorporate artificially intelligent (AI) and robust segmentation algorithms for robustness of lines and regions (Nazif 84).

3) To design an AI system that selects appropriate segmentation and image processing algorithms given a scene. The algorithms selected will be different for each new scene or object presented to it.

4) To extract geometric information from segmented data. This will include size of object in the x and y directions.

5) To design and implement a learning algorithm for objects and their subsequent storage to a database.

6) To develop a matching algorithm for objects in a scene to database objects that will find a combination of orientation and viewpoint which give a close approximation to the actual object.

7) To provide a description of the scene in terms of a user defined goal, eg find fault x.

8) To test the performance of the system with respect to the aims set out above, on a real practical problem.

Most of the objectives have been met. A prototype vision system has been developed on a Sun IPX workstation in C using the X11 Window system (Heller 90, Nye 90). The image acquisition programs were written in Turbo C (Borland International 88) utilising the Matrox frame grabber (Matrox electronic systems Ltd. 91).

1.3 Significance of the Work

The summary below shows the significance of the work undertaken. This is based on the review of current vision systems, their limitations and drawbacks. (See chapter 2 and 3).

1) It is the first implementation that uses a planning technique (Wilensky 83) to select and change segmentation algorithms for a particular scenario. This is done through

planning, replanning different actions, and monitoring the system dependant on a user goal.

2) A rule base is used to provide robust segmentation algorithms, i.e. using rules to decide whether to join lines, delete lines, split or join regions.

3) The system can be easily updated by adding new segmentation algorithms, and the knowledge of when they are to be used. This means that the system can be used in other vision domains, and not just for one specific domain as current vision systems are designed and implemented.

4) The system proposed here could can be extended towards the more general area of vision recognition, Allowing recognition systems to adapt to their environment.

1.4 Vision Systems and their Problems

Computer Vision has progressed considerably on many fronts over the past 20 years (Brady 81). There has been a change in the style of research as well as in the substance. However, most issues are poorly understood, from the exact form of representation, through to the detailed understanding of the individual modules, to topics that have so far received little or no attention. A sample of problems follows in the next few paragraphs. It is by no means exhaustive.

Faster algorithms need to be developed for image processing. This is due to the size of the images produced by the frame grabbing equipment. Traversing such images with different filters and edge detectors takes time. However, the rapid development in Very Large Scale Integration (VLSI) technology has further motivated research in to what is referred to as Local Parallel Programming Architectures (Hennessy 84). It is likely that our conception of computation will change as a result of such developments. Vision will be one of the first areas to benefit from such advances.

There seems to be a lack of methodologies and formal methods in all parts of vision research, from image processing all the way to image understanding. Each program developed to date is application specific. That is, each programmer applies the algorithms that best suit the environment they are working in, thus vision systems become uncertain and unreliable.

Once objects have been segmented and geometric structures have been produced, the next step is to match the objects presented to the vision system to objects in a database. This works by trying to find a combination of orientation and viewpoint of the object. Matching can converge efficiently if the object has strong features. When dealing with geometrically well-defined objects analytical techniques may be deployed. However, most objects in the world are not so well defined and thus more pragmatic rules of thumb are used. This gives rise to problems in database structures and searching methods. Results become erroneous and probabilistic which in many vision applications is not sufficient. Efficient database design which incorporates artificial intelligence is needed, such as that being developed by Pearsons (Pearsons 90), where a technique for inducing recognition or discrimination rules are part of the database.

1.5 Recognition

As the goal of all vision systems is to recognise objects and understand scenes, then some prior imagery of them is needed. If the computer vision problem were merely to recognise or classify a scene from one of several candidates, then we could employ special purpose hardware such as the WISARD system (Aleksander 84). This system can be trained in literally a few seconds to discriminate between a small set of different scenes. It's discrimination abilities far outweigh conventional vision systems in terms of speed. The problem with the special hardware approach occurs if discrimination between larger sets of scenes is required. When there are several objects, each in any orientation and at any relative position, combinatorial explosion is rapidly encountered.

The search space within the database has to be cut down. This can by done in a number of ways: 1) Through the transformation of the original object presented to the system, into other compact representations (Lindly 91). 2) Through the deployment of physical knowledge such as perspective (Foley 82). 3) Through the deployment of knowledge about the scene (Learning by example) (Winston 80).

1.6 Levels of Processing

Image processing can be considered to take place at three different levels, as described below.

1.6.1 Low Level Processing

Low level routines are used to clean noise from an image and detect edges or highlight features we are interested in. This area of vision is also known as image processing.

Image processing has been around for many years and is well established in the field of vision. There are many references on the subject such as (Pavlidis 82, Gonzalez 87, Lindly 91).

Low level processing is more of an art then a science, as it largely consists of methods for the extraction of the important intensity changes in an image that are a prior not known. The approach mostly consists of *convolving* images with local operators, *Convolution Filter* (typically 3*3 pixels) (Lindly 91) to estimate the position, contrast, and orientation of the important intensity changes at each pixel position in the image. A typical edge detection convolution filter is shown in Figure 1.1.

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

Edge Operator

10	10	11	12	10	15	10
10	11	10	11	11	14	10
10	90	90	90	90	90	10
11	91	89	90	90	88	11
10	90	90	89	91	90	12
12	10	15	13	11	10	10

Pixel Value

Image

Figure 1.1 Edge operator and image.

However, it does raise the question, which operator works well with the scene and what parameter values to use such as thresholds? This is a major problem in the vision world. Mistakes are often introduced at this fundamental level by the programmer because they have selected an inappropriate algorithm or threshold value. It is therefore a necessity to acquire some form of statistical knowledge from the image, such as signal-to-noise ratio.

1.6.2 Medium Level Processing (Segmentation)

The purpose of this level is to pass onto subsequent algorithms a symbolic representation of the scene rather then the pixel grid which the low level routines output. The exercise we wish to perform here is called *segmentation.* The original image is split into vertices, lines, facets and regions which we hope represent surfaces in the real world from which they came.

Segmentation is an active area of research. The aim of this project is not to produce new segmentation algorithms but to select appropriate algorithms for a given scene. The segmentation algorithms go through a number of mutations to find an efficient segmentation algorithm for a given scene. The algorithms selected and changed are again dependent on the scene presented to it and a specific goal given by the user. (A basic outline of an Adaptive Image Analysis system is shown in Figure 1.2)

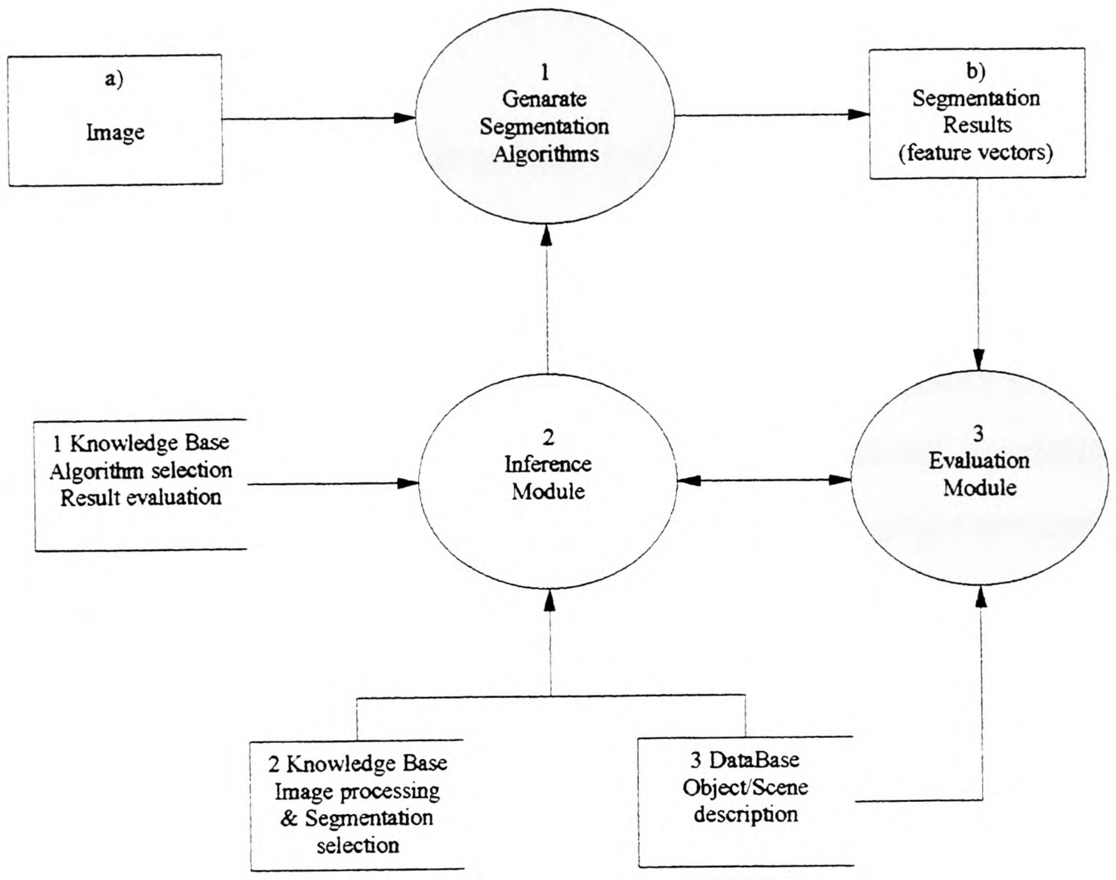

a) Image
1 Genarate Segmentation Algorithms
b) Segmentation Results (feature vectors)
1 Knowledge Base Algorithm selection Result evaluation
2 Inference Module
3 Evaluation Module
2 Knowledge Base Image processing & Segmentation selection
3 DataBase Object/Scene description

Figure 1.2 Adaptive image analysis system.

1.6.3 High Level Processing

It has already been mentioned that to recognise something requires a prior imagery and what is done at this stage is to work backwards. A hypothesis that an object is in the scene is generated and then a search is performed to find the orientation and view point of the actual object.

There are two main functions that this stage carries out. These are: object learning, and scene description. Each object is taught to the system and stored in a database. Each object has to be shown as clearly as possible in order for it to be saved as a perfect model in the database. However, this simple approach has several drawbacks. First, the teaching process tends to be rather time consuming. If a large number of different objects must be inspected, the time consuming teaching must be repeated for each object. Secondly, since every object recognition procedure has a given error rate, the model obtained through teaching may be incorrect.

Once a number of objects have been saved in the database, the next step is to try and describe the scene. This part of the process is a *bottom up analysis* of the scene. First the objects are matched to the database objects (this is not such a simple task as will be shown later). Then their relationship to each other and to the viewer is calculated and given. Finally the system should be able to adapt, that is learn constant relationships, i.e. the sky is always over the horizon. This allows the system to make assumptions about *occluded* objects.

1.7 Knowledge and Vision Systems

If the aim of the vision system is to describe a scene with its own assumptions, then a *knowledge base* is needed to control the process. There are a number of Artificial Intelligence formalisms available for this purpose, for example predicate calculus (logic), production systems (Hayes-Roth 85), and semantic networks (Levesque 86, Nilsson 82).

The advantages of a formal, high level, well behaved knowledge representation system is that it is possible to describe very rich scenes and make useful deductions about them. In principle it should be possible to deduce what some partially visible object might be if there exists a model of what it looks like. In practice this type of common sense reasoning is very hard to build.

Identification can be done in several ways as we have seen from the discussion above. However we have done nothing as yet to teach the system the objects and the scenes we are looking at. A learning process must be put in to the system because identification is the implied purpose for learning. According to Winston (Winston 77) learning can be done at several different levels in the vision system. These are shown in Figure 1.3.

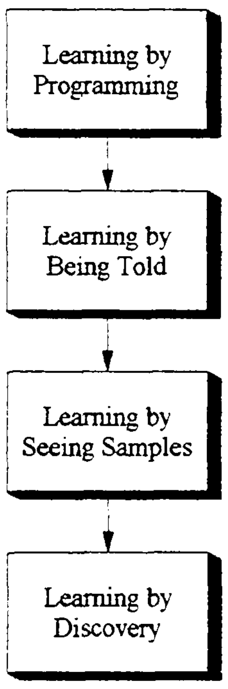


Figure 1.3 Vision systems learning hierarchy.

As we move from the bottom of the hierarchy (learning by being programmed) to the top, the process of learning becomes harder and more ad-hoc. Many structures have to be developed and maintained at each level of learning.

Artificial intelligence can also play a major part in the segmentation of images. There are many possible ways in which an image can be segmented. For example should emphasis be placed on *boundary analysis* or *region growing*, or within segmentation should priority at some particular stage be given to *splitting regions* or *merging regions*?

The idea behind the current work in this project at this stage is to control such possible low level operations within a *production system.*

Clearly there has been an evolutionary process leading from a purely model based image analysis system to rule based systems, with explicit representations of knowledge bases and rule bases.

1.8 Structure of this Thesis

Chapter one discusses the aims and objectives of this thesis and introduces the subject of Vision Systems, and its difficulties. Levels of processing are introduced and some techniques used at each level. Finally Artificial Intelligence techniques are introduced and how they might aid in the vision problem.

Chapter two contains a review of the current approaches in the world of vision. It then narrows to discuss algorithms, their reliability, and their usefulness.

Chapter three explores the range of vision systems. It reviews some actual industrial inspection vision systems, how and why they were implemented and their efficiency. The chapter ends with a statement of the questions addressed in this thesis.

Chapter four introduces the concept design of the vision system proposed. It gives an overview of the knowledge needed to produce a domain independent visual planning system for segmentation algorithm selection.

Chapter five shows what modules the visual planning system proposed here possesses, and how such a system achieves the goals specified.

Chapter six gives an in depth explanation of the design and subsequent implementation of the system. It explains how the objects, algorithms and, knowledge acquisition have been modelled for storage in the subsequent database or knowledge base. This chapter also explains how intelligence has been added for algorithm selection and modification.

Chapter seven shows the validation of the system, how well it works and how updateable it is. The application domain selected is based on x-ray data of metal castings. A description of the knowledge acquisition process is given and performance values depending on the number of faults found within these castings.

Chapter eight critically discusses the current system as defined by the aims of the project, weighing what has been achieved against the imitations and drawbacks.

Chapter nine outlines possible further extensions to this work. It also proposes new questions raised by this research.

Chapter 2
A Review of Vision Systems

2. Introduction

This chapter reviews the present state of the art in vision systems and algorithms associated with such systems. It has become a known fact that any new vision system must incorporate some artificial intelligence. The degree of AI used will depend on the application and the environment in which the vision system is working. AI has played a major role in the image understanding stage.

From the research carried out on vision systems it has become clear that vision systems are uncertain and unreliable. It is therefore an important task to analyse each step of the vision cycle to obtain constraints in order to improve the visual task through knowledge obtained from the domain.

The visual life cycle can be separated for convenience into three main parts, *Low, Intermediate,* and *High* levels of data processing. This chapter also looks at work related to these classes. Low level data processing has many algorithms (see appendix C) that have been available now for many years. These routines however are inherently unreliable due to the underlying pixel structure and digitisation techniques. As data is transformed from one stage to the next it remains unreliable. It is also a difficult process to select a particular set of algorithms that would work well in all circumstances without changes, and thus the algorithm selection is left to the judgement of the programmer. This introduces yet another factor in to the visual life cycle, *uncertainty.*

2.1 Low, Intermediate, and High Level Data Processing

Low, intermediate, and high level data processing perform tasks on different data types within the visual system. Information is passed through this hierarchy of data processing functions until the object or scene of interest may be interpreted.

2.1.1 Low Level or Feature Enhancement Functions

Low level data processing deals with image acquisition, modelling, and feature enhancement functions of images from the real world. This part of the image analysis procedure is called the *iconic phase*. Images go through several transformations to enhance the areas of interest (Signal), such as lines, corners and facets. They are also used to suppress unwanted features such as noise within an image (Clutter).

It must be emphasized at this point that the decision as to what is signal and what is clutter very much depends on the task to be performed. The algorithms selected here may not perform well on different types of images or indeed a changing environment. Therefore, the algorithms selected at this stage are for a specific task, where the domain and environment are static and well known.

2.1.2 Intermediate or Segmentation Functions

Intermediate or Segmentation functions are operations applied on grey scale images or binary representations of grey scale images. The result of a feature extraction function is not an image but a set of numerical values called *feature vectors*. Appropriate feature extraction functions will lead to feature vectors that are characteristics for the object from which the features were extracted. The reason many vision systems use binarisation is that it usually simplifies feature extraction. Many industrial vision systems binaries the image acquired from the beginning; That is, such systems do not allow for the representation and processing of grey scale images. Consequently, vision systems of that type, called *Binary Vision Systems*, cannot perform feature enhancement functions.

Many important computer vision tasks, however, cannot be carried out at all without elaborate enhancement filtering. Other tasks may be feasible on a binary system; however the *robustness* (immunity against changes in the environmental conditions) of purely binary algorithms is typically much poorer then the robustness of the corresponding algorithms performed on grey scale images.

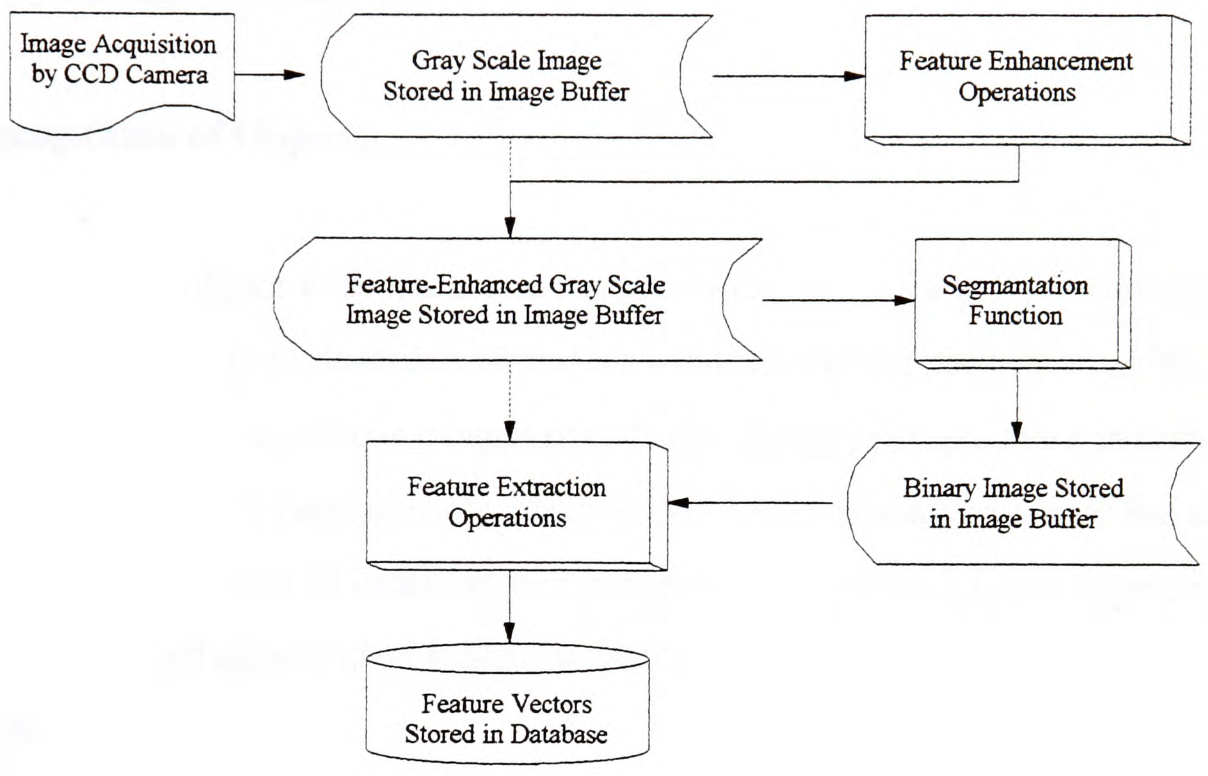


Image Acquisition by CCD Camera
Gray Scale Image Stored in Image Buffer
Feature Enhancement Operations
Feature-Enhanced Gray Scale Image Stored in Image Buffer
Segmantation Function
Feature Extraction Operations
Binary Image Stored in Image Buffer
Feature Vectors Stored in Database

Figure 2.1 Steps of the iconic phase of image analysis.

Generally we can say that the feature extraction tasks become easier, the better the proceeding feature enhancement filters work. It is therefore imperative that these filters work to their maximum capability at all times. Figure 2.1 illustrates the steps of the iconic phase of image analysis.

2.1.3 High Level or Symbolic Phase

The symbolic phase of image analysis requires methodologies for pattern recognition or object classification, symbol manipulation, heuristic search, conflict resolve, etc. These are exactly the methodologies offered by AI. Research has therefore concentrated on developing the symbolic phase with AI techniques. Much research has been carried out in this area leaving the other two stages lacking. The aim of this project is to apply AI techniques in the low and intermediate levels of image processing to aid in symbolic reasoning. If the two lower levels produce better results then it follows that the symbolic phase will also become easier and more able to classify or recognise objects, making the whole system more robust.

2.2 Recognition of Objects

To recognise an object within a scene requires some sort of a prior knowledge of the objects we are trying to describe. Based on what the viewer sees, each of the various parts (segments) of the scene is examined to find a plausible explanation in terms of the coordinate systems for particular objects. It is therefore necessary to *match* the segments of objects to some sort of database description of the object. That is segments of the image are matched against ideal models to give a world model of what the scene actually contains.

Much of the pioneering work in image processing and computer vision has been carried out by Roberts (Roberts 65). His scene consisted of polyhedral blocks such as a cube, a brick, a wedge and a hexagonal prism. The method consisted of edge detection using a derivative technique (Roberts-Cross Operator). These edges were grouped together into lines. Objects were recognised by scaling and rotating the models and matching them with the lines found in the image. These early methods were data-driven (bottom-up) processing and involved little or no goal-driven (top-down) processing strategy. However, this technique was to be used in later methods such as SCORPIO (Lowe 87).

Image analysis involves segmentation by either detecting regions of change or detecting regions of uniformity in grey level scenes. Region based segmentation by region growing, where smaller regions are merged into larger ones if they have similar properties, has been presented by Brice and Fennema (Brice 70) and Yakimovsky (Yakimovsky 73). The opposite to this is to start with large regions and split-and-merge them to form uniform regions. A technique for region based segmentation has been presented by Horowitz and Pavlidis (Horowitz 74). Edge based segmentation is usually easier because it is often easier to model. The discontinuities are detected by various forms of differential operators. Two recent trends in image processing have been rule based segmentation (Nazif 84), and segmentation using texture and fractal dimension (Pentland 84).

Once regions and surfaces have been extracted, the object recognition tasks need to be carried out. Thus given a set of observed features we need to determine which model produced these features and recover the *pose* of the object, i.e. what is the relationship between the object, the model, and the viewing direction. The object can be represented by a world model in which each object is signified by its position and orientation:

$$W = \{(O_i, P_i, \Theta_i)\}_{i=0}^{N_{obj}}$$

Where O_i is an object, P_i is its position, and Θ_i is its orientation. Objects can be represented using either geometric or non-geometric models. In a geometric representation, often simple multi-view models are used when an object only has a small number of stable surfaces. In a non-geometric representation models are represented using there features such as intensity, corner points, relationships of objects, etc.

2.2.1 Matching

The usual objective of matching is to find the optimal location and orientation. If one exists, of objects in the scene based on what the viewer sees or *viewer centred model* (segments), to that of the world model found in the database. This is carried out as part of the larger problem of object recognition.

Feature Spaces and Their Attributes	
Raw Intensity	Matching not tolerant to noise, mainly used as a heuristic match.
Edges	Easy structures to model, give accurate orientation and positioning of objects, less sensitive to noise then intensity information. Edges (Nack 77), Contours (Medioni 84), Surfaces (Pelizarri 89).
Salient Features	Structure harder to model, gives very accurate positioning and orientation of object, can be computationally expensive, can be extended to work on 3D objects, more robust then edge matching. Points of locally maximum curvature on contour lines (Kanal 81) Line intersections (Stockman 82) Closed contours (Price 84) Centres of gravity of closed-boundary regions (Goshtasby 86) Recognition and matching of occluded objects (Griffin 89) Local axes of symmetry (Seitz 89)
Statistical Features	Good for extracting size of object or region, good for rigid transformations, assumption concerning spatial scattering. Moment invariants (Goshtasby 85) Centroid / principal axes (Rosenfeld 82)
Higher-Level Features	Use of relations and other higher-level information, good for inexact and local matching, good for inferring objects. Structural features: graphs of subpattern configurations (Mohr 90) Syntactic features: grammars composed from patterns (Bunke 90) Semantic networks: Scene regions and relations (Faugeras 81)

Table 2.1 Feature spaces used in image matching.

The first step in matching viewer centred images to that of the world model is to select the *feature space* to use. The feature space is the set of features we wish to use from the segmented image to match against the object model. This may be the raw pixel values, i.e. the intensities, edges, contours, surfaces, salient features such as corners, line intersections, points of high curvature, statistical features such as moment invariants or centroids, and higher-level structural and syntactic descriptions. Table 2.1 shows different feature spaces used in image matching. This table briefly describes the attributes of each technique and gives references to works which discuss their use in more detail.

The point is that, by choosing the best feature space it is possible to significantly improve matching. Features can be found on each object independently in the segmentation stage, and this in turn reduces the amount of data to be matched. It is often possible to choose a feature space (dependent on the domain) which will eliminate uncorrected variations that might otherwise make matching unreliable. This is done by extracting only structures of interest. By this we mean finding the pixels in the images that accurately represent significant physical locations in the world as opposed to lighting changes, shadows or changes in reflectivity. This can be achieved by image enhancement routines (Gonzalez 87). Typical enhancement techniques include contrast enhancement, which increases the range of intensity values, image smoothing, which removes high frequency noise, and image sharpening, which highlights edges. These routines force matching to optimize structural similarity and reduce the corresponding data to be matched.

2.2.1.1 The Local-Feature-Focus Method

A method for matching due to Bolles and Cain (Bolles 82) known as the Local-Feature-Focus method is described in some detail here. It is this algorithm, or parts of it that are used within the development of the prototype described in later chapters.

The method is used to locate partially visible two-dimensional industrial objects. The key features of each object such as corners, lines, intensity values, and holes are matched against the computer aided design (CAD) models of the objects. In their implementation there are two types of features, regions and corners. A region is described with respect to its colour (black or white), area, and axis ratio (minor and major axes). A corner is characterised by the size of its angle. Each object will then contain several of these features known as the local features of the object. Each local feature is given a name and its position and orientation with respect to its object. A description of an object also contains a description of its boundary. This boundary is then translated and rotated according to the orientation of the local features, and matched against the CAD database. The Local-Feature-Focus method consists of three processing steps. These are described below:

Step 1: Reading task information

a) A statistical description of the object is given, i.e., the object model, a list of focus features and their nearby features.

b) An Analytical description of the object is given. Here each feature of the object is converted in to clusters of objects, used for matching against The CAD database.

c) A strategy for locating objects is implemented. By this we mean what features are used to match the object, colour, corners, lines etc.

Step 2: Local-Feature Location

a) Locate regional features by finding white regions on a binary image whose properties such as area, minor and major axis are sufficiently close to the modelled values.

b) Finding corners. The system locates corners by moving a jointed pair of chords around the boundaries and comparing the angles between them to find different types of corners.

Step 3: Hypothesis Generation

a) The system hypothesizes objects by recognizing clusters of image features that match clusters of object features, and thus producing a list of such features.

b) Given a list of all possible features, a graph-matching technique is used to locate the largest cluster of mutually consistent features.

The Local-Feature-Method produces efficient results in finding occluded objects and it can reason about features that are not in the field of view. However, such a system needs detailed information on objects and their local features. This means that it can be used for locating parts that can be described in a CAD database, however, it will not work in locating faults on an object unless they can be modelled. Chapter six shows how this method is implemented within this prototype, and the modifications needed to make it more general for finding objects that cannot easily be modelled.

2.2.1.2 Automatic Selection of Matching Algorithm

We have emphasised above that by selecting the best feature space it is possible to reduce the amount of data to be matched. Therefore, the question arises: is it possible to select the best feature space given the domain knowledge? i.e. the environmental attributes of an image. As yet no attempt to answer this question has been made.

Each feature space is selected dependent on the features the segmentation algorithms produce, which in turn are selected for a specific domain. As this project is domain independent then it follows that the segmentation algorithms selected for each task in turn can select the appropriate matching algorithm dependent on their attributes. (See chapter on further extensions).

2.2.1.3 Similarity Measures

After selecting the feature space to use for matching, the second step is selecting how we are to measure the similarity of object within the database and scene. This step is closely related with the selection of matching features since it measures the similarity between these features. Once features have been extracted from an image, the similarity measures evaluate the information in the object model with respect to actual segments.

The criteria used by the similarity measures determine what types of matches are optimal. For example, if grey values are used, instead of features, a similarity measure might be selected to be more noise tolerant since it was not done during feature detection. Table 2.2 gives some similarity matrices used in image matching, along with their advantages and references to works within this field.

Correlation is optimized for exact matches, therefore requiring image processing if too much noise is present. Edge correlation, i.e. correlation of edge images, is a standard approach. Fourier methods (Kuhl 82), such as phase correlation, can be used on raw images when there is frequency-dependent noise. Another possible similarity measure, suggested by Venot. (Venot 84), is based on the number of sign changes in the pointwise subtraction of the two images. This is most advantageous in comparison to classical techniques when the images are dissimilar. This technique allows for background subtraction and noise elimination and modelling.

Similarity Metric	Advantages
Normalized cross-correlation function (Rosenfeld 82)	Accurate for white noise but not toleran of local distortion sharp peak in correlation space difficult to find.
Correlation coefficient (Svedlow 76)	Similar to above but has absolute measure
Statistical correlation and matched filters (Pratt 78)	Good if noise can be modeled
Sum of absolute differences of intensity (Barnea 72)	Efficient computation, good for finding matches with no local distortions
Contour/surface differences (Pelizzari 89)	For structural matching
Number of sign changes in pointwise intensity differences (Venot 89)	Good for dissimilar images
Higher-level matrices: structural matching: tree and graph distances (Mohr 90),	Optimizes match based on features or relations of interest

Table 2.2 Similarity Metrics used in image matching

2.2.1.4 Search Space and Strategy

Because of the large computational costs associated with many matching features and similarity measures, the last step in the design of a matching method is to select the best search strategy. The search space is generally the class of transformations from which we would like to find the optimal transformation to match the images. We can evaluate each transformation candidate using the similarity measures on the preselected features.

Search Strategy	**Advantages and Reference Examples**
Decision Sequencing	Improved efficiency for similar optimisation for rigid transformations (Barnea 72)
Relaxation	Practical approach to find global transformations when local distortions are present, exploits spacial relations between features (Price 85, Shapiro 90)
Dynamic Programming	Good efficiency for finding local transformations when an intrinsic ordering for matching is present (Maire 87, Millios 89)
Hough Transform	For shape matching of rigidly displaced contours by mapping edge space into *dual-parameter* space (Ballard 84, Davis 82)
Linear Programming	For solving systems of linear inequality constraints, used in finding rigid transformations for point matching with polygon-shaped error bounds at each point (Baird 84)
Hierarchical Techniques	Application to improve and speed up many different approaches by guiding search through progressively finer resolutions (Davis 82, Paar 90)
Tree and Graph Matching	Uses tree and graph properties to minimize search, good for inexact matching of higher-level structures (Gmur 90, Sanfeliu 90)

Table 2.3 Search strategies used in Image registration.

Often, the search space is the space of all possible transformations. Examples of common search strategies include hierarchical or multi resolution techniques, decision sequencing, relaxation, generalized Hough transforms, linear programming, tree and graph matching, dynamic programming, and heuristic search. Table 2.3 shows some search strategies used in image matching.

2.3 Knowledge-Based Systems for Vision

Knowledge based techniques will be used (in this thesis) to extract any techniques that lend themselves well to algorithm selection dependent on the domain.

There are currently several ways of representing knowledge commonly used in AI vision systems, each with advantages and disadvantages. Semantic networks and frames have been used in describing binary predicate calculus and graphical representations by Nilsson (Nilsson 82). Semantic networks have been used by Levesque (Levesque 86) and production systems or rule based systems by Hayes-Roth (Hayes-Roth 85). These AI formalisms have their advantages and disadvantages. Logic is formal, with well defined semantics, however, it is not always a very natural tool for scene descriptions while production systems are flexible but lack structure. Semantic networks are intuitively appealing but usually lack a formal semantic basis so that it is not always easy to predict the various side-effects of computations; nevertheless this shortcoming has not mattered for small, experimental systems built to pursue limited objectives.

The advantages of a formal, high level, well behaved knowledge representation system is that it is possible to describe very rich scenes and make useful deductions about them. In principle it should be possible to deduce what some partially visible object might be if there exists a model of what it looks like.

The following discussion gives us some idea of the AI techniques used in the vision world.

2.3.1 Semantic Networks and Frames

Semantic networks have been used and implemented by many AI developers. In this section we describe a formal type of semantic network based on Nilsson's work (Nilsson, 82). This work is important within this project as he presents a way of incorporating the optical properties of objects. These optical properties may be used to select the appropriate segmentation and image processing algorithms and in turn the matching algorithms.

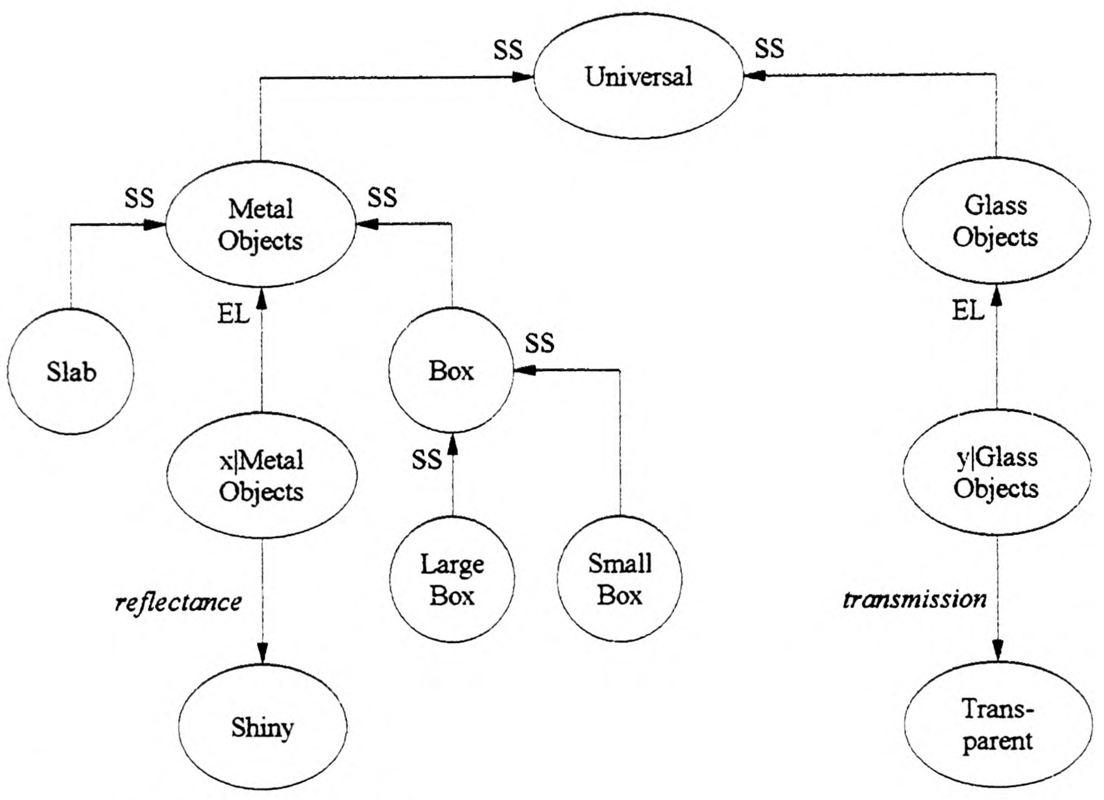


Figure 2.2 The annotated network.

The network is a directed graph of nodes and arcs in which each arc connects exactly two nodes. Nodes represent constants or variables, while arcs show the predicates, EL (elements of), SS (subset of) or else a unary function. Figure 2.2 gives an example of such a network.

When trying to identify objects it will be important to know something about the optical properties of these objects, for example metal has a shiny reflectance, or that glass is transparent etc. The net can easily be annotated to show these, although the provision of corresponding low-level processing techniques are not considered at this stage.

2.3.2 Generic Object Descriptions

The frame concept has been adopted for the ACRONYM model based vision system (Bruods 79). There is also extensive use of generalised cones, and production systems. The use of geometric reasoning to handle spatial relationships is also important. In the present discussion we are interested in the representation of objects in a hierarchical fashion starting with a coarse description and working down to finer detail. The objects are frames in an object graph. (Brooks 81, Brooks 83).

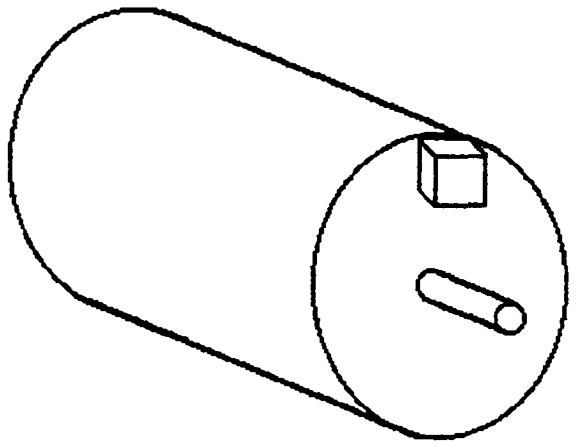


Figure 2.3 A generalised cone motor with shaft and one flange.

The generalised cone is defined by a spine, a cross section and a sweeping rule. A simple electric motor, with one flange, is shown in Figure 2.3.

We could define a specific electric motor as a frame.

Node: Electric-Motor

Class:	SIMPLE-CONE
Spine:	STRAIGHT_LENGTH_8
Sweeping-Rule:	CONSTANT
Cross-Section:	CIRCLE_RADIUS_2.5.

2.3.3 Winston's Arch

Another example of a semantic network that is often quoted concern the structural description of an arch. The importance of the work is that the semantic network was learnt by a program devised by Winston (Winston 70, Winston 75, Winston 84). The system was given an example of an arch, and then several more examples and counter-examples (near-misses). From each image it would learn that an arch in the network must exist, must not exist, or other possibilities. It would learn that an arch consists of a brick or wedge supported by two bricks that must not touch. The work shows how difficult it is to learn explicit knowledge.

In the long term, the best chance of progress in the area of learning may come from research into Connectionist Machines (Neural Networks), which consist of a richly interconnected network of tiny, McCulloch Pitts type, processors. During their learning phase, the strength of interconnections is determined by reference to a teaching set (Rumelhart, 86).

2.3.4 Syntactic Descriptions

An alternate approach to semantic networks is to describe the syntax of a scene in terms of small primitive elements that can be combined according to rules specified in a grammar. Consider the simple example of the staircase structure in Figure 2.4. The primitives could be lines of unit length in the horizontal and vertical directions, denoted by *h* and *v* respectively. Appropriate parsing techniques, perhaps using some production system rules might generate *hhvhvhvh* as a description of the staircase, low level processing, having identified the primitives in the image.

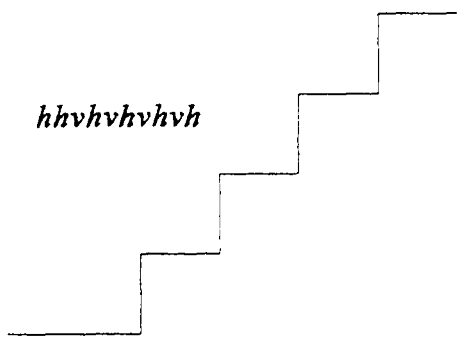


Figure 2.4 A staircase structure.

An attempt to model 3D objects in syntax was undertaken by Duda and Hart (Duda 73), however, this proved too complex to implement. Large sets of information were developed, many with similar features that made the syntax approach ambiguous. For limited domains such as the classification of fingerprints or chromosomes, syntax based methods do work well (Gonjale 78, Miclet 86).

2.3.5 Rule Based Systems

Production systems are widely employed in vision systems and AI. They can be employed to process syntactic definitions of languages (Aho 86), or as the basis of an expert system (Jackson 86). The basic components of a production system are: a set of rules, a database, and an interpreter (sometimes called a control system).

Object recognition tasks consist fundamentally of searching and matching features, this task requires a priori knowledge about such entities as geometry, size, contrast, colour and texture. This knowledge can be represented either in procedural or declarative form.

The procedural knowledge is most effective in this project in representing how certain tasks may be carried out, such as image processing or segmentation.

A rule-based system (RBS) has the following properties (Hayes-Roth 85):

1) Incorporates practical knowledge in a production rule system.
2) Knowledge is incremental, skill is proportional to the knowledge base.
3) Solves a wide range of complex problems by choosing and combining results in an appropriate fashion.
4) A scheduler adaptively determines the execution sequence.
5) Trace facility and natural language interface.

An RBS is therefore a module based knowledge system, with the knowledge being in the following forms:

1) Specific inference from observations,
2) Abstraction, generalisation and categorisation of given data,
3) Necessary and sufficient conditions for achieving a given goal,
4) Likeliest position to look for relevant information,
5) Arbitration.

A production rule has the form:

IF (condition) **THEN** (action)

Where the condition is called the antecedent and the action the consequent. The interpretation of the rule is, if the consequent defines an action then the system executes the specified action, if the consequent defines a conclusion, then the system infers a conclusion.

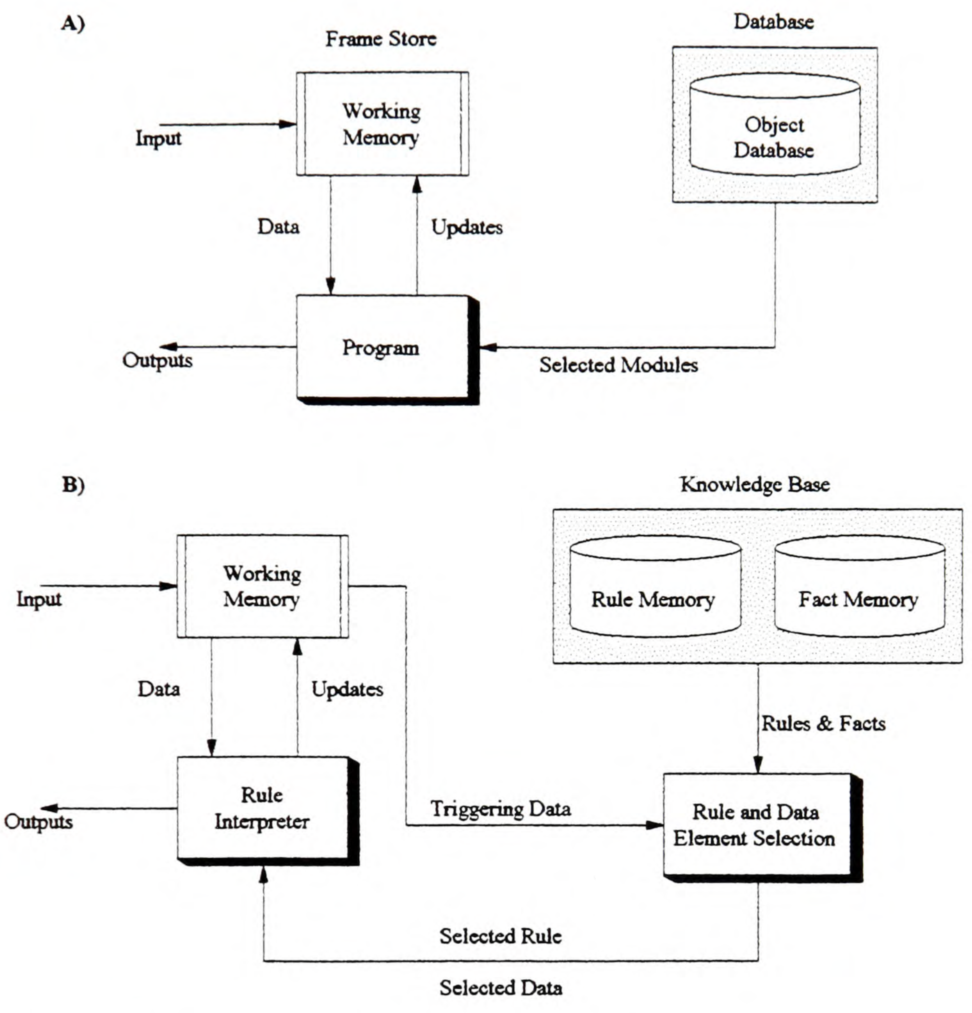


Figure 2.5a Architecture of a simple image processing program.
Figure 2.5b Architecture of a rule based system.

Some examples of rules are given below. These rules have been applied by Nazif and Levine (Nazif, 84) in low level image processing. This work has been extended, and included within this project. Typical rules in the system are:

RULE (801)

IF: (1) There is a Low DIFFERENCE in REGION 1

(2) There is a Low DIFFERENCE in REGION 2

THEN: (1) MERGE THE TWO REGIONS.

RULE: (901)	
IF:	(1) The REGION HISTOGRAM IS BIMODAL
THEN:	(1) SPLIT THE REGION ACCORDING TO THE HISTOGRAM.

A traditional image processing program differs from a RBS in that it lacks a rule and selection subsystem. However, it often has a knowledge database containing model information (not rules). Instead of a rule interpreter, there is a user defined program consisting of one or more algorithms with control and data flow built into the system. These programs lack the flexibility of a RBS. For a modification of its task execution, a new program may need to be written because the control and data flow may be changed by a new item of data. A RBS is more flexible and thus better suited for implementing an adaptive image processing system. The only change needed if more information or knowledge is entered, is the addition of a new rule. Figure 2.5a and 2.5b show the difference between a traditional image processing architecture and a rule based architecture.

Advantages and Disadvantages of Rule Based Systems	
Main advantages of a RBS.	
1)	It presents problem-solving methods in a way which is suitable for computers,
2)	It is modular,
3)	It is incremental,
4)	It is explainable,
5)	It provides a framework for conceptualising computation,
6)	It provides parallel methods for problem solving, and
7)	It makes distinction between analytic and imperative know-how.
Main disadvantages of a RBS.	
1)	There is no analytical foundation for deciding which problems are solvable,
2)	There is no methodology or technique to test consistency and completeness of rule set,
3)	There is no theory of knowledge organisation,
4)	There are no good rule compilers or specialised hardware and,
5)	There is no easy way to integrate RBS into data processing.

Table 2.4 Advantages and disadvantages of a RBS.

There are several advantages and disadvantages to using a RBS. These must be considered when developing a vision system depending on the application. These are shown in Table 2.4.

2.4 Adaptive Vision Systems

A group at the University of Massachusetts (Kohl 87) have been arguing for some time that one must be goal directed in order to do low-level image processing and/or segmentation. They argue that the failure of general segmentation techniques can be traced to the following:

1) The image is too complex because of the physical situation from which the image was derived and/or the nature of the scene, or
2) There is a problem in evaluating different regions/segments.

Images are complex, but some image acquisition process can be modelled, and therefore, we can predict the variability of the acquisition process as shown by Krotkov (Krotkov, 87). The authors suggest that there are no good segmentation evaluation functions. It is known that the segmentation process is not unique given any number of parameters (Anderson 87).

2.4.1 Anderson's Segmentation System

Anderson (Anderson 87) has considered a modular, context independent approach to the problem of 2D segmentation. The modules are edge and region formation modules as shown in Figure 2.6.

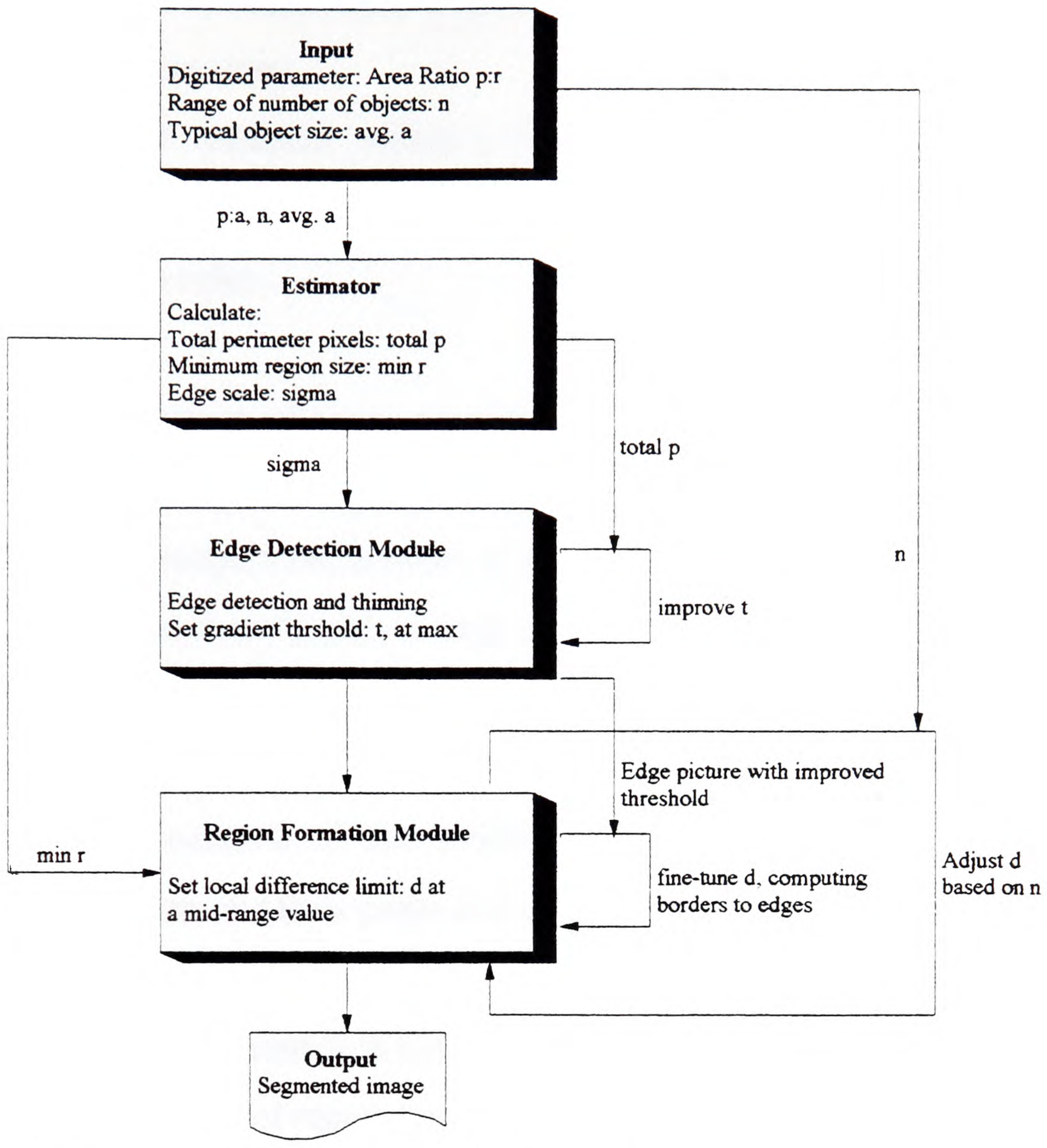

Input
Digitized parameter: Area Ratio p:r
Range of number of objects: n
Typical object size: avg. a
p:a, n, avg. a
Estimator
Calculate:
Total perimeter pixels: total p
Minimum region size: min r
Edge scale: sigma
sigma
total p
Edge Detection Module
Edge detection and thinning
Set gradient thrshold: t, at max
improve t
n
Edge picture with improved threshold
Region Formation Module
Set local difference limit: d at a mid-range value
min r
fine-tune d, computing borders to edges
Adjust d based on n
Output
Segmented image

Figure 2.6 2D edge and region segmentation system.

It is a well known fact that one can get very different segmentation from a picture by just changing the parameters. The most important results of his work so far are in the authors' view the following:

1) Definitions of the task/goal and parameters for each module in terms of general, geometric goals rather than with respect to semantic information,
2) An extensive analysis was done on the relationship between the parameters and the false detection errors and the dismissal errors of a true object boundary,

3) The detection of boundaries is an independent process between edge and region formation processes,
4) The idea of feedback within a module and the interdependency between modules, implies multiple outputs and therefore the need for fusion, i.e. combination rules.

Items' 1 and 2 are issues of local models while items' 3 and 4 are aspects of global models. The most popular approach to global models in the context of image segmentation is the cooperative network or relaxation approach (Rosenfeld 76). Another such model is the Random Markov Fields model used by several researchers (German 84, Derin 87, Cross 83). The principle of this model is that the effects of members of the field upon each other are limited to local interaction as defined by the neighbourhood. This is also the weakness of this model because it assumes an a priori spatial arrangement of objects and their projection on the image. This assumption is too strong, and applicable only in a few, highly controlled experiments. The work of Anderson shows that for image segmentation the global models should represent topological and integral (size and number of regions/edges) properties that are positional invariant rather than neighbourhood dependent.

Experiments undertaken in this research (see chapter 7 and Appendix B) have shown that the performance of the image processing and segmentation algorithms cannot be improved beyond a limited domain by only adjusting their parameters. The reason for this is that many basic assumptions made in the design of such a system are violated when different scenarios are encountered.

2.4.2 The Blackboard Processing System

The Blackboard concept for vision systems was first used in the VISIONS program (Hanson 78) developed for interpretation outdoor scenes. The VISIONS system consists of knowledge sources (KSs) which are all independent modules. The blackboard is a shared memory structure to which all the KSs have access. When a KS is activated it

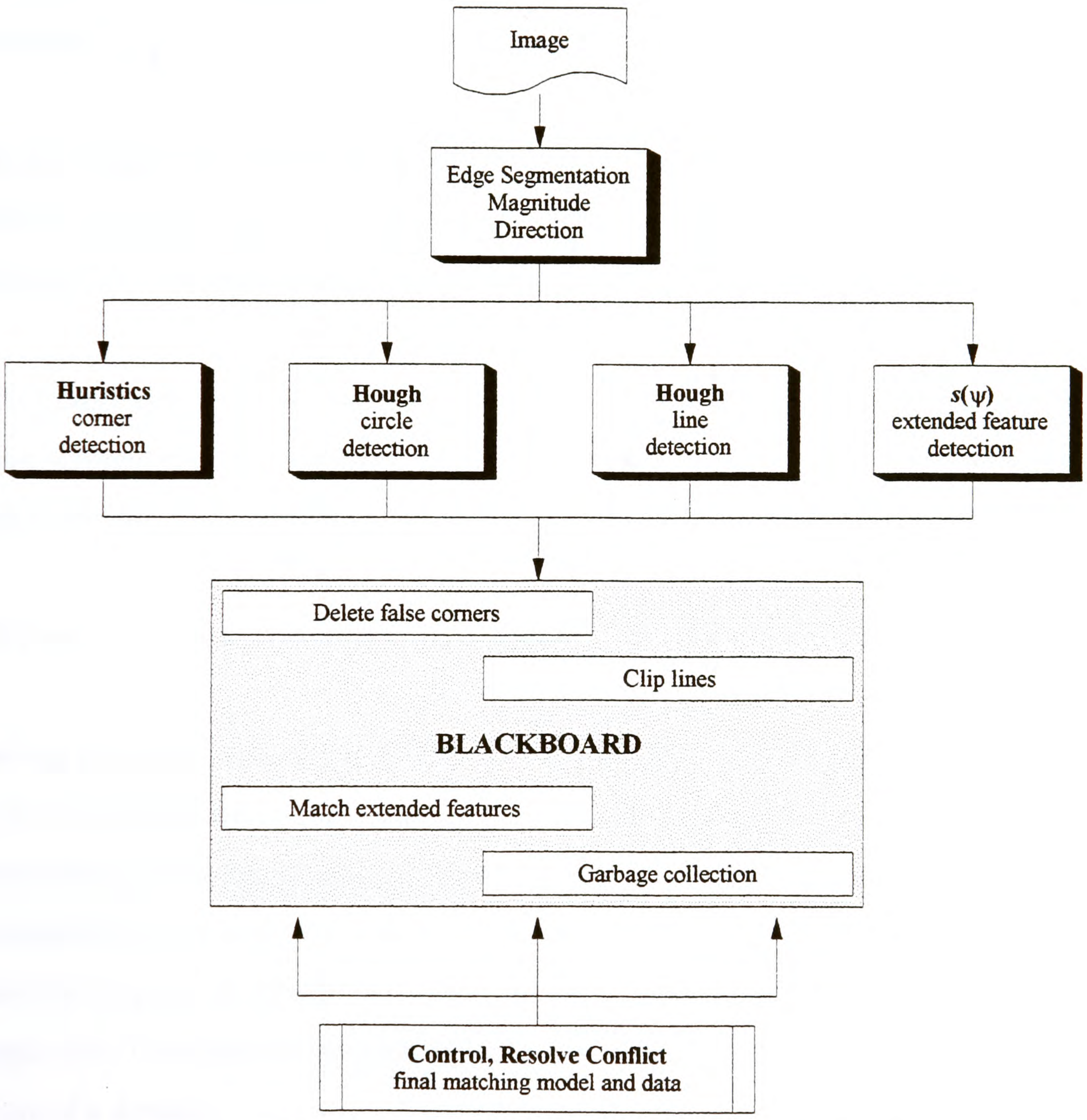


Image
Edge Segmentation Magnitude Direction
Huristics corner detection
Hough circle detection
Hough line detection
s(ψ) extended feature detection
Delete false corners
Clip lines
BLACKBOARD
Match extended features
Garbage collection
Control, Resolve Conflict final matching model and data

Figure 2.7 A schematic diagram of the blackboard processing system.

uses the knowledge residing on the blackboard to create a new hypothesis and write the results back on the blackboard, or it modifies existing knowledge. Furthermore, the KSs operate asynchronously, once activated they continue to operate until the task is finished. The overall goal of the system is to use the blackboard to generate a single hypothesis.

A simple blackboard (Nagao 82) system is shown in Figure 2.7. The line and circle detection are via the use of the Hough transform (Hough 62). These three features can be considered as being generic. Other features are also extracted using s-graph ($s(\psi)$) (Perhins 78).

For this system, two generic features are extracted using the Hough transform, lines and circles (or arcs). The circle detector used is essentially that described by Kimme (Kimme 75). The results are plotted directly into the image space (Davies 86).

This system cannot be termed truly adaptive as it does not react to changes directly in the environment or manufacturing process. However, it does hypothesise about objects and learns the best possible solution to be used in future recognition procedures.

2.5 Future Trends

Perhaps the most fundamental differences between present computer vision, compared to the old model based systems, stem from the current concentration on topics aimed at identifying and understanding the human visual system and reasoning. There is still a considerable amount of work oriented towards applications, but it is also increasingly based on detailed and precise analyses of specific visual abilities (Marr 82, Gregory 71, Gupta 89). The focus of research is more defined in terms of visual abilities than in terms of a domain.

One obvious effect has been a sharp decline in the construction of entire vision systems. Most AI vision researchers have abandoned the idea that visual perception can profitably be studied in the context of a commitment to a particular program or machine architecture. There is for example, no reason to believe that one algorithm is always better than another.

We shall expect to see segmentation algorithms that pass their first approximations to some knowledge based part of the system, which will in turn produce feedback based on domain knowledge and allow intelligent updates to the segmentation.

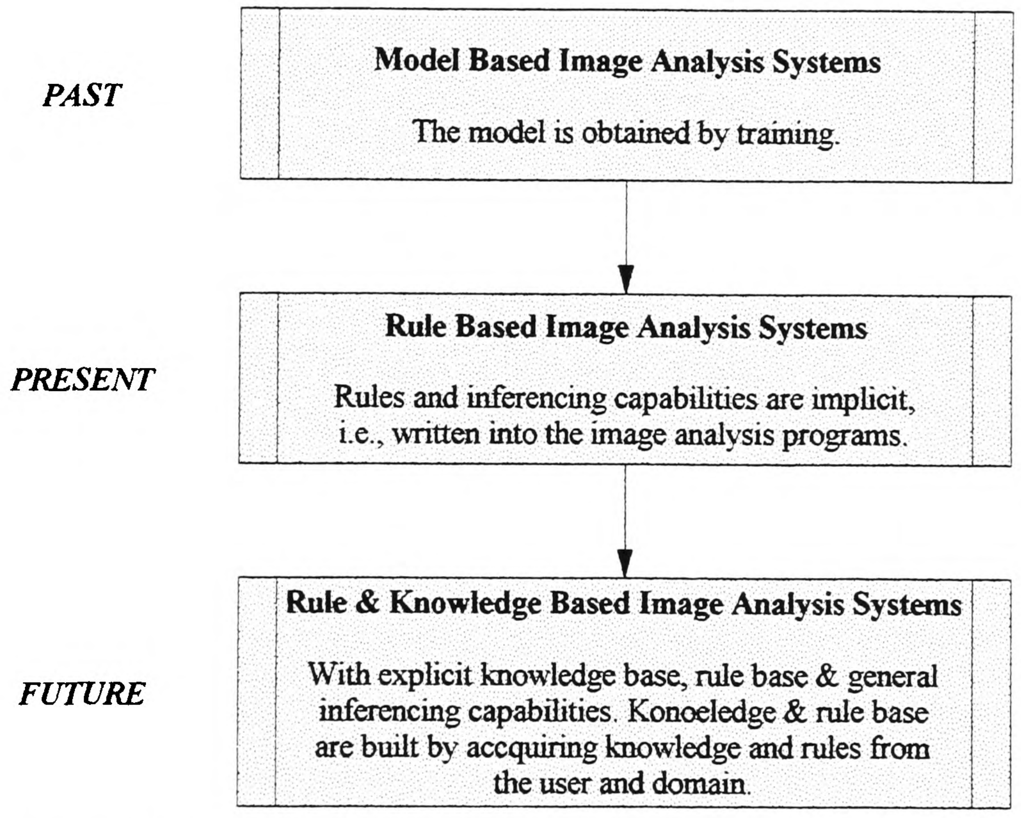

PAST
Model Based Image Analysis Systems
The model is obtained by training.
PRESENT
Rule Based Image Analysis Systems
Rules and inferencing capabilities are implicit, i.e., written into the image analysis programs.
FUTURE
Rule & Knowledge Based Image Analysis Systems
With explicit knowledge base, rule base & general inferencing capabilities. Konoeledge & rule base are built by accquiring knowledge and rules from the user and domain.

Figure 2.8 Expert vision systems - The evolutionary approach.

In addition to the use of a priori knowledge, rules may be employed to improve the performance of the system. In particular, decision rules can considerably enhance the correctness of the recognition process. Figure 2.8 indicates the steps in the evolutionary process leading from a purely model based image analysis system to rule based systems with explicit representation of knowledge bases and rule bases. All the models we have seen have been hand generated, but in the future, automatic conversions from CAD system output will be common. Hence there will be a link into automatic parts handling and inspection. The future, commercially, lies in these applications.

Perhaps the biggest challenge is for systems to be able to make intelligent deductions and observations about situations they have never seen before, based on what they have previously learnt.

Chapter 3
Industrial Inspection Vision Systems

3. Introduction

This chapter describes some systems that are available or under production for industrial inspection. It looks at the trends of industrial vision systems within the market to see why such vision systems are not yet widely acceptable. This chapter also helps to identify current industrial vision systems along with their strengths and shortcomings. The main object of this chapter is to see whether the proposed system has been implemented already by another researcher either in its overall aim or in its design.

This chapter concludes by summarising the faults in current industrial inspection systems and a statement of the questions addressed in this project.

3.1 Industrial Inspection Systems

The probably most frequently given goal (if not the only one) of industrial image analysis is to detect and locate objects of given types in the image. That is, an image may contain objects of different types, and the goal is to identify parts of the image, if existing, which correspond to such objects or pieces of objects.

An industrial vision system must be able to carry out a few "simple" tasks:

1) It must process simple images. A simple image is one captured under controlled illumination from a fixed view point such that three-dimensional objects in some instances can be regarded as being two-dimensional.

2) The system must operate in real-time (this implies use of specialised hardware). It is observed that pipeline processors provide a cost-effective solution for

speeding up the computation, although in many circumstances multiple pipelines may be required (Hennessy 84).

3) The inspection system must produce a low false alarm rate, i.e. the algorithm must be robust. The few false positives the system may produce could, if required, be checked manually.

4) The system must be flexible so that it can adapt easily to changes in the inspection task, or change in the manufacturing process.

An industrial inspection system should fulfill all the above requirements. However, this is rarely the case. The usual approach employed in industrial inspection systems is to give the system a model of the object it is to detect by teaching it. First the systems are shown the bare object, then slowly any other features the object may have associated with it, e.g. chips on a PCB (Hara 83). The system is initially shown the unpopulated board and then the board with the chips on, so that it can learn which chips to look for and on which position of the board. As a result of the teaching process, the system had a model of a correct board, obtained solely by utilising the database and matching capabilities of the image analysis system.

This simple approach has several serious drawbacks, it is therefore slowly being abandoned. First, the teaching process tends to be rather time consuming. If a large number of different objects must be inspected, the time consuming teaching must be repeated for each object. Secondly, since every object recognition procedure has a given error rate, the model obtained through teaching may be incorrect.

There is a more reliable and much faster source from which the model of the object in a correct form can be obtained. The object with all it dimensions can be placed in a CAD database. This database contains all the information about the object and the part placements in a totally correct form. In other words, there exists a priori knowledge in the database that can be utilized to facilitate and improve the inspection task.

Where a prior knowledge is not as readily available as in the example above, the a prior knowledge used could take the form of general knowledge about a given class of objects. This information should consist of how the image was acquired from the scene (lighting, imaging geometry and quality, etc.) (Rosenfeld 86, Amir 90). In most existing image analysis systems, knowledge of this kind is implicitly "embedded" into the algorithms employed in the system. Such an approach assumes that all these parameters are invariant. However, in complex applications this may not be the case, so that adjustments must be made from time to time to ensure a satisfactory performance of the system. A system that allows for such adjustments to be made, e.g. interactively, must comprise the explicit representation of the a prior knowledge utilized in the image analysis procedure. Consequently, the system must encompass means of knowledge representation. Since the a prior knowledge represented in the system must first be acquired, the system must also have means (software modules) for knowledge acquisition, e.g. by conducting a dialogue with the human operator.

Any object recognition procedure has a given error rate which can manifest itself in two directions, false acceptance, or false rejection. False acceptance means for example that a part is falsely accepted as flawless while in reality it has some flaws. False rejection means the reverse, the rejection of a correct item. False rejection is also called a false alarm.

There is a reciprocity between the false acceptance rate and the false rejection rate. By moving up the acceptance threshold, the probability of false acceptance is lowered but, by the same token, the probability of a false rejection is increased, and vice versa. It is usually a matter of priorities as to which end of the spectrum one wants a vision system to operate, e.g. in the aerospace industry where quality standards are very high the priority is usually on a minimal false acceptance. In contrast, in the manufacturing of consumer goods one usually wants to minimise the false alarm rate. Alarms call for the intervention of a human operator, often halting the production process. This can be rather costly and, thus, a too high false alarm rate may not be tolerable.

This dilemma can be avoided or at least mitigated if the recognition algorithms are employed that are so robust that the probability for a false recognition is sufficiently small in the first place. The robustness of the recognition algorithms may be increased by making the image enhancement and segmentation algorithms more adaptive and efficient. This can be done be selecting the appropriate algorithms and thresholds for a given domain.

3.2 The Industrial Vision Market

One reasons for the slow penetration of the use of machine vision in industrial manufacturing has been the very strict requirement on both the hardware and the software for the task. However the leading industrial vision systems supplier, Computer Recognition Systems Ltd. (CRS) strongly believes that industry is teaming with potential industrial vision applications and that CRS is set for a period of sustained high growth. In the context of the machine vision market in the current climate, these comments appear to be far fetched. Industrial companies have not adopted vision technology on a large scale and very few suppliers have made any money in the period 1990-1993 (DTI 93).

Experts on the vision market argue that the reason so few suppliers have managed to make a profit is not due to the failure of vision systems as a technology, nor is it due to a lack of interest among users. Rather it is because too many suppliers jumped into the market on the basis of there (relatively thin) understanding of the relevant computing and optical technologies. They failed to understand that the vision system market is not one diversified market, but hundreds of different niche markets. Each market had its own domain with its own requirements on the vision tasks that need to be performed. They also needed software that could be used again if the manufacturing process changed.

The success of CRS is due to four factors: i) it does not take on jobs where there is no clear profit, ii) it markets not to industry as a whole but to niches where no one else is competing, iii) it has developed a clear technology leadership over some of its rivals, iv) it will only design systems using existing software modules which it already has available. The last of these factors is very important, as it selects algorithms that are known to work well in certain domains.

The failure of vision systems to live up to expectations of market researchers, was not just due to management incompetence among suppliers and developers. Technology, such as optical character recognition and parts inspection, had been improving rapidly throughout the 1980s but failed to reach levels of reliability and reusability for users, for the amount they were willing too pay. New research suggests that the market for vision systems will grow dramatically when users appreciate the improvements that have been achieved. It is also easy to see that unlike many computer-based applications, some of the financial payback of vision systems can quickly be ascertained. In inspection systems, for example, it is easy to produce figures showing whether or not quality has improved. Other benefits, such as improved company profile, are obviously more difficult to cost-justify.

3.3 Working Industrial Inspection Systems

The following section is a brief account of some industrial inspection systems in working environments or available on the market. However, the companies contacted did not want to disclose the full workings of their systems due to the nature of their work or for fear of industrial competition. We would like to thank and acknowledge the companies involved for their time and information provided by them.

3.3.1 Visual Control by Image Processing

Image Processing (IP) techniques, which permits the conversion of scan pattern signals from video and thermal imaging sensors into useful input for a diverse range of control applications, are being developed by Ferranti Instrumentations Ltd. Bracknall, UK. Image recognition is being used increasingly in situations where observation must be conducted under difficult conditions or when tedious surveillance can induce boredom or fatigue in human operators. The techniques can also be employed where decision making is mechanized or integrated with artificial intelligence systems. Applications range from automated instrumentation to robotics and process control systems.

The Ferranti algorithms are based on the latest available IP hardware, using multi-purpose digital frame stores with a range of optional configurations for micro computer or main frame interfaces. Functions include image stabilization, freeze frame and auto-cuing. An engineering service is available for the adaptation of control algorithms and hardware for both slow speed and real time applications.

The algorithms available are both efficient and fast, as special purpose hardware is used. However, there are only a certain amount of algorithms available, which cannot be adjusted. The system itself displays no artificial intelligence, but can be connected to such a system for recognition and event and action processing.

3.3.2 The Intelledex 386HR

A vision system produced by Intelledex Inc., is a Personal Computer (PC) system with user selectional levels of both resolution and grey scale. Higher resolution and finer grey scale definitions are used for more complex inspection tasks, while higher processing speeds are realized with applications requiring less resolution and grey levels. This capability also enables applications with varying inspection requirements to be accommodated without operator intervention or the need for multiple systems. Company officials believe the Intelledex 386HR is suited for inspection and robot guidance tasks in a wide range of industries including electronics, food processing, and pharmaceuticals.

The 386HR includes a library of propriety vision algorithms that is the result of over six years of development by Intelledex. The Pattern Transform functions can identify objects even in situations with widely varying factory lighting. Ledges (edge learning program) uses light-to-dark transitions to find edge position, magnitude, and direction of all edges between two user defined points. This function is especially useful for applications where the edges of objects or faults must be located. Other standard functions include windowing, mathematical morphology, connectivity analysis, Roberts and Laplace operators , and linear and nonlinear array processing.

Intelledex Inc. is a manufacturer of intelligent robot and vision products. However, to-date all their applications have been domain specific, with their algorithms selected for a specified application. The Intelledex 386HR has intelligence built in to it, but this resides within the algorithms themselves, which are implicitly coded into the algorithms.

3.3.3 Computer Vision and Safety Shield for Robots

Perceptive Systems Ltd. has developed a computer vision system with particular applications to robot safety. Vision analysis within the system, called 3DIS, is based on aspects of human vision. Just as the eye and brain respond to visual changes or movement, so 3DIS can respond, automatically analysing changes that occur in its field of vision.

In its production form, 3DIS will give the user the ability to create three-dimensional, "intelligent" sensor spaces within the field of view of the system's attached video cameras. Depending on the particular application, any movement of a subject into the sensor will control the actions of a robot, for example sound an alarm or take a photograph.

This system has no bearing on industrial inspection systems, however, it uses many well-known algorithms in optical flow (Sabbarao 88).

3.4 Industrial Applications

Most of the systems reviewed in this section for industrial inspection involve a model-based approach. However, the models are often not geometrical ones. The model-based approach needs to address the following questions:

1. What features need to be extracted to describe an object?
2. How do we match the located features with the models?

3. How do we represent and group features to describe classes of objects?

To answer these questions we can look more closely at published work within the field of industrial inspection. The review is not exhaustive but it will give the reader an idea of how industrial inspection systems are being developed.

3.4.1 Metal Surface Inspection

Several systems have been described in the literature (Sardis 79, Mundy 80, Suresh 83). In steel making a continuous casting method is employed. This involves the molten steel being continually poured into a water-cooled mould. As the steel solidifies, it is drawn out in a ribbon along the roll table. The ribbons are cut into sections of predetermined length to form slabs. However, these slabs have imperfections that need to be analysed before further processing of the slabs can be carried out. Typical types of faults found in a slab are shown in Figure 3.1.

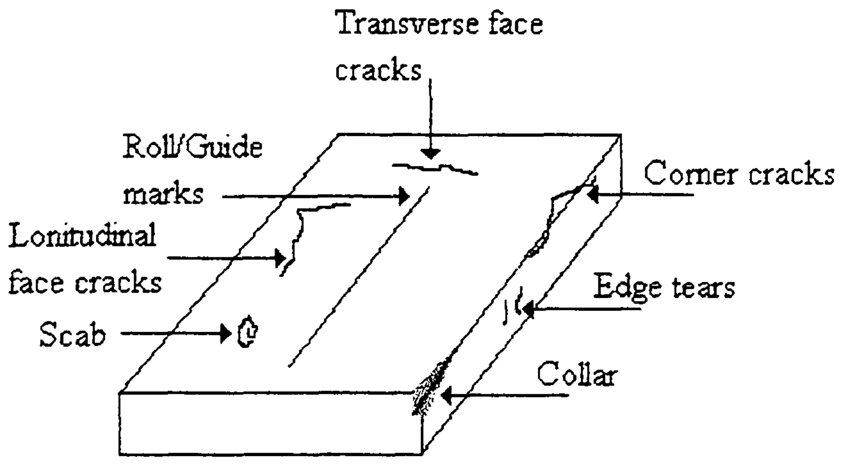


Figure 3.1 Faults found on a steel slab.

An automated system will have to locate the various defects, size and width to determine the slab's condition, whether it is satisfactory for further processing or requires further conditioning, or has to be rejected. Even if the slabs have few imperfections, they tend to have many features that must be inspected, such as texture, intainsity, etc.

Suresh describes a system built by Honeywell, (Suresh 83) which works with images in the visible band. Images are captured using a camera with an infrared blocking filter. Two cameras are used. The data camera views orthogonally to the slab motion, the picture is generated using a CCD array.

Once the data has been captured, the images are segmented and the features extracted using an array processor. The incoming slab image data is first processed by an edge enhancement operator. The Roberts gradient operator (Pratt 78) was chosen for its performance in this domain relative to its low computational overload.

The edge enhanced image is then binarised using a threshold to segment out the object edges. However, they claim that the selection of a proper threshold is crucial in order to reduce the clutter while retaining the primary objects of interest. They decided by examining the slab's imagery, which could be divided into three homogeneous zones on the imagery statistics. The first zone consists of a narrow, bright strip at the edge of the slab. The second zone consists of all vertical striations. Each zone required its own threshold, from the image statistics it was found that zones one and two were constant, however for the third zone it was necessary to use an adaptive threshold algorithm as shown below:

$$SS(i) = (1 - a)SS(i-1) + a \left[\sum_{j \in \text{zone } 3} E_j \right]$$

For each scan line j, they compute a smoothed sum of the edge enhanced image along the scan line. If i denotes the pixel count on the scan line for the third zone this gives the smoothed sum SS(i). E_j denotes the edge enhanced image intensity at pixel position j on the scan line i. Also a is a suitable constant such that $0<a<1$. The threshold Ta (i) applied to the scan line is then determined as:

$$Ta(i) = CK_1\ SS(i)$$

Where coefficient K_1 is a constant and C is a coefficient that varies. Most of the faults are basically fine-line features, the segmentation process consists of edge-enhancement and thersholding the image.

During the segmentation process a set of features is computed for each object detected. These features are then labelled using a semantic structure of longitudinal and transverse edges. The structures are passed to a rule base where object size thresholding takes place. Small objects are rejected as noise, while reasonably large objects are presumed to be imperfections. The imperfections are then classified in a component classification structure.

It is clear from the discussion above that this system has several interesting components such as the adaptive thresholding technique, its use of a rule base to eliminate noise and finally its use of statistical as well as syntactic/semantic classification techniques. However, the system has embedded knowledge of its domain. There is no way of adding new imperfections to the list of current imperfections, except by reprogramming the system. Finally the results obtained are extracted under very strict and controlled conditions, which is very hard to achieve in an industrial environment.

3.4.2 Timber Defects

The problem in processing lumber images is the great variability amongst them (Conners 83), even within the same species, no two knots are the same and each type of wood has a unique grain pattern. The defects in lumber arise from either biological or manufacturing defects. There are a variety of techniques for detecting these defects. Laser scanners have been used to detect splits and knots of varying sizes (King 78, Mathews 76). The tonal property of the lumber can be used to detect certain types of defects: for example clear wood is lighter than knots, holes or cracks. These techniques need to be combined with pattern recognition techniques, i.e. holes and knots are circular while checks and splits are narrow and long.

It is important to note at this stage, that the defects have been classified with their respective attributes. This means that the appropriate algorithms are implemented to detect these attributes. This form of knowledge acquisition and matching to algorithms is done in all vision systems. However, this knowledge is embedded into the system, which means that if a fault that has not been considered arises the system would fail to recognise it. Later we present a way of detecting an unknown fault by comparing it with perfect object or image attributes if it cannot be classified. The system would then undergo a learning process to classify and extract the object attributes and thus select an appropriate algorithm for detecting it.

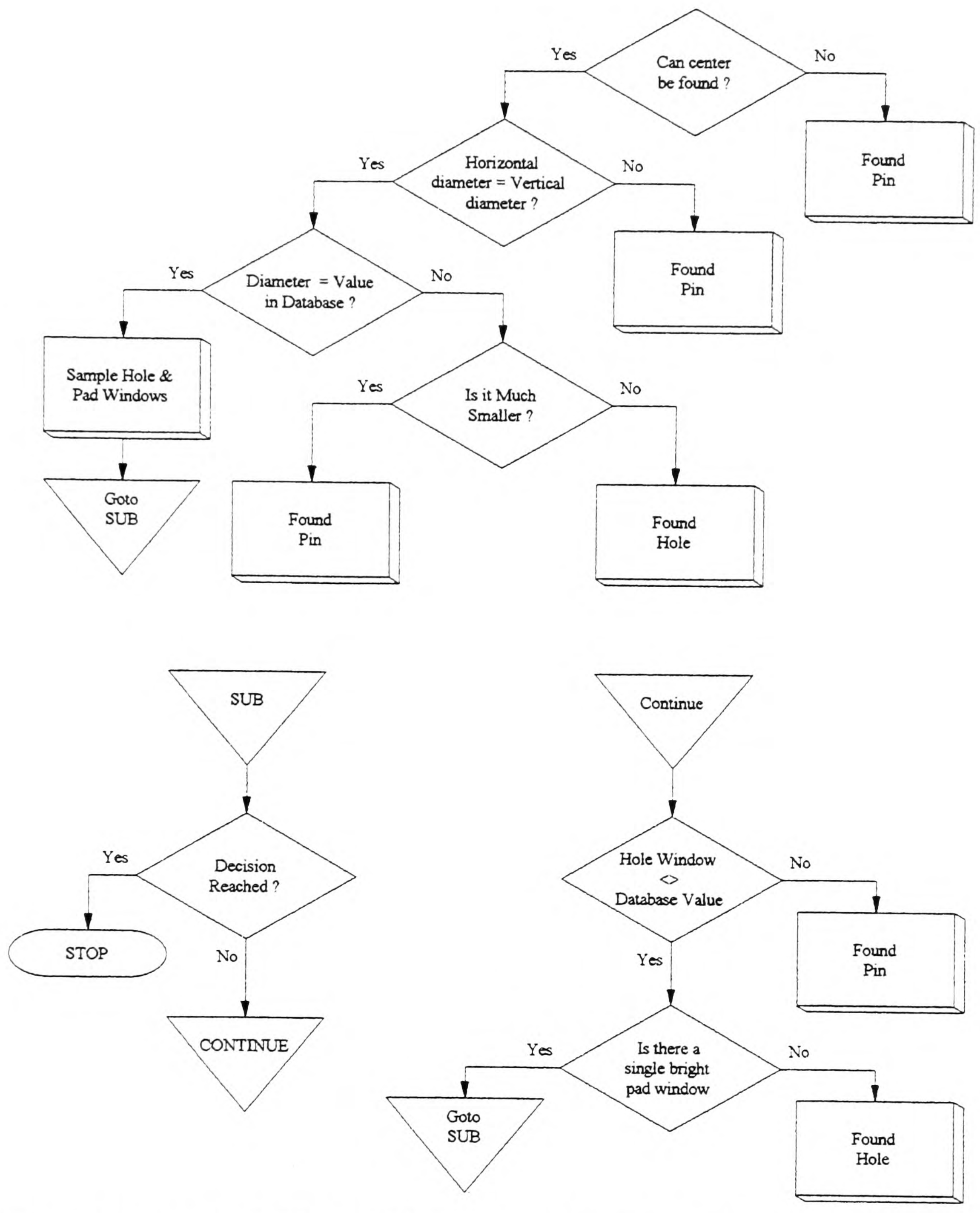
Parallel processing of lumber has been proposed by Conners by subdividing the image into a number of rectangles (Conners 83). For each of the image patches, the tonal property of the lumber is measured by computing i) the mean, ii) the variance, and iii) the skewness. Also, the co-occurrence matrix is used in the texture analysis.

In Conner's initial feasibility study, the combined measures yielded an overall 88.3% correct classification on the eight defects most commonly found in lumber. To minimise the number of calculations needed to make the required classification he uses a sequential classifier based on a production system. The classifier in this case cannot be extended to accommodate more faults without again reprogramming the system.

3.4.3 Printed and Integrated Circuit Board Inspection

We are continuing to observe the growth in density of both integrated circuits (ICs) and complex multi-layer printed circuit boards (PCB). This has led to problems in using traditional techniques for testing. The ICs and PCB fail because of (Pav 83):

1) corrosion or micro cracks, which lead to open or short circuits,
2) oxide breakdown by such processes as static discharge,
3) surface defects such as dust,
4) dirty photo masks,
5) die cracks,
6) packing defects, and
7) thermal mismatch.



igure 3.2 Decision tree in PCB and IC Bottom-side inspection.

hese problems can be detected by visual techniques such as stereo microscope, x-rays nd election beams. However, it is very difficult to manually check for defects in such boards. It has been reported that humans can detect about 90 percent of faults in a single layer PCB. This detection rate goes down to about 50 percent for six layer boards (Yu 88). Even when fault free power and ground layers have been established using electrical testing, the success rate is no better then about 70 percent.

As we can see the inspection of ICs and PCBs requires some sort of optical inspection to determine that the line widths, the line spacing, voids and pinholes are within the specification. These tasks cannot be carried out by electrical testing. Further advantages of optical testing are that it can check the ICs and PCB against its CAD model and thus the method is non-intrusive and hence avoids mechanical damage. A typical decision tree in PCB and IC inspection is shown in Figure 3.2 (Hara 83). If a trustworthy decision can be reached right away, then there is no need to go deeper into the tree. If not, then the decision criterion is refined by asking additional questions until a decision is reached with a sufficiently high confidence level.

It can be seen that even in the relatively simple application of PCB and IC inspection, high level processing is needed. The PCB alphabet comprises of circles, pins, holes and lines. However, most problems will have much larger alphabets of objects and sub-objects.

While some ideas expressed in detecting PCB faults can be implemented using lower level algorithms (Hough 62) more inferencing capabilities are needed. This will allow more robust techniques to be used.

3.5 Common Problems in Industrial Inspection Systems

A common problem in vision is to try to match some structure in the processed image with a corresponding part of a semantic network. Let the goal network be some structure to be matched with some fact network in the database. Each goal or fact network contains one or more nodes and arcs. In the simplest case a given goal can be matched against a fact in the following way: every node in the goal network is unified with a node in the fact network, each arc in the goal is then paired with an arc between corresponding fact nodes, Figure 3.3 shows this graphically.

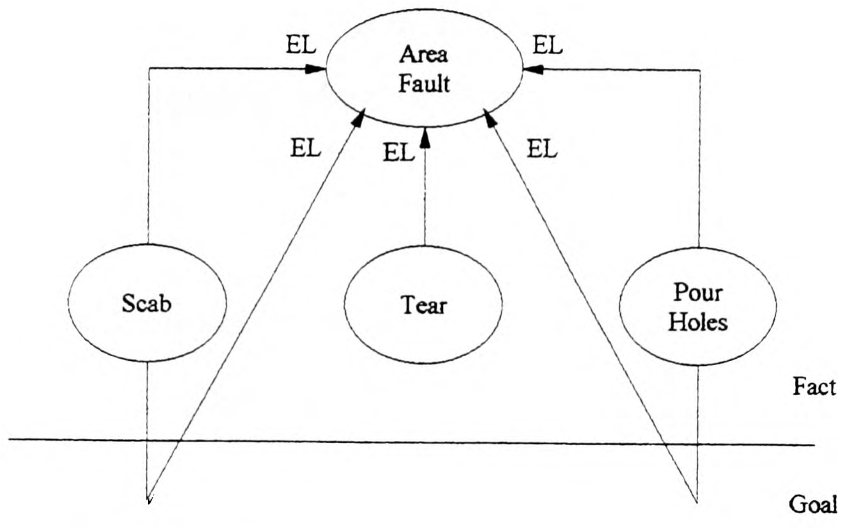


Figure 3.3 Goal and fact network.

There is a problem for the designer of computer vision systems when considering the level of domain dependent structures to put in the representation system. This is exacerbated when trying to develop a general purpose vision system. The more structure there is, the quicker the matching but generality is lost. The chapter on further work explains how such structures can be selected dependent on the domain knowledge. In simple cases the matching process can often be reduced to finding an isomorphism between graphs. However, this process has a high computational overhead. The complexity can be reduced if we can make sensible guesses about which substructures are the most important in matching. Simon provides a good treatment of the issues (Simon 86).

There are other ways to reduce the matching problem. For instance, the degree of a match could be determined on a probabilistic basis (Shapiro 83), or operators such as "greater than" could work on properties like length or area (Adorni 85). In these cases numerical processing may have advantages over symbolic processing in vision (Ambler 75).

Problems also arise at the lower levels of vision, image processing and segmentation. It is very difficult for the developer to select the appropriate algorithms for the task at hand. The main considerations, when selecting an image processing or segmentation algorithm are:

1. The information needed to be extracted from the image, i.e. lines, areas, syntactic representations etc.
2. To select the thresholds for the algorithms to gain maximum performance,
3. To incorporate any relaxation techniques needed to make the system robust,
4. To consider any computational restraints,
5. Selection of operators that best suits the needs of the system.

It can be seen that selecting the appropriate algorithms is no easy task as there are many factors that have to be considered. For instance, knowledge of the domain has to be acquired and analysed. Once this is done, the algorithms need to be selected and fine tuned to give the best possible results and the correct features need to be extracted for the higher level processing routines.

As we can see tuning is an interaction between the domain, image processing and segmentation algorithms, and the higher level, symbolic processing algorithms. With all this to contend with we are still at the mercy of the programmer's knowledge, have they selected the appropriate algorithms? Is the system robust? etc. There are many questions that must be answered, which need extensive tests. Evidently there is a great deal of uncertainty and unreliability in the whole process.

3.5.1 Unreliability of Low Level Processes

Low-level processes, such as region, line and surface extraction, vary in unreliable and unpredictable ways. This is due to a variety of confounding factors that include surface colours and markings, texture, occlusion, the complexity of 3D shapes, uncontrolled lighting producing highlights and shadows (Novini 90), and resolution and digitisation effects.

3.5.2 Uncertainty

There is inherent uncertainty in every stage of processing. Sensory data is usually locally ambiguous and constraints must be introduced at many levels to reduce uncertainty in an effective manner. As the data is transformed, it remains unreliable to varying degrees. The process of hypothesis generation and validation must work in the face of incomplete and inconsistent information.

Another major problem in these systems is the representation and appropriate use of all available sources of knowledge during the interpretation process. Each of the many different kinds of knowledge that may be relevant at various points during interpretation impose different kinds of constraints on the underlying representations. In general, they must be sensitive enough to capture subtle differences and variations among objects in the same object class, yet be robust enough to capture broadly applicable sketches of objects and expected scenarios.

3.6 Questions Addressed

Several questions have come to light during the review of vision systems, both in chapters two and three. These questions encapsulate low, medium and high level tasks.

Clearly there are many possible ways in which an image can be segmented. For example, should emphasis be placed on boundary analysis or region growing, or within segmentation should priority at some particular stage be given to splitting regions or merging others? The idea behind the current work is to control such possible low-level operations using the domain knowledge and target matrices within a production system.

Once the image is segmented, what is the best possible structure to model the data? Do we save geometric features, or salient features etc.? It is true that the more features we model the easier the matching problem becomes, but as stated above generality is lost. Some objects may not have well defined geometric features for instance. To have a truly adaptive system each structure found must be modelled uniquely. This implies that the segmentation algorithms applied to that object must also pass the structure back to the inference module, which in turn will select the appropriate matching algorithm known to work well with those features. (See chapter on further extensions.)

Current industrial vision systems are developed for the specific domain in which they will operate, with the domain knowledge being acquired at that moment in time. This raises the question of how flexible is the system? That is, what happens if the inspection task or manufacturing process changes? Do we scrap the system, or is it rewritten to incorporate the changes? This suggests that the system must be updatable, and thus how easy is it for such systems to be updated? The system proposed here is updatable by adding the new domain knowledge and if necessary updating the rules and segmentation algorithms. At the moment the prototype developed here requires a programmer's intervention, however a system is proposed where this can be done interactively by the user (see further extensions).

3.7 Summary

The prime objective in undertaking this review was to see if the goal of the proposed system has already been achieved in an existing system. The review has shown that no existing system has the same aim as the proposed system. Only the metal surface inspection system offers an adaptive thresholding technique (Suresh 83), other systems developed are purely domain specific. The knowledge required about the objects', noise, and environmental conditions are embedded within the system. This makes the system inflexible and unable to adapt to change within the inspection task.

We can deduce that, basically, industrial machine vision is a collection of techniques (both image/segmentation processing and recognition/matching procedures) specifically designed to operate on a given application. Recognizing that machine vision is a collection of techniques, and that there is no universal best technique, one can perceive a technology for selecting and refining appropriate algorithms where the application domain knowledge is given. This in fact is done by human programmers and experts in the appropriate domain. Knowledge is acquired by the programmer on the domain to be modelled (object description, faults, and environmental conditions) from the expert. The programmer then selects the appropriate algorithms to carry out the task. However, extensive testing and remodification is carried out until the system meets an acceptable performance level. This type of system then becomes inflexible and thus unreliable.

Chapter 4
The Visual Planner's Data and Knowledge Representation

4. Introduction

This chapter presents a technique for the automatic design of image segmentation algorithms. It describes the data and knowledge needed to implement such a system and the transformation of the image from a pictorial representation to a symbolic one. We describe a way of linking the information retrieved from the image to algorithms that may act on such an image using contextual information of the objects.

It has become clear that image segmentation is an essential step in every practical image processing system (see chapters 2 and 3), not just industrial inspection applications. Current vision systems and in particular their segmentation algorithms suffer from a well known problem, they have a poor performance rating on images different from the ones used in their initial development and training stage.

We therefore present a system concept for the automatic selection and design of segmentation algorithms based on image and object characteristics, and knowledge of image processing primitives (*the Visual Planner*). The proposed system concept makes use of Planning techniques, discussed in Artificial Intelligence (Wilensky 83, Georgeff 87). This concept provides the essential elements of an automatic segmentation design system that will not be scenario dependent and will be adaptive in nature.

4.1 Motivation Behind the Work

The main problem with existing segmentation algorithms is their poor performance on a set of images different from the ones used in their initial development and training stages. Changes in the image can be defined in terms of content and quality. This limitation has created a major bottleneck in most vision systems. It has been this limitation that has been the basic motivation behind the research and development of a new segmentation approach.

This new concept selects the segmentation algorithms, and sets their parameters, depending on the global image and target characteristics, and contextual scene information. To this end an adaptive system has been produced that not only changes parameters such as the ones proposed by Anderson (Anderson 87) or Pavlidis (Pavlidis 82) but also selects the appropriate algorithms available within its database for that type of scene.

In designing such a system we must keep in mind that the performance of the segmentation algorithms cannot be improved beyond a limited domain by (only) adjusting their parameters, as experiments have shown (Chapter 7 and Appendix B). The reason for this is that many basic assumptions made in the design of the segmentation algorithm are violated when different scenarios are encountered.

To overcome this problem we could use a knowledge based segmentation technique, whereby we attempt to select the appropriate segmentation algorithm from a database of two or three existing algorithms. However, again experiments have shown that these techniques also suffer from a similar problem, i.e. performance cannot be improved beyond a limited domain. This is because image processing and segmentation algorithms work on a narrow domain and even in their own domain their performance is rather unstable. Yet another problem with such a technique, is that most of the existing or available algorithms have a large area of overlap, and they do not span the space of useful application domain. Again, all these algorithms fail dramatically in many instances because they are based on fixed assumptions that in many cases are not valid. Consequently, selecting among a set of given algorithms cannot lead to a meaningful improvement in performance.

It has become clear that what is needed is a new approach to segmentation, illustrated in Figure 4.1, whereby the algorithm design is a planned sequence of different image processing primitives. Such a design depends on several inputs.

1. A desired goal of the segmentation process (i.e. find *x* fault or find *y* target, etc.),

2. The input image and target characteristics called the *Domain Information* (i.e. contrast, target size, signal to noise ratio, target average intensity etc.).

3. The contextual information about the objects (i.e. targets are geometric, linear, circular, etc.).

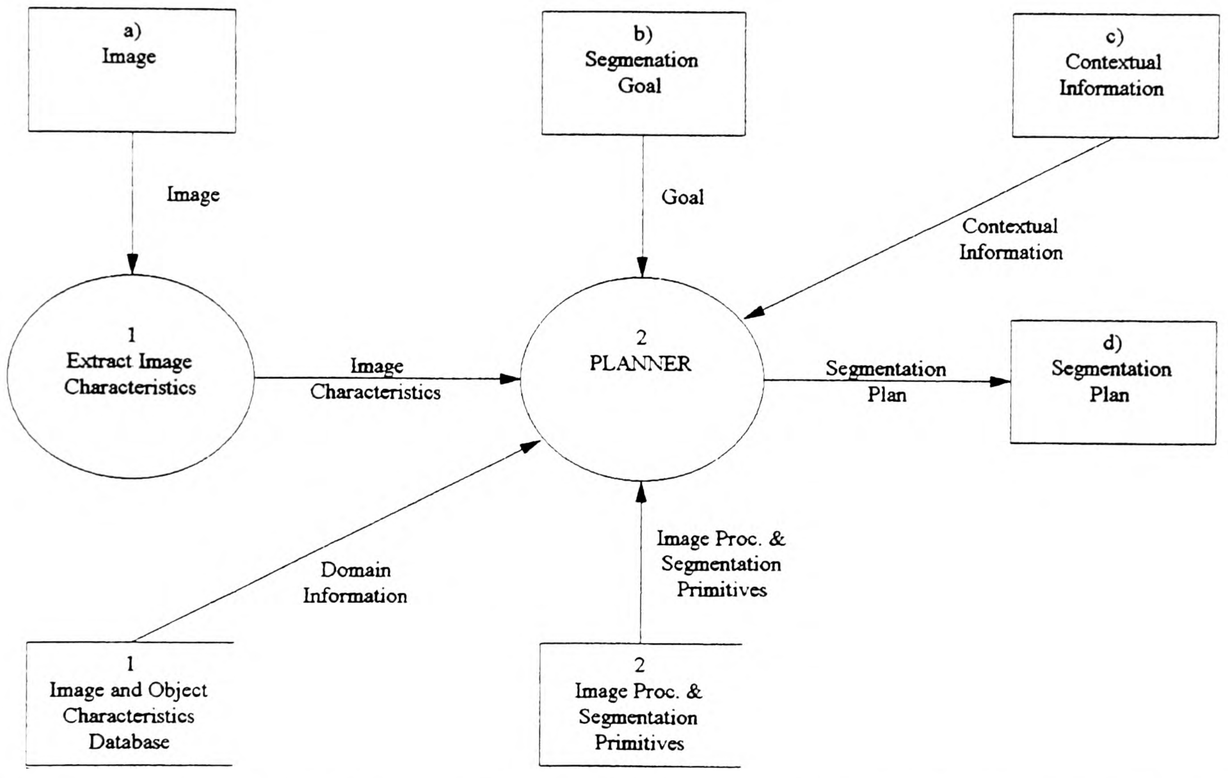

a) Image
b) Segmenation Goal
c) Contextual Information
Image
Goal
Contextual Information
1 Extract Image Characteristics
Image Characteristics
2 PLANNER
Segmentation Plan
d) Segmentation Plan
Domain Information
Image Proc. & Segmentation Primitives
1 Image and Object Characteristics Database
2 Image Proc. & Segmentation Primitives

Figure 4.1 Concept diagram for automatic segmentation algorithm design.

It can be seen that a different scenario, in terms of goal, image and target characteristics, and contextual information will lead to the generation of different plans, based on information available from the scene domain.

The more information and knowledge a planner is given, the more accurate are its results. This leads us to the question of, how complex can such a visual system become? Although it is easy to update algorithms or add new ones do the database, where they are used, and under what circumstances they are used in, is a more difficult task to achieve. This is due to the fact that a new algorithm may work well together with some algorithms and not so well with others. Conflicts may arise due to the nature of the algorithms, such conflicts may give unexpected results. The algorithms may require information that is not modelled within the object database, this may have to be added. As can be seen, the complexity of such a system is unbounded. However, constraints can be applied which will make the system practical to use.

4.2 Adaptive System: Overview

A detailed block diagram is illustrated in Figure 4.2. The input to the system would be a raw image, a set of image and target characteristics, contextual information, and an explicit segmentation goal. The output is a segmentation algorithm specification that is a collection of sequential image processing steps. The selected algorithms are tailored to a particular scenario. For each different scenario a different plan is created. In planning, we start with an initial state and a desirable state or a goal, and attempt to devise a plan that can achieve the goal (Wilensky 83, Genersereth 87).

The planning concept is illustrated in Figure 4.3. The initial state in this case is the raw image. The goal is the desired segmentation result that can be expressed at different levels of abstraction, find target, find straight lines, find x fault, etc. The plan is a set of ordered sequential image processing operations. The plan is generated by the planner that has access to the knowledge-base. The knowledge base contains information about which operations to perform given certain images these rules are called the *domain rules*. The knowledge-base is represented in a rule-base structure, however, other structures may be used.

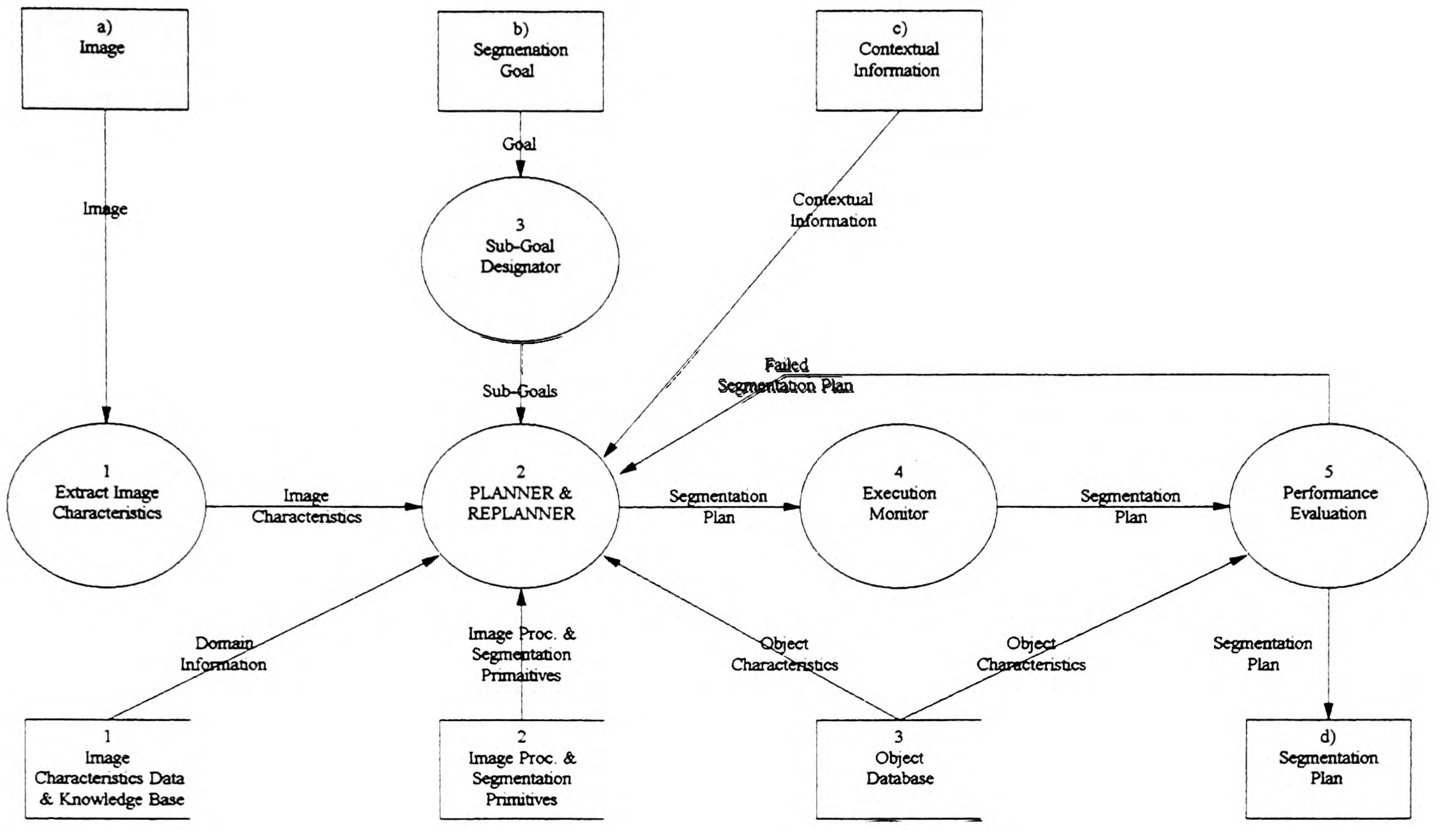


Figure 4.2 Detailed diagram of adaptive segmentation design system.

4.3 The Planner and Planning

The classical definition of the planning problem assumes a state-based representation of the world. This means that the world is represented by taking a snapshot of it at one particular time and describing the world as it appears in this snapshot. This description is generally a set of sentences in some formal language describing what is true in the snapshot. The sentences within this project take the form of two types of knowledge: contextual information, and image and object characteristics.

The inputs and the outputs of the planning process are illustrated in Figure 4.3. While the inputs (initial state and goal) to the planner are described as being real world problems, current AI techniques cannot handle the full complexity of our everyday world. The actual problem being solved is always in a limited environment, but the term real world indicates the environmental goal representing this full complexity.

The planner's output will be correct only to the extent that the representation modelled in the knowledge base correctly reflects the real environment. The output is a sequence of primitive actions, i.e. actions for which the planner has not been given any knowledge about the details of the means of execution. If the representation is appropriate, these actions will be meaningful in the environment and might be carried out by whatever agents are accepting instructions from the planner.

All inputs to the planner are in a language provided by the planner, whether it be pictorial or as a script of sentences. The three inputs illustrated in Figure 4.3 are described bellow:

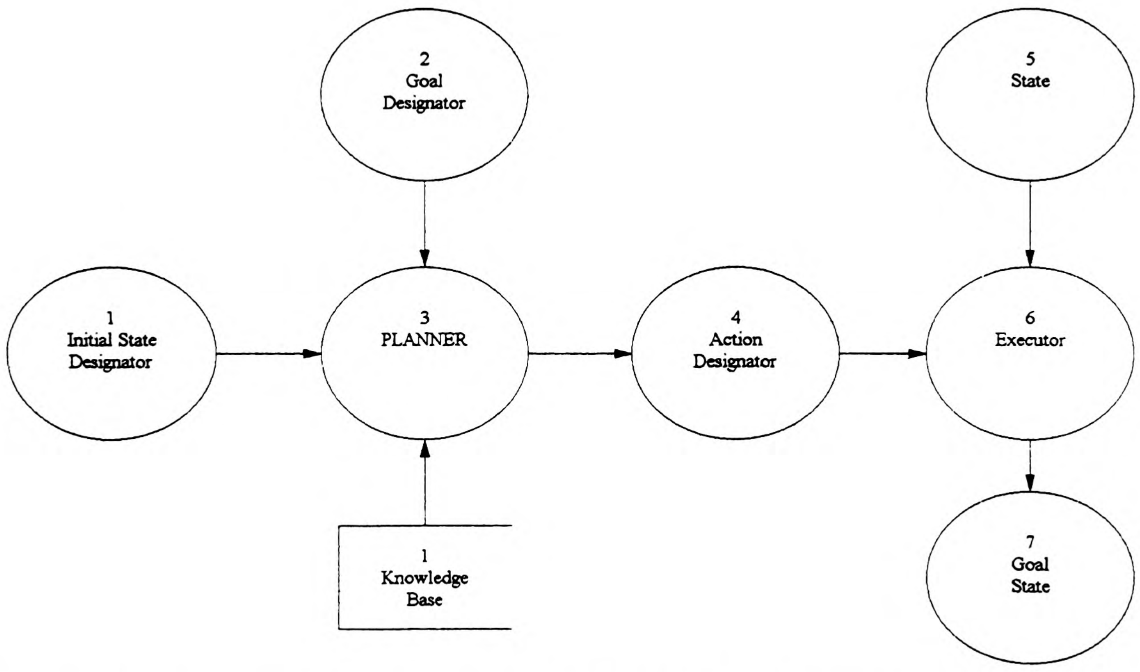


Figure 4.3 General planning concept in Artificial Intelligence.

1. The initial world state is generally described by a set of sentences, (from an iconic representation to a symbolic one in this case) although additional sentences may be deduced by the planner from domain constraints that are also provided.

2. The actions that can be taken in the world must be represented in such a way that the planner can take the state of the world in which the action is performed and map it into

the state of the world that will exist after the action is performed. This information is kept in the knowledge base.

3. A set of sentences that describes the goal to be achieved is also given. These sentences will be interpreted by the planner, and thus take any form desired, for example structured language (find crack > 5mm). The planner is then required to produce, as output, a description of a sequence of actions that, when applied to the initial state, will result in the goal being achieved.

4.4 Capabilities of a Planning System

The sections below describe some features that are desirable in a planning system. Most of these capabilities are necessary if a planner is to achieve heuristic adequacy. (i.e. be efficient enough to be useful in practice). AI planning systems generally have several of these features, although very few systems have them all.

4.4.1 Domain Knowledge Acquisition

A practical planning system must have some means of acquiring knowledge of their domain. This is mainly contextual information of the domain. Dean (Dean 92) was one of the first to propose the use of contextual information in practical planners.

4.4.2 Nonlinear Plans

A plan is nonlinear if it contains actions that are unordered with respect to each other, i.e. actions for which the planner has not yet determined an order. If a planner can represent and reason about nonlinear plans, it can avoid committing to a particular order of actions until information for ordering the actions has been accumulated. This implies that actions may be run in parallel. In image processing this capability is essential for fast execution of algorithms. This makes nonlinear plans very attractive in image processing and segmentation.

By allowing nonlinear plans we pose many problems for basic planning algorithms. How does the basic planning algorithm reason about how actions that may take place concurrently interact with each other? This type of knowledge must be incorporated within the knowledge base primitives, for example if one sub-goal is used before another what effect will it have on the initial state. The planner must be able to reason about this.

4.4.3 Hierarchical Planning

In complex domains it is crucial to plan at different levels of abstraction. For example, in planning to detect an object within a scene, the planner should first reason that the image processing routines should be executed before the segmentation routines, which in turn should be executed before image understanding procedures. This is an abstract level of reasoning, where at the most detailed level, the plan for executing segmentation algorithms will include specifications of which algorithms work well on particular objects. A useful domain independent planning system must support planning at different levels of abstraction. This is referred to as hierarchical planning.

Until now image processing and segmentation algorithm implementation within a vision system have adopted a bottom up approach, i.e. first the image processing routines were implemented and tested, then the segmentation algorithms, and finally the image understanding algorithms. This approach the author believes has many serious drawbacks: a) such a system development is domain dependent, and cannot work with different images, b) the system may not work well if changes occur in the environment, such as a change in lighting or if some form of noise is added, and c) appropriate algorithms may not have been selected for the task at hand. If errors are introduced at the lower stages of development, then they will be inherited and magnified by the higher level algorithms casing system failure.

4.4.4 Constraints and Variables

Constraints are used to limit the search space for a proposed plan. Constraints may take many forms such as variable limits, e.g. threshold value from 10-20, or constraints on different operators, e.g. operator one may not work well with operator two, or constraints on the domain, e.g., the average intensity of the domain must be within these limits. The method of constraints eliminates a huge search space of different algorithms and numerical values such as thresholds, making the planner more practical and reliable.

4.4.5 Replanning

When plans are executed in the real world, events rarely proceed exactly to plan. If a planner can change or alter its original plan after an unexpected event (such as an environmental change, or modified goal) and continue using the modified plan, it can potentially save considerable effort in replanning. This would make the system adaptive within its environment. Of course, the number of modifications one might make to an existing plan is enormous, so the problem is difficult. This project addresses this problem.

4.4.6 Domain Independence

AI planners are domain independent so their techniques are readily available for new problems. This means that any image processing domain in this case may be modelled. Thus if the manufacturing process changes or a new domain needs to be modelled, then we should only need to populate the knowledge base with the new domain knowledge. Such a planner provides the user with a knowledge representation module for encoding domain-specific knowledge. It is the quality of the knowledge so encoded that determines how well the planner performs.

What the domain-independent planning system provides is a representation that is particularly geared towards representing actions and reasoning about how their effects change the state of the world. By forcing the domain knowledge to be encoded in its formalism, the planner is able to use its solutions too many problems within the given domain the user chooses to encode.

4.5 The Visual Planner's Knowledge Bases

The Visual Planner developed within this project uses many of the techniques set out above. However, it differs from classical planners in that it is geared to work for industrial visual applications. Trying to make a complete planner for all world situations would be a complex task and beyond the scope of this project, it would also make such a planner impractical for the application domain selected (visual segmentation). However, within the world of visual inspection it is domain-independent and many situations may be modelled and implemented. As the development of such a system is a modular approach consisting of the planner and its *agents* (external modules), there has been no need to implement a full blown visual system. This property also has the desirable effect of making the system highly updatable and thus increasing its application domain.

4.5.1 The Visual Planner's Domain Information

While the desirability of having domain information is obvious for orchestrating initial plans and refining plans thereafter, previous classical planners or indeed adaptive visual systems have not incorporated such a capability. This is due to the fact that planners tried to be too general in nature and thus such capabilities became too complex

(Waldinger 81), and vision algorithms work on a method of refinement until some maximum or minimum is reached (Pavlidis 82). The Visual Planner presented here deduces many constraints based on the contextual information of the domain.

4.5.2 Domain Information Acquisition

The Visual Planner provides a method for acquiring domain information by interacting with the user. The Domain information acquisition process is a simple module that collects characteristics from the images presented to the system and as already stated from the user. The diagrams below (Figure 4.4a and 4.4b) shows the image and object characteristics acquisition process, and the transformation applied to the data before subsequent storage to the relevant databases. The diagram is presented at a number of different levels. The data extracted and stored is shown in the Table 4.2.

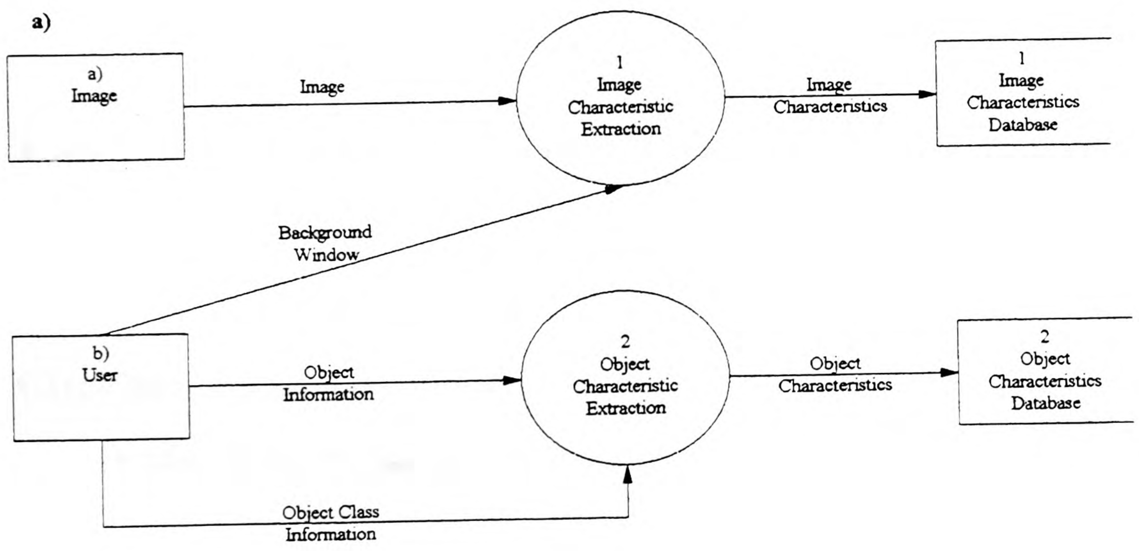
Attribute Name	Description
Average Intensity Histogram	Average intensity histogram of all images shown to the system in learning mode.
Maximum Background Threshold	Maximum intensity value of background given by user in sample window of image.
Minimum Background Threshold	Minimum intensity value of background given by user in sample window of image.
Maximum (+) Noise Deviation	Maximum positive noise deviation of noisy image to clean reference image.
Maximum (-) Noise Deviation	Maximum negative noise deviation of noisy image to clean reference image.

Table 4.2 Image characteristics Information.

The rules which act on domain information (see chapter 6) are called the *domain rules*. The Visual Planners domain rules allow expression of domain constraints. By accessing different images, the system can react to changes between two states, thus allowing the plan to change on domain knowledge only. For instance, if lighting conditions change then the planner can react to the change in the domain and thus alter the plan accordingly using the domain rules.

Domain information is gained by the Visual Planner by scanning the image as it is presented to it, and detecting any changes in the signal to noise ratio, average intensity, etc. If a change is detected, the domain rules specify what plan of action may be taken. Chapter 6 shows how the domain rules are used to create an initial image enhancement plan.

a)


a) Image
Image
1 Image Characteristic Extraction
Image Characteristics
1 Image Characteristics Database
Background Window
b) User
Object Information
2 Object Characteristic Extraction
Object Characteristics
2 Object Characteristics Database
Object Class Information

b)

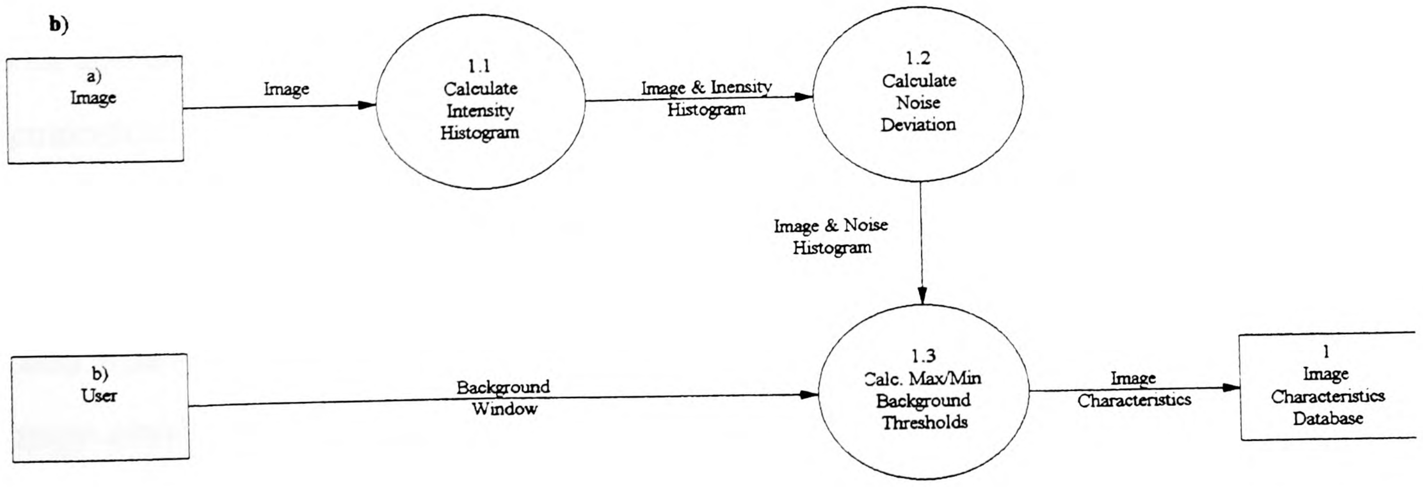

a) Image
Image
1.1 Calculate Intensity Histogram
Image & Inensity Histogram
1.2 Calculate Noise Deviation
Image & Noise Histogram
b) User
Background Window
1.3 Calc. Max/Min Background Thresholds
Image Characteristics
1 Image Characteristics Database

Figure 4.4a Level 1 diagram for domain knowledge acquisition.
Figure 4.4b Exploded process 1 from Figure 4.4a "Extract Image Characteristics".

4.5.3 Object Characteristics

Once the Visual Planner has extracted the domain information from a set of images, we must then extract object knowledge. Actual physical attributes (name, size, average intensity, colour, etc.) of objects may be stored in a CAD database, however more abstract information is also needed. We must be able to describe the objects in terms of their general shape, i.e. the object is linear or circular. This type of description is necessary when deciding which algorithm or set of algorithms to apply to the object that will best describe its segments. The Visual Planner has four predefined descriptions shown below, this type of information is known as the contextual information of the objects.

1. **Linear:** This is any object that has straight line segments within it. For example a cube is linear, a crack is linear etc.

2. **Circular:** These objects have a circular or elliptical shape, by this we mean that an object or part of an object has a radius and a centre.

3. **Geometric:** An object is geometric if the object is always of a predefined size, that is its size does not vary in any way, and the objects will always have the same lines of symmetry.

4. **Blob:** A blob is an object that has no predefined shape, may vary in size, and has no fixed lines of symmetry. This type of object is useful when trying to find objects on image attributes such as colour or average intensity only.

The contextual information described above imposes constraints for the object when selecting initial algorithms for a plan. The constraints listed here allow us to limit the search space for algorithms and produce hierarchical plans (see hierarchical planning chapter 5).

4.5.3.1 Extracting Object Characteristics

When extracting object characteristics from a scene, we need as accurate a description of the object as possible. This reduces any errors that may be caused due to a poor description. The Visual Planner acquires knowledge about an object by interacting with the user and inheriting contextual information as described above from other objects.

General information on the object such as name, class of object, etc. is given by the user. This information is summarised in Table 4.3.

General Information	Description
Main Class	The main class describes the object in general. e.g., object is crack, cube, resistor, etc.
Contextual Information	General Attributes of object class. e.g. Linear, Circular, Geometric, or Blob.
Name	Name of object. e.g. Surface crack, red cube, 500 ohm resistor etc.
Not Subclass	Exception constraint. Current object will not inherit attributes from the object specified. e.g. Surface Crack is not of type Internal Crack.

Table 4.3 General information held on an object.

The next step is to extract the physical object attributes. Ideally they should be stored in some form of CAD database, however, the Visual Planner learns the objects by applying segmentation algorithms to a selected object. Each image may have several objects of interest, however, only one object at a time may be considered. Once the general information on the object is given, the user selects a single object from the image by placing a window around it. The object characteristics are then extracted depending on the contextual information given previously. For instance if we describe an object as being linear the object characteristics extracted are lines and corner-points. Table 4.4 shows all the physical attributes of an object and what contextual information will apply to each attribute. This is then reconfirmed by the user, or if errors have been found it is amended by the user.

Object Attribute	Contextual Information	Description
Average Intensity	All Classes	Average intensity of object calculated in all cases.
Max Intensity Value	All Classes	Maximum intensity value of object calculated in all cases.
Min Intensity Value	All Classes	Minimum intensity value of object calculated in all cases.
Area	All Classes	Area of object calculated for all classes of objects.
Vertices	Linear & Geometric	Number of vertices counted if class attributes are linear and geometric.
Lines	Linear or Geometric.	Number of lines between vertices calculated.
Facets	Linear & Geometric.	Number of facets calculated within object.
Major Axis	All Classes	The major axis of each object is calculated. This allows the object to be in any orientation.
Minor Axis	All Classes	Same as above but for minor axis of the object
Radius	Circular, Geometric	If an object is circular and geometric then its radius is calculated..
Centre	Circular	Centre value calculated for circular objects.

Table 4.4 Physical attributes and the contextual information used to select them.

4.5.3.2 Object Hierarchy

Objects extracted from the images are stored in an object hierarchy. This allows the Visual Planner to inherit contextual information and any plans that may exist from other objects. This type of structure also allows us to declare an object such that it is not a member of a given object class. These are implemented as constraints on objects, and thus will allow the Visual Planner to work more efficiently by not producing initial plans if one exists for a similar object. The two class constraints are described below:

1. Class: This constrains specific classes in the object hierarchy. This allows the propagation of other constraints down to the subclasses, It also aids in the search for objects if the general class is known. An example of a class constraint can be given as:

Face Crack **OF TYPE** Crack.

The contextual information and an initial plan for Face Crack are inherited from a similar object of type Crack in this case. The plan may be refined at a later stage if needed.

2. Not-Class: This type of constraint is implemented for exceptions. Here we declare an object such that it is not a member of a given class. For example, we could declare that a Face Crack be a Crack except a member of the Class Internal Crack, more formally:

Face Crack **OF TYPE** Crack.
Face Crack **IS NOT OF TYPE** Internal Crack.

Here Face Crack inherits all the constraints of class Crack, and excludes all the constraints of class Internal Crack. Once again an initial plan for Face Crack is inherited, if one exists from a subclass of crack as long as it is not of type Internal Crack. Figure 4.5 shows how the class structure of objects is implemented and how contextual information is inherited.

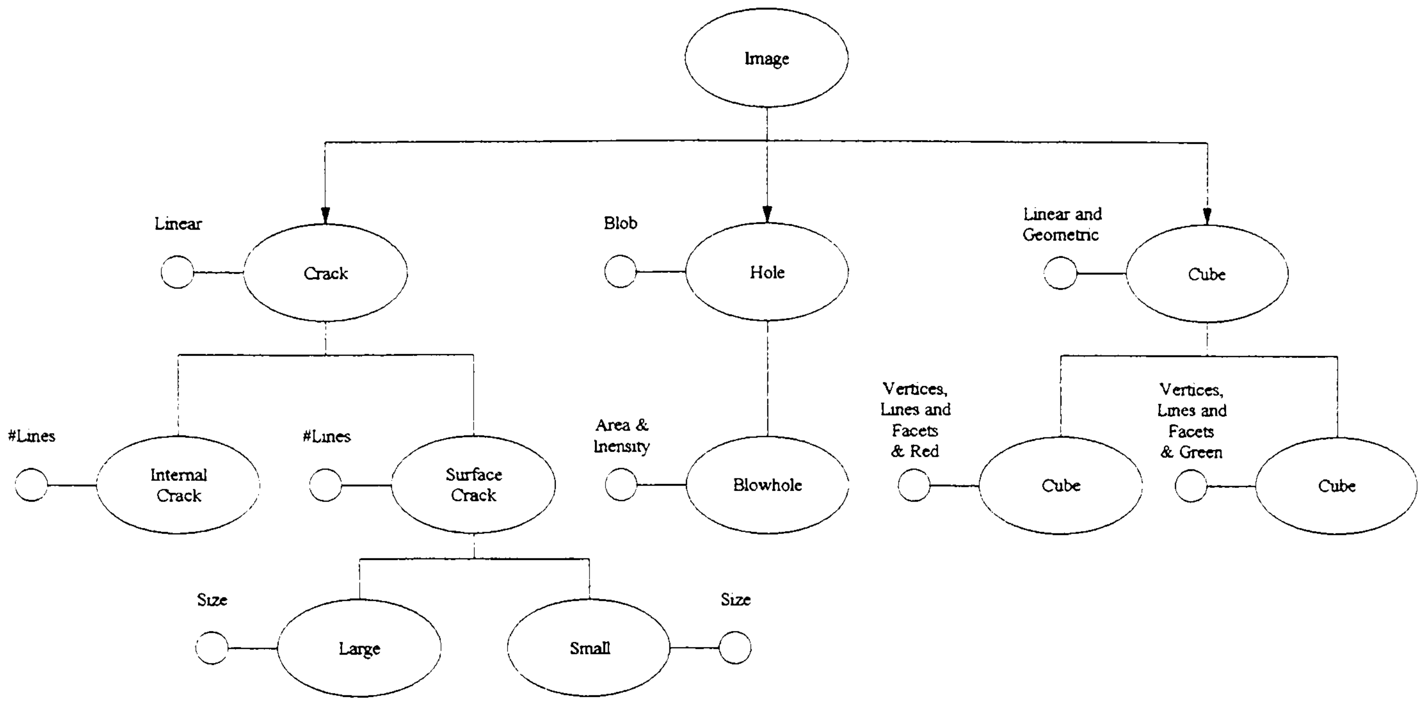


Figure 4.5 Object Hierarchy: sub-class inherits contextual information and existing plan.

4.5.4 Image Processing and Segmentation Database

Appendix C reviews a number of image processing and segmentation techniques. It is by no means an exhaustive list but gives the reader an idea of the algorithms implemented in this project. More importantly this type of review was undertaken to acquire the knowledge needed on the characteristics of such algorithms to classify them in terms of where they can be applied, with what thresholds (if any) under different scene conditions. This is necessary if algorithms are to be selected automatically and then adapted to suit the environment.

It is clear that image processing as opposed to segmentation algorithms perform different functions. Image processing routines enhance an image by reducing noise or highlighting particular features, they should not change the scene information in any destructive way. Whereas segmentation algorithms produce a more symbolic description of a scene, giving for example different regions, lines, corner points, etc., in terms of a coordinate system.

So how do we know which image processing and segmentation algorithms we need to employ at any moment in time? and how do we model this knowledge?

4.5.5 Image Processing and Segmentation Algorithm Knowledge

The algorithms employed within this project are divided into two sections, image processing algorithms, and segmentation algorithms. This is due to the fact that different types of knowledge are used to select the appropriate algorithms. Domain rules are used to select image processing algorithms, such as noise reduction, image enhancement, and background subtraction. The information used to initiate the image processing algorithms is extracted from the image and the image characteristics database. As opposed to the contextual information used to select segmentation algorithms. It then follows that any plan produced will be in two parts: a) the enhancement plan, made up of image processing algorithms, and b) the segmentation plan.

The enhancement plan precedes the segmentation plan, and each plan can change independently of the other, however, a change in the enhancement plan may effect the segmentation plan. Thus replanning is used if the sub-plans do not achieve the goal or sub-goal.

Clearly different types of knowledge needs to be kept for each plan. Image processing algorithms are selected by domain rules and thus no contextual information on such algorithms needs to be stored as image processing algorithms are used to highlight and improve the quality of the image, and thus do not extract features of objects. However, segmentation algorithms are selected according to the contextual information of the domain. If for example we are looking for linear objects such as a crack in an object, then only edge detectors and boundary following algorithms may be employed, with an appropriate threshold to highlight the crack specified. Table 4.5 shows the knowledge stored for each algorithm within the image processing and segmentation knowledge base. The object's contextual information maps directly onto the segmentation algorithms contextual information.

Image Processing Algorithm	**Description**
Name	Image processing algorithm name.
ID	Algorithms identification number.
Succeeding Algorithm ID	A list of algorithm identification numbers that can succeed the current algorithm.
Threshold Max Value	Threshold Maximum Value if applicable.
Threshold Min Value	Threshold Minimum Value if applicable.

Segmentation Algorithm	**Description**
Name	Segmentation algorithm name.
ID	Algorithm identification number.
Priority Number	What Priority this algorithm has within the order of the plan.
Succeeding Algorithm ID	A list of algorithm identification numbers that can succeed the current algorithm.
Threshold Max Value	Threshold maximum value if applicable.
Threshold Min Value	Threshold minimum value is applicable.
Contextual Information	This is a table of available contextual constraints, i.e., Linear, Geometric, Circular, Blob.

Table 4.5 Knowledge held on image processing and segmentation algorithms.

4.5.6 Algorithm Design Knowledge Base

Once a plan has been orchestrated for a particular object, whether it has been produced from scratch or inherited from other objects in the same class it must be stored for future use or to be inherited once again by other objects. This is done in the algorithm design knowledge base. For each object, its associated plan is stored in its complete form and order of execution, with specific thresholds set for each relevant section of the plan. This can be summarised in Table 4.6.

Data	Description
Object Class	General object class such as crack.
Object ID	Object identification number.
Algorithm Table [Max-algorithms]	Table of associated algorithms.
Algorithm ID	Algorithm identification number.
Threshold Value	Threshold value of algorithm if required.

Table 4.6 Algorithm design knowledge base structure.

4.6 Goal Specification

Once all the objects within the domain have been stored in the object characteristics database, the user must specify a goal for detection. This can take many forms and works on any attribute of the objects modelled within the object characteristics database. It is from this goal that an initial segmentation plan is orchestrated to segment any object within this database. The goal within the Visual Planner is selected from a list of available attributes found within the object characteristics database. The goals can be at any level of abstraction, i.e. **FIND ALL** or **FIND SURFACE CRACKS**. These goals move from a very general find to (find all objects) to a more specific find (find surface cracks). They may also be introduced with constraints, such as **FIND SURFACE CRACKS < 5CM**. The plan will be orchestrated on the specific user goals. A problem arises on how we divide these goals into more specific goals? This task falls to the Sub-Goal Designator that searches the database for the object or objects selected and extracts their attributes to make sub goals. For each sub-goal a different sub-plan is created.

The vocabulary used in the Visual Planner is simple but very powerful as any attribute may be selected. The vocabulary is shown below, but may be extended if needed to accommodate more elaborate search routines.

6.6.1 Visual Planner Goal Vocabulary

FIND	This is the main statement within the Visual Planner. The aim of any visual system is to find the object of interest, whether it is in a bin of parts or a fault on an object.
ALL	This will search for all objects if a specific object name is not given.
AREA	This puts a limit on the area of objects, e.g. **FIND HOLES AREA < 10 mm**. Area is used with some size constraint and object of interest.
<, >, =, <=, >=	These are constraints used to limit the search for size, intensity values, etc. Any combination may be used to specify ranges.
Any Attribute	Any attribute that exists in the objects knowledge base may be used to search for objects, e.g. **FIND ALL INTENSITY < 100 AND >50.** This will look for all objects with an average intensity of 100 or less but grater then 50. Any Attribute may also include the name of the object or its main class, e.g. **FIND ALL CRACKS**.

The Visual Planner prototype developed here uses only the **AND** logical operator, but could be extended to use **NOT** and **OR**, or any other logical operator.

4.6.2 Sub-Goal Designator

The sub-goal designator takes the main goal and starts to split it into sub-goals. For instance if **FIND ALL** has been specified it will search through the object characteristics database and start extracting each object in turn with its attributes, constraints and contextual information. Each of these descriptions is passed to the Visual Planner where an initial plan is orchestrated for that object producing a sub-plan for that object. The sub-goal designator traverses the object characteristics database until no more objects exist. The goal to sub-goal split can be shown in Figure 4.6.

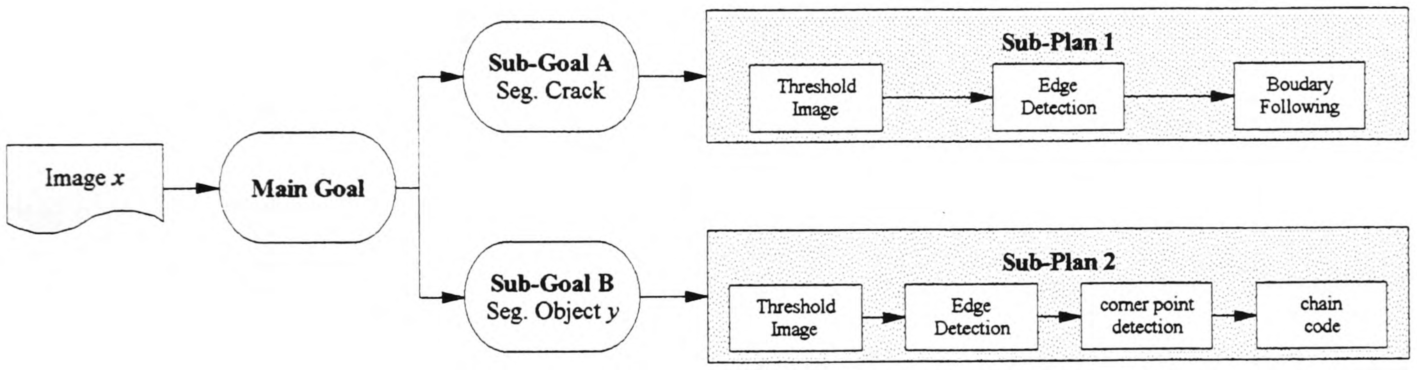


Figure 4.6 Main goal to sub-goal split and then to sub-plan creation.

If constraints are added (see chapter 5, Constraints), whether they are size constraints, specific objects, or attribute constraints, then only those objects that fall within the constraints are considered. More specific constraints reduce the search for objects, and reduce the time the Visual Planner takes to sort and select the algorithms.

4.7 Summary

This chapter addressed the motivation behind the work developed within this project. It presented a system for the automatic design of image processing and segmentation algorithms using a planning technique (The Visual Planner). It described the knowledge and data needed to produce such a design (the plan), how this data is acquired, and extracted from images presented to the Visual Planner. It showed how the contextual information of the object mapped directly onto the contextual information in the image processing and segmentation primitives knowledge base. It showed how contextual information is inherited from different classes, in order to reduce the time needed in creating a plan if one exists for a similar object in the same class.

Finally it describes the goal specification entered by the user to initiate the design of the plan, and how the goal can be specified at different levels of abstraction. We showed how the main goal is reduced to sub-goals by the sub-goal designator, creating a sub-goal for each object found in the object characteristics database that falls within the main goal specification.

Chapter 5
An Adaptive Visual Planning System

5. Introduction

Once all the knowledge and data necessary has been acquired by the Visual Planner we may proceed to orchestrate a plan for each object presented to the system. A goal is given by the user at some level of detail, such as **FIND Surface Crack**. The Visual Planner then has the image characteristics, contextual information, object attributes and, a goal to proceed with the plan. An initial plan is orchestrated for that object comprising an enhancement plan and a segmentation plan to provide a complete plan that achieves the overall goal.

Once the plan has been formed, it is executed to see if the goal has been satisfied by the Execution Monitor. If the goal has failed then the Visual Planner replans by using different algorithms if available, or different thresholds depending on v/hich sub-goal has failed.

This chapter gives an overview of how initial enhancement and segmentation plans and refined plans thereafter are orchestrated. It also shows how the Visual Planner compares to other planners and visual systems available today.

5.1 The Initial Plan

Once the Visual Planner has acquired all the data needed and a specific segmentation goal, we may start the planning process. The first step for the Visual Planner is to produce an initial plan. The initial plan consists of two sub-plans:

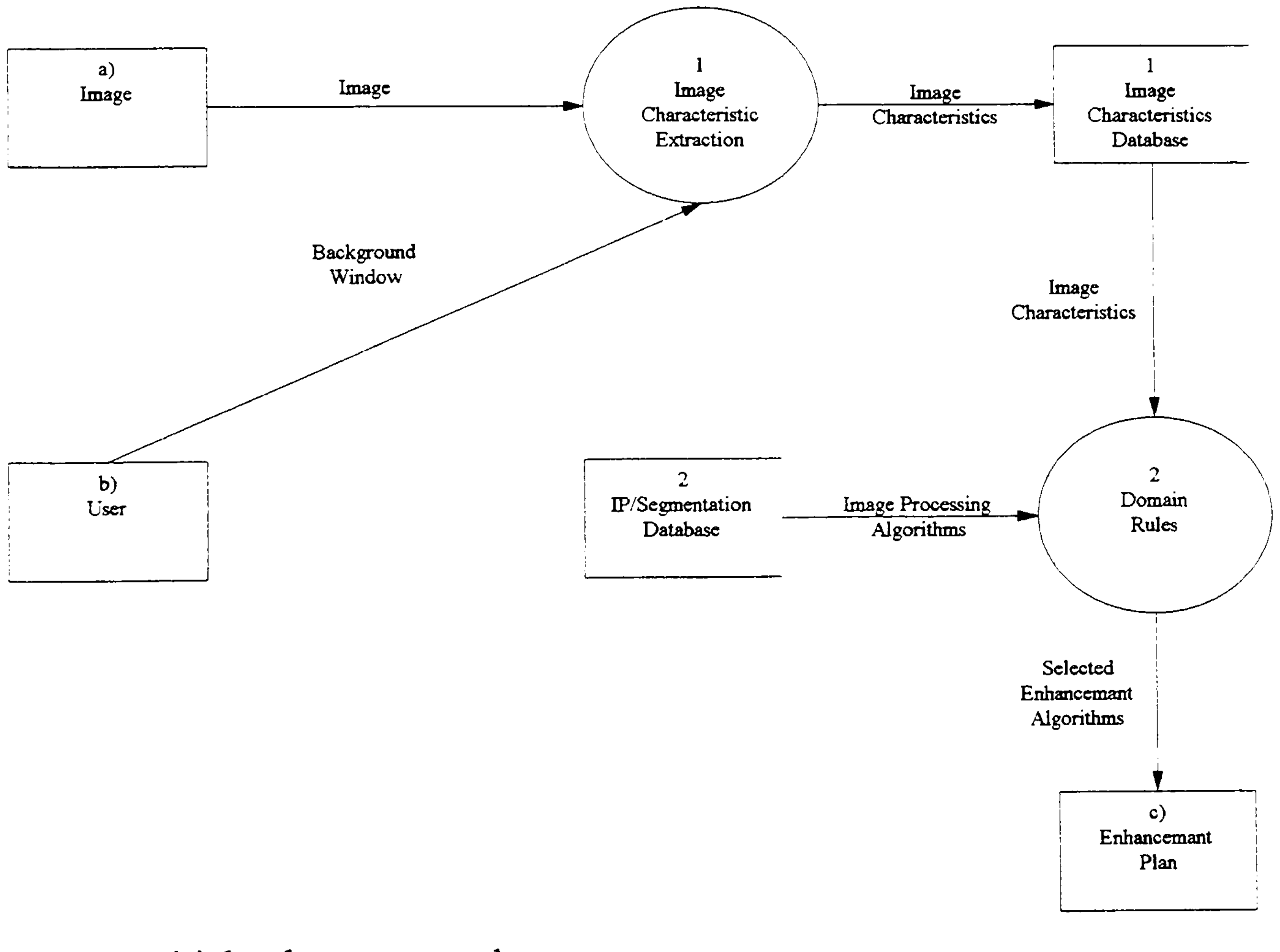
a) The Initial Enhancement Plan: This is composed of image processing algorithms (Lindley 91), and is used to enhance the image by reducing noise, removing the background, and generally highlighting features in an image we are interested in.

b) The Initial Segmentation Plan: This plan is composed of segmentation algorithms (Pavlidis 82), and we hope will extract the features we are interested in from the image as a symbolic description.

The two stages of the initial plan are described below, with an outline of the data needed to create such plans, and how they are achieved by the Visual Planner.

5.1.1 The Initial Enhancement Plan

The enhancement plan is orchestrated by the Visual Planner by acquiring knowledge of the domain from the image characteristics database (see chapter 4). The *domain rules* will then select the appropriate algorithms to enhance the image. This is shown in Figure 5.1.



a) Image
Image
1 Image Characteristic Extraction
Image Characteristics
1 Image Characteristics Database
Background Window
Image Characteristics
b) User
2 IP/Segmentation Database
Image Processing Algorithms
2 Domain Rules
Selected Enhancemant Algorithms
c) Enhancemant Plan

Figure 5.1 Initial enhancement plan.

The Visual Planner's domain rules (see chapter 6), are a set of rules that select appropriate image processing algorithms depending on the information held in the image characteristics database. Domain rules are also used to monitor the image and make any changes to the enhancement plan if there is a change in average intensity, or noise within the image. The initial enhancement plan will however remain the same for all images within that domain unless some change has occurred, such as a change in the overall average intensity value, which may have occurred for example due to a change in the lighting conditions.

5.1.2 The Initial Segmentation Plan

The initial segmentation plan for a given goal is orchestrated by using contextual information of an object and matching it to the same contextual knowledge for the algorithms. An initial segmentation plan may also be orchestrated using an existing plan for an object in the same class.

The goal specified by the user can be at different levels (**FIND ALL = Intensity Value < 100** or **FIND Internal Crack**). However, initial plans are only orchestrated for individual objects. The goal is then split up by the sub-goal designator to accommodate only individual objects in turn (see sub-goal designator, chapter 4)

Once a goal or sub-goal is given to segment a specific object, for example given the goal **FIND Crack x**, its attributes, contextual information and class are extracted form the object characteristics database. This is shown in the structure below.

MAIN CLASS:	Crack	
NAME:	Crack x	
CONTEXTUAL INFORMATION:		Linear
ATTRIBUTES:		
AVERAGE INTENSITY:		100
MAX INTENSITY VALUE:		150
MIN INTENSITY VALUE:		50
VERTICES		1
MAJOR AXIS		100 Pixels
MINOR AXIS		3 Pixels

The attributes extracted from the object database will depend on the contextual information of the object. In this case a linear object has been given as the goal, and therefore no size information needs to be extracted unless specifically stated in the goal. The class type of the object is also extracted such as crack. This is a simple hierarchical object class structure that helps the planner select an existing plan if one exists for a similar object with the same class structure. These classes are updatable as more objects are added to the object database.

5.1.2.1 The Initial Segmentation Plan Using an Existing Plan

Before an initial plan is orchestrated from scratch for a given object, the Visual Planner checks to see if one exists for an object in the same class, with the same contextual information (in this case **Class Crack**, with **Linear** contextual information.). The algorithm design knowledge base is traversed until the same class of object is found. If a match exists, then that object's contextual information is extracted by the Visual Planner. If they are the same then a match has been found. (See object hierarchy chapter 4).

If a plan exists for that object class, then the thresholds are changed according to the attributes of Crack x. This plan then becomes the initial plan for Crack x. Figure 5.2 shows how an initial plan may be selected from a set of existing object plans with the same class as long as it is not an exception class.

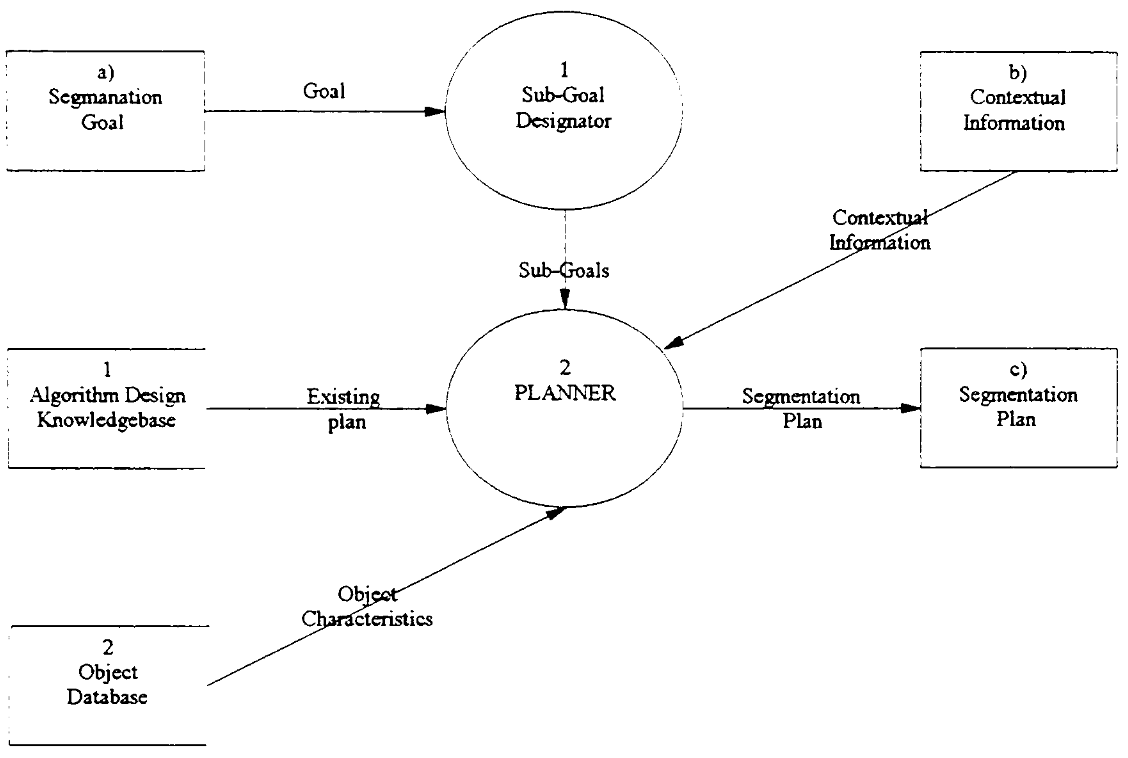


Figure 5.2 Initial segmentation plan orchestrated from an existing plan of a similar object.

5.1.2.2 The Initial Segmentation Plan using Contextual Information

If no plan exists, then one must be orchestrated from scratch. This is done by using the contextual information and attributes of the object as specified by the goal or sub-goal. The Visual Planner starts by matching the contextual information of the object to the contextual information of one algorithm within the image processing and segmentation primitives database. If many matches are found then it selects the one with the lowest priority level. The priority level being the order in which the algorithms must be executed. If two or more algorithms exist with the same contextual information and priority level, then only one algorithm is selected, and the rest are marked for use by the replanner if the algorithm selected fails (see replanner below).

Once an initial algorithm is selected, the plan is orchestrated by taking succeeding algorithms in turn and adding them to the plan until no more succeeding algorithms exist with the same contextual information. However, as many succeeding algorithms may exist for one particular algorithm one is selected and the rest are marked for reuse if the plan fails. The Visual Planner uses a backtracking algorithm to consider all possible plans.

If an algorithm requires a threshold value to be set, then the difference between the object's background value and average intensity is set as the threshold, however, maximum and minimum threshold values are set, according to the background's average intensity and the object's average intensity. This is a form of constraint within the Visual Planner known as the range constraint (see constraints below). This constraint works on algorithms that use variables as parameters. A variable can be considered to lie within a certain range, e.g. **Threshold Min Value = 10, Threshold Max Value = 100.** This type of limit reduces the number of computations and mutations an algorithm would go through in the replanning process (see replanner below). The structure below shows how this information is represented.

ALGORITHM NAME:	Sobel Edge Detector
CONTEXTUAL INFORMATION:	Linear
PRIORITY LEVEL:	3
SUCCEEDING ALGORITHM ID:	7 ("Boundary following")
THRESHOLD:	Object Background Intensity - Object Intensity
MIN THRESHOLD:	Object Background Intensity
MAX THRESHOLD:	Max Object Intensity

As a different initial plan is created for each sub-goal, it follows that each sub-plan can be executed in any order. This makes the plans nonlinear, and they may be executed in parallel.

Figure 5.3 shows how an initial segmentation plan is orchestrated using a given user defined goal, the contextual information and characteristics of the object to be segmented, and the image processing and segmentation primitives.

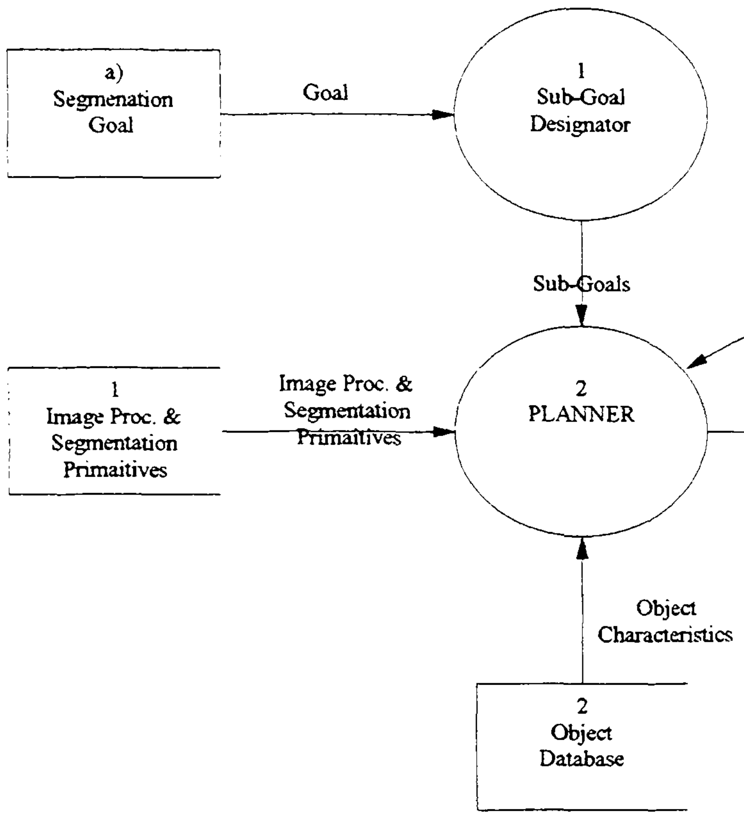


Figure 5.3 Initial segmentation plan.

5.2 Replanning

The Visual Planner provides a replanning module. This carries out two main tasks: Re-creating and refining initial sub-plans and, replanning after an unexpected event has occurred.

The problem is the following: given a plan, a world description, and some appropriate description of an unanticipated situation that occurs during execution of the plan (goal failure or environmental change), our task is to transform the plan, retaining as much of the old plan as is reasonable, into one that will still accomplish the original goal from the current situation. This process can be divided into three steps:

1. Discovering or entering information about the current situation.

2. Creating "fixes" that change the old plan, possibly by deleting part of it and inserting some newly created sub-plan.

3. Determining whether any changes effected by such fixes will conflict with the remaining parts of the old plan. For example if the environment changes due to an increase in lighting then other sub-plans that have already succeeded may fail with this change.

The replanning problem as described above has not been fully addressed by any system to date. Very few planners provide a replanning capability and, visual systems cannot easily adapt to changes in their environment. Since the Visual Planner has a domain-independent replanning capability, we will describe this capability in some detail.

5.2.1 Recreating and Refining Initial Plans

When selecting an initial plan for a given goal or sub-goal, whether it is inherited from another object in its class, or created using the contextual information and attributes of the object and mapping them to the appropriate algorithms, we cannot be sure that they will achieve the overall goal. The sub-plans are executed individually by the Execution Monitor to see if they extract the appropriate information on the object, and if this object can be matched to the same object in the object database. If this is the case the sub-plan is valid, and thus stored in the algorithm design knowledge base for that object. The sub-plan selected may not be the best plan, but it has extracted enough segments to identify the object. The object recognition module within the Execution Monitor (see execution monitor functions chapter 6) has an acceptance threshold for each segment modelled which may be set as high as 100% if we require all segments to be extracted and matched for a given object. This however, will cause more changes in the sub-plan, and may cause the sub-goal to fail if no sub-plan can reproduce all the segments of an object. A more realistic figure of 90% may be given. The acceptance threshold given will depend on the application domain.

The problem arises when a sub-plan fails, and thus the sub-goal fails. This must be fixed in order for the overall goal to be achieved. The Visual Planner starts by looking for an alternate algorithm to start with. As we have already marked alternate algorithms the search space is reduced, if the algorithms were inherited then the initial planning process is started form scratch, but excluding the current plan. The whole process is repeated until one of two things occur: a) the plan has achieved the goal and extracted the object of interest or, b) there are no more algorithms to consider.

If the new sub-plan has succeeded then it is stored in the algorithm design knowledge base, and the Visual Planner moves onto the next sub-plan until all sub-goals have been achieved. If however, all the new algorithms have failed, then the Visual Planner starts with the original algorithm and starts to change the threshold values where appropriate within the range specified, from minimum to maximum threshold values. The minimum and maximum threshold values are used as constraints, thus limiting the search space. By changing the initial thresholds different effects are encountered, and over-segmentation may take place. Over-segmentation is the process by which the objects being segmented from the background are themselves segmented or fractured into sub-components. However, with over-segmentation we are still able to match the object of interest.

If all algorithms and threshold changes have failed then the Visual Planner will interrupt the user and ask whether to continue with the other sup-goals or not in order to achieve a partial goal. The overall plan will still be able to work, but not at 100% efficiency, for example only six sub-plans have succeeded form 10. It will be up to the user to decide if this is desirable or new algorithms must be introduced into the Visual Planner. Figure 5.4 shows how the replanner makes changes to the initial sub-plan.

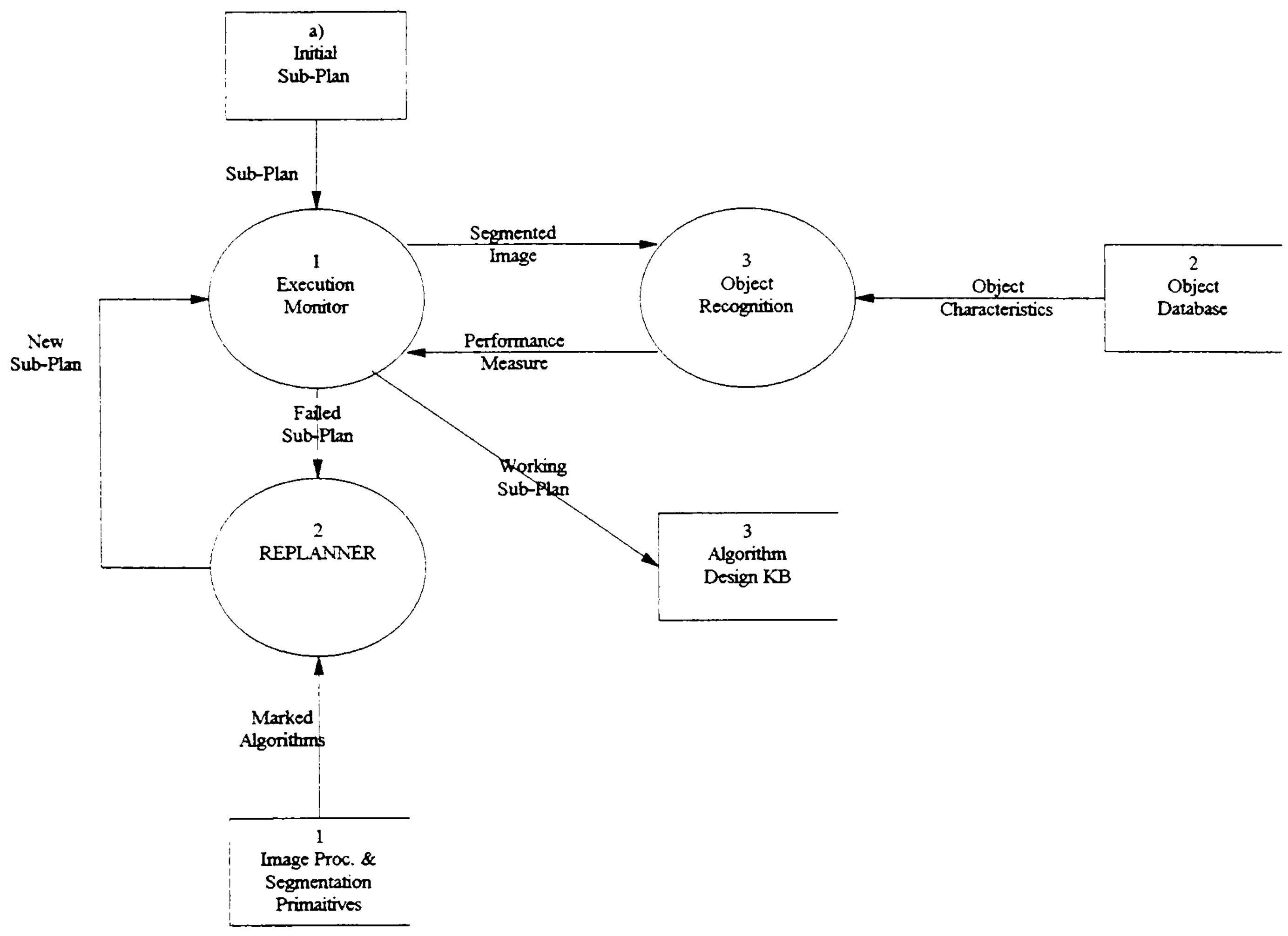


Figure 5.4 Replanning process for the initial segmentation sub-plan.

5.2.2 Replanning After an Unexpected Event

In real-world domains such as quality control, things do not always proceed as planned. Therefore, it is necessary to monitor the execution of a plan and to replan when things do not go as expected. In complex or time critical domains, it becomes increasingly important to use as much as possible of the old plan, rather then to begin anew when new situations arise.

In general, optimal recovery from an arbitrary error poses a difficult problem. Often very little of the existing plan can be reused. One can always fall back on solving the original problem in the new situation, ignoring the plan that was being executed. Since the problem is so difficult, one would not expect impressive performance, in terms of producing optimal plans that reuse the original plan, from a domain-independent replanner. Producing optimal plans requires domain-specific information for dealing with errors. In many domains, the type of errors that are commonly encountered can be predicted (e.g. The lighting conditions change, or photographic equipment is out of focus).

The Visual Planner uses domain rules to monitor the state of the environment and compensates for these by making changes to the enhancement plan using further domain rules. As this plan is common to all sub-segmentation plans this need only be done once. A problem arises when the changes to the environment change the threshold values in the image characteristics. This information must be passed to the algorithms in order to change the range (see constraints below) threshold values for future plans. An ideal situation would be where the threshold values change automatically both in the object knowledge database and any existing plan as environmental changes take place. Active database technology would allow for this as events occur (see future work).

5.3 Nonlinear Plans

The sub-plans produced by the Visual Planner are initially nonlinear as each sub-plan is unordered and they may be executed in any sequence with respect to each other, this is advantageous as sub-plans may be made to run in parallel if required. As sub-plans are nonlinear they may be changed independently without affecting the overall plan.

Once the sub-plan has been selected, it may be executed in its own right. This causes a segmented sub-image to be produced which we hope will have the features necessary for matching the object of interest to the object Data base, in order to achieve the sub-goal. For each sub-goal a sub-plan is produced which works on the original unsegmented image independently. These algorithms may then run in parallel giving an effective segmentation and recognition solution. The Visual Planner prototype does not execute these sub-plans in parallel, however, it does not matter which sub-plan is executed first or last.

5.4 Hierarchical Planning

It is generally recognized that planning in realistic domains requires planning at different levels of abstraction (Hobbs 85). This allows the planner to orchestrate plans more efficiently. The combinations of concatenating the most detailed possible descriptions of actions would be overwhelming without the use of more abstract concepts.

To see how hierarchical planning can help to avoid the combinatorial explosion involved in reasoning about primitive actions, consider trying to find an object x in a scene. The highest abstract level might be to match x to a database of objects. The Visual Planner can plan sequences of these steps without considering the detailed actions of noise reduction, edge detection, or segmentation. Each of these steps can be expanded into more detailed actions, finally getting down to which edge detector or noise reduction filter to use. Hierarchical abstraction levels provide the structure necessary for generating complex plans at the primitive level.

Within the Visual Planner this hierarchical level of abstraction is achieved by looking at the object's characteristics and contextual scene information. The hierarchical plans are extracted by the sub-goal designator (see module functions). Such a hierarchical plan for finding a cube is shown in Figure 5.5.

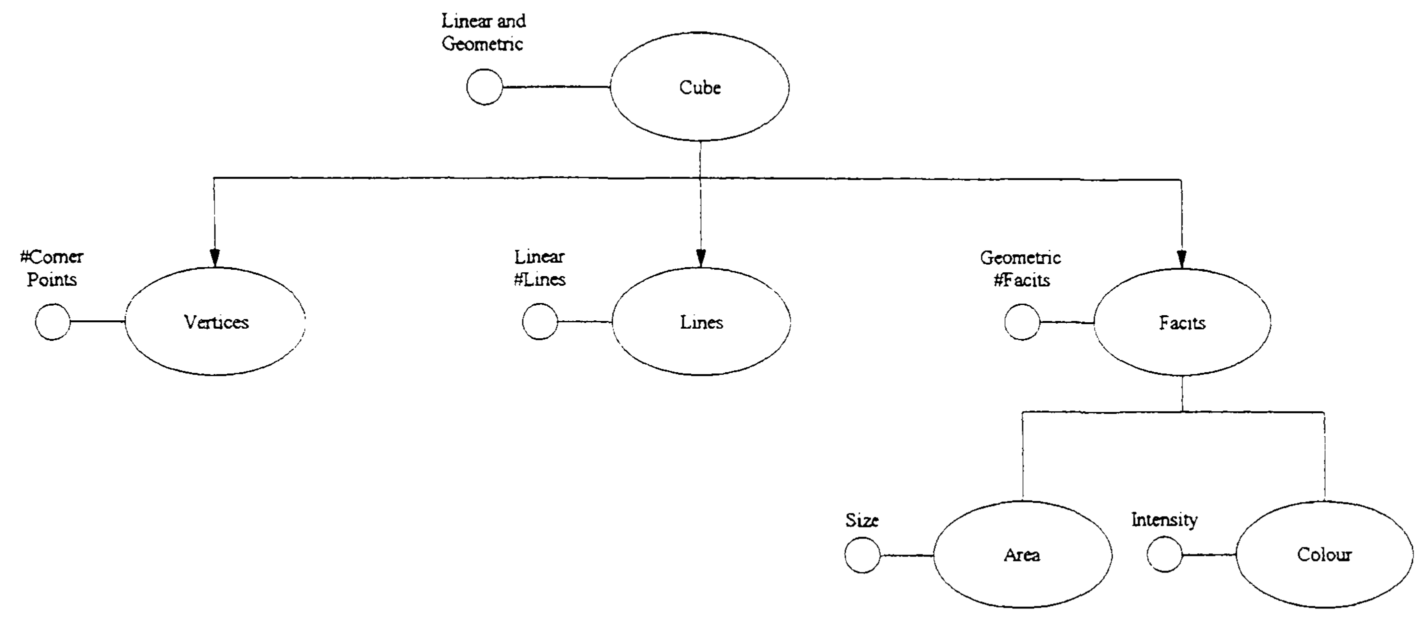


Figure 5.5 A hierarchical plan for finding a cube.

The cube is described in terms of vertices, and edges. The hierarchical plan will now know only to select edge detectors and corner-point detectors as a cube's contextual knowledge would be linear and geometric. Extra scene information is taken directly from the scene, and extra object attributes are taken from the database of objects, e.g. height, width, average intensity etc. There has been no mention of which algorithms to use, however we have limited the search space of algorithms to (in this case) edge detectors, corner detectors, and scene enhancement operations. If these operations do not achieve the sub-goal then we may replan using similar operators.

As can be seen the Visual Planner is capable of simple image understanding, to see if a sub-goal has been achieved. However, more complex scene understanding, including 3D and occluded objects will have to be carried out by dedicated algorithms, which may be selected by the Visual Planner (see future work).

5.5 The Execution and Monitoring System

The Visual Planner monitors the world directly using domain rules, it can call the replanner after it has detected a change in the environment or a sub-goal has failed. Often, it is able to retain most of the original overall plan by making some modifications to the sub-plans.

This capability extends the capabilities of previous general planning systems by exploiting the rich structure in the system's plan representation using nonlinear sub-plans and integrating the replanner with the planning system itself. This integration provides a number of benefits, of which the most important follow: contextual knowledge is used to provide a reasonable solution to potential fixes quickly, and the replanner can be called as a subroutine to solve problems after the execution monitor has detected a sub-goal failure. The last effectively eliminates the problem of needing to check interactions between fixes and the overall plan as only independent sub-plans need to be changed.

The replanning part of the Visual Planner tries to change the old sub-plan, using heuristics, contextual knowledge, and image attributes. If all possible changes to the sub-plan fail, then the overall plan may still work by achieving the other sub-goals. The sub-plan that has failed is completely discarded and the performance measure of the overall plan is reduced according to the number of sub-goals achieved. For instance, if there are ten objects to be found, and one sub-plan fails, then only nine objects have been detected successfully, the performance measure will fall to 90%. This may be acceptable to the user depending on what object has not been detected and how critical the domain they are working in is. The performance measure can be given more formally as:

pm% = (nsp-nf) * (100/nsg)

where:	nsg = Number of sub-goals
	nf = Number of failures
	pm% = Performance measure

The execution monitor also scans the images as they are presented to the Visual Planner. It is during this scan that any environmental changes are detected, that is, if the image characteristics presented to the Visual Planner do not match the image characteristics in the image characteristics knowledge base, then domain rules are called to either fix the problem, or interrupt the user for further information.

5.6 Planning Constraints

One of the Visual Planner's most important advantages over many previous adaptive vision systems is the ability to use constraints when producing a plan. This ability is important both for domain representation and for finding solutions efficiently. While constraints have been used in domain-specific planning systems (Stefik 81), and in virtually all complete visual systems, the constraints themselves have been domain specific, making them less useful for solving different problems. The Visual Planner presented here is the first visual system to use domain-independent constraints.

There are many types of constraints within the Visual Planner described previously. Constraints may place restrictions on the properties of an object (e.g. requiring certain attribute values for that object). They may also require that certain relationships exist between an object and other objects (e.g. crack x is of class crack). The Visual Planner provides a general language interface for expressing these constraints and relationships.

Constraints improve efficiency because large parts of the search space are reduced, since only predictions that are consistent with the constraints will be true. Other planning systems such as NOAH (Sacerdoti 77) would choose algorithms that may fail, even with a backtracking capability, it would still have to search the whole space in the worst case.

Much of the Visual Planner's expressive power and efficiency is rooted in the ability to reason about constraints. However, constraints add considerably to the complexity of the planner because they interact with all parts of the system. The allowable constraints implemented in the Visual Planner are listed below:

1. **Class:** This constrains specific classes in the sort hierarchy. This allows the propagation of other constraints down to the subclasses, It also aids in the search for objects if the general class is known. An example of a class constraint can be given as:

Face Crack **OF TYPE** Crack.

2. **Not-Class:** This type of constraint is implemented for exceptions. Here we declare an object such that it is not a member of a given class. For example, we could declare that a face crack be a crack except a member of the class internal crack, more formally:

Face Crack OF TYPE Crack.
Face Crack IS NOT OF TYPE Internal Crack.

Here face crack inherits all the constraints of class crack, and excludes all the constraints of class internal crack.

3. **Any Attribute:** This requires a specific value for a specific attribute of an object. Any attribute is a constraint when specifying a goal. This reduces the search space of objects, and will only consider objects with that attribute. For example **FIND object x**

WITH intensity = 100. This would constrain the algorithms for selecting only object x with its intensity = 100. Numeric values can also be compared with greater then or less than.

4. Range: This constraint works on algorithms that use variables as parameters. A variable can be considered to lie within a certain range, e.g. **Minimum Threshold = 10, Maximum Threshold = 100.** This type of limit reduces the number of computations and mutations an algorithm would go through if we know the range the threshold works well within. This allows the replanner to consider only thresholds within this range.

It can be seen that constraints are a major part of the system, they aid in efficiency, both in terms of search time, and quality of output. However, the collection of such constraints may not be as complete as possible due to the complexity of algorithms and objects. It is for this reason that the Visual Planner tries to propagate and inherit as many constraints as possible given the contextual knowledge.

5.7 Domain Independence

Since we view domain-independent search control as necessary in complex domains, the Visual Planner provides for this in several ways. This chapter has described several ways that properties of the domain can be used to control the automatic search, planning and execution. The domain knowledge and contextual knowledge available are used by the system to control the search for a correct solution, in different modules in the system. These constraints make the whole process quicker when producing a plan.

These domain constraints are used to select appropriate algorithms as they are needed in the current plan. This permits the plan to shift focus easily among alternatives, which cannot be done in systems that use a backtracking algorithm alone.

5.8 Comparison with Other AI Planning Systems

A planning system must provide at least four of the features in table 4.1 (chapter 4) to make it a planner. In particular, an AI planning system is domain-independent, supports hierarchical planning, and permits nonlinear plans. Planning is viewed as a search through the space of operator applications and plan orderings.

Planner	NonL	Hier	Var	Const	Repln	DI
STRIPS						#
HACKER						#
ABSTRIPS		#				#
NOAH	#	#	#			#
NONLN	#	#	#			#
DEVISER	#	#	#	#	#	#
MOLGEN	#	#	#	#		
Visual Planner	#	#	#	#	#	#

Key: NonL = Non liner, Hier = Hierarchical, Var = Variables, Const = Constraints, Repln = Replanning, DI = Domain Independent

Table 5.1 Features of existing systems compared to the Visual Planner.

Sacerdoti's NOAH (Sacerdoti 77) and Fikes and Nilsson's STRIPS (Fikes 81) mark the beginning of this approach; their ideas inspired most planning research. Many systems developed this paradigm further, e.g. Tate's NONLN (Tate 77) and Vere's DEVISER (Vere 83). Table 5.1 summarizes the previously discussed capabilities (chapter 4) provided be each of these systems as compared to the Visual Planner within this project.

These systems are dedicated planners as compared to the Visual Planner which works only in a visual domain and thus the comparison is by no means used to show that the Visual Planner is better than these systems. The comparison of features is used to show that the Visual Planner is truly an adaptive, domain-independent planning system. These planning systems have other features that far surpass the Visual Planner's capabilities.

Since STRIPS solved such a simple problem, it technically does not fit the definition of an AI planning paradigm. However, it marks the beginning of this approach in AI planners. STRIPS restricted itself to linear plans at the same level of abstraction. These restrictions greatly reduce the planning problem, but also make the representation so weak that it is not useful for practical problems. ABSTRIPS (Lifschitz 87), a descendent of STRIPS did permit hierarchical planning, but inherited the other limitations of STRIPS. It kept track of abstraction levels by assigning each predicate name a level number, and then formed a complete plan based on all predicates with less then a given level number.

The first planner to introduce multiple goals was the HACKER system (Sussman 75). The planner was still linear, but this research showed that interaction between goals can be complex. HACKER was initially forced to produce incorrect plans for such problems, and it corrected these by having a plan modification technique for fixing bugs in its plans.

NOAH was the first system to qualify as an AI planner under the definition, since it produced nonlinear, hierarchical plans, while avoiding some weaknesses of linear planners such as HACKER. NOAH could not avoid producing incorrect plans because of the complexity introduced by non-linearity. The major limitation in NOAH was that it did not backtrack so could only find a solution if it happened to guess correctly at every choice point within the hierarchy.

NONLIN further extended NOAH, and the most important addition within this system was backtracking to modify already created plans. DEVISER added temporal reasoning capabilities to the NONLIN planner. While DEVISER did not allow general constraints, it did permit specification of temporal constraints. This can be very advantageous for real time applications, however, this may be at the cost of efficiency.

MOLGEN (Stefik 81) is a planner that is domain-specific for planning experiments in molecular genetics. However, it is important in the history of planning as it introduced the use of constraints.

5.9 Summary

In this chapter we explored a new system concept and its building blocks for the design of a scenario and domain independent, imaging segmentation system (the Visual Planner).

We described how initial enhancement plans are produced, and how such plans may be changed by monitoring the world using domain rules. We also described how initial segmentation sub-plans are selected and changed to achieve a given sub-goal.

The main advantage of the Visual Planner is its ability to react to new or unexpected events. In many cases it is able to retain most of the plan and change the plan to avoid problems caused by these unexpected events. It cannot solve difficult problems involving drastic changes to the expected state of the world, whatever the domain may be at the time. However it does handle many types of small errors (bad lighting, more noise, etc.) that may crop up frequently in the domain in which it is working.

Chapter 6
A Working Design for the Visual Planner

6. Introduction

This chapter describes the design and specification of the Visual Planner and the modules used within it. It attempts to give the reader a good understanding of the system, the structures it uses, and how we use a planner to select appropriate algorithms and thresholds within an adaptive system.

The Visual Planner operates in three modes: image processing mode, learning mode, and goal specification and detection mode. The image processing mode allows the user to become familiar with the image processing functions implemented within the Visual Planner. It has no influence on the Visual Planner, but will give the user an idea of how different algorithms have different effects on images. In the learning mode the system extracts knowledge from the image, objects, and the user to orchestrate an initial enhancement and segmentation plan for each object. The initial plan is developed over a number of algorithm and threshold changes based on performance measures given by the user and those calculated by the execution monitor. During the detection of objects the planner monitors the images to see if any unexpected events have occurred within the image (change in lighting conditions, new algorithm added by the user). These events may be initiated by the execution monitor or the user. The Visual Planner will then try to alter the plan if necessary by replanning to reflect these changes.

6.1 The Visual Planner's Operational Modes

The Visual Planner has two main modes of operation, the learning mode and the goal specification and detection mode. All AI/expert systems must be trained in some way, and then be able to use the acquired knowledge to achieve their goal. The Visual Planner is no exception, and thus learns the necessary information needed from the image and interactively from the user (see chapter 4). Other knowledge such as the image processing knowledge is explicitly written into the code, but can easily be taught to the system (see future work).

6.1.1 The Visual Planner's Learning Mode

The Visual Planner needs to acquire knowledge about the domain it will be working in. This is done in a number of ways: characteristics from the image are extracted, characteristics from the objects are extracted, and general knowledge from the user is acquired. By this we mean that the objects presented to the Visual Planner are classified as being liner, geometric, circular or a blob. The image processing primitives are embedded within the system, however, constraints are used to limit the choice of the image processing primitives and thresholds used by the Visual Planner.

One of the inputs to the system is the information about the scene and image. Image matrices are used to characterise the scene. Image matrices form an essential part of any multi-scenario automated image processing system. The image matrices are divided into two categories: target matrices, and scene matrices.

Each of these metric categories have different influences on different steps in the Visual Planner. For example target matrices have more relevance to the segmentation process as opposed to scene matrices which initiate the image enhancement algorithms. The Visual Planner uses a defined set of matrices (ATRWG 85), which are measures that quantify local and global characteristics of an image.

Different images and targets have different characteristics. For each of these situations a plan is made. The rules for planning are based on the contextual constraints and the goal of the segmentation given by the user. The way the rules for planning are fired, depends on the image matrices. Therefore, the system behaves similarly in similar situations. If the image matrices change, the plan will be forced to change to reflect this.

6.1.2 The Visual Planner's Goal Specification and Detection Mode

Once the system has acquired the knowledge necessary, i.e. image and object characteristics, we can proceed to specify a segmentation goal. This goal can be at different levels of abstraction, e.g. **Find All Where Intensity < 100** or more specifically **Find Surface Crack**. Thus, each goal must be split into sub-goals by the sub-goal designator. Each sub-goal will then correspond to a particular object within the scene, and thus a segmentation sub-plan will be created for that object.

After the Visual Planner has orchestrated a sub-plan to detect the relevant object or target, it automatically scans the images with the image processing primitives selected within the sub-plan. If the goal has been achieved then the sub-plan is stored in the algorithm design knowledge base.

This process, however, can be interrupted by events that occur during the execution of the plan. These events can be both automatic and interactive. The automatic events occur when the planner recognises a problem in the domain. This usually occurs if the image matrices change in any way (such as lighting, nose, unexpected image) or the Visual Planner has failed to achieve the goal. Reactive events occur when the user interrupts the execution and asserts a new situation into the process. This could be any number of things such as: the user is not satisfied with the results (i.e. the goal has succeeded but not to the satisfaction of the user), new image processing primitives are added or updated and the user wants the Visual Planner to take these into account. In any case the replanner is called and the appropriate actions are carried out depending on the problem.

In order for The Visual Planner to see if it has achieved the sub-goal some form of matching or registration must take place. This is carried out by the execution monitor. The execution monitor matches the segments produced by the sub-plan to the object characteristics database. It is to this part of the system that high-level image understanding procedures may be added. The present Visual Planner does not incorporate such procedures, but instead uses a simple matching technique by comparing the segments produced to the local features of an object, such as average intensity, size, corners, edges and area in the object characteristic database. This is a simplified method of that proposed by Bolles and Cain (Bolles 83) known as the *local-feature-focus* method. A measure of how certain the Visual Planner has achieved the sub-goal is also given by the execution monitor. For instance if we are looking for a cube, and the execution monitor gives a matching strength of *x*, and *x* is greater then some predefined threshold then it has achieved the sub-goal.

Selecting a certainty measure for a sub-goal will once again depend on the application domain. The threshold can be as low as 60% in consumer goods and as high as 95% in precision engineering. This threshold measure of certainty may be set by the user. The user must also decide what weighting each segment of an object will have up to 100%. For example a hole may have a specific intensity value within an image, and no predetermined size. Its contextual information is a blob, this means that more emphasis will be placed on finding it by intensity value rather then size. An example of specifying some weights is shown below:

Class		Hole		
Name		Surface Hole		
Contextual Information		Blob		
Attributes:				
	Average Intensity	100	**Weighting**	40
	Min Intensity	95	**Weighting**	20
	Max Intensity	110	**Weighting**	20
	Major Axis	20	**Weighting**	10
	Minor Axis	5	**Weighting**	10

When specifying weights, the user is helped by the contextual information. As in this example, a surface hole is a blob and therefore more emphasis is placed on average intensity values. However, for linear objects more emphasis should be placed on the number of lines, for geometric objects the size is more important and so on.

6.2 Image Characteristic for Domain Rules

Global knowledge about the image (average intensity, lighting conditions, signal-to-noise ratio and, background information) is very important in a domain independent imaging system. This type of knowledge can, by itself orchestrate a simple initial enhancement plan. So what type of global image characteristics are needed? and what can be extracted? In more complex domains detailed knowledge must be given if for instance the global characteristics change in every image. However, in industrial applications the images remain roughly the same. The background is fixed, lighting conditions are fixed, and camera geometry is known. It is therefore relatively simple to extract image characteristics from a given scene.

The Visual Planner extracts four types of characteristics from the scene: signal-to-noise ratio, appropriate thresholds, average intensity, and image background intensity. The signal-to-noise ratio is used to select an appropriate noise suppression algorithm if needed and, the thresholds are used for binarisation and digitisation, Average intensity is used to monitor domain change's form one image to another, and background intensity is used to extract the background or clutter from the signal. The algorithms that extract this information are described in turn and how it effects the image enhancement algorithm selection within the plan.

6.2.1 Noise Detection

Much progress in cleaning noise can be made by making some simple assumptions about the character of the noise and the nature of the uncorrupted image. It is, for instance, reasonable to assume that grey levels can be distorted randomly above and below their true values. However, there are circumstances in which this may not be the case. If the noise is known to behave in a different way then other techniques must be used to clean it. Without some analytic knowledge of its behaviour these algorithms cannot be implemented on all images. To acquire such knowledge needs some pre-processing of the image in its environment. For instance in an industrial inspection system there are many forms of *ambient noise* such as dust, industrial fog, and bad lighting conditions. To clean such an image we have to test the *image quality.*

6.2.1.1 Image Quality

We can define a clean image g(i,j) as being the average of a number of noisy images. The noise can then be defined as being the root-mean-square deviation of a noisy image n(i,j) from the mean g(i,j) image (Pratt 78). Figure 6.1 shows two histograms and a deviation graph of an image. Histogram 'a' shows an individual noisy image, histogram 'b' shows the average of eight similar images, and graph 'c' shows the deviation of the noisy histogram 'a' from the mean histogram 'b'. The graphs were calculated as shown below:

A histogram of an image is calculated by counting all pixels with a particular intensity value. The intensity values vary between 0 and some maximum grey value (0-255).

g(s) = Histogram g(i,j). Calculating the same for n(i,j) = n(s)

$sd[s]=|g(s)-n(s)|$

where sd[s] is the deviation at each intensity level. This gives us values above and below true values. These values are used as reference deviation histograms.

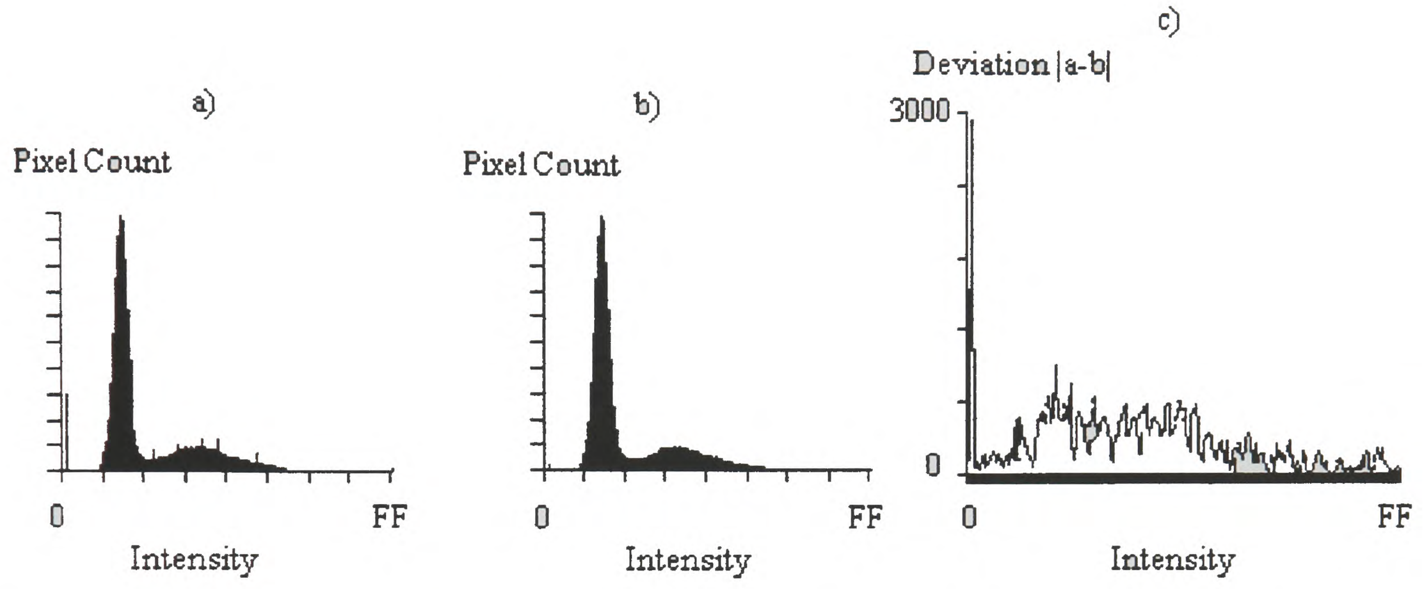


Figure 6.1 a) noisy histogram, b) average of eight images, c) deviation graph.

Such noise measurements can be acquired by the Visual Planner very easily. This knowledge can then be stored in the image characteristics database for use by the Visual Planner when selecting an algorithm for cleaning the noise.

6.2.1.2 Selecting a Noise Threshold

The Visual Planner computes both maximum and minimum noise thresholds and the standard deviation at all intensity levels, it is these values that are used to select an algorithm for noise reduction using the above method. The maximum and minimum threshold values are selected from the deviation graph in Figure 6.1c. The maximum peak is selected for the maximum threshold and the minimum peak for the minimum threshold value. The noise suppression algorithm selected must reduce the noise between these two peaks and reduce the noise over all the deviation histogram values. A measure of performance is calculated by counting the number of times the clean image deviation falls above or below the reference deviation histogram.

The Visual Planner uses two algorithms for noise suppression, the Mean Filter (Gonzalez 84, Pratt 78), and the Median Filter (Huang 78, Lev 76) (see Appendix C). The domain rules will select the algorithm (see domain rules below).

6.2.2 Thresholding

Thresholding an image is an important part of image processing. A grey scale image is binarised to extract, for example, the signal from the clutter. If the objects in the scene occupy a distinctive grey level range then they can be extracted by thresholding. The proper choice of the threshold is very important if one wants to extract the objects correctly. The Visual Planner selects an initial threshold using the knowledge of objects within the scene and the grey level histogram.

6.2.2.1 Threshold Selection

One simple approach to segmentation is to segment the grey-level image *G* at a value *T* to produce a binary image *B*. A threshold operation is defined as:

$$\begin{array}{ll} B_{ij} = 1 & \text{IF } G_{ij} >= T \\ B_{ij} = 0 & \text{IF } G_{ij} < T \end{array}$$

How do we select a single value for *T*? Usually, *T* is obtained from the histograms of the image. If we have a simple image such that the objects have a mean intensity value m_0 with a small variance and similarly for the background m_b, then if m_0 and m_b are sufficiently far apart such that they do not overlap, the histogram is said to be bimodal (Figure 6.2). In this case, the threshold value can be chosen to be the mid-value between these two peaks. This is often called the *mode method* (Prewitt 66).

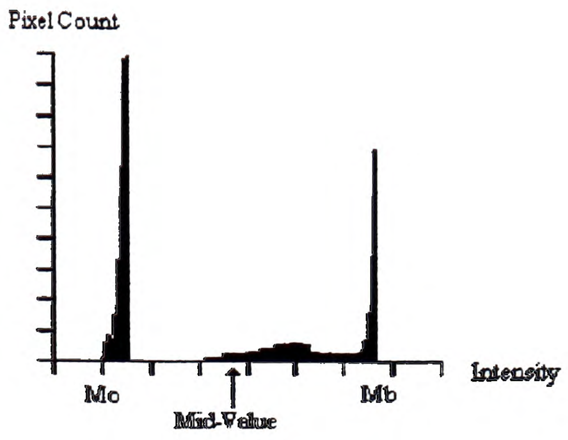


Figure 6.2 A bimodal histogram.

If there are N objects with distinct grey-levels, the image may be segmented by multiple thresholds:

$$\begin{array}{ll} R_{ij} = N & \text{IF } G_{ij} >= T_{N-1} \\ R_{ij} = N\text{-}1 & \text{IF } T_{N-2} <= G_{ij} <= T_{N-1} \\ \vdots & \\ R_{ij} = 1 & \text{IF } T_0 <= G_{ij} <= T_1 \\ R_{ij} = 0 & \text{otherwise} \end{array}$$

The desired object can then be thresholded out of the image R. Several methods have been developed for automatically selecting the threshold values which have been reviewed in (Weszka 78, Rosenfeld 82, Sahoo 88, Pun 80, Pun 81, Dove 62).

6.2.2.2 Visual Planner Threshold Selection

The Visual Planner thresholds the image in two ways as described above, first by using background information, if available or, secondly by using object information. The object information used is the average intensity of each object and its minimum and maximum intensity values to select multiple thresholds. Figure 6.3a shows a binarised image calculated using the mean intensity of the image and Figure 6.3b shows the same image if background knowledge is available.

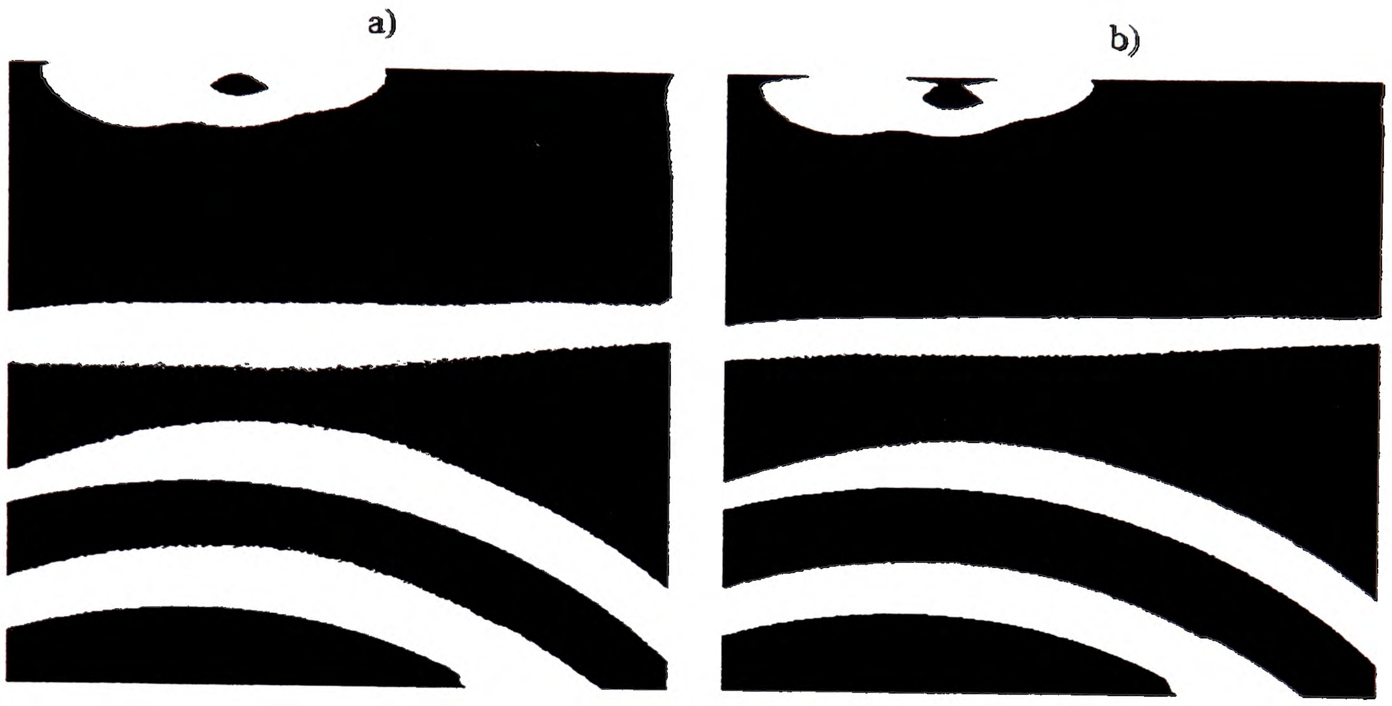


Figure 6.3a Image binarised using mean of image.
Figure 6.3b Image binarised using background to set threshold.

It can be seen that using the background knowledge, better results are obtained. The image is better represented with less blurring and fewer false detections.

If we are to use multiple thresholds then the image is subjected to each object intensity value threshold in turn. This causes a set of sub-images to be generated. These sub-images are then added and a binarised image is produced. However this is a time consuming process if many objects exist within the object characteristics database, therefore the Visual Planner sees if the image can be binarised using the histogram first. The Visual Planner checks to see if the histogram is bimodal, if it is, then it is split at the mid point between two peaks. This can be extended to use multiple thresholds on the histogram.

6.2.3 Background Information

Knowing the background in any image is a very important aspect of a visual system, where the background is any unwanted information (the clutter). If the background is uniform and static, i.e. its average intensity is equal throughout then we can give it this information interactively by teaching it the background. The background is taught to the Visual Planner by giving it samples of the background. A histogram for each sample is calculated at each stage. The histograms are then added and maximum and minimum threshold values are selected from this histogram, using its maximum and minimum histogram range values.

A problem arises however, when using only the background histogram information to segment an image. If the object intensity value overlaps the background intensity value as shown in Figure 6.4 then under-segmentation takes place losing part of the object. This situation must be avoided at all costs, we therefore use the object intensity value to segment the image. This gives us over-segmentation, however, other object attributes such as size, area, and lines can be used in detecting an object. In any case the Visual Planner will not be able to produce a full strength match, and false targets may be picked up. The Visual Planner's match strength will also depend on its contextual information. For instance if the object of interest is geometric and has well defined features then the overlap will not generate a large problem, as matching in this case is determined mainly on size, lines, and corner points.

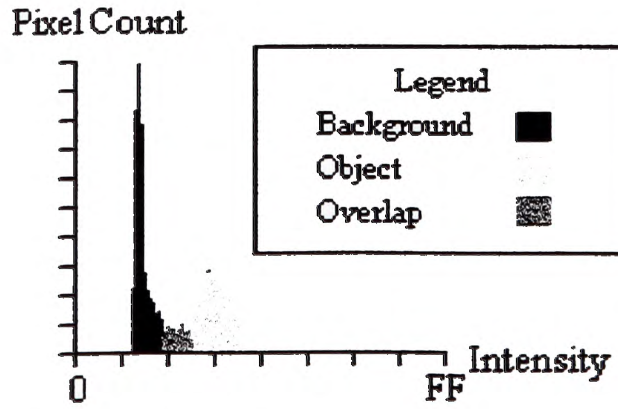


Figure 6.4 Histogram showing object and background overlap.

6.2.4 Average Intensity of an Image

If the image average intensity value changes dramatically when the background should be uniform and static, and does not comply with the object's intensity values, then we can say there has been a change in the environment, a new situation has occurred and we may take appropriate actions to replan using the domain rules.

An average histogram for all images presented to the system both in learning mode and detection mode is shown below (Figure 6.5). The histogram shows the mid value for all images and the overall range of the histogram, both in width and height.

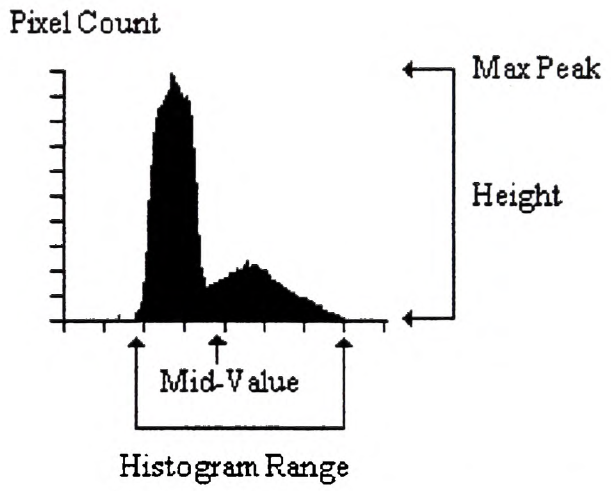


Figure 6.5 Average histogram of all images presented to the Visual Planner.

6.3 Domain Rules

Domain rules are used to select noise reduction and enhancement algorithms from the domain knowledge acquired by the Visual Planner. The domain rules carry out two tasks: a) they orchestrate an initial enhancement plan, and b) they are used to initiate replanning actions if the domain knowledge changes. Each image is scanned as it is presented to the Visual Planner to detect these changes.

6.3.1 Domain Rules for Noise Suppression Algorithm

The Visual Planner starts by selecting the first available noise suppression algorithm (the mean filter). A noisy image n(i,j) which has been previously used in calculating the clean reference image g(i,j) is then subjected to this filter, its deviation is then calculated and compared to the reference deviation of the noisy image. This yields three types of information: a) a new maximum noise peak, b) a new minimum noise peak and, c) a measure of its performance.

We can calculate a measure of performance by comparing each intensity value in the two deviation graphs as shown below:

s_r = number of pixels at grey level r, $0 <= r <= M$
M = number of intensity levels
sd(s) = reference deviation (clean image to noisy image)
g(s) = new deviation graph (filtered image to clean image)

```
WHILE r <= M
{
        IF $g(s_r) < sd(s_r)$ THEN performance = performance + 1
        $s_r = s_r + 1$
}
```

This information is stored by the Visual Planner, and the next algorithm (median filter) is subjected to the same tests. The domain rules below show how the comparison of these results select the algorithm. The rules return an overall measure of its performance. The algorithm with the highest overall performance measure is selected.

```
DRule (1)      IF max_clean_peak < max_noise_threshold
               THEN measure=measure+1
```

```
DRule (2)    IF min_clean_peak < min_noise_threshold
             THEN measure=measure+1

DRule (3)    IF deviation_error < 50%
             THEN measure=measure+1
```

The maximum performance measure any noise algorithm can have using these rules is three. The deviation error must be under 50% this means no noise has been added, but also no noise has been subtracted. However, we do not know exactly how much noise has been cleaned for each algorithm. We therefore compare the error measure (magnitude of error for each pixel value) produced by each algorithm and select the one with the least error. If their overall performance measures are equal then the rules below are fired. These rules select the algorithm with the minimum error measure in the range of 0-number of intensity levels.

```
DRule (4)
    IF noise algorithm 1 performance measure = noise algorithm 2 performance measure
    THEN {
            for each algorithm CALL DRule (5)
            CALL (algorithm selected)
            }

DRule (5)
    IF error_measure < min_error_measure
    THEN {
            min_error_measure=error_measure
            algorithm_selected=current_algorithm
            }
```

6.3.2 Enhancement Domain Rules

Enhancement is a method that attempts to improve the appearance of an image i.e. make the features of an image more apparent for image processing routines, and can provide additional visual information (Andrews 74). The Visual Planner uses histogram equalisation (Gonzalez 87) to highlight features. Histogram equalisation is a nonlinear operation, it involves pointwise mapping of each intensity value to another globally.

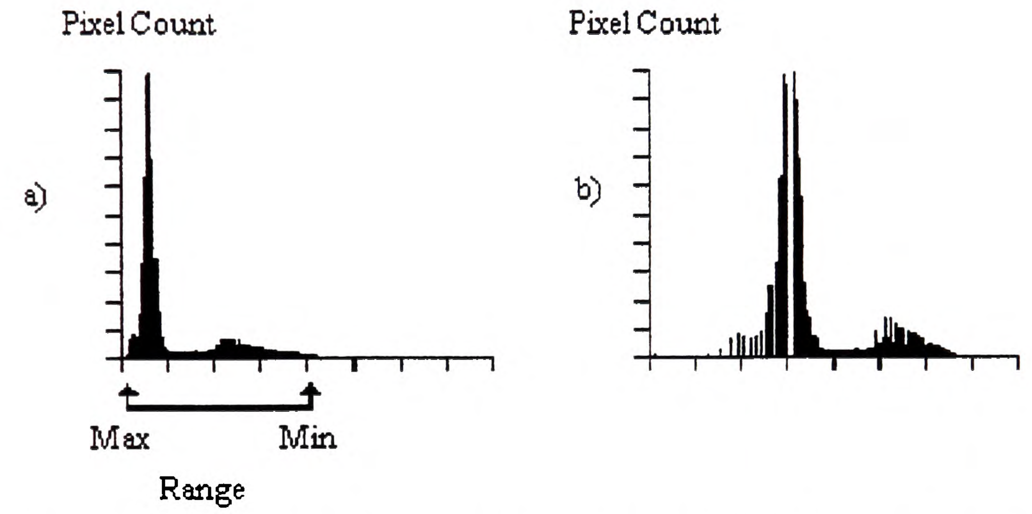


Figure 6.6 a) image histogram with low contrast, b) histogram equalised.

The idea behind this technique is to derive a histogram from an image by recording the number of pixels at particular grey levels. The histogram often yields useful information about the nature of the image. If there is a bias towards the lower or higher intensity grey levels, we could rightly deduce from this that a more equitable sharing of the pixels among the grey levels would affect the image appearance. Figure 6.6a shows a histogram with a low intensity value. Figure 6.6b shows the same histogram after the enhancement operation. The domain rules that achieve this operation are shown below:

DRule (6)

IF Histogram Range Max Value < Mid Histogram Value for all images

THEN Call (histogram equalisation)

DRule (7)

IF Histogram Range Min Value > Mid Histogram Value for all images
THEN Call (histogram equalisation)

The domain rules above check to see if the current image histogram is bunched below or above the average histogram mid-value position. If so, the histogram is equalised to produce a better spread of pixel information more in line with the average histogram of all images. The domain rules that enhance the image are very important within the Visual Planner, as lighting conditions may change the image average intensity values.

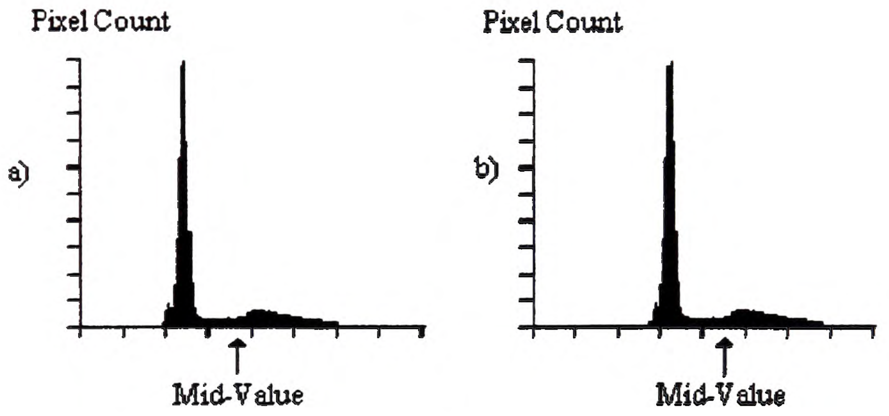


Figure 6.7 a) Normal image histogram. b) Same histogram with constant light change.

The Visual Planner also provides rules if the histogram range has moved to the left or right of the average mid-value histogram. This means that a constant light source has been added or subtracted from the image globally. Figure 6.7a shows the original histogram of an image and Figure 6.7b shows the same histogram with a global constant light change.

DRule (8)

IF Histogram Range <= Histogram Range Value for all images **AND**
IF Mid Histogram Value > Mid Histogram Value for all images
THEN Call (Darken Histogram(Mid Histogram Value for all images - Mid Histogram Value))

DRule (9)

```
IF Histogram Range <= Histogram Range Value for all images AND
IF Mid Histogram Value < Mid Histogram Value for all images
THEN Call (Brighten Histogram(Mid Histogram Value for all images - Mid Histogram Value))
```

Rule 8 darkens the image, and moves the histogram to the left, whereas rule 9 brightens the image, and moves the histogram to the right. These transformations are global changes to the image. They change the brightness of an image.

6.3.3 Average Intensity Change Domain Rules

These rules monitor the histogram changes within an image. Rule 10 sees if the histogram range has increased for some reason. It could be that a new object has been encountered or more light from one source has been added, or some type of unknown noise has been introduced. In any case this type of problem cannot be solved by the Visual Planner and thus user intervention is required. The 10% is added to the histogram as a relaxation figure. Selecting a relaxation figure is a matter of trial and error depending on how many false alarms are raised. This may be increased or decreased any time dependent on the accuracy required for the application domain. Rule 11 Monitors' the height of the histogram, once again if the height has changed then a new object has been added within the histogram range or another event has occurred, control is passed to the user to resolve the situation. The user may ask the system to ignore the event or halt the process, and add more information to the Visual Planner.

DRule (10)

```
IF Histogram of ALL Image Range + 10% < Current Histogram
THEN CALL (USER)
```

DRule (11)

IF Histogram of ALL Images Max_Peak +10% < Current Histogram Max_Peak
THEN CALL (**USER**)

These rules can be applied in any order, except for rules that change the image attributes such as histogram equalisation. If this rule is applied at the beginning then the rest of the rules cannot be fired as equalisation produces unpredictable histograms. If the rule is fired at the end then a corrupt image is subject to the rest of the rules, once again rasing false alarms and unpredictable transformations on the image. It is therefore better to apply this rule at the beginning of the rules, this will bring the histogram close to the average histogram, if information has not been lost.

6.3.4 Threshold Domain Rules

These rules binarise the image according to the histogram. If the histogram is bimodal then we can split the image according to the histogram, thus separating the regions of interest. If however the histogram is not bimodal the image is split using multiple thresholds selected form the target characteristics.

DRule (12)

IF Histogram is BIMODAL
THEN CALL (Binaries Image, Threshold=Max Background Intensity Value)

DRule (13)

IF Histogram is **NOT** BIMODAL
THEN CALL (Multiple Threshold Selection Function)

These rules just separate the foreground from the background, and are used to reduce the search space for objects within an image, as we need only consider objects in the foreground.

6.4 Sub-Goal Designation

Before we can continue to produce segmentation sub-plans, we must first produce the sub-goals from the overall goal. As has been stated the goal can be given at any level of abstraction, with many types of search criteria. The sub-goal designator splits the main goal into sub-goals according to the objects that meet the search criteria. Each sub-goal corresponds to one object in the object characteristics database, and can be best illustrated using some examples as shown below:

1. Goal Find Cubes:

This is the main goal to find all cubes. In this case cube is the main class of object, all objects with the class cube will be extracted from the object characteristics database. For each object within the class cube, its domain information is extracted and object characteristics such as threshold values. For each object a sub-goal is then created. Find cube would then be separated into:

Sub-Goals: Find Cube *x*, Find Cube *y*, Find Cube *z*, etc.

The cube's attributes, and contextual information are extracted for use by the Visual Planner for producing initial plans. If no objects exist within the objects characteristics database the goal is invalid.

2. Goal Find Intensity Value <= 100:

Here no objects are specified but we are interested in all objects that have an average intensity value of less then or equal to 100. The object characteristics database is searched for objects that have this attribute and they are once again extracted with their contextual information. Sub-goals are then deduced depending on the objects' contextual information and other attributes.

As stated earlier any attribute that exists for an object can be used as a goal, we can even search for individual objects, by making the goal very specific, such as giving the name of the object. Here the goal will become the sub-goal as only one object of that name will exist in the object characteristics database. The sub-goal designator also checks that a goal can be met, i.e. there exists an object in the object characteristics database that matches the search criteria. This process can be shown in Figure 6.8.

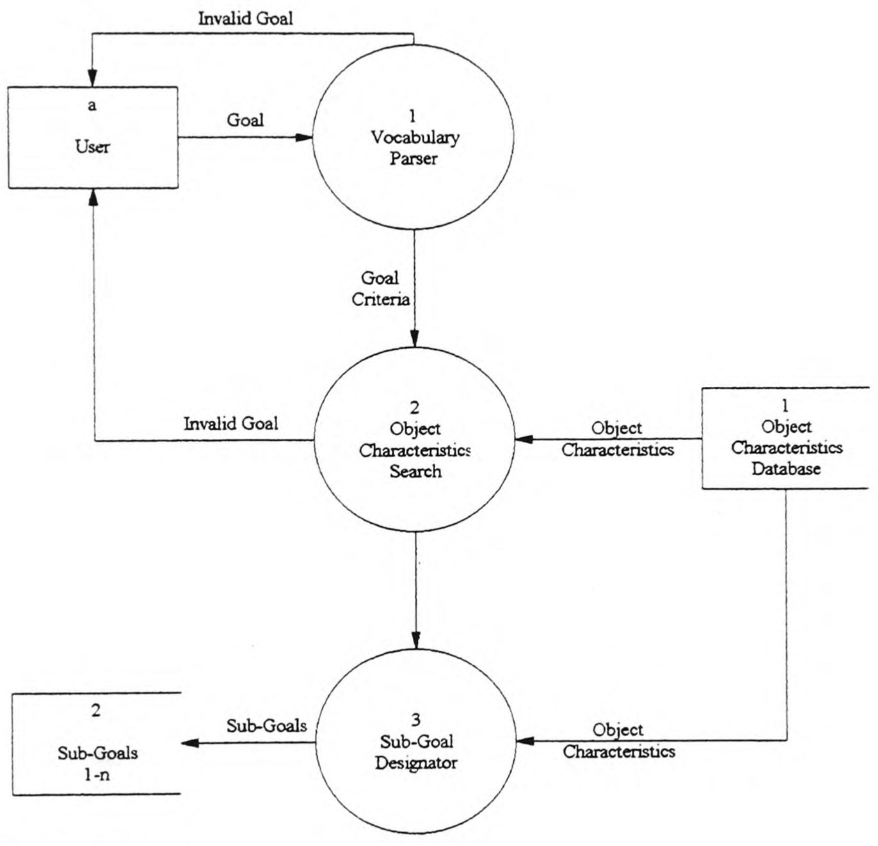


Figure 6.8 Sub-goal creation.

The vocabulary parser takes the goal specified by the user and makes sure that the goal is valid semantically, e.g. **Find X Intensity <100 AND Intensity >100**. We cannot have an object X that has an intensity value < 100 AND > 100. This is an invalid goal. Also the Visual Planner's interface prevents illegal syntax, as the vocabulary is limited, it thus becomes an easy task to check for errors.

6.5 The Initial Segmentation Planning Process

Armed with the knowledge necessary and a user defined goal we may start the planning process. It has been stated that for each sub-goal an initial sub-plan is created, This can be inherited or created.

Each sub-goal is passed to the Visual Planner along with the enhanced image to be segmented. Each sub-goal presented to the system will have to be achieved, it is therefore important to have the objects to be segmented in the image, and that the object has been registered in the object database before a sub-plan can be formed.

We can also have more then one object in the image, that is, we could have three types of cube, by specifying **Find all Cubes** as the main goal, a sub-goal will be created for each cube in the object characteristics database. This saves time compared to creating sub-plans one by one. One advantage of the Visual Planner is that it can inherit sub-plans for objects, so if we specify **Find all Cubes**, the first sup-plan for cube *x* will be created, and the rest (cube *y*, cube *z*, etc.) will be inherited with appropriate threshold values changed, as long as there are no exception classes. The planning rules specified below show how initial sub-goals are changed into initial sub-plans:

PRule (1)

```
    IF Sub-Goal.Object_Class = Algorithm_Design.Object_Class
    AND Sub-Goal.Object_Class.Exception <> Algorithm_Design.Object.Class
    AND Sub-Goal.Object_Class.Contextual.information=Object_Class.Name.Contextual.information
    THEN {
                Inherit Sub-Plan from Algorithm Design Knowledge base
                Change Threshold Values
         }
```

Rule 1 will attempt to inherit an existing sub-plan, if one exists in the algorithm design knowledge base. This is selected based on the class of the object. It will not inherit a sub-plan if the exception class has been set e.g.,

Class	Cube
Name	Cube *x*
Exception	Cube *y*

Here Cube *x* will not inherit any sub-plan from Cube *y*, if one exists, it will instead attempt to inherit another sub-plan of class Cube, if one exists. If the classes match and there is no exception constraint, we then look at the contextual knowledge of the object that is to inherit the sub-plan, and try to match it to the object that will give the plan. If they are the same then Cube *x* will be able to inherit the sub-plan. The sub-plan thresholds are then changed according to the new object's intensity values. The new sub-plan is then executed to see if it has achieved the sub-goal (see execution monitor below).

If the object extracted from the sub-goal cannot inherit a plan, then one must be created, using the contextual information of the object and its threshold values (object attributes). This is done by the Visual Planner using the rules below:

```
PRule (2)
    Initial Algorithm = Call Initial-Algorithm-Selection()
    IF Initial Algorithm = NULL THEN
        ERROR "Cannot Initiate Sub-Plan"
    ELSE
        CALL PRule(3)
```

```
PRule(3)
    IF Initial Algorithm.Contextual Information = Sub-Goal.Object Name.Contextual Information
    THEN
        {
            CALL Make_Sub-Plan (Initial Algorithm)
            Set Threshold Values
            Save_plan In algorithm table
        }
    ELSE
        ERROR "No plan can be created for object"
```

Rule 2 looks in the image processing and segmentation database to see if an algorithm exists with the same contextual information that matches the contextual information of the object. Initially only one algorithm is selected which has a priority level 1. This algorithm is then passed to Rule(3). Rule(3) then checks to see that the initial algorithm's contextual information matches the objects contextual information. If it does then procedure *Make_Sub-plan (Initial algorithm)* is called, which makes the sub plan for the object by extracting the succeeding algorithm form the current algorithm. This is done until no succeeding algorithm exists, and that all possible sub-plans have been created for a given object.

The algorithm below is used for selecting an initial algorithm for each sub-plan created known as the initial algorithm selection planning algorithm:

Variables used:

Min-C-I-Count:	A count that holds the minimum number of contextual information items found in any list of contextual information items.
Alg-C-I-Count:	A count that holds the number of contextual information items in the list of contextual information for a given algorithm.
Weighting:	Used to select algorithm according to best weighting factor.
Initial-Algorithm:	Initial algorithm selected for each sub-plan.

Preconditions for each initial sup-plan algorithm:

Min-C-I-Count = 5	Initially this count is set to 5 as their are four types of contextual information i.e., Linear, Geometric, Circular and, Blob. This makes sure that at least one algorithm is selected.
Priority-level =1	Only consider priority level one algorithms.
Weighting = 0	Make weighting for each algorithm = 0.
Initial-Algorithm = Null	

```
FOR each segmentation algorithm in database = Priority-level DO
{
        Rule 1: IF Algorithm.Contextual-Information = Object.Contextual-Information THEN
                        Weighting = Weighting + 1
                ELSE
                        Break
        Rule 2: IF Alg-C-I-Count < Min-C-I-Count THEN
                {
                        Weighting = Weighting + 1
                        Min-C-I-Count = Alg-C-I-Count
                }
                ELSE
                        Break
        Rule 3: IF Weighting = 2 THEN
                        Initial-Algorithm = Current-Algorithm
}/* End FOR*/
Return (Current-Algorithm)
```

For each initial algorithm selected according to the sub-plan an initial segmentation plan must be created. This algorithm is called Make-Sub-Plan and is shown below:

Variables used:

Current-Algorithm:	Algorithm currently being considered by the planner.
Next-Algorithm:	Algorithm that will be appended to the plan.
Succeeding-Alg-List:	Succeeding algorithm list.
Succeeding-Alg:	Succeeding algorithm under consideration by planner.

```
Precondition per Initial-Algorithm:
Current-Algorithm=Initial-Algorithm
Weighting = 0
Min-C-I-Count = 5

WHILE Current-Algorithms has Succeeding-Alg-List DO
{
          FOR each Succeeding-Alg in Succeeding-Alg-List DO
          {
          Rule 1:  IF Succeeding-Alg EXISTS in Sub-Plan THEN
                             Break
          Rule 2:  IF Succeeding-Alg.Contextual-Information = Object.Contextual-Information THEN
                             Weighting=Weighting + 1
                   ELSE
                             Break
          Rule 3:  IF Succeeding-Alg.Alg-C-I-Count < Min-C-I-Count THEN
                   {
                             Weighting = Weighting +1
                             Min-C-I-Count = Succeeding-Alg.Alg-C-I-Count
                   }
                   ELSE
                             Break
          Rule 4:  IF Weighting = 2 THEN
                             Next-Algorithm = Succeeding-Alg
          } /*End FOR*/
          Current-Algorithm = Next-Algorithm
          APPEND (Current-Algorithm, Sub-Plan)
          Min-C-I-Count = 5
} /*End WHILE*/
```

If any part of the sub-plan requires threshold values, then these are set according to the object and background intensity values. This sub-plan is then saved in a table and the rules are recursed to see if another sub-plan can be made for the object, and so on until no more plans exist. All possible plans are created at this stage for efficiency reasons when replanning. The sub-plan that is initially selected will be the one that has the lowest priority sum for all algorithms within that sub-plan. This ensures that a short plan is selected. The initial planning process is shown in Figure 6.9.

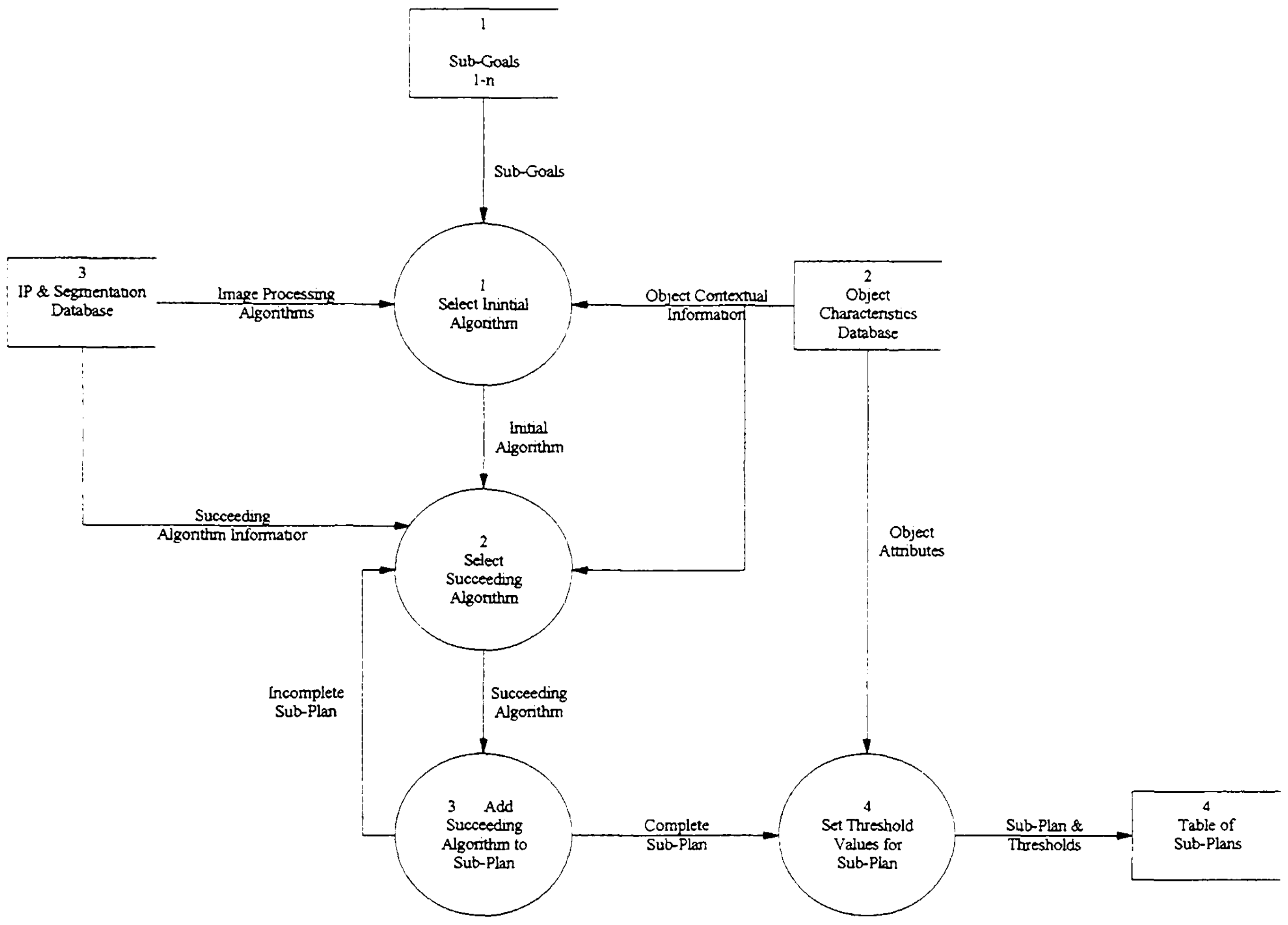

1
Sub-Goals 1-n
Sub-Goals
3
IP & Segmentation Database
Image Processing Algorithms
1
Select Inintial Algorithm
Object Contextual Information
2
Object Characteristics Database
Initial Algorithm
Succeeding Algorithm Information
2
Select Succeeding Algorithm
Object Attributes
Incomplete Sub-Plan
Succeeding Algorithm
3 Add Succeeding Algorithm to Sub-Plan
Complete Sub-Plan
4
Set Threshold Values for Sub-Plan
Sub-Plan & Thresholds
4
Table of Sub-Plans

Figure 6.9 Sub-Plan Creation.

After a number of plans have been created for one object the initial sub-plan selected will be executed to see if it has achieved the sub-goal. If it has, then the sub-plan is added to the algorithm design knowledge base. This is done by the execution monitor.

6.6 Execution Monitor Functions

The execution monitor carries out a number of functions that are summarised below:

1) Executes the sub-plan on the image.

2) Gives a performance measure for the sub-plan. i.e., found object x with 95% matching strength using simple object recognition.
3) Initiates a replan if the sub-goal fails.
4) Gives a performance measure of the overall plan functionality (see execution and monitoring system chapter 5).
6) Interrupts the user if a sub-goal cannot be achieved.

6.6.1 Executing a Sub-Plan

After the image has been enhanced by the domain rules a segmentation sub-plan is executed by the execution monitor. The algorithms within the sub-plan are executed and a symbolic representation of the image is extracted. The feature vectors extracted will depend on the contextual information of the object.

6.6.2 Matching Feature Vectors to the Object Database

Once the plan is executed, we wish to see if it has achieved the sub-goal. It is therefore necessary to match the features extracted by the sub-plan to the features in the object database. As we know what object we are looking for, we can go directly to that object in the object database. The local-feature-focus method (Bolles 83) is used to match features in the scene to the object database. It is at this stage that any matching algorithm can be chosen to match features in the object database.

The local-feature-focus algorithm starts by trying to match intensity areas that match the object, this would be the object size and area (average intensity, max intensity value, min intensity value, area, major axis, and minor axis). If an area that meets these criteria is found, then we try to find vertices, lines, facets, radius, and centre dependent on the contextual information of the object, i.e. what has been modelled. If a match is found that meets all the criteria then the object has been found, and a 100% matching strength is returned and the sub-plan is stored in the algorithm design knowledge base. However, if one or more of the criteria fail then the image is scanned again to see if another area exists. If it does then this area is subjected to the same tests. If no area is found that matches the criteria 100% then a reduced matching strength is returned, dependent on the number of attributes matched. If this is above the performance threshold value specified by the user then the sub-plan is stored in the algorithm design knowledge base. If not, a replan (see replanning below) is initiated until the sub-goal is achieved or fails completely.

If the sub-goal cannot be achieved then the user is interrupted and informed which sub-goal has failed. The user is then asked if the system should continue with the other objects. If so, the overall performance measure is reduced and the process starts again for the next sub-plan.

6.7 The Replanner and Replanning Actions

It is the replanners job to resolve any problems within a sub-plan. As there are a number of different problems encountered there are a number of replanning rules that must be fired depending on the problem.

If a sub-goal fails to recognise the object it was intended to, then a number of actions are executed to remedy this problem. A backtracking technique is used to alter the sub-plan, this technique is good for an image processing system, as the sub-plan need not be executed form the start if a change has occurred, but only form the new branch. This saves time executing the image processing and segmentation algorithms. By using backtracking, only in the worst case will the whole sub-plan have to be executed.

```
BACKTRACK
IF Sub-Plan Failed THEN {
        Current-Algorithm = Preceding-Algorithm
        REMOVE(Used-Algorithm from Current-Algorithm.Succeeding Algorithm List)
        IF Current-Algorithm.Succeeding Algorithm List = NULL THEN
                        Call BACKTRACK
        ELSE{
                Call Make_Sub-Plan(Current-Algorithm)
                APPEND(New Sub-Plan, Old Sub-Plan)
                EXECUTE (Sub-Plan)
        }
}
```

The algorithm Backtrack tries to find an alternate sub-plan if a goal has failed for some reason. It starts by backtracking to the preceding algorithm and placing this algorithm in the variable Current-Algorithm. It then removes the succeeding algorithm that was used in the sub-plan from its succeeding algorithm list. If there are no more succeeding algorithms in the Current-Algorithm Succeeding Algorithm List, Then this path is exhausted and the algorithm is recursed, backtracking again to the previous algorithm. If however, there is a succeeding algorithm, Make_Sub-Plan(Current-Algorithm) is called and a new section of sub-plan is created.

This new section is then appended to the old truncated section, and the sub-plan is executed once more. If it fails again the whole process is repeated. Figure 6.10 shows the original path of a sub-plan (1,2,4,6) all the possible sub-plans that can be created using the backtracking algorithm from this network is shown in Table 6.1.

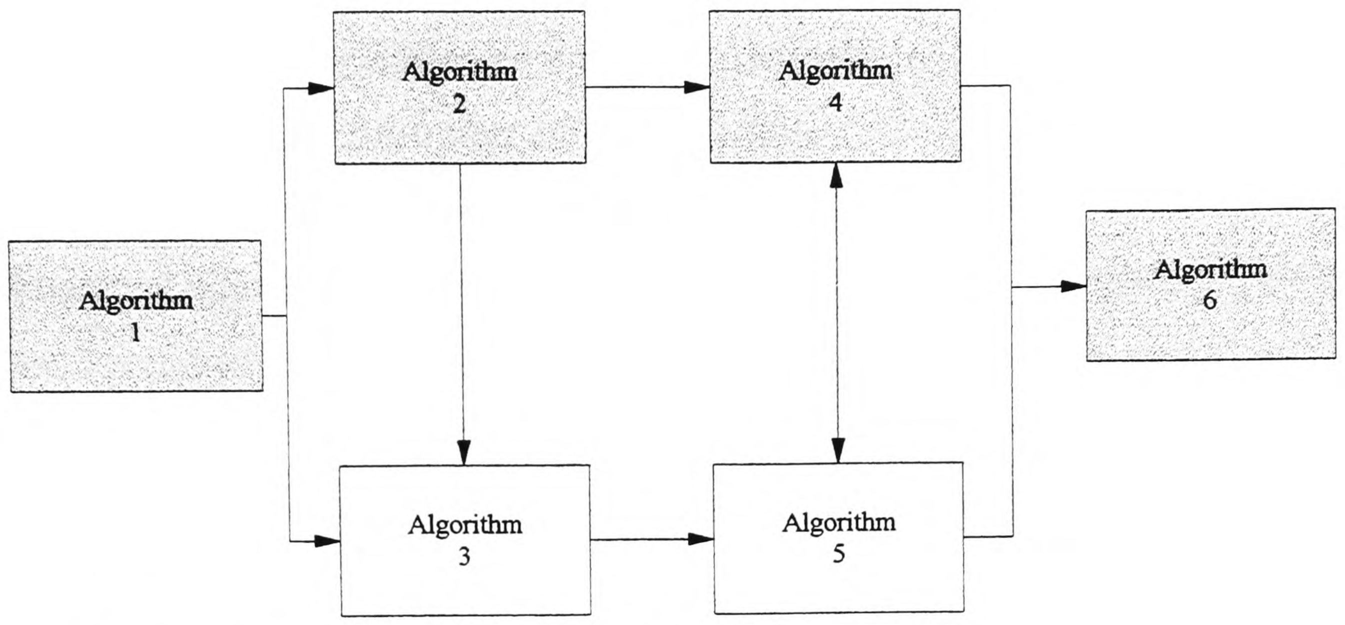


Figure 6.10 Algorithm network with original path (sub-plan).

Sub-Plan ID	**Sub-Plan**
1	1, 2, 4, 6
2	1, 2, 4, 5, 6
3	1, 2, 3, 5, 6
4	1, 2, 3, 5, 4, 6
5	1, 3, 5, 6
6	1, 3, 5, 4, 6

Table 6.1 Possible sub-plan creation by replanning.

```
Threshold Algorithm
        IF No Alternative Algorithm Exists AND
        IF Threshold_Change_Count <20 AND
        IF Threshold < Max_Threshold Constraint AND
        IF Threshold > Min_Threshold Constraint
        THEN RETRY (Algorithm)
        ELSE Object Not Found
```

The threshold algorithm is a set of conditions that refines the threshold of the algorithm if a threshold exists, for instance it may change the threshold of an edge detector in the positive and negative directions, both directions being tested in turn. This could be done in parallel to save time. It is limited by a number of constraints, first the number of changes that may take place on an algorithm, and secondly if the threshold change is within the limitation constraints of the algorithm. This stops the algorithms from changing the threshold forever. This algorithm is only applied to sub-plans if the backtracking algorithm has failed to achieve the sub-goal. It could have been done during the planning process, but once again this would have been inefficient, and reduced the effect of the backtracking technique used. Also, the thresholds selected initially form the domain knowledge produce a best first approximation.

```
Failed
        IF No Alternative Algorithm Exists AND
        IF No Threshold Change Can be Made
        THEN {
                Object Not Found
                CALL (User)
                }
```

If an alternative algorithm cannot be found or no threshold exists within the algorithm for tuning, then we can assume that the object of interest in the scene does not exist or cannot be found using the available algorithms. In this case the user should add more algorithms to the image processing and segmentation database and start the replanning process again.

6.8 Summary

This chapter has explained the main components that form the Visual Planner. It has described how they act and react to certain events that occur within the domain, and intervention from the user, how a goal is given to the system and how sub-goals are created and satisfied. Some example rules and algorithms have been given in an intuitive form to aid in the understanding and development of such a system.

We have shown how the planning rules traverse through a network of image processing and segmentation algorithms using contextual information of the object and the algorithm to select a path (sub-plan). We have also described the execution monitor, which executes sub-plans, checks that they have succeeded, and initiates replanning. The next chapter shows the results obtained on a set of x-ray images (appendix B) and the validation of the system based on how well it has performed.

Chapter 7
Test Domain,
Results, and Validation

7. Introduction

This chapter presents the test data used to validate the system and the results produced by the Visual Planner. The system was tested as a whole and the results compared to an expert's interpretation of the images. A Database driven test design (Beizer 83, Hetzel 73) was used as the Visual Planner is too complicated (Hanson 78) to test at the component level. This means that we have not tried to formally prove the system module by module, but have instead tested the Visual Planner on the results it gives as a complete system.

Comparisons of the results produced by the Visual Planner and those given by the experts were measured using the Chi Square Test (Owen 90). This test is appropriate because it allows us to calculate the deviation of the observed and expected values of our results, thus giving a measure of accuracy. The Visual Planner was also compared to, and tested against Anderson's (Anderson 87) Parameter tuning system.

7.1 Validation Aim

Validation within the Visual Planner was carried out to assure the quality of software, and to prove that it is a viable alternative for manufacturers. The aim of the validation was to determine whether the Visual Planner can reach a 95% accuracy level within the visual task, and if not what level can it reach within a given domain? The Visual Planner may then be used by programmers as a tool for larger and more specialized image understanding systems in the appropriate domain.

7.2 Image Acquisition

Images must somehow be captured and converted in order for the Visual Planner to perform all the necessary processing. A device called a *frame grabber* is employed to take an analogue signal and digitise it into discrete picture elements, or pixels, for subsequent transfer to computer memory. The laboratory set-up employs a CCD cameras and a frame grabber with adequate RAM to store a digitised images, having a resolution of 512*480 pixels with 0-255 shades of grey. As each image is 512*480 pixels with 0-255 shades of grey, the storage requirements are very high. Typically on image will require about 250 K Bytes of memory.

7.3 Test Domain

The test domain used for the system was based on x-ray data of metal castings provided by the x-ray department at Stone Foundries Ltd. A sample of the images used to test the system are shown in appendix A. The data is separated into several categories shown in the tables below. At present, these castings are x-rayed and then the images are manually inspected by a domain expert.

The system was taught using a sample of x-rays from the data provided by Stone Foundries. 16 clean images were used to extract domain knowledge and reference information. Pratt (Pratt 78) used 8 images to calculate image quality, however 16 (above this number the histogram did not change drastically) images were used as we also needed to calculate the average intensity histogram. 240 images were used to teach the system different types of faults (see Appendix B), once again their intensity histograms where added to the average intensity histogram. This was done as it gave a better indication of the environment we were working in. The Visual Planner was then tested using 200 (20 images for each fault tested, see Appendix B) images with different faults. These results were then compared to the expert's interpretation of the data. The results were compared in two ways: a) to see if the Visual Planner compared to the experts in detecting the correct fault by name, b) to see if the visual planner picked up a fault at all or no fault, or was uncertain if a fault existed. Uncertain in this case means that the Visual Planner may have found an object but could not match as many attributes as would be necessary to make a positive identification.

Another 200 tests were carried out on a random sample of the 200 (20 for each fault tested, see appendix B) test images, but with the environment slightly changed in some way. In this instance lighting condition were changed by adding and reducing light, and noise was introduced to the image manually by adding a filter in front of the camera lens, and finally the lens focus was changed to give a blurred image. For each change in the environment 5 images were tested from each group of faults.

Finely these results were compared to Anderson's Parameter tuning system, to see how well the Visual Planner performed, as a result of knowledge acquired from the domain. This test also proves that the Visual Planner's success is not only attiibuted to the matching algorithm used, as the same matching algorithm was used for both systems.

Table 7.1 shows an expert's classification of defects found on the x-ray images. These are not only defects in the metal, but defects in the images themselves and objects that are not part of the x-ray, but added manually.

Category One: Spurious Images	**Defect Code**	**ID No.**
Artifacts	A	1
Emulsion Marks	FM	2
Film Fog	FF	3
Diffraction mottle	DM	4
Image Caused by Scatter	SC	5
Coarse Grain	C/G	6
Excess Metal	E	7
Surface Irregularities	SF	8
Identification	ID	9
Pick Up (Lead, Ceramic, tape, etc)	PU	10

Table 7.1 Category One Defects.

Film fog for instance, is caused by the lens of the x-ray camera not being clean or the quality of the film itself. This can effect and change the enhancement plan if it is encountered as it makes the image blurred. Artifacts on the image can range from dust particles to finger prints. In any case they are unwanted noise, once again this may have an effect on the plan.

Identification marks are put onto the x-ray so that each image has an identification of what it is. This usually takes the form of letters and numbers to uniquely identify each x-ray. These marks must be considered, and taught to the system by showing them to the Visual Planner. This makes sure that their intensity values are added to the average intensity histogram of the image characteristics, and thus become part of the background.

Category Two:

Images due to Internal Defects	**Defect Code**	**ID No.**
Gas Hole or Pore	Gas or G	11
Pinhole/Gas Porosity	Por	12
Blowhole	B/H	13
Shrinkage	SH	14
*Microshinkage	M/SH	15
*Filamentary Shrinkage	F/SH	16
Sealed Hot Tear	S/HT	17
Flowline	FL	18
*Oxide	OX	19
Dense Inclusion	DI	20
Internal Crack	CR	21
Hot Tear	HT	22
Broken Core	C/Br	23

* Faults not tested due to lack of data or too small to be tested due to resolution.

Table 7.2 Internal defects found in x-ray images.

Table 7.2 shows internal defects found in the castings. It is this data used to test the system. However, due to the resolution of the camera as described above some of these defects were too small to pick up and are only included here for completeness. Some defects described here ware also not tested because no x-ray images of the faults were available.

The information available on these faults from the experts at Stone Foundries is that there are two main types of faults: a) *most internal faults are linear,* and b) *pore holes and other holes are mostly undefined in shape, but can be recognised by their size, intensity, and grouping*. This information is very limited, but will suffice to give the Visual Planner contextual domain information when selecting algorithms. We can specify that cracks and tears are linear and pore holes and gas holes are blobs.

An x-ray image showing features of both categories listed above is shown in Figure 7.1. Here we can see that the image has identification marks, a crack of some type and some pore holes.

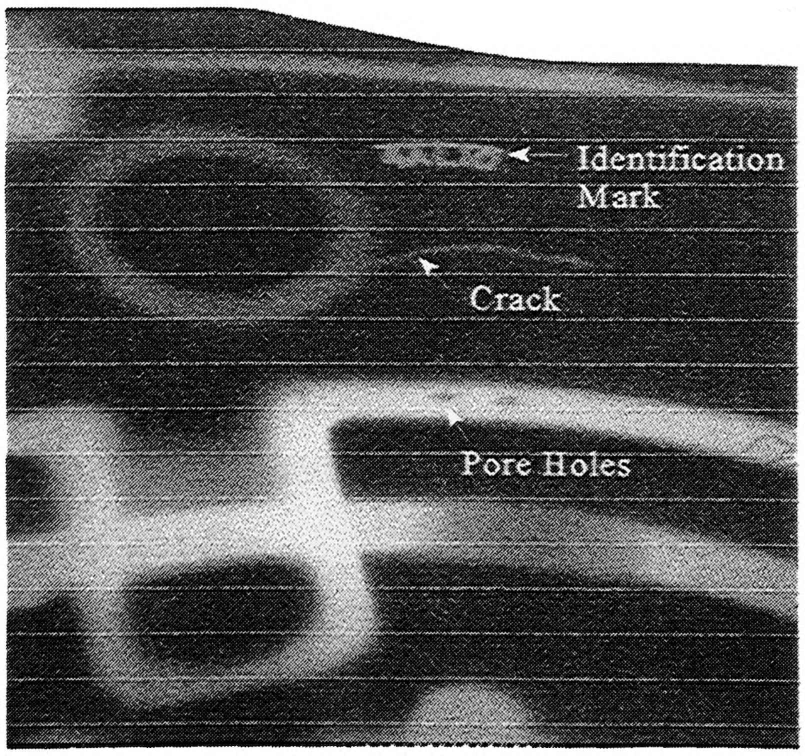


Figure 7.1 x-ray image with some faults.

7.4 Domain Information Acquisition

A number of images (16) that did not contain faults (as agreed by the experts) were shown to the Visual Planner in its learning mode. These images were used to extract the domain information. A small set of these images are shown in Appendix A. These images were used to calculate average intensity of the domain, signal-to-noise ratio, distribution of average intensity histogram, and background intensity information. These images are used as reference images, and thus would be the normal working environment (image characteristics) for the system. It is the deviation from the average reference histogram (see chapter 6) that will cause an event within the Visual Planner that may require a replan or some sort or user intervention if the change cannot be resolved.

The characteristics extracted form the images (average reference histogram and noise deviation histogram) were used in an attempt to determine using the domain rules the following:

a) If low-level enhancement would benefit the segmentation process as the image presented may already be a binarised, well defined image.

b) If the objects to be identified were well defined compared to the background data of each object, i.e., the histograms of the foreground and background do not overlap.

c) If thresholding techniques could be used for segmentation or elimination of background noise, and data.

d) Determining intensity characteristics of the object and the background for multiple threshold selection.

7.5 Sample of Results

The results given here show a sample of the images used to test the system. More sample images can be seen in appendix A along with their attributes. The samples selected here show two types of faults, Blowholes (Figure 7.2a) and Cracks (Figure 7.2b). In the following sections we describe the image characteristics, object characteristics, and the algorithms and thresholds selected to achieve the goal.

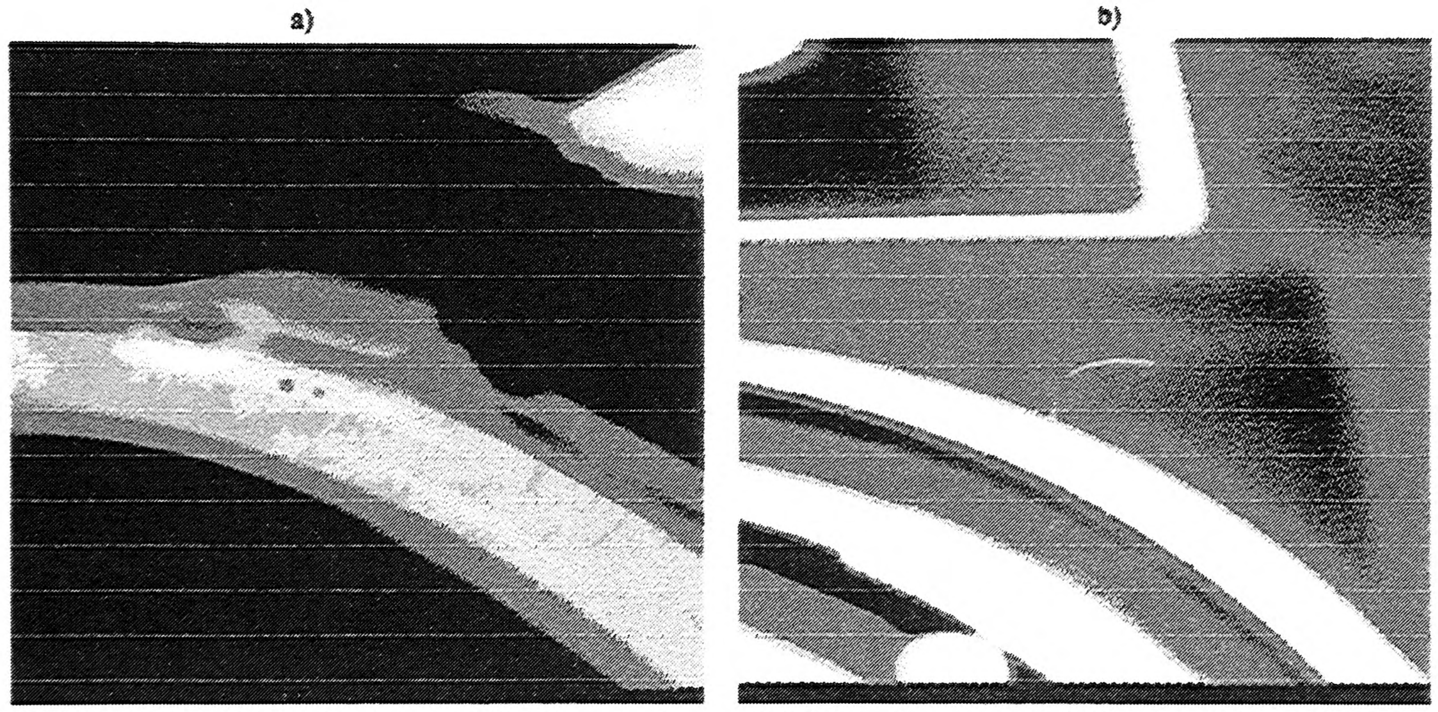
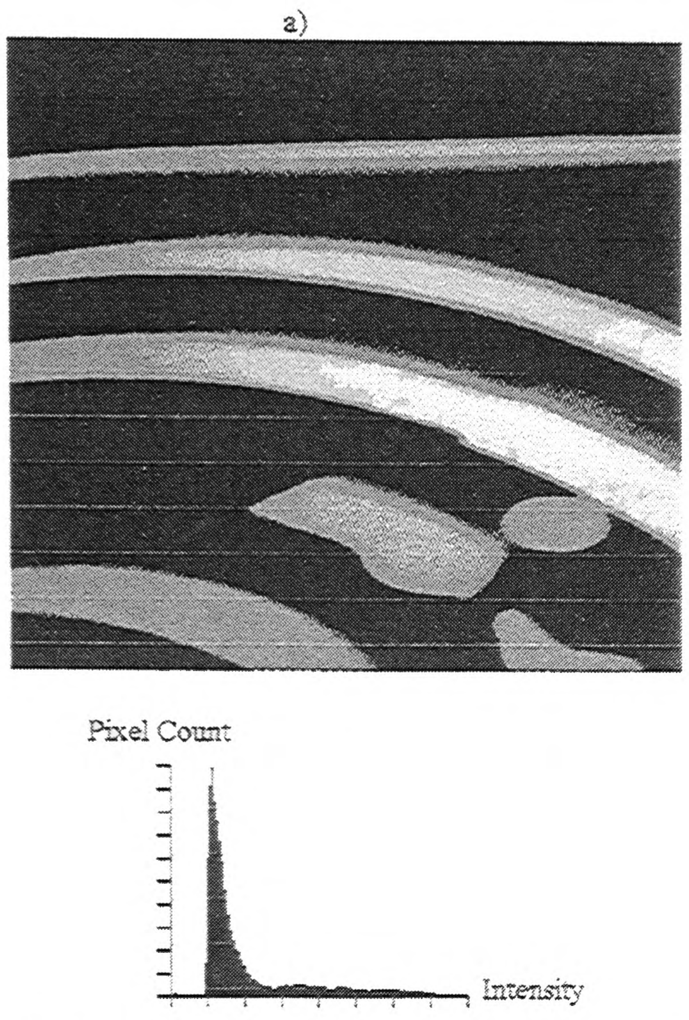


Figure 7.2 a) Test image with Blowhole. **b)** Test image with Internal Crack.

The tests were carried out twice, once under normal working conditions, as taught to the system and once with the domain slightly changed. The domain was changed in this case by increasing the light source by adding a constant increase in light below the x-rays.

The histograms below show how such a change effected an x-ray. Figure 7.3a shows an x-ray in its normal environment with its associated intensity histogram and Figure 7.3b shows the same image with a constant light source added, along with the new histogram.



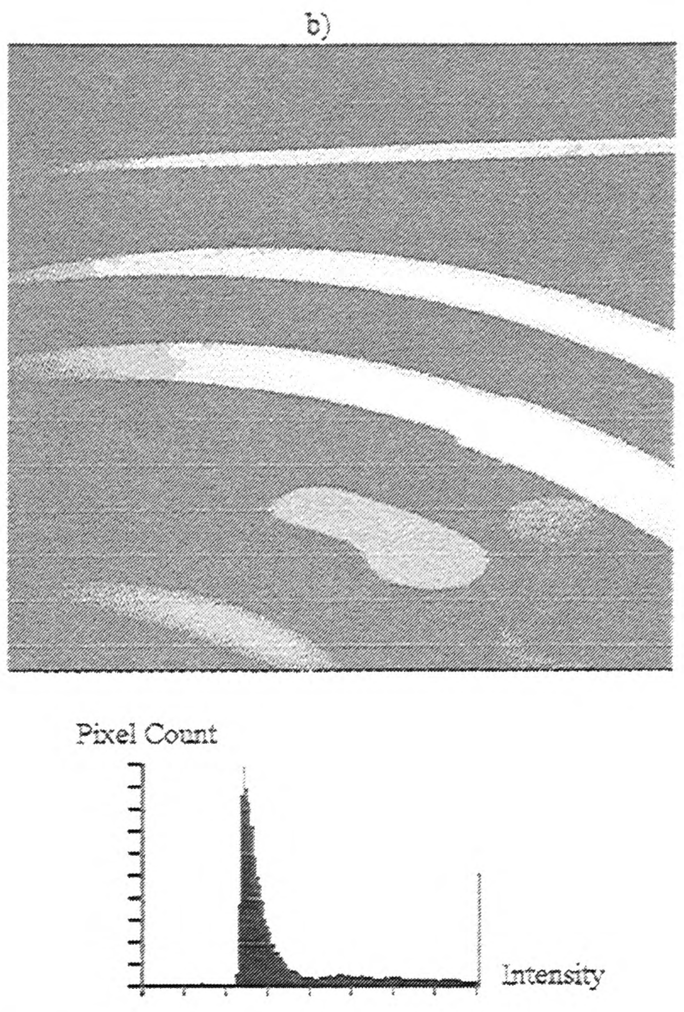


Figure 7.3a x-ray in its normal environment and its associated intensity histogram.
Figure 7.3b x-ray with constant light source added and its associated histogram.

7.5.1 Image Characteristics

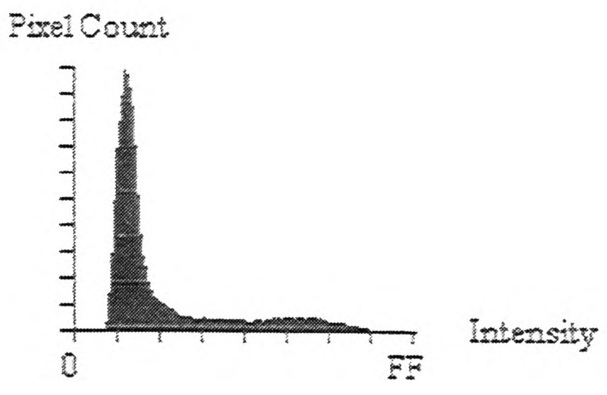


Figure 7.4 Average intensity histogram of all images.

The domain information extracted from the sample images (Figure 7.3a Blowhole and Figure 7.3b Internal Crack) and other images used to train the system will be used to select the appropriate image enhancement and noise suppression algorithms. Table 7.3 shows the image characteristics (see chapter 4) extracted from the training set of images. Figure 7.4

shows the average histogram of all images shown to the Visual Planner after noise suppression has taken place on all these images.

Attribute Name	**Value**	
Maximum Background Threshold	94	Intensity Value
Minimum Background Threshold	20	Intensity Value
Maximum (+) Noise Deviation	1586	Pixel Count
Maximum (-) Noise Deviation	-1529	Pixel Count

Table 7.3 Image Characteristics

7.5.2 Object Characteristics

The object characteristics of the Blowhole and the Internal Crack are shown in Table 7.4. This includes contextual information given by the user and object attributes extracted from the image.

General Information	**Value**	
Main Class	Crack	Hole
Contextual Information	Linear	Blob
Name	Internal Crack	Blowhole
Not Subclass	Flowline	Gas Hole
Object Attribute	**Value**	
Average Intensity	127	31
Max Intensity Value	132	35
Min Intensity Value	120	29
Area	43	32
Vertices		
Lines	1	
Facets		
Major Axis	16	10
Minor Axis	3	7
Radius		
Centre		

Table 7.4 Object characteristics and contextual information for Internal Crack and Blowhole.

7.5.3 Sub-Plan Creation

Once the image and object characteristics have been acquired by the Visual Planner it starts to create sub-plans given a user goal. As we want to segment two types of faults (Internal Crack and Blowhole) two specific goals were given as shown below:

Goal a)	Find All Internal Crack
Goal b)	Find All Blowhole

After the goal has been given the sub-goal designator takes over. As the goals were specific the sub goals came out to be the same as the main goal. However, when given a more general goal the sub-goal designator produced more sub-goals. For instance when asked to Find All Objects Average Intensity <= 35 the sub-goals produced were as follows:

Sub-goal a)	Find Gas Hole
Sub-goal b)	Find Pore Hole

Any object held in the object characteristics database that had an average intensity value less then 35 was extracted. For each sub-goal a sub-plan is created or inherited dependent on the deductions made by the Visual Planner, e.g. Gas Hole is a Blob of class Hole.

7.5.3.1 Enhancement Plan Creation

The enhancement plans for the Internal Crack and the Blowhole was created using only image and object attributes based on their intensity values. The Median filter was selected for noise reduction as it performed better then the Mode filter in terms of suppressing noise (see chapter 5). This filter was selected in the training stage by the noise domain rules. The results of the Median and Mode filter are given below:

Median Filter

Maximum noise deviation = 1529	Pixel Count
Minimum noise deviation = -2635	Pixel Count
Performance error per intensity value = 24	Intensity Count

Mode Filter

Maximum noise deviation = 1563	Pixel Count
Minimum noise deviation = -3421	Pixel Count
Performance error per intensity value = 72	Intensity Count

As we can see both noise suppression filters failed to reduce the noise deviation on the negative scale, as compared to Table 7.3. However, there were less errors per intensity value produced by the Median filter as only 24 out of the 256 intensity values fell outside the maximum and minimum noise deviation thresholds values, and thus this filter was selected.

Table 7.5 shows' the enhancement plans and threshold values set for both the Internal Crack and the Blowhole. The enhancement plans were not changed in any way after the initial selection.

	Internal Crack	**Blowhole**
Enhancement Plan	Median Filter	Median Filter
	Digitisation	Digitisation
	*Min=143, Max=191	*Min=29, Max=35
* Threshold values selected for algorithms		

Table 7.5 Initial enhancement plan and threshold values.

The initial enhancement plan selected to digitise each image as only one object of interest was given in each of them. The domain rules, in this case have selected to leave all intensity values in the image that are between the maximum and minimum threshold values of the object. All values out of this range were set too black (intensity value 0).

7.5.3.2 Segmentation Plan Creation

As each object used in this test has different contextual information (one being a blob and the other linear), a different segmentation sub-plan is orchestrated. The initial segmentation sub-plan orchestrated by the Visual Planner for the Blowhole was sufficient in order to segment the image and thus no replanning was needed. The Internal Crack sub-plan, however, underwent a mutation from its initial plan (Sobel edge detection, Robust line rules, and Boundary following) to the sub-plan and threshold values shown in table 7.6.

	Internal Crack	**Blowhole**
Segmentation Plan	Sobel Edge Detector	Region Splitting
	*157	*Min=29, Max=35
	Robust Line Rules	Robust Region Rules
	Dilate	Calculate Area of Regions
	Erode	Calculate Major Axis
	Boundary Following	Calculate Minor Axis
	Calculate Major Axis	
	Calculate Minor Axis	

* Threshold values selected for algorithm

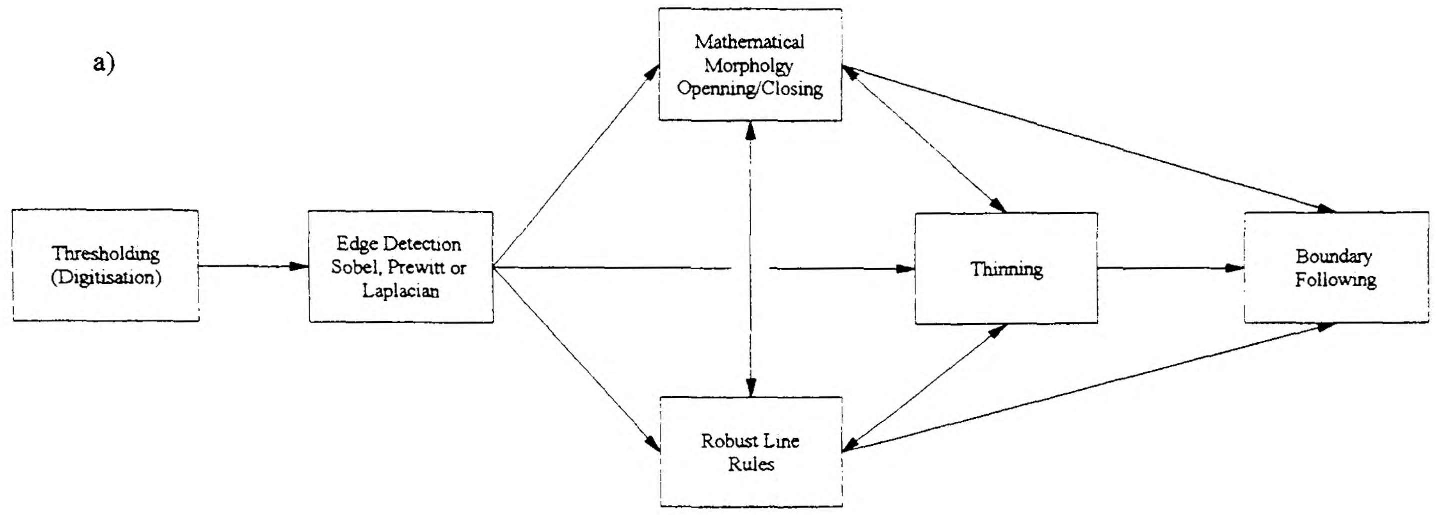
Table 7.6 Initial segmentation sub-plans and threshold values.

The Internal Crack is Linear, so only algorithms that work on linear objects are used. In this case the Internal Crack is Linear but not Geometric and thus the Internal crack has no specific size, corners or facets. Therefore the major and minor axis are used as constraints, if an object found falls below or equal to these constraints within the boundary then the sub-plan has achieved its sub-goal. The Sobel edge detector (Ballard

82) was used to find the edges of the Crack, it was then subjected to the robust edge detection rules (Panayiotou 93) which join broken lines and delete unwanted ones. This was succeeded by a dilation that fills troughs on edges and then Erode that erodes the edges reducing the spikes produced by the edge detector (Lindley 91). Boundary following (Dudani 76) is then used to detect all closed regions produced by the edge detector. Finally each area within a closed boundary is calculated giving the major and minor axis.

The contextual information used when describing the Blowhole was Blob. This means that the object has no predefined size, but can be matched using its area and intensity values. In this case a Region Splitter (Gonzalez 87) was used to segment regions that fell within the maximum and minimum threshold values of the object. The area, major axis and minor axis of each region detected was then calculated. If these values match or fall above or below the object's major and minor axes respectively, compared with the values in the object database then the Blowhole has been detected The area of a Blowhole may vary, however. When we give the Visual Planner the attributes of the Blowhole, we attempt to give it the smallest and largest possible values for both its minor and major axes.

a)


Mathematical Morpholgy Openning/Closing
Thresholding (Digitisation)
Edge Detection Sobel, Prewitt or Laplacian
Thinning
Boundary Following
Robust Line Rules

b)

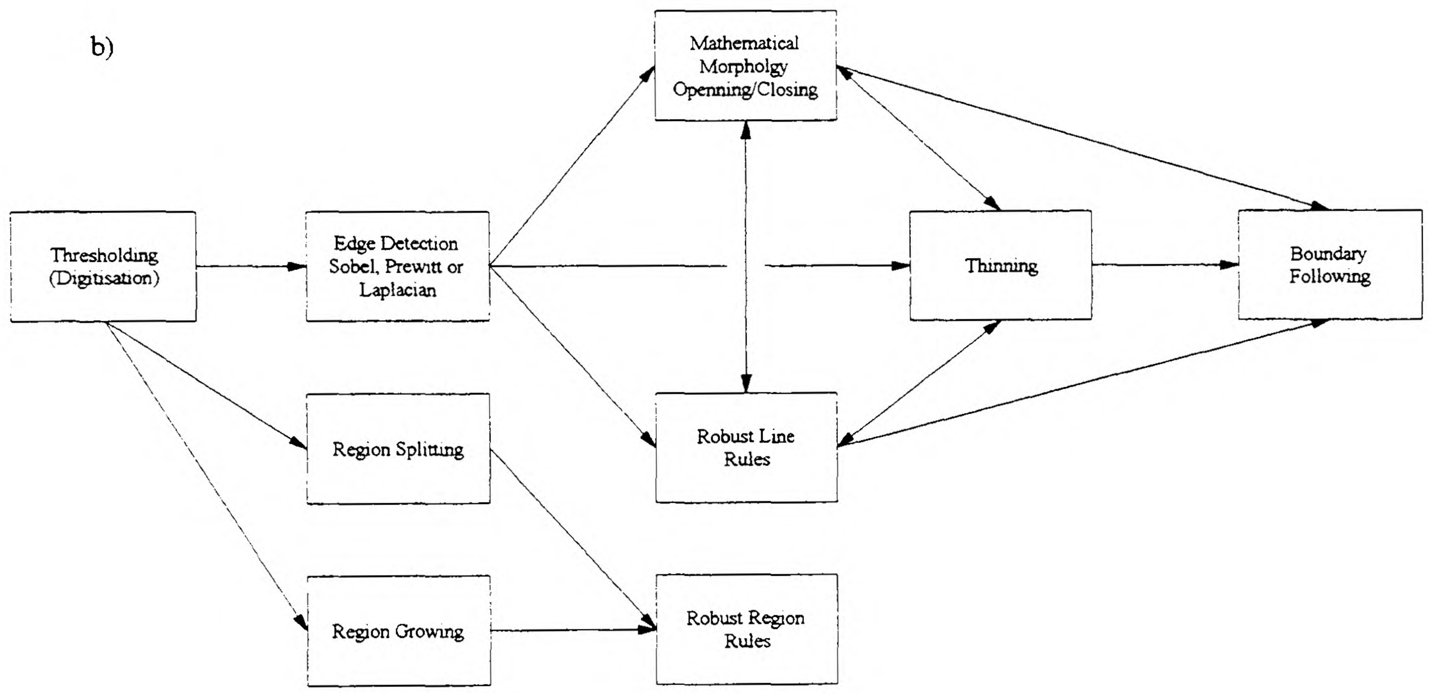

Mathematical Morpholgy Openning/Closing
Thresholding (Digitisation)
Edge Detection Sobel, Prewitt or Laplacian
Thinning
Boundary Following
Region Splitting
Robust Line Rules
Region Growing
Robust Region Rules

Figure 7.5a All possible plans for Internal Crack.
Figure 7.5b All possible plans for Blowhole.

Figure 7.5a and 7.5b shows all the possible segmentation plans that can be selected from the algorithm and image processing database, given the contextual information of the object (Crack is Linear and Blowhole is a Blob). The sub-plan used is selected by the planning rules as shown in chapter 6.

7.5.4 Replanning Events

The Visual Planner was shown images with a slightly different light intensity to see if the domain rules and replanning actions achieved their tasks. The histograms below (Figure 7.6a and Figure 7.6b) show the average histogram of all images taught to the Visual Planner and a histogram of an image with lighting conditions increased uniformly across the image.

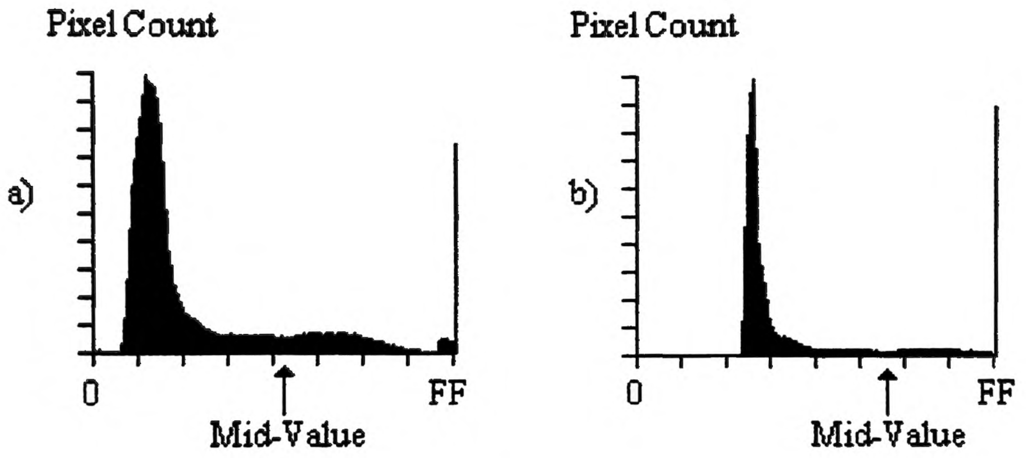


Figure 7.6a Average histogram of all images in normal environment.
Figure 7.6b Histogram of image with lighting conditions changed.

As we can see the histogram has moved to the right increasing the intensity values. The algorithms selected for the initial plan would no longer work as the object's intensities have changed. The enhancement and segmentation plans for the Internal Crack and Blowhole are shown in Table 7.7.

	Internal Crack	**Blowhole**
Enhancement Plan	Darken Image	Darken Image
	Median Filter	Median Filter
	Digitisation	Digitisation
	*Min=200, Max=220	*Min=29, Max=35
Segmentation Plan	Prewitt Edge Detector	Region Splitting
	*150	*Min=29, Max=35
	Robust Line Rules	Robust Region Rules
	Dilate	Calculate Area of Regions
	Erode	Calculate Major Axis
	Boundary Following	Calculate Minor Axis
	Calculate Major Axis	
	Calculate Minor Axis	

* Threshold values for algorithms

Table 7.7 Enhancement and segmentation sub-plans after replanning.

As we can see the initial enhancement plan has changed, and now includes the algorithm that darkens the image globally. The Blowhole algorithms and thresholds have not changed beyond this point. However, the sub-plan for the Internal Crack has changed both in algorithm structure and threshold values. This may be due to the fact that the crack had a higher intensity value then the Blowhole, and thus may have lost some information when the lighting conditions changed.

7.5.5 The Replanning Path

In order to see more clearly how a replan takes place we will follow through some of the possible paths that were considered by the Visual Planner in order to produce its amended plan for the Internal Crack. For each image processing algorithm we have an ID number associated with it as shown below:

ID	**Algorithm**
1	Sobel Edge Detector
2	Prewitt Edge Detector
3	Laplacian Edge Detector
4	Robust Line Rules
5	Mathematical Morphology
6	Thinning
7	Boundary Following

It is these numbers that show how the paths are formed. Table 7.8 show's some plans the Visual Planner considered and failed on before changing the threshold value for each failed plan.

Plan 1	1 - 4 - 5 - 6 - 7	Plan 18	2 - 5 - 7
Plan 2	1 - 4 - 6 - 7	Plan 19	2 - 6 - 4 - 5 - 7
Plan 3	1 - 4 - 7	Plan 20	2 - 6 - 4 - 7
Plan 4	1 - 5 - 4 - 6 - 7	Plan 21	2 - 6 - 5 - 4 - 7
Plan 5	1 - 5 - 4 - 7	Plan 22	2 - 6 - 5 - 7
Plan 6	1 - 5 - 6 - 7	Plan 23	3 - 4 - 5 - 6 - 7
Plan 7	1 - 5 - 7	Plan 24	3 - 4 - 6 - 7
Plan 8	1 - 6 - 4 - 5 - 7	Plan 25	3 - 4 - 7
Plan 9	1 - 6 - 4 - 7	Plan 26	3 - 5 - 4 - 6 - 7
Plan 10	1 - 6 - 5 - 4 - 7	Plan 27	3 - 5 - 4 - 7
Plan 11	1 - 6 - 5 - 7	Plan 28	3 - 5 - 6 - 7
Plan 12	2 - 4 - 5 - 6 - 7	Plan 29	3 - 5 - 7
Plan 13	2 - 4 - 6 - 7	Plan 30	3 - 6 - 4 - 5 - 7
Plan 14	2 - 4 - 7	Plan 31	3 - 6 - 4 - 7
Plan 15	2 - 5 - 4 - 6 - 7	Plan 32	3 - 6 - 5 - 4 - 7
Plan 16	2 - 5 - 4 - 7	Plan 33	3 - 6 - 5 - 7
Plan 17	2 - 5 - 6 - 7		

Table 7.8 Some replans produced by the Visual Planner to find Internal Crack.

7.6 The Chi Square Test

The aim of this test is to see if a 95% detection in faults can be reached and at what accuracy level determined by the user. The matching strengths used range from 60% (consumer products) to 100% (critical domains). With the matching strength set to 60% in the object recognition module, we are telling the Visual Planner, that it need be at least 60% certain that it has segmented and matched the object to the object database, and thus giving a low false alarm rate. However, if a matching strength of 100% is given, the Visual Planner must match all the features segmented from the object to the object database. This will produce a high alarm rate as the Visual Planner will fail to achieve the desired matching strength.

For each certainty threshold or similarity measure (60%-100%) we would like to reach a 95% confidence level. We thus can formulate a null hypothesis, that the expected frequency of correctly identified faults or objects will by 95% and 5% for incorrectly identified objects, and plans that, although produced a match, were under the matching strength.

The Chi Square tests in (Appendix B) are based on a set of 100 randomly selected images, and use two degrees of freedom with the critical value set at the 5% level = 5.99. This value reduces the risk of the plan working by chance. If X^2 is less then the critical value, then the Visual Planner has reached a 95% confidence level at the relevant matching strength criterion. The tests were carried out on test data that did not differ from the training set used. By this we mean that environmental changes did not take place.

Although the test is a measure of the overall performance of the system, in terms of matching the segments to the object database in order to achieve the goal, we cannot be certain on how well the system will work if other matching algorithms are used. It must be made clear that we are not trying to find a best matching algorithm this can be proved in a comparison (see appendix B) with Anderson's parameter tuning system (Anderson 87).

7.7 Anderson's Parameter Tuning System (Comparison)

The use of Anderson's parameter tuning system, within this project, is to prove that the Visual Planner's results are not dependent completely on the efficiency of the matching algorithm. Moreover on the ability of the Visual Planner to produce enhancement and segmentation plans with appropriate thresholds, which correctly reflect the state of the domain.

The tests we.e carried out on one class of fault, that of Crack and more specifically on the following faults Sealed Hot Tear, Flowline, Internal Crack, and Hot Tear, these are all liner faults, with different attributes (see Appendix A). The algorithms selected for this test are those initially produced by the Visual Planner for Internal Crack (Sobel Edge Detector, Robust Line Rules, Dilate, Erode, and Boundary Following). These algorithms seem to be a fair set of algorithms used by the Visual Planner to detect linear faults.

The threshold value for the Sobel Edge Detector changed from 255 down to 0 in steps of 5. This was done until the matching algorithm found a match. If a match was found within the specified matching strength, then it was recorded as correct. If a match was found, but was under the specified matching strength, then this was recorded as uncertain. The test was carried out on 100 images with no domain change and on a 100 images with the domain changed.

7.8 System Validation

By looking Figure 7.7 at the results we can say that the Visual Planner reached a 95% confidence level at approximately the 90% matching strength level. After this point the uncertainty factor rose dramatically, i.e. more false alarms. Even though the system could find the errors it was not certain of the results, and thus did not achieve the goal.

Figure 7.7 shows that if the Visual Planner is uncertain a false alarm has been raised. If the Visual Planner is incorrect then it has failed to detect the object, or it has detected a false target, or it has identified the fault wrongly. Correct means that the Visual Planner has detected a fault correctly, and has matched its feature vectors above the matching strength given by the user.

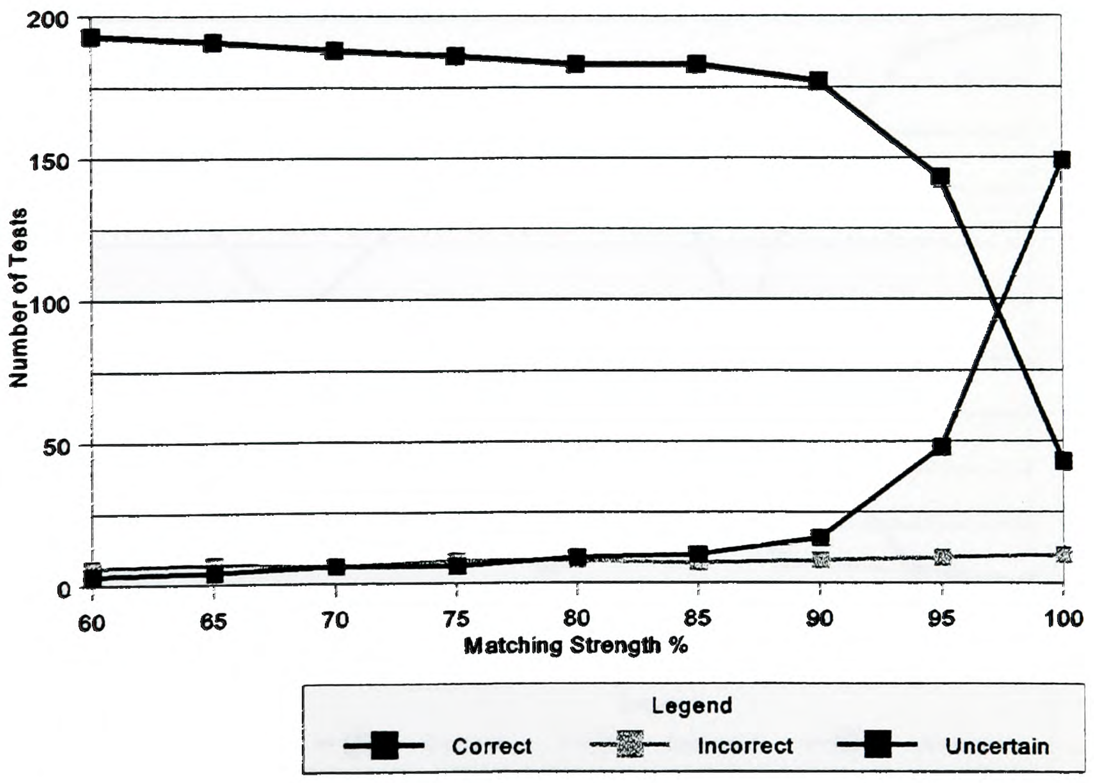


Figure 7.7 Results obtained from the Visual Planner with no environmental changes.

By using Anderson's parameter tuning system that used the same sub-plan created by the Visual Planner initially for Internal Crack (Anderson 87), we achieved an 85% accuracy level at approximately 80% (Figure 7.8) matching strength on the same set of images, with the same matching algorithm.

For each match weighting used with the test, the threshold changed from 255 down to 0. Before approximately the 70% mark (within this domain) the Anderson system does not produce enough threshold changes in order to find a correct match, as the changes stop when the matching algorithm has found a match (correct or incorrect). Before this point under segmentation takes place, and the matching algorithm does not need to be all that certain of a match, and thus usually a false target is detected. By about the 85% mark the threshold values have changed so many times that over-segmentation is encountered, and thus the likelihood of a false target being detected is increased until there comes a point where it's inevitable.

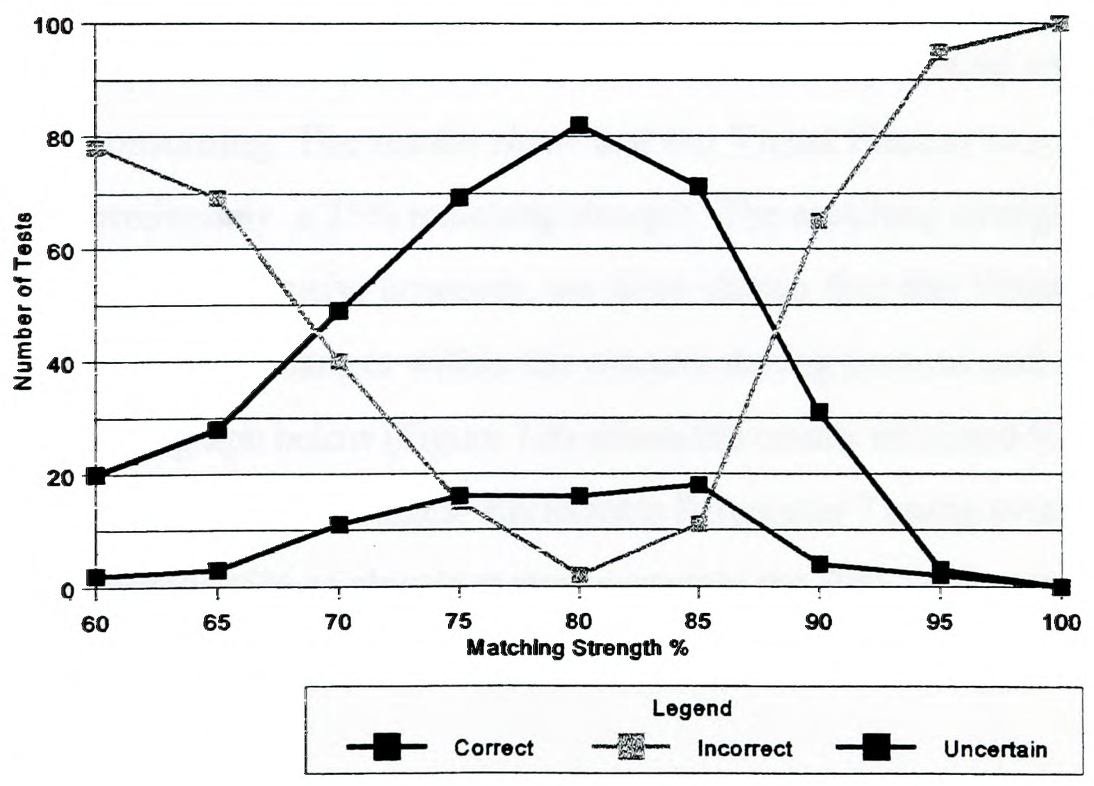


Figure 7.8 Results obtained with Anderson's Parameter Tuning system.

We can therefore conclude that algorithm parameter tuning has a limited success in improving the segmentation process. Moreover, algorithm and parameter tuning, along with domain information has a better performance measure.

7.9 System Validation With Environment Changes

Using a sample of data (200 x-rays) with the environment changed in some way, the system was once again tested (Appendix B). The aim of this test was to see if the domain rules worked in changing the image characteristics. It is these tests that validate the domain rules and show how complete these rules are. The same criteria was once again used to test the system. i.e. a 95% accuracy level with a matching strength range of 60%-100%.

With the environmental conditions slightly changed (Light increase, light decrease, Random noise added, and image out of focus. Appendix B) the Visual Planner seemed to be more uncertain (more false alarms) of its results. The number of erroneous identifications has risen to approximately 5%-7%. This may not seem to be a drastic change from the previous results of 2%-3%, however, more replanning was needed and this was time consuming. The results show that the Visual Planner can detect 95% of errors with approximately a 75% matching strength. The matching strength has reduced to 75% from 90% previously, however, we have shown that the Visual Planner can adapt to environmental changes within the manufacturing process and still be able to identify faults. The graph below (Figure 7.9) shows the results obtained from the Visual Planner with environmental changes. Anderson's Parameter Tuning system could only match, approximately 25% of objects at approximately the 70%-80% matching strength mark.

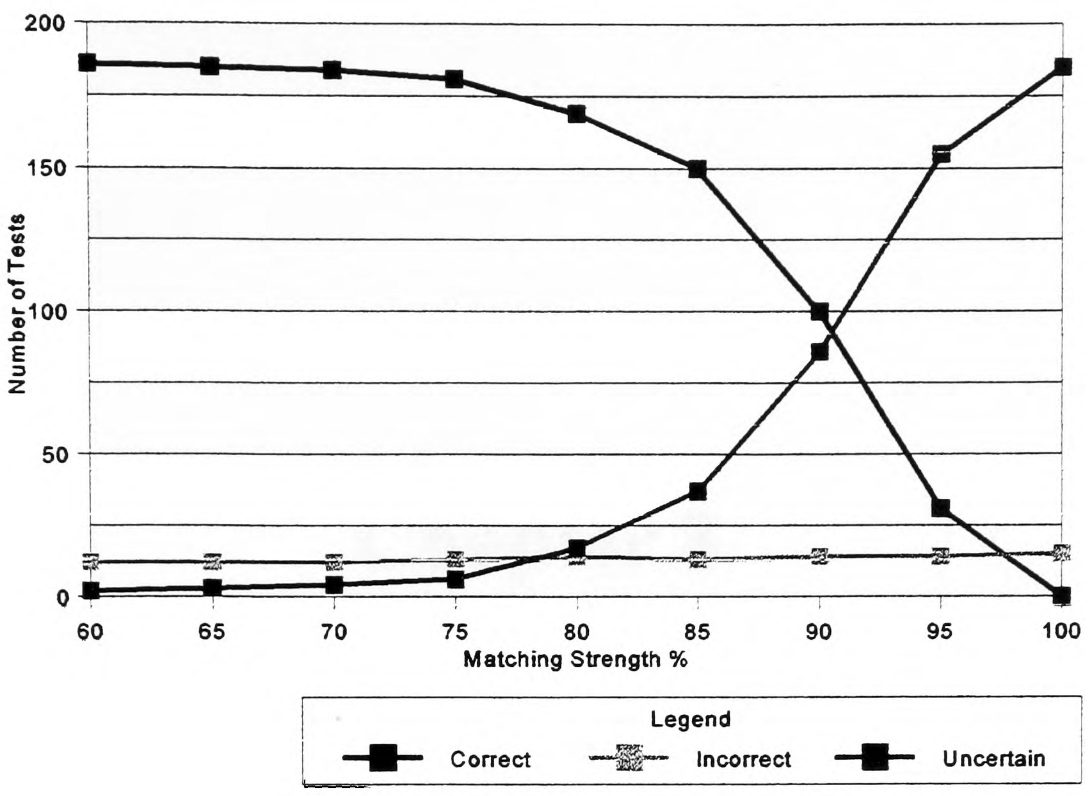


Figure 7.9 Results obtained from the Visual Planner with environmental changes.

7.10 Summary

This chapter described the test domain used to test the Visual Planner. The test data used based on x-rays of metal castings was not used in the development of the system. This shows that the Visual Planner is truly domain independent. We have shown that the Visual Planner can detect 95% of errors with a matching strength level of 90%. We can thus say that the Visual Planner can be used to produce segmentation algorithms in domain critical applications.

The Visual Planner is able to detect objects even if the environment has changed in some predefined way, i.e. lighting conditions have changed or more noise has been introduced in some way. The results here show that The Visual Planner is able to adapt and compensate for changes in the environment using the domain rules and replanning actions. Even if the environment changes the Visual Planner is still able to detect 95% of errors with a matching strength of 75%.

Chapter 8

Discussion of Application and Results

8. Introduction

This chapter presents an assessment of the applications' domain in which the Visual Planner may work well in given the results. It also presents the achievements and limitations of the Visual Planner as described by the project aims.

8.1 Applications for The Visual Planner

It has become clear that the Visual Planner should work well in many industrial application domains. It has shown that it is able to detect objects and faults from a set of images different from the ones used for its initial training and development. The simple prototype developed here is limited in many ways (see below), but it has been shown that the technique used works.

In its current form the Visual Planner may be used to select appropriate algorithms for a full blown visual inspection system. This eliminates the programmer's need for selecting the appropriate algorithms, and thus reducing uncertainty and unreliability within the system.

If the Visual Planner were to be incorporated within a full visual inspection system, in order to take advantage of its replanning capabilities, the system would have to be expanded and made more configurable (see chapter on further work).

The review of visual inspection systems undertaken (chapter 2 and chapter 3) indicates that the field is very active. Image recognition and understanding has an appeal to researchers and companies interested in its practical applications. The Visual Planner can be used by researchers, as a prototype in the field of active vision due to its ability in selecting threshold values and different algorithms. (Ballard 87, Poggio 87, Bajcsy

88), or by companies interested in having a system that is updatable, and will not become redundant if their domain changes in any way.

8.2 Implementation

What has been achieved is a small research prototype that produces segmentation algorithms for given objects. The algorithms selected for this task, and their thresholds if any, have been selected and changed in order for them to achieve their goals. The results and validation show that different algorithms and thresholds need to be selected, and not just the parameters of given algorithms changed (Anderson 87). The ability to acquire knowledge about the domain and image processing characteristics has shown that the system is domain independent and thus adaptive in nature.

8.3 Significance of the Work

The significance of the work in the authors' opinion is that:

1. It is the first implementation, to the writer's knowledge, which uses a planner with contextual information on the domain and image processing and segmentation primitives to select such algorithms for the visual task.

2. The fundamental design and implementation of the system does not restrict its use to any one visual domain. This is due to the modularity of the Visual Planner and to the fact that the system is oriented solely around the concept of planning.

The clear distinction between the inference engine and the knowledge base, makes it possible to remove one knowledge base from the Visual Planner and replace it with another.

3. The Visual Planner provides a domain-independent formalism for describing a domain at different levels of abstraction listed below:

a) High Level - Object contextual information describing the general shape of the object i.e. Blob, Liner, Geometric, Circular.
b) Medium Level - size, area, number of facets, number of lines etc.
c) Low Level - average intensity, background intensity, intensity threshold values domain contextual information such as lighting conditions, noise etc.

At each level we also include actions that can be taken to achieve a given goal. For example given a geometric object, initially select algorithms known to work well in calculating area, size, edges etc. At the second level of abstraction rules are used which select appropriate segmentation primitives given the object characteristics, i.e. lines are well defined, fault x is variable in size with an intensity value between min and max. The third level of abstraction is used to select algorithms in order to extract the intensity characteristics, by adding nose reduction algorithms based on the noise model taught to the system, or changing the contrast of an image based on the average intensity of all images.

The Visual Planner is more advanced then other adaptive vision system, primarily because it uses constraints, and employs contextual information to represent and reason about different world states.

4. The Visual Planner uses production rules to carry out its actions. The use of production rules (Davis 82) seems to be a natural way of expressing things about the domain, and the use of such rules seem to be comprehensible and intuitive to programmers. Thus, the general form of the representation and the way it is employed should not be unfamiliar to the average user.

More specifically, however, consider the source of the knowledge. It is supplied by human experts who are attempting to formalize their own domain knowledge. As such, the rules within the Visual Planner embody accepted patterns of human reasoning, implying that they should be relatively easy to understand.

5. In real-world domains, things do not always proceed as planned, making it necessary to monitor the execution of a plan and to replan when things do not go as expected. In complex visual domains it becomes increasingly important to use as much as possible of the old plan, rather than to begin again. The Visual Planner's execution monitor detects unexpected events within the domain, and can determine how they affect the plan being executed. In many cases, it can retain most of the original plan by making changes in that plan to avoid problems caused by these unexpected events. For example when lighting conditions change, we are able to keep most cf the plan by inserting an algorithm that would change the contrast in an image.

If a new object is added to the inspection line then we can also inherit existing plans based on the object characteristics and match them against existing objects. If we decide to produce object x in a different colour then no part of en existing plan needs to be changed, as intensity value does not matter except for unexpected events.

6. The Visual Planner provides a menu driven image processing interface in which the user experiments with image processing algorithms, which are applied to the image. This can aid the user in deciding what algorithm constraints to put on the objects of interest. This section could be extended so that a user could construct and execute sequences of algorithms.

7. The most important features of the Visual Planner are summarised below:

1) Domain independence.
2) Different abstraction levels.
3) Nonlinear actions.
4) Constraints.
5) Replanning.

8.4 Limitations

The Visual Planner has several limitations both as a planner and as a visual system. In this section we summarize many restrictions incorporated within the Visual Planner. These restrictions are categorised in two sections: a) the planning system, and b) the visual problems.

Despite these limitations, there are useful problems that can be addressed within them. Visual inspection is one of them, as shown by the results. These limitations permit an efficient system that works well in a practical domain. However, many of these limitations allow the system to come to a solution in a reasonable amount of time, otherwise we could encounter combinatorial explosions. For example if every combination and permutation of algorithm was used within the Visual Planner to achieve a given goal, then the system would become unusable as the number of algorithms grew. It would be far better to relax the constraints on a few algorithms within the Visual Planner, this would depend on the domain.

There are, naturally, some domains that the Visual Planner would not work well in. For instance a surveillance system that detects if any object has entered its boundary. The simplicity of these domain means that little use is made of the Visual Planner, although changes in lighting conditions may be monitored. However, many of the more complicated (and computationally expensive) features go unused. All the work carried out by the Visual Planner is unnecessary when something far simpler would do.

8.4.1 Planning Limitations

1. Like other classical planners, the Visual Planner employs a state-transition approach to representing a dynamic world. Actions change the world from one discrete state to another. The Visual Planner extends previous adaptive vision systems by allowing replanning after unexpected events, and being hierarchical, by saving search time as part of the planning process. However, sophisticated reasoning about time and modelling of dynamic processes are not possible within the present framework.

2. Having discrete actions means that the effects of an action occur instantaneously as far as the Visual Planner is concerned. However, it is not designed to monitor the world as it is planning, and therefore cannot react immediately to a changing environment.

3. A major limitation of Visual Systems is that they require complete and correct knowledge of the world. This is, of course, unrealistic in the real world, although there are certainly useful problems to be solved in domains where the state of the world is known. The Visual Planner alleviates this problem somewhat by analysing the state of the world by itself, but the system does no sophisticated reasoning about uncertainty of an object, i.e. The Visual Planner will not deduce that it is probably looking at object x and thus change the plan to give a stronger match. This limitation is also extended into the execution monitoring and replanning modules, which require correct information about unexpected events, such as changes in lighting conditions and noise.

4. Some rules, such as the domain rules, and the image processing primitive selection rules are implicitly embedded within the system. This is a limitation, as the rules encoded may not work well for all domains, and changing such rules requires

programming effort. It would be better to use an interactive method (see future work) for specifying the rules, and which primitives work well in what situations.

5. Since the Visual Planner is intended for use by programmers and users alike, this meant a restriction had to be placed on its vocabulary in order to make it easy to use. However, such a restriction may limit the search for objects in complex domains.

8.4.2 Visual Limitations

1. As with most visual systems, the Visual Planner is limited by the hardware used for image acquisition. High resolution cameras are available but are expensive. If cheaper cameras with less quality are used then some details within the image may be lost. Selecting a camera will depend on the domain. If we are looking for small cracks, or the granularity and texture of some object, then a high resolution microscope camera may be needed, if we are looking for heat traces then an infrared camera may be required. The tests carried out within this project used one CCD camera with 512*480 resolution, and 256 shades of grey to take pictures of x-rays. In fact we should have used an x-ray camera directly to give a better quality image. It is therefore safe to assume that the same results may not have been acquired if the appropriate camera was used, and that better results may have been given by the Visual Planner as the x-rays used suffered from different types of unwanted and unmodelled noise (finger prints and dust).

2. If a system is to be truly domain independent, then the best characteristics of each object must be extracted and saved. The question arises of how we model all these characteristics and attributes within a variant data structure? It is known that the more information we have on an object the better we can select segmentation algorithms. The Visual Planner uses a variant structure that models only certain information, such as average intensity, size, lines, facets, vertices, and class of object depending on the objects' contextual information. This is very limiting, we may want to model texture or

any number of other attributes. The difficulties here are how many attributes within different domains exist, and what effect would all these attributes have on the planning process i.e. the time it takes to plan and replan.

3. If the average intensity histogram differs from image to image too much, i.e. the background is not uniform or static, then the image intensity threshold values calculated may not be appropriate for use on an image acquired for segmentation. Therefore new objects may be lost in this information unless they really stand out, i.e. domain rules become useless. However, objects may still be extracted on their characteristics and attributes by the Visual Planner, but not with such certainty and accuracy.

4. Sometimes, it is not possible to set segmentation process thresholds such that all the objects of interest are located with respect to the background without over-segmenting the image. Over segmentation is the process by which the objects being segmented from the background are themselves segmented or fractured into sub-components. For example if two objects exist within an image (one is dark and one is light) by trying to extract the darker object we may over-segment the lighter object. In such cases background subtraction may be needed, or completely segmenting the image into regions, using both splitting and merging methods based on object characteristics.

5. The opposite to over-segmentation is under-segmentation. Most often with under-segmentation, objects of interest are separated from the background, but are adjacent or overlapping and are therefore segmented into only one composite object rather then discrete components. The solution to under segmentation is to avoid it at all costs. This however, is not always possible. The Visual Planner may use multiple thresholds, based on the object intensity information, but if two objects exist with approximately the same average intensity then the same threshold will be selected, and thus it will not be able to separate the objects.

8.5 Conclusion and Summary of Results

Most segmentation schemes incorporated into vision systems today are based on the principle of boundary detection (Dudani 76) or region splitting and merging (Gonzalez 87). These techniques assume that a significant grey-level change occurs between the signal and clutter. However, this is most often an erroneous assumption, and so these techniques are fragile, and quite commonly require well controlled conditions or human supervision. Effects of uneven sampling, lighting, shadowing, partial occlusion, clutter, noise, object-to-background changes, etc., contribute to errors in these segmentation processes that are manifest as false segmentation, as shown by the test carried out on Anderson's parameter tuning system (Anderson 87). Errors in the segmentation process can produce partial segmentation of the objects of interest, such as when two overlapping objects may be segmented as one (clumping of objects), and poor segmentation of objects and background, which may result in the calculation of erroneous features.

This thesis examined whether it was possible to select appropriate algorithms for image enhancement and segmentation dependent on the contextual information of the domain, object characteristics, and knowledge about image processing primitives. A domain independent Visual Planner was proposed and built for this purpose. Clearly such a system is only as good as the knowledge acquired by the system. The more information the Visual Planner has, the better its performance. The results show that the Visual Planner was able to select algorithms, form plans, and refine such plans for this purpose.

The final results obtained in the previous chapter were of a sample of 400 x-ray images shown to the Visual Planner at random; 200 of which had different domain changes within them, notably lighting changes, noise, and the image being out of focus. It is, however, not clear that the population was high enough to give a complete validation of the system. There may have been images that could not be taken into account which would have caused unusual effects in the system. Although every possible care has been taken to test the system to the full, using man made data, and real data, the system cannot be formally proven due to its complexity.

Selecting a different matching algorithm may have changed the results, however, the validation undertaken was not used to validate the matching algorithm, but to see if the Visual Planner could produce sub-plans that achieved a given goal. We have shown that the Visual Planner can adapt and change sub-plans to achieve sub-goals, while retaining most of the original plan. For a true validation to take place the system would have to be tested under different domains, although the test data used to develop the system was different from the test data used to validate the Visual Planner. A comparison with Anderson's parameter tuning system was undertaken to prove that the Visual Planner's results were not just based on the efficiency of the matching algorithm.

The results show that the Visual Planner can detect 95% of faults taught to the system with approximately 90% matching strength given the object attributes weighting, using the current matching algorithm and 95% of faults with approximately 75% matching strength with the domain changed in some way. What was interesting however, was the fact that incorrect identifications were constant throughout. This implies that the Visual Planner could identify 95% of faults overall, but was not confident of the results. If a better matching algorithm was used we could increase the accuracy level. We could also use different matching algorithms dependent on the features segmented, i.e. algorithms known to work well with specific features (see future work).

The Visual Planner has shown improvement over current domain dependent visual systems both in its flexibility by being upgradable and adaptable, and in its not being written and trained for any specific domain and making it domain independent and reusable. We have shown that the Visual Planner can be used by programmers as a tool when designing a visual system for a specific domain. This reduces uncertainty and unreliability as to which algorithms and thresholds to select when designing a system. The Visual Planner's domain rules and replanning actions can be used within any visual system to adapt to changes in the environment.

The next chapter tries to suggest solutions to the limitations given above and answer some of the questions raised during this work.

Chapter 9
Further Work

9. Introduction

It can be seen that the Visual Planner proposed is an adaptive and expandable visual system, as it is able to use contextual information to create initial plans and refined plans thereafter. Such plans use image processing and segmentation routines to achieve a given goal. The system can also react to changes in the environment and replan for a given situation. However, in the previous chapter a number of limitations came to light within the Visual Planner. These limitations were in two parts:

1. The actual planning concept.
2. The visual part of the system.

To overcome these problems further experiments and research would be useful in the following areas:

1. Developing an interface for adding and amending domain rules, planning and replanning rules.
2. Implementing the Visual Planner using an Active Database.
3. Adaptively selecting a matching algorithm for a given object in a scene.
4. Extending the Visual Planning concept.

These topics will be discussed in turn, with some suggestions and examples on how such extensions might be incorporated within the Visual Planner. The chapter ends with the new questions that have come to light during this research.

9.1 Further Experiments and Research

A number of extensions needed to the Visual Planner have come to light during this research. However, it is not clear what side effects these actions will have on the overall system performance, as no experiments have been conducted. These extensions are discussed below, giving some examples on how they may be implemented.

9.2 Adding and Changing Rules Interactively

At the moment the rules for selecting appropriate sub-plans and refined plans in the Visual Planner are encoded within the system. There is no easy way of adding new rules or even amending rules within the Visual Planner. As there are two main types of rules within the Visual Planner (Domain Rules, Planning and Re-planning Rules), we could have two different interfaces for specifying and modifying rules. One would be for adding and making changes to the Domain Rules, and handling environmental change, the other would be an interface for the Planning and Re-Planning Rules.

9.2.1 Adding and Changing Domain Rules

Domain Rules mainly work on a histogram of an image. At the moment the Visual Planner learns the domain it is working in by accumulating an average histogram of all the images presented to it. The variation of the histogram of a given image from the average histogram is then calculated. If the variation falls outside a predefined threshold value, then a re-plan is initiated. However, the problem lies with these threshold values, simply they cannot be changed without some programming effort, and even if they were, we could not be sure we had set appropriate thresholds given a particular domain.

To overcome this problem the Visual Planner should present a histogram of an image and the average histogram of all images to the user. The user may then interactively specify maximum or minimum threshold values on the average histogram, exclude part of the histogram altogether on the image histogram, move the histogram of the image to compensate for light changes, or equalise the image before it is added to the average histogram. The Visual Planner would then use the values on the average histogram as constraints for that domain. If the constraints are violated in any way, than the Visual

Planner will initiate an alarm and interrupt the user. The changes made to the image histogram should then be made to all images presented to the Visual Planner thereafter. As the user makes changes to the histogram, or sets threshold values, then these should be reflected in the current image. This would allow the user to see what effects his changes would have on the image, thus allowing them to highlight particular objects or compensate for bad lighting conditions due to poor equipment such as camera, or light source.

A simple example could be as follows: The user wants to reduce the alarm rate initiated if the maximum height value on the average histogram is violated. This could be done by moving the maximum value pointer on the histogram to a higher position as shown in Figure 9.1.

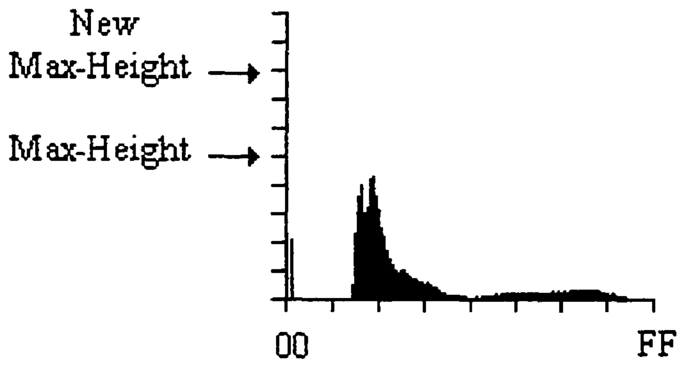


Figure 9.1 Average histogram with selected height value.

The change will not affect any image presented to the Visual Planner, however, it would modify the Domain Rule (DRule 11, see chapter 6) that detects the violation from:

```
DRule (11)
        IF Histogram of ALL Images Max_Peak + 10% < Current Histogram Max_Peak
        THEN CALL (USER)
```

to:

```
DRule (11)
        IF Histogram of ALL Images Max_Peak + New Max_Height < Current Histogram Max_Peak
        THEN CALL (USER)
```

9.2.2 Adding and Changing Planning and Re-Planning Rules

The rules for planning and selecting initial sub-plans, and refined plans thereafter are embedded within the Visual Planner. This however, is very restrictive as the rules developed here requires considerable programming effort and are difficult to test. The Planning and Re-planning Rules traverse their way through a network of algorithms, driven by the contextual information of the object and how well the algorithm works on each object, i.e. the space of useful application domain overlap is reduced. However, we may wish to traverse the network given other restrictions and constraints, or even relax the rules so that author algorithms that would not be considered, will be. For example suppose we wish to select the fastest possible segmentation process for an object. At the moment the rules work in such a way that a sub-plan is created by moving to succeeding algorithms until no more algorithms exist. The sub-plans created can therefore be longer than may actually be needed. It is feasible for example that an object may be segmented on just a thresholding technique (a priority level 1 algorithm).

In this case the rules that drive the search must be changed in some way so as to allow the Visual Planner to execute the sub-plans as each new algorithm is added to a sub-plan. Once the goal has been achieved, we do not need to consider any more algorithms for that particular object. A user interface could be added that gives the user a list of current Planning Rules and a list of structures that could be used on these rules as shown in table 9.1 below:

By editing the rules using the structures available, different search strategies could be used. In our example we could change Rule 4 to execute the algorithms as they are added to the sub-plan. If the execution is successful then the goal has been achieved, and thus the sub-plan would be saved. This type of interface would require some expertise to use, however, scripts of such rules could be written and saved for future use.

Planning Rules

```
RULE 1:     IF Succeeding-Alg EXISTS in Sub-Plan THEN
                                Break
RULE 2:     IF Succeeding-Alg.Contextual-Information = Object.Contextual-Information THEN
                    Weighting=Weighting + 1
            ELSE
                    Break
RULE 3:     IF Succeeding-Alg.Alg-C-I-Count < Min-C-I-Count THEN
            {
                    Weighting = Weighting +1
                    Min-C-I-Count = Succeeding-Alg.Alg-C-I-Count
            }
            ELSE
                    Break
RULE 4:     IF Weighting = 2 THEN
                    Next-Algorithm = Succeeding-Alg
```

Structures Available

IF, THEN, ELSE, BREAK
EXECUTE, CALL, USER
REPLAN

Table 9.1 Planning Rules and user commands that ma be added to such rules.

9.3 Active Database Extension

It is proposed by Panayiotou and Naqvi (Panayiotou 95) that the development of such a system should be undertaken on an Active Database (Naqvi 93). The philosophy of an active database is the provision of a flexible, adaptive, and active capability. An active database system has the notion of event-condition-action. Such a philosophy is ideal for a system such as the Visual Planner as it is stimulated by mainly undefined external events such as a change in lighting, noise, etc. The Visual Planner would then allow us to deal with the event accordingly, depending on the domain rules. The major change here is that events can have priority levels, thus if the environment changes in any way the Active Visual Planner would stop the current planning process and deal with the event.

An active database management system (DBMS) manages knowledge as well as providing the traditional database functionality. Thus, the architecture of an active database management system, in contrast to that of a passive DBMS, also has the capability to deal with rules. Rules such as Domain Rules can thus be easily changed, added, or deleted, much the same way as data can in a passive DBMS. This capability would allow us to change rules interactively as described above.

An active database provides a premium over the knowledge based approach in that it allows fast evaluation of knowledge in response to a change in the application domain space, i.e. external environmental events would be handled almost instantaneously by an active database implementation of a Visual Planner. The use of an active database would allow the domain knowledge to be stored in a central repository making it easily accessible, it would then be easy to compare an external event with a similar event that occurred previously. Re-planning would thus become easier as we could take the course of action that was taken when a similar event occurred as an initial plan template. An active database also supports other types of knowledge representation schemes (not just production rules), such as frames and semantics. This is an important factor within a visual system, as different representation schemes may be used for an appropriate search or goal match (Minsky 75).

An active database is highly concurrent, therefore such a system can process many images concurrently. This eliminates the limitation within the current visual planning prototype where each sub-plan is executed sequentially. Executing sub-plans on the same image concurrently can course problems, however, an active database has the capability of managing resources, in this case the image. Sub-plans cannot work on the same copy of an image, as algorithms within each different sub-plan may produce undesirable results causing the sup-plans to fail for no apparent reason. An active database would make copies of each image for a sub-plan to work on.

9.4 Adaptively Selecting Matching Algorithms

As we have seen in chapter seven, the success of the Visual Planner is influenced by the matching algorithm selected, in this case the local-feature-focus method (Bolles 83) has been used. However, This algorithm works well on specific features, and occluded objects are not considered within this implementation of the method.

We have seen from the review in chapter two that there are many ways an object can be recognised, and different similarity measures can be given dependent on the noise and the features extracted. Here we give an extension to the Visual Planner, which we believe is not only feasible, but more practical. Selecting a matching algorithm will depend on what features have been extracted by the sub-plan and what features are modelled within the object database. The matching algorithm can be selected from one of two places, a) the image processing and segmentation primitives database, where the appropriate matching algorithm is extracted with the segmentation algorithm selected, or b) from the object database, dependent on the contextual information of the object and thus the features to be extracted.

The matching algorithm is selected dependent on what feature space we need to compare. For example if an object is Linear then we may wish to match the object on it's edges (Nack 77), contours (Medioni 84), or surfaces (Pelizzari 89).

Selecting an algorithm to compute the similarity measure as already stated in chapter two is closely related with the selection of matching features since it measures the similarity between features extracted from the image and the model of the object. The selection of the similarity measure algorithm will then depend on which matching algorithm was selected. We could also use the image noise characteristics in selecting a similarity measure algorithm as we already have a model of the noise within the image characteristics database. This would make the algorithm selected more tolerant to noise if needed.

9.5 Extending the Visual Planning Concept

The Visual Planner developed here does not have the complexity of most classical planners, in that such complexity is not needed as we are not trying to solve the general problem of planning. However, extensions to the Visual Planner, would allow it to work more efficiently and increase its application domains. The list below gives some ideas which could be implemented to produce a more adequate system.

1. One advantage of the Visual Planner over other visual systems is that it could work in parallel as each sub-plan is nonlinear i.e. each sub-plan can be executed independently to achieve a partial goal. However, many algorithms selected may be common too many sub-plans. A resource manager could be added which detects this and produces plans that may share the image. This would make the execution of the overall plan much quicker and more efficient. Figure 9.2 shows how a resource manager could be used to produce composite sub-plans.

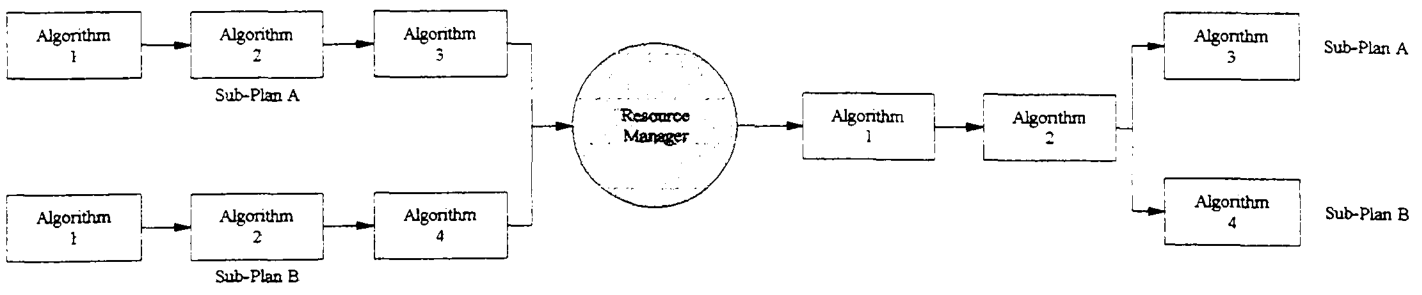


Figure 9.2 Resource Manager produces composite plan.

In this example the Resource Manager uses sub-plan A and sub-Plan B to produce a composite sub-plan as Algorithms 1 and 2 are common. We have effectively reduced the execution time by *Algorithm 1 Execution Time + Algorithm 2 Execution Time.*

2. The Visual Planner does not detect changes in the environment as it is actually planning and replanning. It is therefore not a real-time system and cannot work on continuous moving images. We have proposed a system using active database technology (Naqvi 93, Panayiotou 95) which can monitor the world as changes occur by calculating the number of sign changes in pointwise intensity values (see chapter 3) from the image currently being processed to the image that has just been captured (Venot 89). If the change is greater then some predefined threshold value, then an event could be initiated which interrupts the planning process and resolves the difference.

3. The Visual Planner may not select the best possible plan to achieve a sub-goal, it will select a sub-plan that satisfies the goal, but it may not be the most efficient. To select the best possible plan it would require every plan selected to be executed with different threshold values within the threshold range constraints. Each sub-plan would then have to be compared to all other possible sub-plans' performance measures. This would clearly be time consuming, thus making the Visual Planner inadequate. However, this type of search may be needed in safety critical domains where only the beast possible plan would suffice. This could be implemented as an option within the Visual Planner only to be used within such domains. Changing the planning and replanning rules may also produce better results, such as finding the shortest route through the network of succeeding algorithms. Whatever the method, better search techniques will be needed to make the Visual Planner more efficient.

9.6 New Questions Addressed

1. It has been said that we cannot have a truly adaptive system (Kohl 87) due to the fact that an image is too complex either because of the physical situation from which the image was derived, the nature of the scene, or there is a problem in evaluating different regions or segments. The results presented here however, showed that we can have a semi-adaptive system that can overcome problems within a scene by using domain rules to adjust for the physical situation from which the image was derived. As to the fact that

there is a problem evaluating different regions or segments, we have proposed a system that uses different matching algorithms dependent on the contextual information and characteristics of the objects to interpreting the scene.

The author has shown that we can at least have a semi-adaptive system, which acquires knowledge about its environment, and uses this knowledge to achieve a given goal.

2. The Visual Planner uses production rules to reason about changes in the environment, objects and in planning (Hayes-Roth 85, Nazif 84). However, is this the best possible method of implementing such a system? Chapter two has given some advantages and disadvantages of production systems. It has shown that other AI techniques (Semantic networks and frames, Syntactic descriptions, etc.) may be more useful for limited domains such as the classification of fingerprints. Selecting an AI technique to use will then depend on the domain we wish to model. The question then is, can we select an appropriate AI technique that works well given the domain?

Selecting among a number of techniques to produce a sub-plan, may at first seem not so difficult, if for example the Visual Planner was used to count a staircase structure we would say that a syntactic description (Miclet 86) of the object need only be calculated and compared to the syntactic description of the model. This is most efficient and very robust as we are basically matching the alphabet of the image to the alphabet of the model. However, The object has to be modelled using this alphabet, and if we wish to use for instance a production system, then the model will also have to be contained in a different form, one of lines, vertices and facets. The point is, that mixing different representations of knowledge and models will produce a far too complicated system that cannot by easily updated. However, for well defined domains where we know that the domain can be modelled using a syntactic description, or frames, or rules, then we should be able to select between the different techniques.

3. The Visual Planner does no high level reasoning about its domain, its aim is to produce segmentation algorithms given a predefined goal and the knowledge and data needed to produce such algorithms. However, we have seen that some sort of matching algorithm is needed to see if it has achieved its goal. The question then arises whether the Visual Planner can be made to understand a given scene and reason about this scene? If so we can imagine a system that produces plans for identifying objects related to other objects, without having to fully detect them in a scene, such objects may be occluded or out of the field of view that may have to be inferred by the Visual Planner (Mohr 90, Bunke 90). However, without closer inspection of the object occluded we cannot infer that it does or does not have a fault.

9.7 Summary

This chapter presented some extensions that could be carried out within the current framework of the Visual Planner to produce a more efficient and reliable system. It shows how active database technology could be used to control events, conditions, and actions with greater ease and reliability. We also discussed how we could adaptively select a matching algorithm given the features extracted from a sub-plan. We also showed how such an algorithm became part of that sub-plan, and thus the overall plan.

Finally the new questions that have come to light during this research were addressed and some possible solutions were given.

Bibliography

[Adorni 85] Adorni G., Massone L., Sandini G. and Trucco E. "Reasoning about iconic data in artificial vision." *Proceedings of the Society of Photo-Optical Instrumentation Engineer.* **595 (Computer Vision for Robots)**, pp. 245-255, 1985.

[Adorni 87] Adorni G., Massone L., Sandini G. and Immovilli M. "From Early Processing to Conceptual Reasoning: An Attempt to Fill the Gap." *Proceedings of the 10th International Joint Conference on Artificial Intelligence.* **IJCAI**, pp. 775-778, 1987.

[ATRWG 85] "Image Metrics for Automatic Target Recognition." **ATRWG**, Evaluation Committee, 1985.

[Aho 86] Aho A.V., Sethi R., and Ullman J.D. *Compilers: Principles, Techniques and Tools.* Addison Wesley, Reading, Massachusetts 1986.

[Aleksander 84] Aleksander I., Thomas W.V. and Bowden P.A. "WISARD - A Radical Step Forward in Image Recognition." *Sensor Review,* pp. 120-124, 1984.

[Aloimonos 88] Aloimonos J., "Visual Shape Computation." *Proc IEEE,* **Vol.76 No. 8**, pp. 899-916, 1988.

[Ambler 75] Ambler A.P., Barrow H.G., Brown C.M., Burstall R.M., and Popplestone R.J. "A Versatile Computer-Controlled Assembly System." *Artificial Intelligence,* **6(2)**, pp 129-156, 1975.

[Amir 90] Amir N.R. "The Lighting and Optics Expert System for Machine Vision." *Proceedings of the SPIE Machine Vision Systems Integration in Industry,* **Vol. 1386,** pp. 2-9, 1990.

[Anderson 87] Anderson H., "Edge-detection for Object Recognition in Aerrial Photographs." University of Pennsylvania, Grasp Laboratory Tech. Rep. **MS-CIS-87-96**, 1987

[Andrews 74] Andrews H.C. "Digital Image Resteration: A Survey.", *IEEE Computer*, **Vol. 7**, pp. 36-45, 1974.

[Bajcsy 88] Bajcsy R., "Active Perception." *Proc. IEEE,* **Vol. 76 No. 8**, pp. 996-1005, 1988.

[Ballard 82] Ballard D.H. and Brown C.M. *Computer Vision.* Prentice Hall, New Jersey, 1982.

[Ballard 87] Ballard D.H., *Eye Movement and Spatial Cognition.* Technical Report 218, Comp. Sci. Dept., Univ. of Rochester, 1987.

[Barnea 72] Barnea D.I., and Silverman H.F., "A class of algorithms for fast digital registration." *IEEE Trans. Comput,* **C-21**, pp. 179-186, 1972.

[Beizer 83] Beizer B., *Software System Testing and Quality Assurance.* New York, Van Nostrand Reinhold, 1983.

[Binford 71] Binford T.O. "Visual Perception by Computer." *Proc. IEEE Conf. Systems and Control*, 1971.

[Bolles 83]	Bolles R.C. and Cain R.A. "Recognizing and Locating Partially Visible Objects: The Local-Feature-Focus Method." *The International Journal of Robotics Research,* **Vol 1, No. 3,** pp. 57-82, 1982.
[Borland Int. 88]	Borland International, *Turbo C Reference Guide Version 2,* Borland International Inc. California, 1988.
[Brady 81]	Brady J.M., *Computer Vision,* North-Holland, Amsterdam, 1981.
[Brice 70]	Brice C. and Fennema C,. "Scene Analysis Using Regions." *Artificial Intelligence*, **Vol 1, No. 3**, pp. 205-226, 1970.
[Brooks 81]	Brooks R.A., "Symbolic Reasoning Among 3-D Models and 2-D Images." *Artificial Intelligence,* **Vol. 17**, pp. 285-348, 1981.
[Brooks 83]	Brooks R.A., "Model-based Interpretation of Two-Dimensional Images." *IEEE Transactions on Pattern Analysis and Machine Intelligence,* **5(2)**, pp 140-149, March 1983.
[Bunke 90]	Bunke H. and Sanfeliu A., *Syntactic and Structural Pattern Recognition, Theory and Application,* World Scientific, Teaneck, N.J., 1990.
[Canny 86]	Canny J. "A Computational Approach to Edge Detection." *IEEE Transactions on Pattern Analysis and Machine Intelligence,* **PAMI-8, No. 6**, pp. 679-698, 1986.

[Conners 83]	Conners R.W., "Identifying and Locating Surface Defects in Wood: Part of an Automated Lumber Processing System." *IEEE Trans. Pattern Analysis and Machine Intelligence,* **PAMI-5(6)**, pp. 573-583, 1983.
[Cross 83]	Cross G.R., and Jain A.K., "Markow Random Field Texture Models." *IEEE Trans. Pattern Anal. Machine Intelligence,* **Vol. 5**, pp. 25-39, 1983.
[Davies 86]	Davies E.R., "Image Space Transforms for Detecting Straight Edges in Industrial Images." *Pattern Recognition Letters,* **Vol. 4**, pp. 185-192, 1986.
[Derin 87]	Derin H., and Won C.S., "A Parallel Image Segmentation Algorithm using Relaxation Varying N Neighbourhoods and its Mapping to Array Processors." *CVGIP*, **Vol. 40, No. 1**, pp. 54-78, 1987.
[DTI 93]	DTI report "Profile: Industrial Vision Technology." *Department of Trade and Industry financial report,* 1993.
[Duda 73]	Duda R.O., and Hart P.E., *Pattern Classification and Scene Analysis,* John Wily and Sons, New York, 1973.
[Dudani 76]	Dudani S.A. "Region Extraction Using Boundary Following." *Pattern Recognition and Artificial Intelligence,* pp. 216-232, 1976.

[Espielid 91] Espielid R. and Jonassen I. "A Comparison of Splitting Methods for the Identification of Corner-Points." *Pattern Recognition Letters,* **No. 12**, pp. 79-83, 1991.

[Faugeras 81] Faugeras O, and Price K., "Semantic Description of Aerial Images using Stochastic Labelling." *IEEE Trans. Patt. Anal. Machine Intell,* **PAMI-3**, (Nov.), pp. 638-642, 1981.

[Feldman 74] Feldman J.A. and Yakimovsky Y. "Decision Theory and Artificial Intelligence: I. A Semantics-Based Region Analyzer." *Artificial Intelligence,* **Vol. 5 No. 4**, pp. 349-371, 1974.

[Fikes 81] Fikes R., Hart P., and Nilsson N., "Learning and Executing Generalized Robot Plans." *In Readings in Artificial Intelligence,* Tioga Publishing, Palo Alto, California, pp. 231-249, 1981.

[Foley 82] Foley J.D., and Van Dam A., *Fundamentals of Interactive Computer Graphics,* Addison-Wesley, Reading, Massachusetts, 1982.

[Gannon 76] Gannon J.D., "Data types and programming reliability - some preliminary evidence.", *PIB Symposium on Computer Software Engineering,* Polytechnic Institute of New York, April 1976.

[Geman 84] Geman S., and Geman D., "Stochastic Relaxation, Gibbs Distributing and the Bayesain Restoration of Images." *IEEE Trans. Pattern Anal. Machine Intell.,* **Vol. 6**, pp. 721-741, 1984.

[Genesereth 87]	Genesereth M.R., and Nilsson N.J., *Logical Foundation of Artificial Intelligence,* Morgan Kaufmann, Los Altos, California, 1987.
[Georgeff 87]	Georgeff M.P., "Planning." *Annual Review of Computer Science,* **Vol. 2**, pp 359-400, 1987.
[Gibson 79]	Gibson J.J., *The ecological approach to visual perception.* Houghton Mifflin, Boston, 1979.
[Giloi 88]	Giloi W.K. "Artificial Intelligence in Industrial Machine Vision." *Proceedings of the 4th Annual Artificial Intelligence Conference,* pp. 457-471, 1988.
[Gonzalez 87]	Gonzalez R.C. and Wintz P. *Digital Image Processing [Second Edition],* Addison-Wesley, Reading, Massachusetts, 1987.
[Goshtasby 85]	Goshtasby A. and Stockman G.C., "Point Pattern Matching using Convex Hull Edges." *IEEE Trans. Syst. Man Cybernetics SMC-15,* **Vol. 5**, pp. 631-637, 1985.
[Goshtasby 86]	Goshtasby A., "Piecewise Linear Mapping Functions for Image Registration." *Patt. Recogl.,* **Vol. 19 No. 6**, pp. 459-466, 1986.
[Gregory 71]	Gregory R.L. *The Intelligent Eye.* Weidenfeld and Nicolson, London, 1971.

[Griffin 89] Griffin P.M., Villalobos J.R. and Foster J.W. "Recognition and Matching of Occluded Objects." *The International Journal of Advanced Manufacturing Technology,* **No. 4**, pp 263-268, 1989.

[Gupta 89] Gupta M.M. and Knopf G.K., "The Percept: A Neural Model for Machine Vision." *Proc. of the Int. Conf. on New Generation of Computers*, Beijing, pp. 147-155, April 1989.

[Han 89] Han M., Jang D. and Foster J. "Identification of Corner-Points of Two-Dimensional Images Using a Line Search Method." *Pattern Recognition,* **Vol. 22, No. 1**, pp. 13-20, 1989.

[Hansen 92] Hansen O. "Local Symmetry Modeling in Multi-Dimensional Images." *Pattern Recognition Letters,* **No. 13**, pp. 253-262, 1992.

[Hanson 78] Hanson W.J., "Measurement of program complexity by the pair cyclomatic number, operator count.", *ACM SIGPLAN Notices,* **Vol. 13**, pp 29-32, 1978.

[Hara 83] Hara Y., Akiyama N., and Karasaki K. "Automatic Inspection Systems for Printed Circuit Boards." *IEEE Transactions on Pattern Analysis and Machine Intelligence,* **PAMI-5**, pp. 623-630, 1983.

[Haralick 85] Haralick R.M. and Shapiro L.G. "Image Segmentation Techniques." *Computer Vision, Graphics, and Image Processing,* **No. 29**, pp. 100-132, 1985.

[Hayes-Roth 85] Hayes-Roth B., "A Blackboard Architecture for Control." *Artificial Intelligence,* **No. 26**, pp. 251-321, 1985.

[Hayes-Roth 85] Hayes-Roth F., "Rule-Based Systems." *Comm. ACM,* **No. 28**, pp. 921-932, 1985.

[Heller 90] Heller D. *XView Programming Manual.* **Vol. 7**, O'Reilly and Associates, England, 1990.

[Hennessy 84] Hennessy J.L. "VLSI Processor Architecture." *IEEE Trans. Computer,* **C-33(12)**, pp. 1221-1246, 1984.

[Hetzel 73] Hetzel W.C. (Editor), *Program Test Methods,* Englewood Cliffs, New Jersey, Prentice-Hall, 1972.

[Hobbs 85] Hobbs J., "Granularity." *Proceedings IJCAI-85*, Los Angeles, California, pp. 432-435, 1985.

[Horowitz 74] Horowittz S.L and Pavlidis.T. "Picture segmentation by a direct split-and-merge procedure." *In Proc, 2nd IJCPR,* pp. 424-433, 1974.

[Hough 62] Hough P.V.C. "Method and Means for Recognizing Complex Patterns." US Patent 3069654, 1962.

[Huang 78] Huang T.S., Yang G.J. and Tang G.Y., "A Fast Two Dimensional Median Filtering Algorithm." *Proceedings of the IEEE Conference on Pattern Recognition and Image Processing,* pp. 128-131, 1978.

[Jackson 86] Jackson P., *Introduction to Expert Systems,* Addison-Wesley, Reading, Massachusetts, 1986.

[Kanal 81] Kanal L.N., Lambird B.A., Lavine D., and Stockman G.C., "Digital Registration of Images from Similar and Dissimilar Sensors." *Proceedings of the International Conference on Cybernetics and Society,* pp. 347-351, 1981.

[Kehtarnavaz 88] Kehtaranavaz N. and DeFigueirdo R.J.P., "A 3-D Contour Segmentation Scheme Based on Curvature and Torsion." *IEEE Trans. Pattern Anal. Machine Intelligence,* **Vol. 10**, pp. 707-713, 1988.

[Kimme 75] Kimme C., Ballard D.H., and Sklansky J., "Finding Circles by an Array of Accumulators." *Communications of the ACM,* **18(2)**, pp. 120-122, 1975.

[King 78] King E.A., "Laser Scanning." *In Proc. 4th Nondestructive Testing of Wood Symb.,* pp. 15-22, 1978.

[Kirsh 71] Kirsh R.A., "Computer Determination of the Constituent Structure of Biological Images." *Comput. Biomed. Res,* **Vol. 4**, pp. 315-528, 1971.

[Kohl 87] Kohl C.A., Hanson A.R., and Riseman E.M., "Goal Directed Control of Low-Level Processes for Image Interpretation." *In IU Proceedings DARPA,* **Vol. 2**, pp. 538-551, 1987.

[Kohler 81] Kohler R., "A Segmentation System Based on Threshold." *Computer Graphics Image Processing,* **Vol. 15**, pp. 319-338, 1981.

[Kuhl 82] Kuhl F.P. and Giardina C.R., "Elliptic Fourier Features of a Closed Contour." *Comput. Graph. Image processing,* **Vol. 18**, pp. 236-258, 1982.

[Lev 76] Lev A., Zucker S.W., and Rosenfeld A., "Interactive enhancement of Noisy Images." *IEEE Transactions on Systems, Man and Cybernetics,* **SMC-7(6)**, June 1976.

[Levesque 86] Levesque H.J., "Knowlidge Representation and Reasoning." *Annual Review of Computer Science,* Ed. J.F. Traub et al., pp. 255-287, 1986.

[Lifschitz 87] Lifschitz V., "On the Semantics of STRIPS." *Proceedings of the 1986 Workshop on Reasoning about Actions and Plans,* Timberline Lodge, Timberline, Oregon, pp. 1-9, 1987.

[Lindly 91] Lindly C.A., *Practical Image Processing in C,* John Wiley and Sons, 1991

[Lowe 87] Lowe D.G., "Three-Dimensional Object Recognition from Single Two-Dimensional Images." *Artificial Intelligence,* **Vol. 31**, pp. 355-395, 1987.

[Marr 78] Marr D., Palm G. and Poggio T., "Analysis of a Cooperative Stereo Algorithm." *Biological Cybernetics,* **No. 28**, pp. 223-239, 1978.

[Marr 79] Marr D. and Poggio T., "A Computational Theory of Human Stereo Vision." *Proceedings of the Royal Society of London, Series B,* **No. 204**, pp. 301-328, 1979.

[Marr 82] Marr D., *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information,* W.H. Freeman & Co., San Francisco, 1982.

[Mathews 76] Mathews P.C., and Beech B.H., "Method and Apparatus for Detecting Timber Defects." US Patent 3,976,384, 1976.

[Matrox 91] Matrox Electronic Systems Ltd., *Matrox IP-LIB Reference Manual,* **Rev.1**, Matrox UK. 1991.

[Medioni 84] Medioni G. and Nevatia R., "Matching Images using Linear features." *IEEE Trans. Patt. Anal. Machine Intell.,* **PAMI-6**, pp. 675-685, 1984.

[Michaels 81] Michales C.F. and Carello C., *Direct Perception,* Prentice Hall, Englewood Cliffs, NJ, 1981.

[Mohr 90] Mohr R., Pavlidis T., and Sanfeliu A., *Structural Pattern Analysis.* Worled Scentific, Teaneck, N.J., 1990.

[Mundy 80] Mundy J.L., and Jarvis J.F., "Automatic Visual Inspection." *In Applications of Pattern Recognition,* CRS Press, New York, 1980.

[Nack 77] Nack M.L., "Rectification and Registration of Digital Images and the Effect of Cloud Detection." *Proceedings of Machine Processing of Remotely Sensed Data,* pp. 12-23, 1977.

[Nagao 82] Nagao M., "Control Strategies in Pattern Analysis." *In Proc.6th Int. Conf. on Pattern Recognition,* pp. 996-1006, Munich, Germany, 1982.

[Naqvi 93] Naqvi. W. and Ibrahim M.T., "Rule and Knowledge Management in Active Database Systems", *Proc. of 1st Int. on Rules in Database Systems,* Edinburgh, September 1993.

[Nazif 84] Nazif A. and Levine M., "Low Level Image Segmentation: An Expert System." *IEEE Transactions on Pattern Analysis and Machine Intelligence,* **PAMI-6 No. 5**, pp. 555-577, 1984.

[Nilsson 82] Nilsson N.J., *Principles of Artificial Intelligence,* Springer-Verlag, Berlin, 1982.

[Nye 90] Nye A., *Xlib Reference Manual Second Edition,* **Vol.2**, O'Reilly and Associates, England, 1990.

[Owen 90] Owen F., Jones R., *Statistics 3rd Edition,* Pitman Publishing, London, 1990.

[Paar 90]	Paar G., and Kropatsch W.G., "Hierarchical Cooperation Between Numerical and Symbolic Image Representation." *In Structural Pattern Analysis,* World Scientific, Teaneck, N.J., 1990.
[Pavlidis 82]	Pavlidis T., *Algorithms for Graphics and Image Processing,* Springer-Verlag, Berlin, Heidelberg, 1982.
[Pearsons 90]	Pearsons T.J., "Conceptual Clustering in Relational Structures: An App.lication in the Domain of Vision." *Proceedings of the Fourth European Working Session on Learning,* **EWSC89**, pp. 163-177, Pitman, London, UK., 1990.
[Pelizarri 89]	Pelizarri C.A., Chen G.T.Y., Spelbring D.R., Weichselbaum R.R., and Chen C.T., "Accurate Three-dimensional Registration of CT, PET and / or MR Images of the Brain." *J. Comput. Assisted Tomogr.*, **Vol. 13**, pp. 20-26, 1989.
[Pentland 84]	Pentland A. P., "Fractal based description of natural scenes." *IEEE Trans. pattern Analysis and Machine Intelligence,* **PAMI-6**, pp. 661-674, 1984.
[Perkins 78]	Perkins W.A., "A Model-Based Vision System for Industrial Parts." *IEEE Trans. Computers,* **C-27**, pp.126-143, 1978.
[Poggio 87]	Poggio T., "MIT Progress in Image Understanding." *Proc. DARPA IU Workshop,* Los Angeles, CA, pp. 41-54, 1987.

[Pollard 85] Pollard S.B., Mayhew J.E.W. and Frisby J.P. "PMF: A Stereo Correspondence Algorithm Using a Disparity Gradient Limit." *Perception,* **Vol. 14**, pp. 449-470, 1985.

[Pratt 78] Pratt W.K., *Digital Image Processing,* John Wiley & Sons Inc., New York, 1978.

[Prewitt 66] Prewitt J.M.S. and Mandelsohn M.L., "The Analysis of Cell Images." *Annual New York Acad. Science,* pp. 1035-1053, New York Acadamy of Science, New York, 1966.

[Prewitt 70] Prewitt J.M.S., "Object Enhancement and Extraction." In B.S. Lapkin and A. Rosenfeld, Editors, *Picture Processing and Psychopictorics,* Academic Press, New York, 1970.

[Price 84] Price K.E., "Matching Closed Contours." *Proceedings of the Seventh International Conference on Pattern Recognition* pp. 990-992, 1984.

[Pun 80] Pun T., " A New Method for Gray-Level Picture Thrsholding Using the Entropy of the Histogram." *Signal Processing,* **Vol. 2**, pp. 223-237, 1980.

[Pun 81] Pun T., "Entropic Thrsholding: A New Approach." *Computer Vision, Graphics and Image Processing,* **No. 16**, pp. 210-239, 1981.

[Roberts 65] Roberts L.G., "Machine Perception of Three-dimensional Solids." In J.P. Tipp.et et al., Editors, *Optical and Electro-*

Optical Information Processing, MIT Press, Cambridge, MA, 1965.

[Rosenfeld 82] Rosenfeld A. and Kak A.C., *Digital Picture Processing.* **Vol 1 & 2**, Academic Press, New York, Second Edition, 1982.

[Rummel 84] Rummel P., and Beutel W., "Workpiece Recognition and Inspection by a Model-Based Scene Analysis System." *Pattern Recognition,* **No. 17**, pp. 141-148, 1984.

[Sacerdoti 77] Sacerdoti E., *A Structure for Plans and Behaviour,* Elservier, North-Holland, New York, 1977.

[Sardis 79] Sardis G.N., and Brandin D.M., "An Automatic Surface Inspection System for Flat Rolled Steel." *Automatica,* **No. 15**, pp. 505-520, 1979.

[Seitz 89] Seitz P., "The Robust Recognition of Object Primitives using Local Axes of Symmetry." *Signal Processing,* pp. 89-108, 1989.

[Shapiro 83] Shapiro L.G., "Computer Vision Systems: Past, Present, and Future." *in NATO ASI F-4: Practical Data Analysis,* Springer-Verlag, pp. 199-237, Berlin 1983.

[Shapiro 90] Shapiro L.G., and Haralick, R.M., "Matching Relational Structures Using Discrete Relaxation." *In Syntactic and Structural Pattern Recognition, Theory and Applications,* World Scientific, Teaneck, N.J., 1990.

[Stefik 81]	Stefik M., "Planning and Metaplanning." *In Readings in Artificial Intelligence,* Tioga Publishing, Palo Alto, California, pp. 272-286, 1981.
[Stockman 82]	Stockman G.C., Kopstein S., and Benett S., "Matching Images to Models for Registration and Object Detection Via Clustering." *IEEE Trans. Patt. Anal. Machine Intell.,* **Vol. 4**, pp. 229-241, 1982.
[Subbarao 88]	Subbarao M. *Interpretation of Visual Motion: A Computational Study,* Pitman Publishing, 1988.
[Suresh 83]	Suresh B.R., Fundakowski R.A., and Levitt T.S., "A real-Time Automated Visual Inspection System for Hot Steel Slabs." *IEEE Trans. Pattern Analysis and Machine Intelligence,* **PAMI-4**, pp. 242-249, 1983.
[Sussman 75]	Sussman G.J., *A Computer Model of Skill Acquisition,* Elsevier, North-Holland, New York, 1975.
[Svedlow 76]	Svedlow M., McGillem C.D., and Anuta P.E., "Experimental examination of similarity measures and preprocessing methods used for image registration." *In The Symposium on Machine Processing of Remotelt Sensed Data*, pp. 4-9, June 1976.
[Tate 77]	Tate A., "Generating Project Networks." *Proceedings IJCAI-77,* Cambridge, Massachusetts, pp. 888-893, 1977.

[Turney 85] Turney J.L., Mudge T.N. and Volz R.A., "Recognising partially occluded parts." *IEEE Trans. Pattern Analysis and Machine Intelligence*, **PAMI-7**, pp. 410-421, 1985.

[Venot 84] Venot A., Lebruchec J.F., and Roucayrol J.C., "A new class of similarity measures for robust image registration." *Compt. Vision Graph. Image Process.*, **No. 28**, pp. 176-184, 1984.

[Vere 83] Vere S., "Planning in Time: Windows and Durations for Activities and Goals." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **Vol. 5,** pp 246-267, 1983.

[Waldinger 81] Waldinger R., "Achieving Several Goals Simultaneously.." *In Readings in Artificial Intelligence,* Tioga Publishing, Palo Alto, California, pp. 250-271, 1981.

[Weszka 78] Weszka J.S., "Survey: A Survey of Threshold selection techniques." *Computer Graphics and Image Processing,* **Vol. 7**, pp. 259-265, 1978.

[Wilensky 83] Wilensky R., *Planning and Understanding A Computational Approach to Human Reasoning,* Addison-Wesley, Reading Massachusetts, 1983.

[Winston 75] Winston P.H., "Learning Structural Descriptions from Examples." *In The Psychology of Computer Vision,* McGraw Hill, New York, 1975.

[Winston 77]	Winston P.H., *Artificial Intelligence, Second Editio,.* Addison-Wesley, Reading, Massachusetts, 1977.
[Yakimovsky 73]	Yakimovsky Y.Y. and Feldman J., "A Semantic-Based Decision Theoretic Region Analyzer." In *Proc. 3rd Int. Conf. Artificial Intelligence,* pp. 580-588, 1973.
[Yu 88]	Yu S.S., Cheng W.C., and Chiang C.S.C., "Printed Circuit Board Inspection System PI/1." *In Automated Inspection and High Speed Vision Architecture II,* pp. 126-134, 1988.

Published Work.

[1] Panayiotou S., "Use of Multi-Scanner Input in 3D Artificially Intelligent Imaging Systems." *Proceedings of the Institute of Measurement & Control in Instrumentation and Control,* **IMC-2**, pp. 150-167, Nottingham, March 1993.

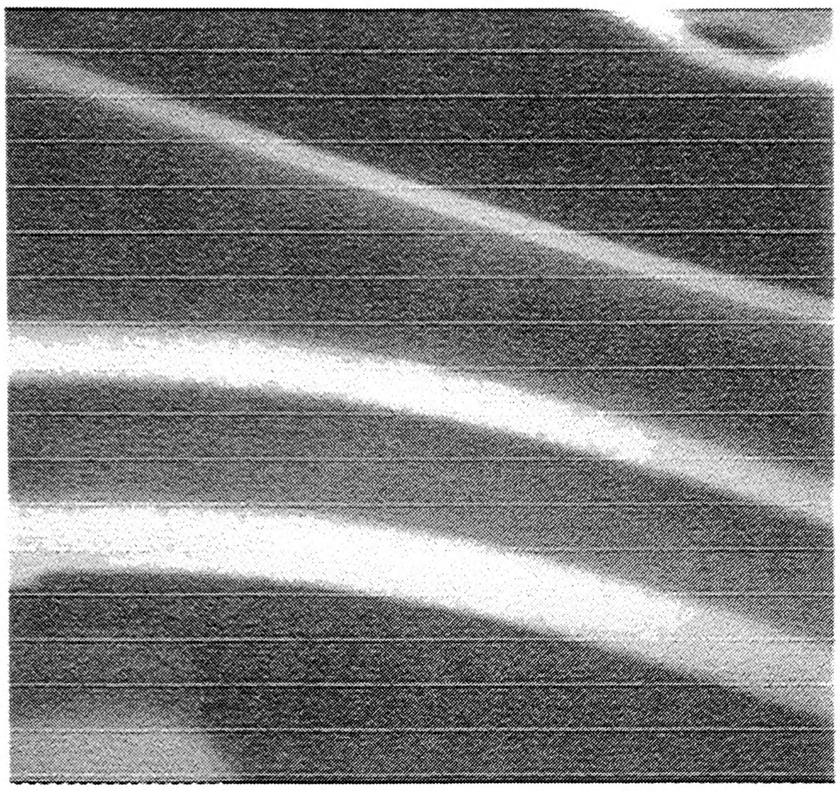
[2] Panayiotou S., and Soper A., "Artificially Intelligent 3D Industrial Inspection System for Metal Inspection." *Proceedings of the IEEE Southwest Symposium on Image Analysis and Interpretation,* pp. 130-136, Dallas, Texas, April 1984.

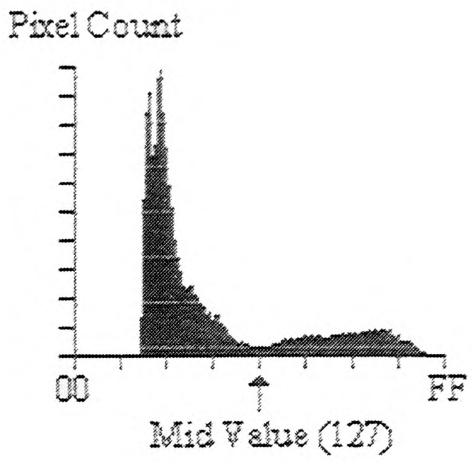
[3] Panayiotou S., and Soper A., "An Adaptive Visual Planning System Applied to the Detection of Faults in Casting." *In Applications of Artificial Intelligence in Engineering X,* Editors G. Rzevski, R.A. Adey and C. Tasso, pp. 163-170, Computational Mechanics Publications, Southhampton, Boston, 1985.

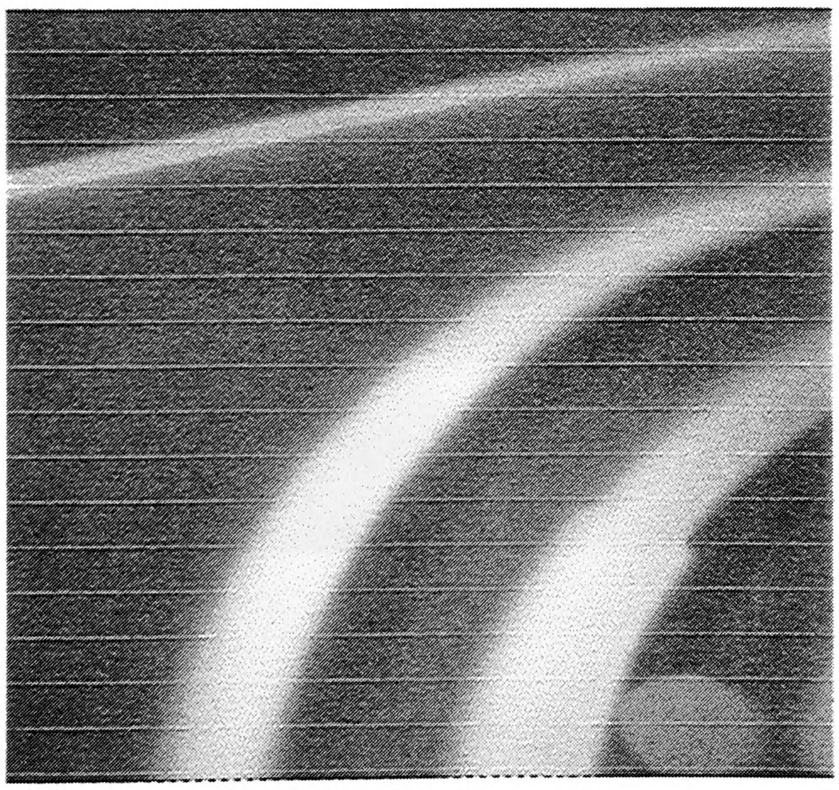
[4] Panayiotou S., and Naqvi W., "Applied Active Databases for Evolving Image Processing Algorithms." **DEXA95** in Press 1995.

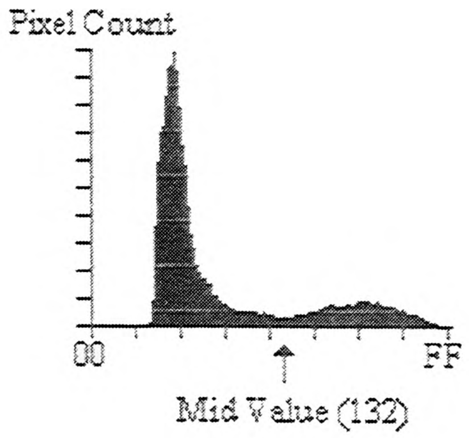
Appendix A
Test Images and Domain

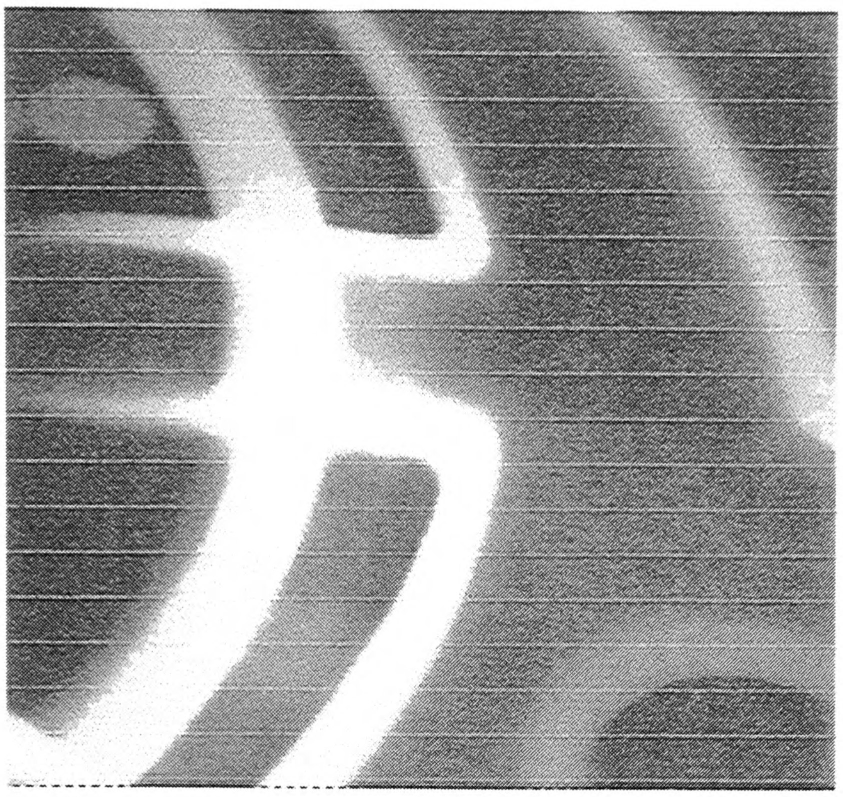
A1. Sample of Domain

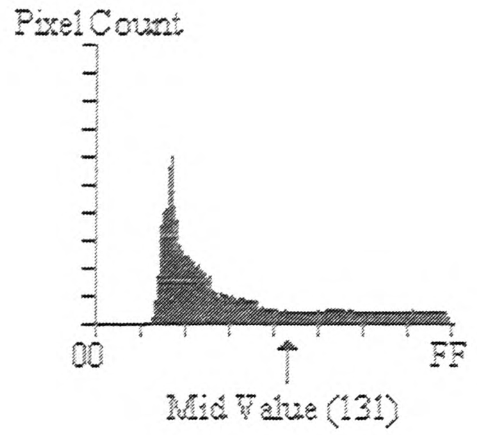


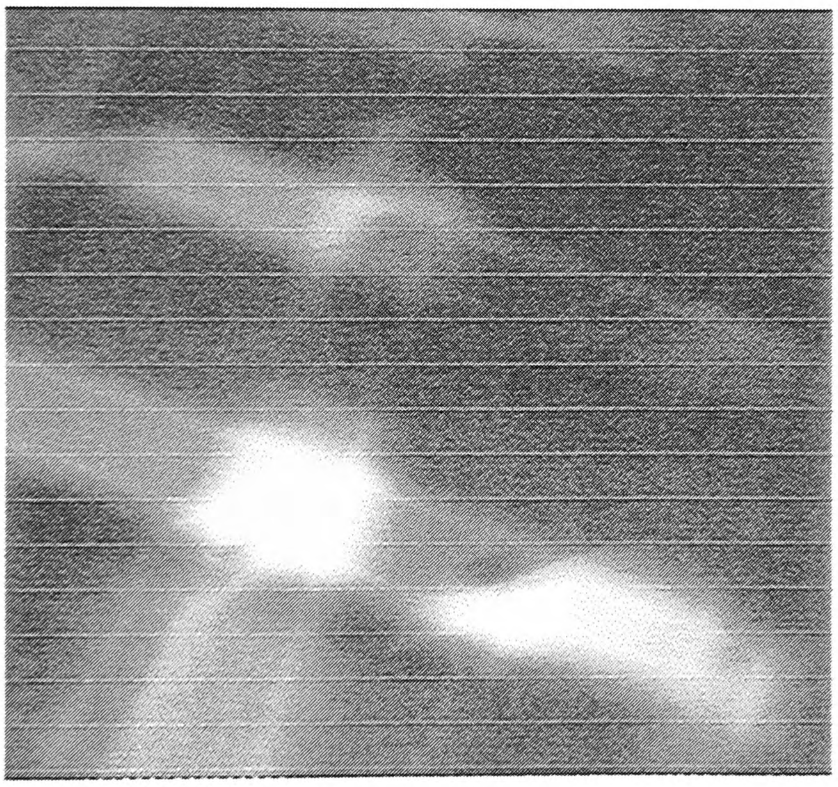


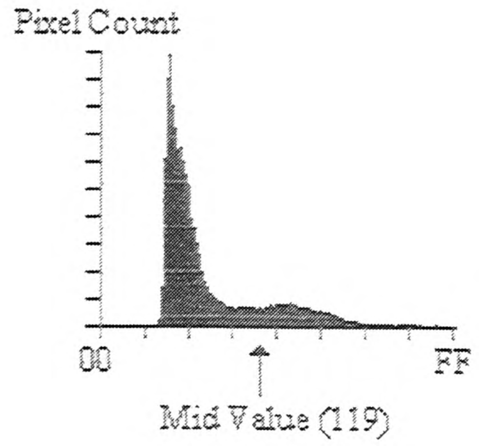







Pixel Count
00
FF
Mid Value (131)




Pixel Count
00
FF
Mid Value (119)

The x-rays above show a sample of clean (no faults) images and their associated histograms. The histograms show that the images are bimodal, and thus with the knowledge of the background intensity values we can segment the background from the foreground using a simple thresholding technique. Figure A1 shows the average histogram of clean images, the background intensity range, and foreground intensity range.

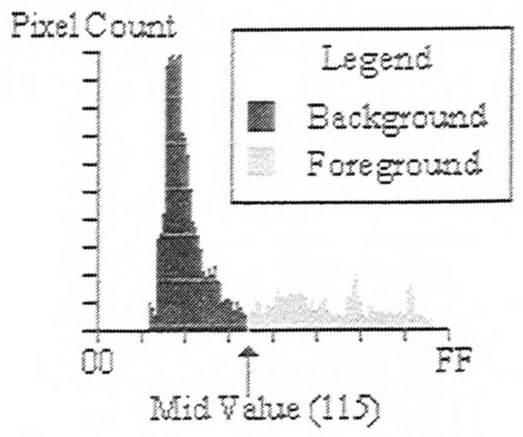


Figure A1 Average histogram.

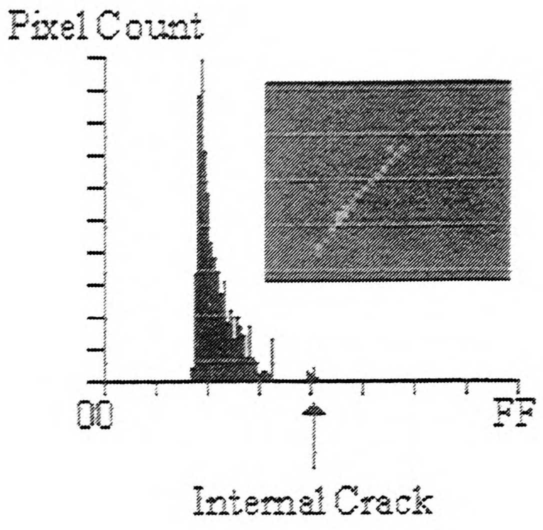
The table below (Table A1) shows the image characteristics for this domain, based on the set of clean images used in the training stage of the Visual Planner.

Attribute Name	**Value**	
Maximum Background Threshold	136	Intensity Value
Minimum Background Threshold	17	Intensity Value
Maximum (+) Noise Deviation	1570	Pixel Count
Minimum (-) Noise Deviation	-1522	Pixel Count

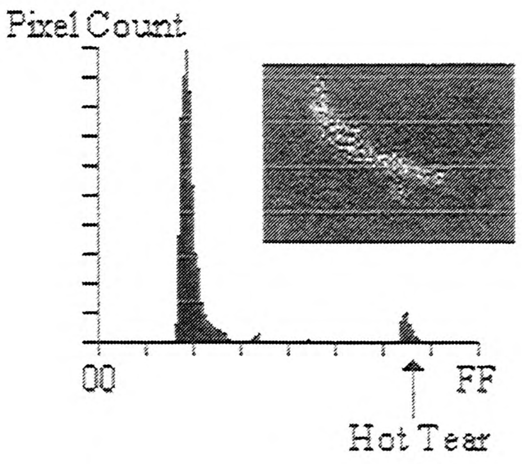
Table A1 Image Characteristics.

A2 Sample of Object Attributes

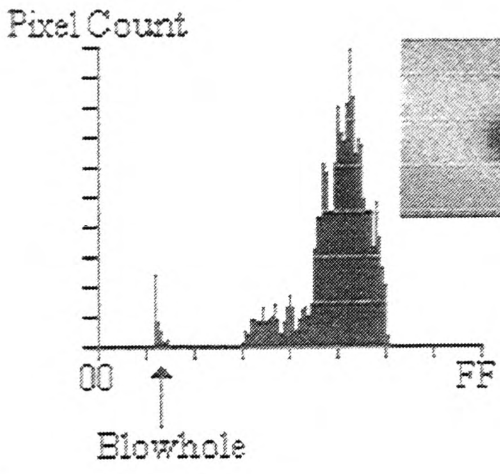
Object Attribute	Value
Main Class	Crack
Contextual Information	Linear
Name	Internal Crack
Average Intensity	127
Max Intensity Value	132
Min Intensity Value	120
Area	43
Lines	1
Major Axis	16
Minor Axis	3

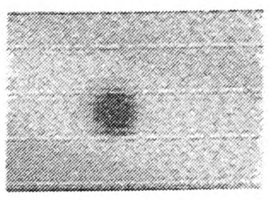


Object Attribute	Value
Main Class	Crack
Contextual Information	Linear
Name	Hot Tear
Average Intensity	239
Max Intensity Value	241
Min Intensity Value	234
Area	65
Lines	1
Major Axis	26
Minor Axis	10

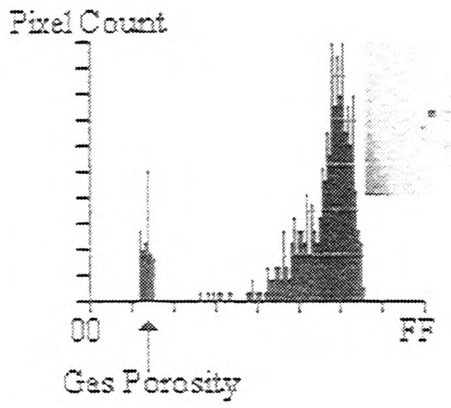


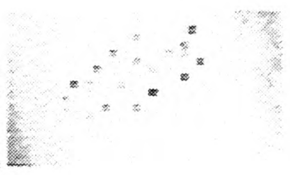
Object Attribute	Value
Main Class	Hole
Contextual Information	Blob
Name	Blowhole
Average Intensity	35
Max Intensity Value	39
Min Intensity Value	32
Area	67
Major Axis	17
Minor Axis	14





Object Attribute	Value
Main Class	Hole
Contextual Information	Blob
Name	Gas Porosity
Average Intensity	40
Max Intensity Value	45
Min Intensity Value	34
Area	5
Major Axis	3
Minor Axis	2





Please note: Figures of faults are not to scale.

Appendix B
Test Results

B1 Test Results on 200 Normal Images

Defect Code and ID No.

	Gas 11	Por 12	B/H 13	SH 14	S/HT 17	FL 18	DI 20	CR 21	HT 22	C/Br 23
Tsize	30	10	50	20	20	20	10	60	10	10
Test	20	20	20	20	20	20	20	20	20	20

Results at 60% Matching Strength

✔	20	19	20	20	20	20	20	20	18	19
✖	0	1	0	0	2	0	0	0	2	1
?	1	0	0	0	2	0	0	0	0	0
i	d	b			c,c				e,e	e

Results at 65% Matching Strength

✔	20	19	20	20	20	20	18	20	19	19
✖	0	1	1	0	0	1	2	0	1	1
?	0	0	1	0	0	2	0	0	0	1
i		b	c			c,a	b,b		e	c

Results at 70% Matching Strength

✔	19	18	19	20	20	20	19	20	19	20
✖	1	2	1	0	0	0	1	0	1	0
?	0	2	2	0	0	2	0	0	0	0
i	e	c,c	c,d			d,d	b		d	

Results at 75% Matching Strength

✔	18	19	20	18	20	19	19	19	20	20
✖	2	1	0	2	0	1	1	1	0	0
?	1	0	1	2	0	2	0	0	0	0
i	c,e	b	d	a,c,d		b,d,d	b	a		

Results at 80% Matching Strength

✔	19	19	18	19	19	18	20	20	20	20
✖	1	1	2	1	1	2	0	0	0	0
?	2	0	2	2	0	2	0	1	0	0
i	d,d,e	a	b,c,d	c,d,e	e	a,d,d, e		d		

Results at 85% Matching Strength

✔	19	19	20	18	20	19	19	19	20	20
✖	1	1	0	2	0	1	1	1	0	0
?	1	2	0	1	1	1	2	1	1	0
i	b,d	c,d		a,d,e	d	d,e	c,d	a,d	c	

Results at 90% Matching Strength

✔	20	19	20	19	20	18	19	20	18	20
✖	0	1	2	1	0	2	1	0	2	0
?	3	1	2	2	1	3	2	0	2	0
i	d,d,d	a,c	b,c,d	e,d,d	d	c,d,d, e	a,d,d		a,d,d, e	

Results at 95% Matching Strength

✔	18	19	19	19	19	18	20	20	19	20
✖	2	1	1	1	1	2	0	0	1	0
?	5	4	6	6	5	10	6	2	4	0
i	a,c, 4*d	a, 4*d	e, 6*d	c, 5*d	c, 4*d	b,e, 10*d,	6*d	d,d	d,d,d, d,e	

Results at 100% Matching Strength

✔	18	19	19	20	20	18	19	20	19	20
✖	2	1	1	2	0	2	1	0	1	0
?	16	17	18	15	17	18	13	16	14	5
i	a,c, 15*d	b, 17*d	a, 18*d	b,c, 14*d	17*d	a,b, 18*d	13*d, e	16*d	b, 14*d	5*

KEY:

Tsize	: Test Image Sample Size
✔	: Correct
✖	: Incorrect
?	: Uncertain
i	: Information on errors:

a: Wrongly identified fault i.e. picked up another fault
b: Picked up false target
c: Uncertain of object and object absent
d: Uncertain of object and object present
e: Object not picked up i.e. sub-plan failed.

B2 Test Results on 200 Images with Environmental Changes

Change in lighting and noise (out of focus, dirt on lens etc.)

5 images tested with increased lighting conditions

5 images tested with reduced lighting conditions

5 images tested with camera out of focus

5 images tested with a filter in front of lens to introduce random noise.

Defect Code and ID No.

	Gas 11	Por 12	B/H 13	SH 14	S/HT 17	FL 18	DI 20	CR 21	HT 22	C/E 23
Tsize	30	10	50	20	20	20	10	60	10	10
Test	20	20	20	20	20	20	20	20	20	20

Results at 60% Matching Strength

✔	18	16	19	19	19	20	18	20	19	20
✖L+	1	2								
✖L-		1		1			1			
✖N	1	1	1		1		1		1	
✖F										
?	0	2	0	0	0	0	0	0	0	0
i	e,e	a,a,c, c,e,e	b	e	a		a,a		b	

Results at 65% Matching Strength

✔	19	17	19	18	19	20	17	19	20	20
✖L+	1	1		1						
✖L-		1		1	1		2	1		
✖N		1					1			
✖F			1							
?	0	1	0	0	0	1	0	1	0	0
i	b	a,a,b, d	e	b,b	b	d	b,b,e	b,d		

Results at 70% Matching Strength

✔	18	18	19	19	19	19	17	19	20	20
✖L+	2	1	1		1		2			
✖L-		1		1				1		
✖N							1			
✖F						1				
?	0	0	0	2	0	1	1	0	0	0
i	b,b	b,e	b	b,d,d	b	b,d	a,b,b, c	b		

Results at 75% Matching Strength

✔	17	17	19	19	19	19	18	19	20	20
✖L+	1	2	1	1			1			
✖L-	2					1		1		
✖N		1					1			
✖F					1					
?	2	0	1	0	0	1	2	0	0	0
i	b,b,b, d,d	b,b,b	b	a	b	b,d	b,b,d, d	b		

Results at 80% Matching Strength

✔	18	17	19	19	19	19	17	19	19	20
✖L+	2	3							1	
✖L-					1	1	2	1		
✖N			1				1			
✖F				1						
?	3	4	1	1	3	2	2	1	0	0
i	b,b,d, d,d	a,b,b, 4*d	b,d	d,e	b,d,d, d	b,d,d	a,b,b, d,d	b,d		

Results at 85% Matching Strength

✔	17	18	19	18	19	19	18	19	20	20
✖L+	2	1	1		1		1			
✖L-	1	1		2						
✖N							1	1		
✖F						1				
?	4	4	5	5	6	3	5	4	1	0
i	b,b,b, 4*d	b,b, 4*d	b,5*d	a,b, 5*d	a,6*d	b,d	b,c, 5*d	b,4*d	d	

Results at 90% Matching Strength

✔	17	16	19	18	19	20	18	19	19	20
✖L+	3	3					1			
✖L-		1		2			1	1		
✖N			1		1				1	
✖F										
?	9	7	10	12	14	12	10	9	2	1
i	a,b,b, c,8*d	b,b,c, c,5*d	b, 10*d	b,c, 11*d	b, 14*d	12*d	a,c, 9*d	b,9*d	b,2*d	d

Results at 95% Matching Strength

✔	18	17	19	18	19	20	17	19	19	20
✖L+	2	1		1			1	1		
✖L-		2			1		2			
✖N				1					1	
✖F			1							
?	19	16	18	17	18	20	16	15	9	7
i	b,b, 19*d	a,b,b, 16*d	a, 18*d	b,b, 17*d	c, 18*d	20*d	b,b,c, 15*d	b, 15*d	c,8*d	7*

Results at 100% Matching Strength

✔	18	17	19	18	18	19	17	19	20	20
✖L+	1	2		2			1			
✖L-	1	1	1		1	1	1	1		
✖N							1			
✖F					1					
?	18	17	19	17	18	19	17	20	20	20
i	b,b,2 18*d	b,b,c, 16*d	b,19* d	b,c, 16*d	a,b, 18*d	b, 19*d	b,b,c, 16*d	c, 19*d	20*d	20*

KEY:

	Tsize	: Test Image Sample Size
	✔	: Correct
	✖L+	: Incorrect when light is added
	✖L-	: Incorrect when light is subtracted
	✖N	: Incorrect when noise is added
	✖F	: Incorrect when out of focus
	?	: Uncertain
	i	: Information on errors:

a: Wrongly identified fault i.e. picked up another fault
b: Picked up false target
c: Uncertain of object and object absent
d: Uncertain of object and object present
e: Object not picked up i.e. sub-plan failed.

B3 Chi Square Test on Normal Images

60% Matching Strength	**Observed**	**Expected**	**(O-E)**	**$(O-E)^2$**	**$\frac{(O-E)^2}{E}$**
Correct	99	95	4	16	0.168421
Incorrect	1	2.5	-1.5	2.25	0.9
Uncertain	0	2.5	-2.5	6.25	2.5
	100	100	0		3.568421

X^2 = 3.568421

65% Matching Strength	**Observed**	**Expected**	**(O-E)**	**$(O-E)^2$**	**$\frac{(O-E)^2}{E}$**
Correct	98	95	3	9	0.094737
Incorrect	2	2.5	-0.5	0.25	0.1
Uncertain	0	2.5	-2.5	6.25	2.5
	100	100	0		2.694737

X^2 = 2.694737

70% Matching Strength	**Observed**	**Expected**	**(O-E)**	**$(O-E)^2$**	**$\frac{(O-E)^2}{E}$**
Correct	96	95	1	1	0.010526
Incorrect	3	2.5	0.5	0.25	0.1
Uncertain	1	2.5	-1.5	2.25	0.9
	100	100	0		1.010526

X^2 = 1.010526

75% Matching Strength	**Observed**	**Expected**	**(O-E)**	**$(O-E)^2$**	**$\frac{(O-E)^2}{E}$**
Correct	97	95	2	4	0.042105
Incorrect	1	2.5	-1.5	2.25	0.9
Uncertain	2	2.5	-0.5	0.25	0.1
	100	100	0		1.042105

X^2 **= 1.042105**

80% Matching Strength	**Observed**	**Expected**	**(O-E)**	**$(O-E)^2$**	**$\frac{(O-E)^2}{E}$**
Correct	93	95	-2	4	0.042105
Incorrect	3	2.5	0.5	0.25	0.1
Uncertain	4	2.5	1.5	2.25	0.9
	100	100	0		1.042105

X^2 **= 1.042105**

85% Matching Strength	**Observed**	**Expected**	**(O-E)**	**$(O-E)^2$**	**$\frac{(O-E)^2}{E}$**
Correct	93	95	-2	4	0.042105
Incorrect	2	2.5	-0.5	0.25	0.1
Uncertain	5	2.5	2.5	6.25	2.5
	100	100	0		2.642105

X^2 **= 2.642105**

90% Matching Strength	**Observed**	**Expected**	**(O-E)**	**$(O-E)^2$**	**$\frac{(O-E)^2}{E}$**
Correct	90	95	-5	25	0.263158
Incorrect	3	2.5	0.5	0.25	0.1
Uncertain	7	2.5	4.5	20.25	8.1
	100	100	0		8.463158

X^2 = 8.463158

95% Matching Strength	**Observed**	**Expected**	**(O-E)**	**$(O-E)^2$**	**$\frac{(O-E)^2}{E}$**
Correct	76	95	-19	361	3.8
Incorrect	4	2.5	1.5	2.25	0.9
Uncertain	20	2.5	17.5	306.25	122.5
	100	100	0		127.2

X^2 = 127.2

100% Matching Strength	**Observed**	**Expected**	**(O-E)**	**$(O-E)^2$**	**$\frac{(O-E)^2}{E}$**
Correct	17	95	-78	6084	64.04211
Incorrect	4	2.5	1.5	2.25	0.9
Uncertain	79	2.5	76.5	5852.25	2340.9
	100	100	0		2405.842

X^2 = 2405.842

B4 Chi Square Test on Changed Environment

60% Matching Strength	Observed	Expected	(O-E)	$(O-E)^2$	$\frac{(O-E)^2}{E}$
Correct	95	95	0	0	0
Incorrect	4	2.5	1.5	2.25	0.9
Uncertain	1	2.5	-1.5	2.25	0.9
	100	100	0		3.8

$X^2 = 3.8$

65% Matching Strength	Observed	Expected	(O-E)	$(O-E)^2$	$\frac{(O-E)^2}{E}$
Correct	94	95	-1	1	0.010526
Incorrect	5	2.5	2.5	6.25	2.5
Uncertain	1	2.5	-1.5	2.25	0.9
	100	100	0		3.410526

$X^2 = 3.410526$

70% Matching Strength	Observed	Expected	(O-E)	$(O-E)^2$	$\frac{(O-E)^2}{E}$
Correct	93	95	-2	4	0.042105
Incorrect	6	2.5	3.5	12.25	4.9
Uncertain	1	2.5	-1.5	2.25	0.9
	100	100	0		5.842105

$X^2 = 5.842105$

75% Matching Strength	**Observed**	**Expected**	**(O-E)**	**$(O-E)^2$**	**$\frac{(O-E)^2}{E}$**
Correct	92	95	-3	9	0.094737
Incorrect	5	2.5	2.5	6.25	2.5
Uncertain	3	2.5	0.5	0.25	0.1
	100	100	0		2.694737

X^2 = 2.694737

80% Matching Strength	**Observed**	**Expected**	**(O-E)**	**$(O-E)^2$**	**$\frac{(O-E)^2}{E}$**
Correct	86	95	-9	81	0.852632
Incorrect	6	2.5	3.5	12.25	4.9
Uncertain	8	2.5	5.5	30.25	12.1
	100	100	0		17.85263

X^2 = 17.85263

85% Matching Strength	**Observed**	**Expected**	**(O-E)**	**$(O-E)^2$**	**$\frac{(O-E)^2}{E}$**
Correct	77	95	-18	324	3.410526
Incorrect	7	2.5	4.5	20.25	8.1
Uncertain	16	2.5	13.5	182.25	72.9
	100	100	0		84.41053

X^2 = 84.41053

90% Matching Strength	**Observed**	**Expected**	**(O-E)**	**$(O-E)^2$**	**$\frac{(O-E)^2}{E}$**
Correct	50	95	-45	2025	21.31579
Incorrect	7	2.5	4.5	20.25	8.1
Uncertain	43	2.5	40.5	1640.25	656.1
	100	100	0		685.5158

X^2 = 685.5158

95% Matching Strength	**Observed**	**Expected**	**(O-E)**	**$(O-E)^2$**	**$\frac{(O-E)^2}{E}$**
Correct	16	95	-79	6241	65.69474
Incorrect	6	2.5	3.5	12.25	4.9
Uncertain	78	2.5	75.5	5700.25	2280.1
	100	100	0		2350.695

X^2 = 2350.695

100% Matching Strength	**Observed**	**Expected**	**(O-E)**	**$(O-E)^2$**	**$\frac{(O-E)^2}{E}$**
Correct	0	95	-95	9025	95
Incorrect	7	2.5	4.5	20.25	8.1
Uncertain	93	2.5	90.5	8190.25	3276.1
	100	100	0		3379.2

X^2 = 3379.2

B5 Images Showing Environmental Changes

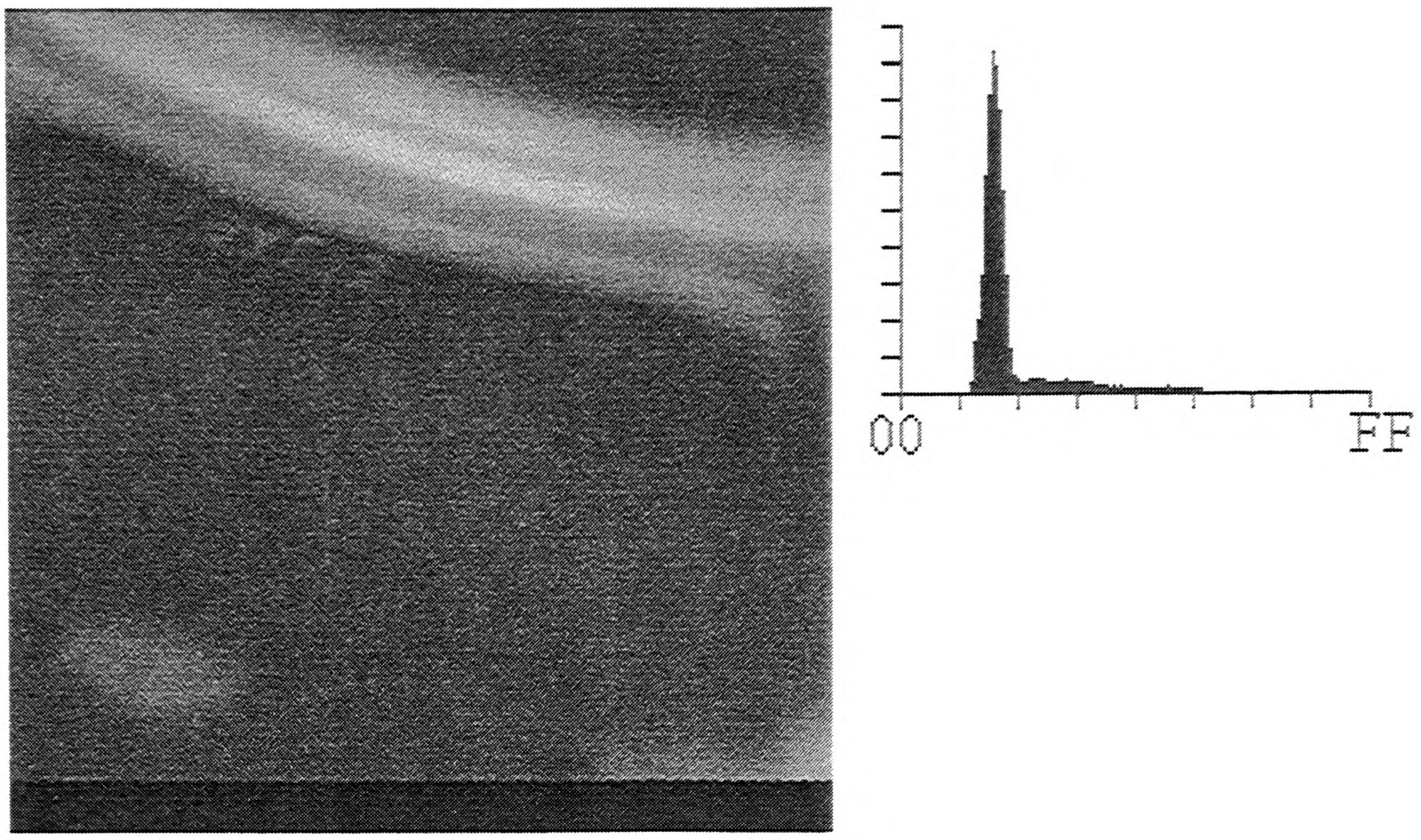


Figure B1 Normal image.

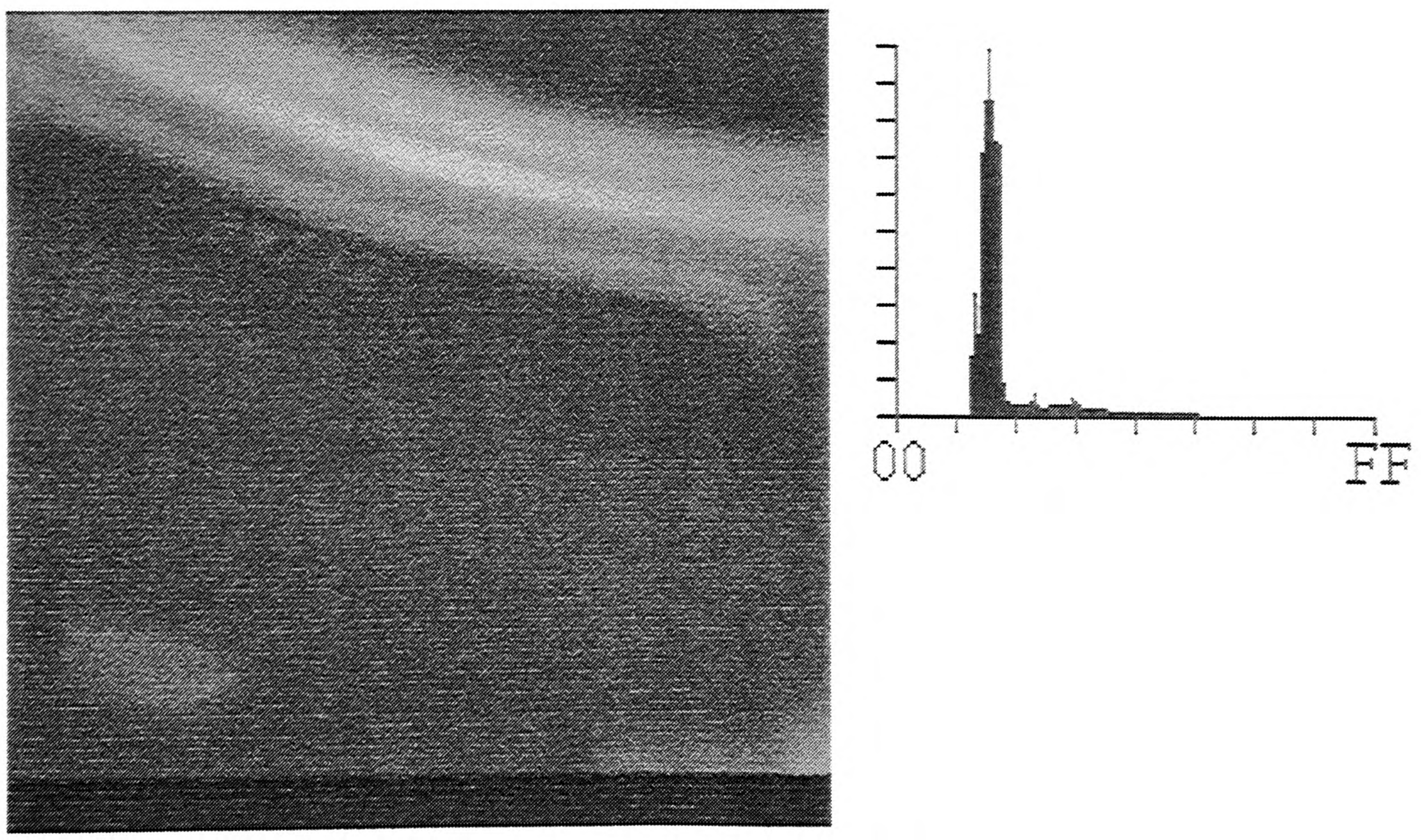


Figure B2. Image out of focus.

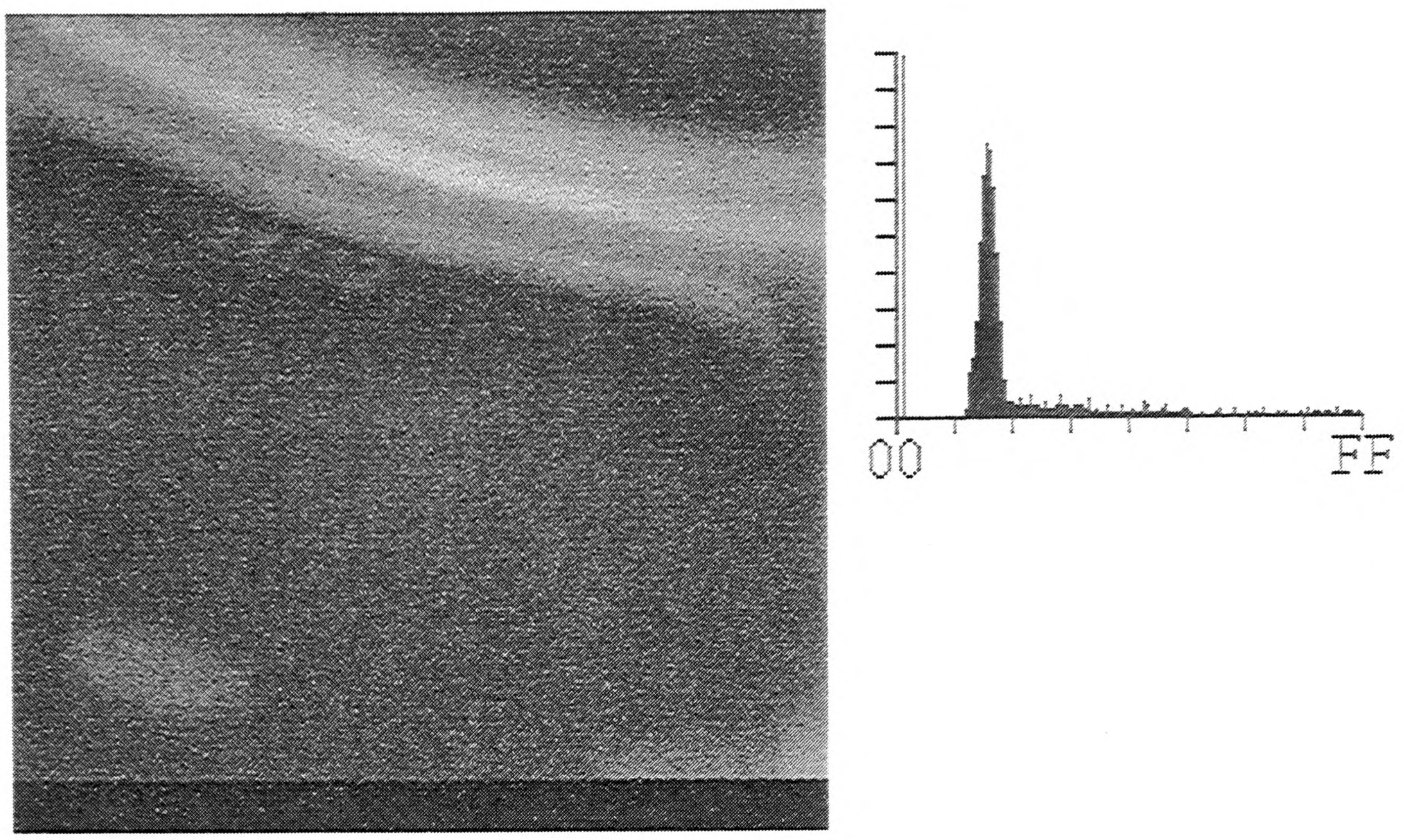


Figure B3. Image with random noise.

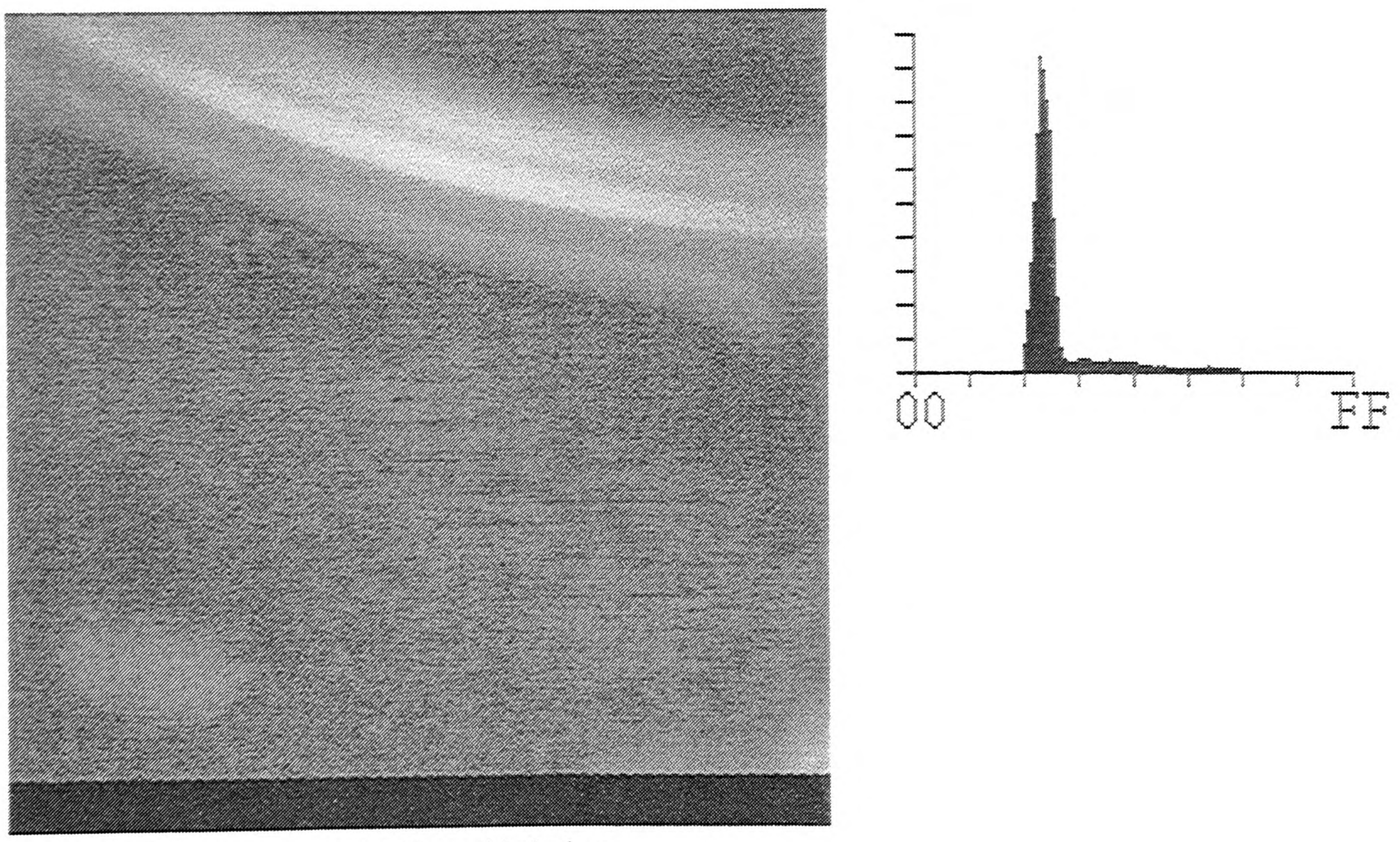
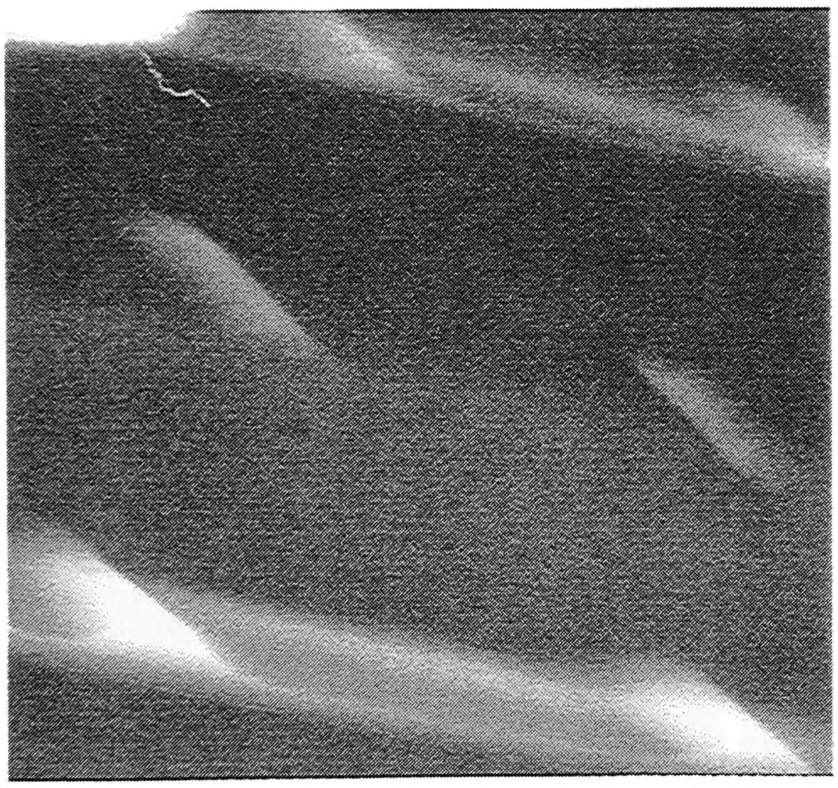


Figure B4. Image with low lighting.

B6 Anderson's Parameter Tuning V the Visual Planner

The figures below show the difference between Anderson's (Anderson 74) parameter tuning system, compared to the Visual Planner. The results show that parameter tuning has a limited success over the Visual Planner. The segmentation process is not as efficient as that produced by the Visual Planner. The parameter tuning system is also slower then visual planner. This is due to the fact that with each parameter change the algorithms had to be executed.

B6.1 Test on Non-Corrupted Image



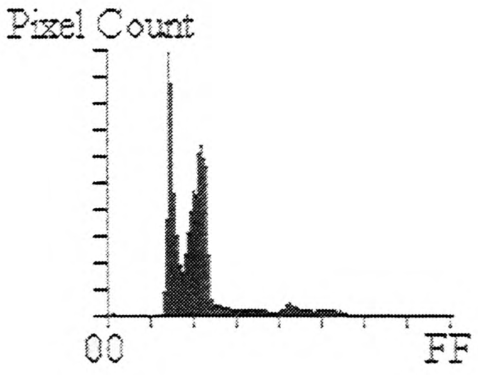


Figure B5 Non-corrupt image with Internal Crack

B6.1.1 Anderson's Parameter Tuning System Results.

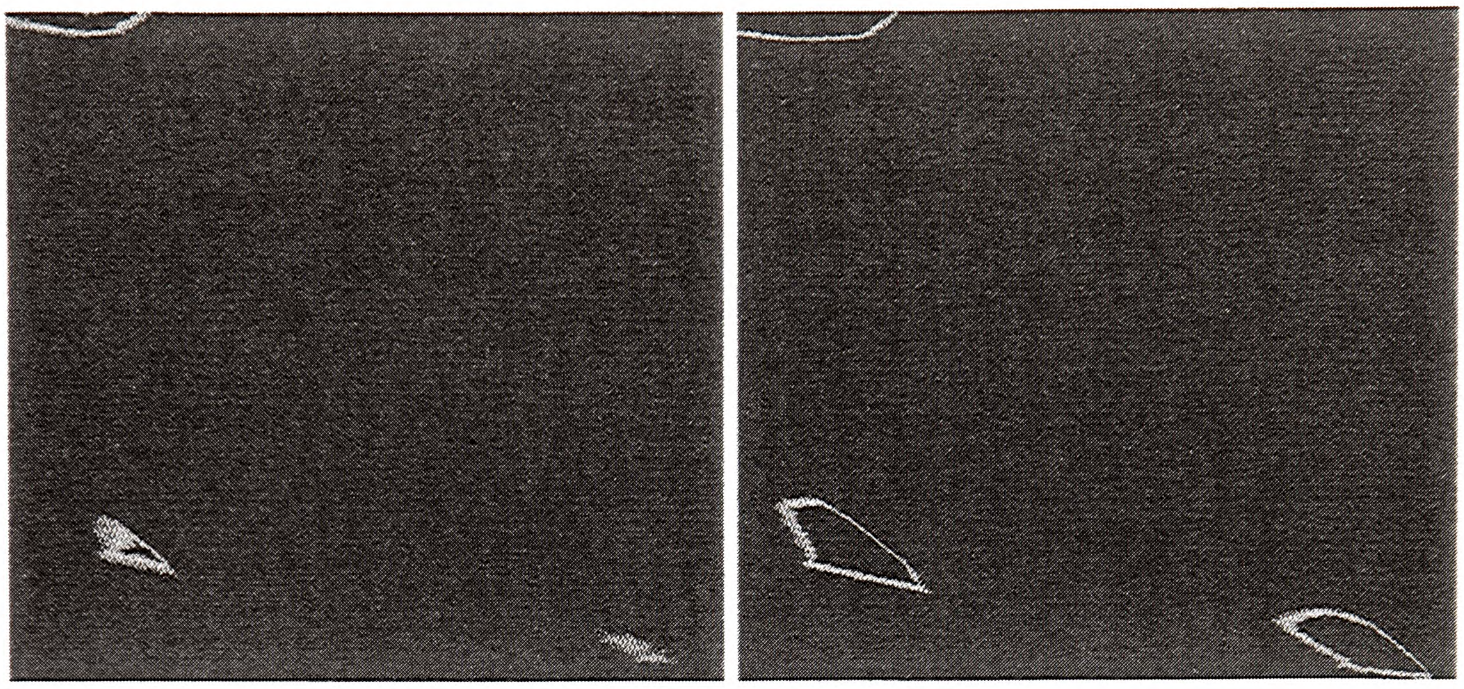


Figure B6 a) Threshold at 240, under-segmentation.
b) Threshold at 200, under-segmentation

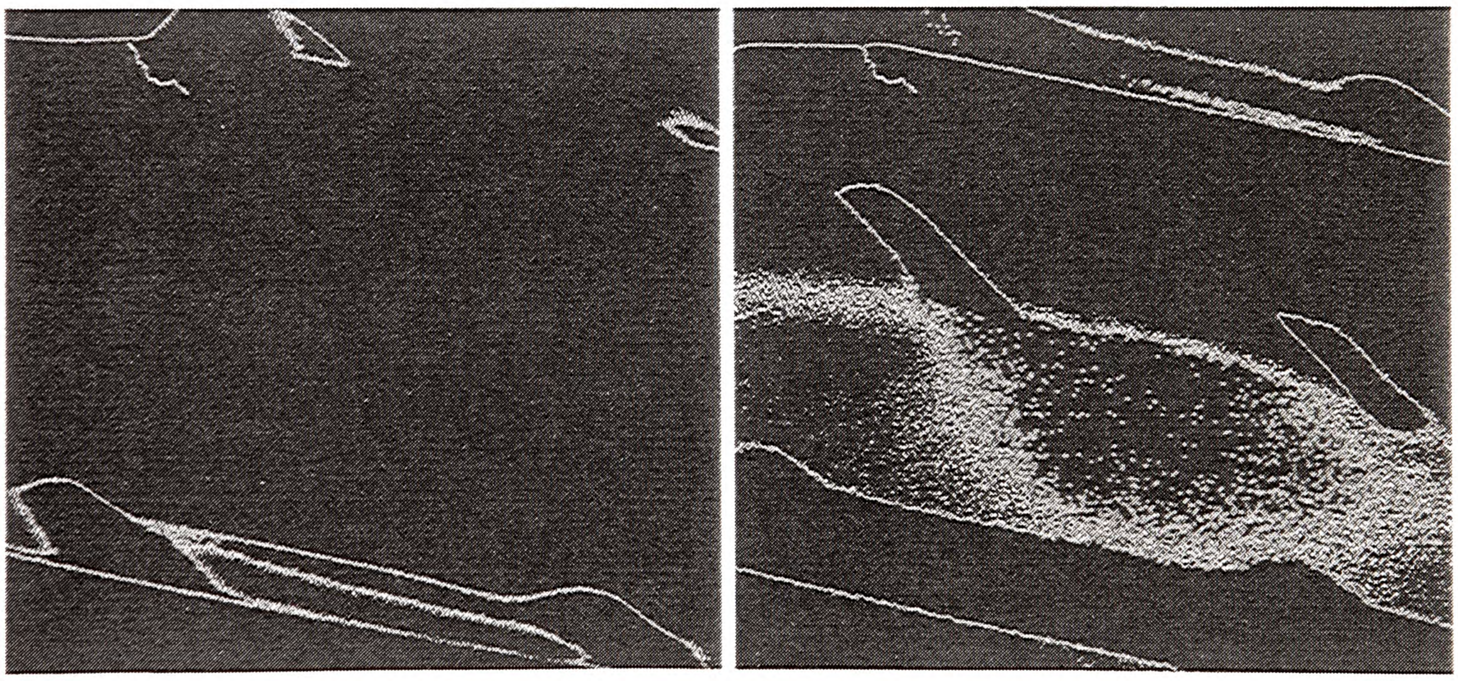


Figure B6 c) Threshold at 150, Internal Crack found.
d) Threshold at 100, over-segmentation.

The matching algorithm found the crack after 12 iteration using Anderson's parameter tuning system from intensity level 255 down to intensity level 195 with a threshold change of -5 for each intensity level. It continued to find the crack for a further 11 iterations down to intensity level 145.

B6.1.2 The Visual Planner's Segmentation Plans.

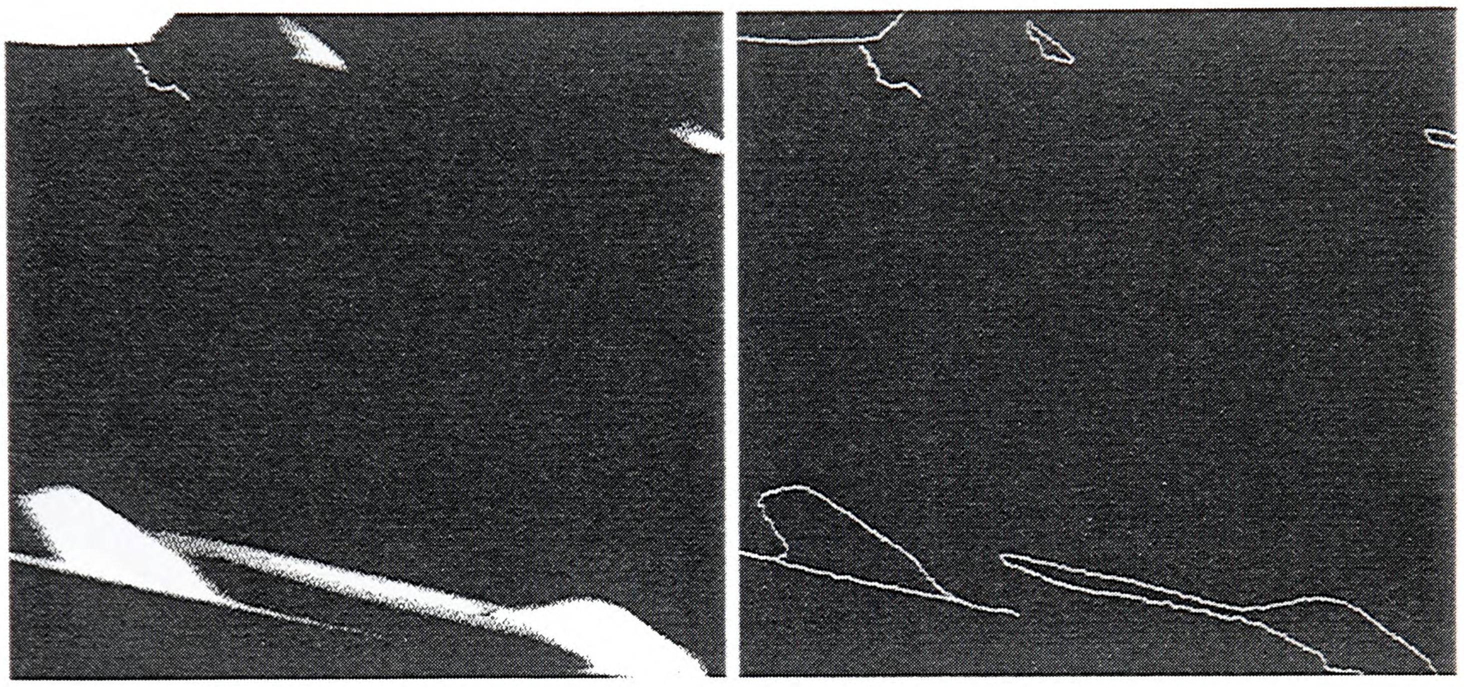


Figure B7 a) Visual Planner's enhancement plan.
b) Visual Planner's segmentation plan.

The Visual Planner Produced the following enhancement and segmentation sub-plans:

Enhancement Plan	**Segmentation Plan**
Median Filter	Sobel Edge Detection *157
Digitisation *Min=143, Max=191	Dilate
	Erode
	Thinning
	Robust Line Rules
	Boundary Following

*Threshold values for algorithms

The Visual Planner went through 2 mutations from the initial algorithm selected before it found a match and produced the segmentation plan above. The digitisation threshold values were set according to the average background intensity value (143) and the maximum object intensity value (191). The threshold value for the edge detector was set at the average object intensity value (157).

B6.2 Test on Corrupted Image

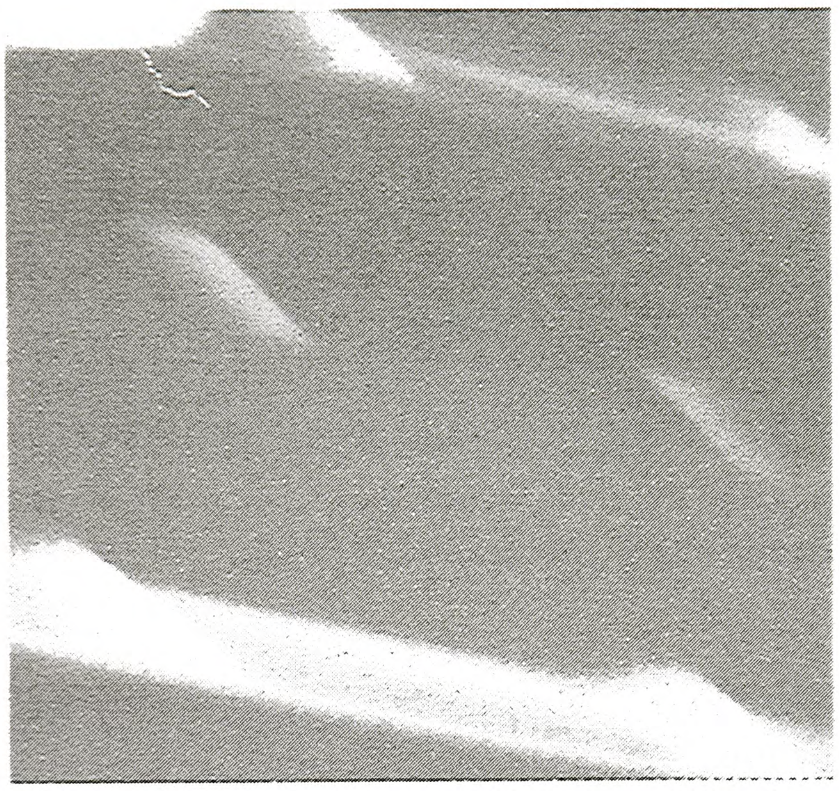
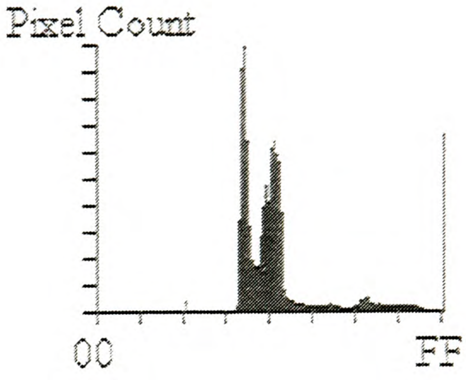


Figure B8 Corrupt image with Internal Crack.



B6.2.1 Anderson's Parameter Tuning System on Corrupt Image.

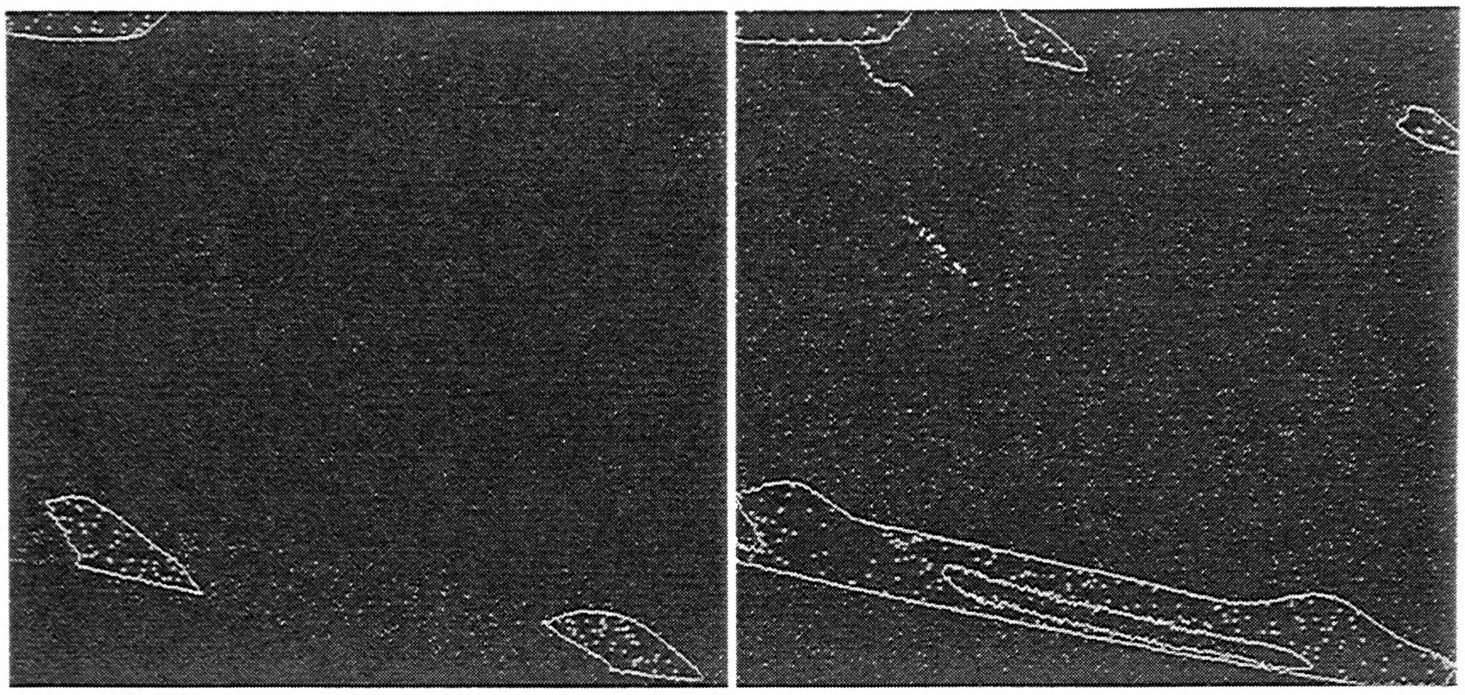


Figure B9 a) Threshold at 240, under-segmentation.
b) Threshold at 200, Crack appears, but image too corrupt.

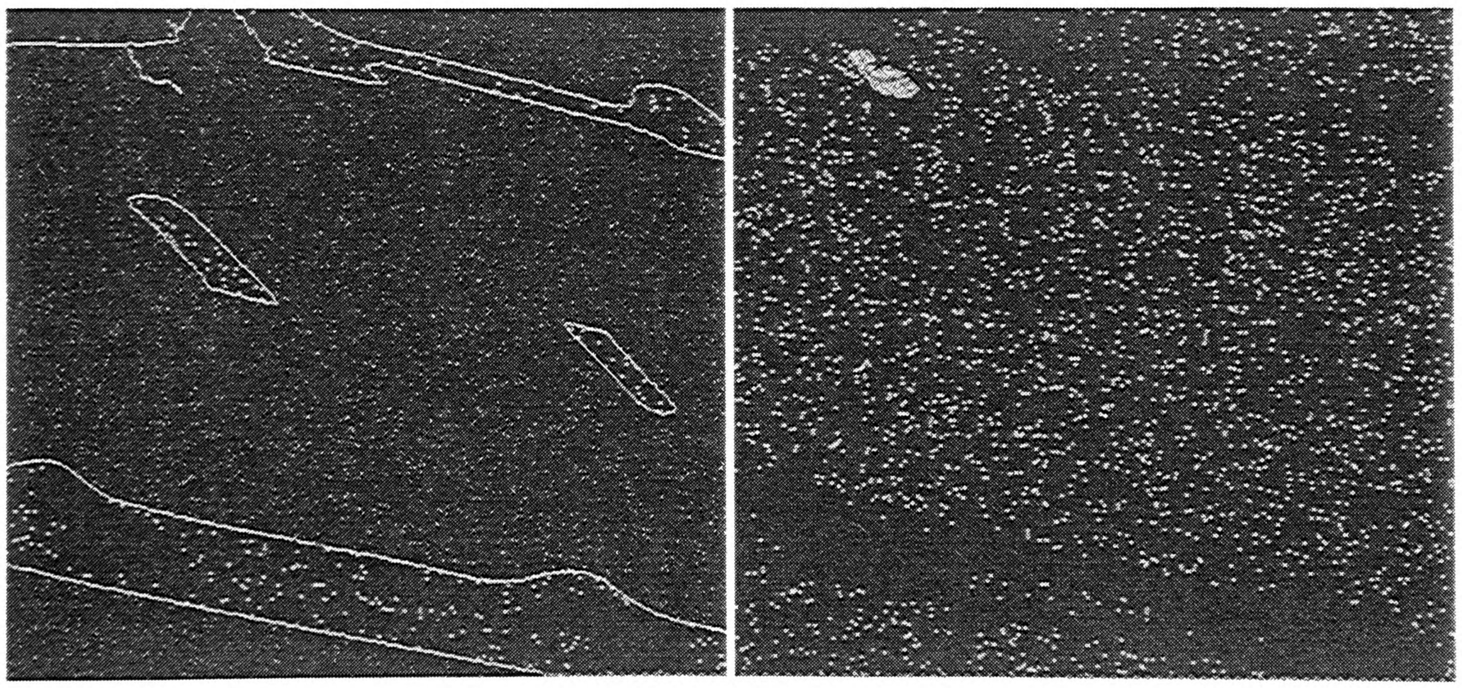


Figure B9 c) Threshold at 150, over-segmentation.
d) Threshold at 100, over-segmentation.

Anderson's parameter tuning system could not produce a strong enough segmentation in order to get a match. the system failed with this image.

B6.2.2 The Visual Planner's Segmentation Plans on Corrupt Image.

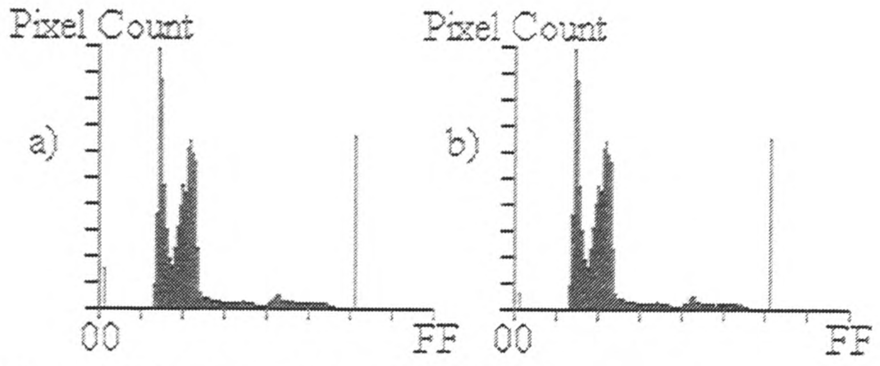


Figure B10 a) Original image histogram.
b) Domain Rules darken image.

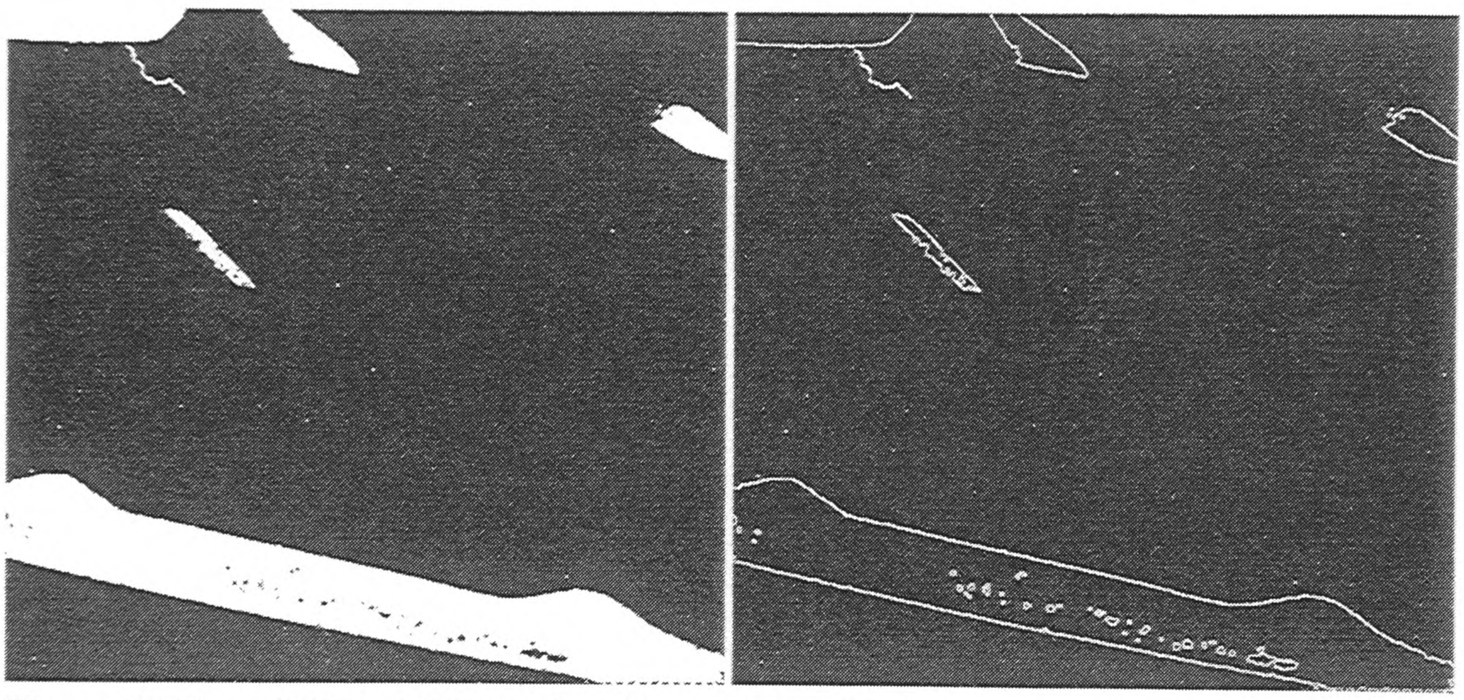


Figure B11 a) Visual Planner's enhancement plan.
b) Visual Planner's segmentation plan.

Enhancement Plan	**Segmentation Plan**
Light Decrease Domain Rule *-25	Sobel Edge Detection *157
Median Filter	Dilate
Digitisation *Min=143, Max=191	Erode
	Thinning
	Robust Line Rules
	Boundary Following

*Threshold values for algorithms

Appendix C
Image Processing and Segmentation Characteristics

Appendix C: Image Processing and Segmentation Characteristics

C1 Introduction

This section discusses and lists the different image processing and segmentation algorithms implemented in this project. It shows the thresholds needed if any and the references where the reader may find the work for a better explanation. It also shows which algorithm succeeds the current algorithm in order for a sub-plan to be formulated. It thus gives the information needed to declare each algorithm to the Visual Planner.

C2 Image Processing Techniques

Here the image processing techniques used within the Visual Planner are given. Image processing enhances an image or the features we are interested in for segmentation to take place.

C2.1 Noise Reduction Filters

Only simple features have been implemented, however, many others exit based on these ideas that work on an iterative scheme (Lev 76). These algorithms are common to all the sub-plans and thus have no declared succeeding algorithms, the image processing algorithms are executed by the domain rules dependent on the quality (noise characteristics) of the image.

Noise Reduction Filter	Attributes
Mean Filter	Simple to implement, easy calculation working out the mean of the pixels (local average). Can Blur edges. (Gonzalez 84).
Mean Filter (Threshold)	Same as above but change only takes effect if T (threshold) > predetermined value. Stops burring but difficult to determine value for T at each step. (Pratt 78).
Median Filter	Calculate median at each step and set value. Requires partial sort (time consuming). Does not blur edges.(Huang 78, Lev 76).

Table C1. Noise reduction filters and their attributes.

C2.2 Histogram Transforms

Histogram Transform techniques use intensity mapping in an attempt to improve appearance as shown in the table below.

Feature Enhancement	Attributes
Histogram Equalisation	Gives a more equatable sharing of pixels, it is monotonic i.e., dark areas remain dark and light areas remain light. It does not alter the information in a scene, just changes grey level gradients. (Pavlidis 82).
Local Enhancement	Same technique as above is used, but used for each pixel in a window W. Enhances detail over a small area to give better quality. It is however computationally expensive, and may not produce the desired result. (Gonzalez 87).

Table C2. Feature Enhancement techniques by histogram equalisation.

C2.3 Edge Detection

Primitive edge detection tries to determine whether a component of a boundary in an image passes through or near a given pixel. This is done by examining the rate of change of intensity near the pixel. Sharp changes (steep gradient) are good evidence of an edge, slow changes will suggest the contrary. Figure C.1 shows some templates used for edge detection.

Sobel Edge Operators

$$\begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array} \qquad \begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline \end{array}$$

Vertical Horizontal

Prewitt Edge Operators

$$\begin{array}{|c|c|c|}\hline -2 & 0 & 2 \\ \hline -2 & 0 & 2 \\ \hline -2 & 0 & 2 \\ \hline \end{array} \qquad \begin{array}{|c|c|c|}\hline 2 & 2 & 2 \\ \hline 0 & 0 & 0 \\ \hline -2 & -2 & -2 \\ \hline \end{array}$$

Vertical Horizontal

Laplacian 1 Operator

$$\begin{array}{|c|c|c|}\hline 0 & -1 & 0 \\ \hline -1 & 4 & -1 \\ \hline 0 & -1 & 0 \\ \hline \end{array}$$

Laplacian 2 Operator

$$\begin{array}{|c|c|c|}\hline -1 & -1 & -1 \\ \hline -1 & 8 & -1 \\ \hline -1 & -1 & -1 \\ \hline \end{array}$$

Figure C.1 Different edge operators..

These operators use a different threshold values dependent on the features we wish to highlight. The table C.3 shows some advantages and disadvantages of such templates.

Edge Operator	Advantages and Disadvantages
Sobel edge templates	Different thresholds highlight different features. Gradient magnitude and gradient direction can be calculated. It is slow as two templates are used (Ballard 82). More accurate then Prewitt operators (Davis 84).
Prewitt edge templates	Same as above, but less efficient as it enhances the effects of noise. (Prewitt 70).
Laplacian	Zero-crossing (Marr 80) avoids thresholding and thinning, gives closed curves, does not give response where intensity is changing smoothly (Robinson 77). However, it responds strongly to noise and corners (Rosenfeld 82) and gives response on both sides of an edge.

Table C.3. Edge Operators.

The Laplacian templates are implemented in the Visual Planner, however, they have not been used in any sub-plan by the Visual Planner, as it can not achieve a given goal when using such templates this is because within the test domain of castings the Laplacian filter detected too many false edges. The Visual Planner has no control over the Laplacian as it does not use threshold values.

C2.4 Mathematical Morphology

After the Sobel or Prewitt edge templates have been applied the image is transformed into a number of white lines on a black background. These lines however, have spikes or troughs that brake the flow of the line. There are two operations known as erosion and dilation (Gonzalez 87). We can use these two methods one after the other to produce different effect on images. For instance erosion followed by dilation gives us an opening effect and thus damps down spikes in an image. Dilation followed by erosion gives us a closing effect and thus fill troughs on an edge.

C2.5 Thinning

After edge detection and opening and closing operations have been applied on an edge, thinning is then applied. Thinning is the process of reducing an edge to a one pixel wide connected frame. Thinned objects are called the skeleton of the body (Sherman 59). Thinning is used here so that boundary following may take place. This is not essential, but allows the boundary following algorithm to work more efficiently, by only considering one pixel rather then a neighbourhood of pixels.

C3 Segmentation Techniques

Segmentation is a form of pixel classification, i.e. we are assigning each pixel into a class of pixels based on some property of the pixel. Segmentation algorithms can be divided into two classes: i) based on pixel similarities, or ii) based on surface discontinuities. The former method involves regions and the latter edges. This section explains some segmentation techniques used within this project. It explains the thresholds used within these algorithms.

C3.1 Thresholding and Digitisation

Thresholding is probably the most simple of all segmentation techniques. It simply sets a region to black if it is below a certain threshold value, usually determined from the histogram of the image (Prewitt 66) or white if it is above a given threshold value. This produces a binary image. Binary images are useful, however, we do lose grey scale information. Several methods have been developed for automatically selecting threshold values, these have been reviewed in (Wezka 78, Rosenfeld 82).

Digitisation is based on the thresholding method, however, grey scale values are left in tact. Here the image is split according to two threshold values as shown below:

$$B_{ij} = 0 \text{ IF } G_{ij} < \mathrm{T}_{\mathrm{min}}$$
$$B_{ij} = G_{ij} \text{ IF } G_{ij} >= \mathrm{T}_{\mathrm{min}} \text{ AND } G_{ij} <= \mathrm{T}_{\mathrm{max}}$$
$$B_{ij} = 0 \text{ IF } G_{ij} > \mathrm{T}_{\mathrm{max}}$$

B_{ij} = New intensity value at a given pixel location (i,j).

G_{ij} = Current image intensity value at a given pixel location (i,j)

It can be seen from this method that we need knowledge of the object of interest to set the threshold values T_{min} and T_{max}. The Visual Planner uses this method as we have object characteristics, and for each object its maximum and minimum intensity values that can be used for T_{max} and T_{min} respectavly. If this algorithm is selected by the Visual Planner then the threshold values are extracted by the knowledge available on the algorithms. This knowledge is embedded within the algorithm i.e., what type of thresholds are needed from the object or image characteristics database.

C3.2 Region Segmentation

Region segmentation (Brice 70, Feldman 74) is used to split an image into connected clusters of pixels. Each pixel can only belong to one region, and all pixels have to belong to some region. Region segmentation achieves the same results as multiple thresholding, however knowledge of the objects is not needed to set the threshold values that will connect adjacent pixels. The advantages of using region based segmentation over edge detection and boundary following are:

1) there are fewer regions then pixels,
2) regions are connected and unique.

The disadvantages of using region segmentation are:

1) a region could be considered to be a single surface (under segmentation),
2) assumptions are made about the uniformity of image features when selecting threshold values i.e. the surface will always bee between T_{min} and T_{max},
3) surface properties or reflection could produce noise, i.e. increase or decrease its average surface intensity above or below a given threshold.

There are two types of region based segmentation, Region Growing and Region Splitting. In Region Growing each pixel is considered to be a separate region and adjacent regions are merged if they have similar properties such as grey level. This is done according to some predefined threshold value. Region Splitting (Robertson 73) is exactly the opposite to region growing. The image is regarded as being a single region, and each region is recusivly subdivided into subregions. This once again can be used with some predefined object or background threshold, or the bimodality of histograms may be used to split regions (Prewitt 70, Nazif 84).

C3.3 Hough Transform

The Hough Transform (Hough 62) aims to bypass the detection of boundaries and generate information about parameter features such as circles and straight lines directly from information about edge locations. The Hough transform works in transform or parameter space. The parameters used describe the features we are looking for. The Hough transform within the Visual Planner is used to locate circular objects, Suppose we wish to locate circles of radius R in an image, such a circle centred at (a,b) has equation

$$(x-a)^2+(y-b)^2 = R^2$$

parametrically

$$(a+R\cos\theta, b+R\sin\theta) \quad 0<=\theta<2\pi$$

Each edge pixel generated by some edge operator is then considered. The edge may be part of a circle, or part of another feature, or just noise. If however, it is part of a circle for which we are searching then we know that the centre of this circle must satisfy:

$$(a,b) = (x+R\cos\theta+R\sin\theta) \quad 0<=\theta<2\pi$$

The Hough transform is clearly computationally expensive, however, it is very good at finding circles or ellipses in the presence of noise, or if an object is partially occluded.

C3.4 Boundary Following

Boundary following is a region extraction method for finding a region outer boundary. The criteria used for defining region boundaries may be based on image properties such as pixel intensities, colour, or texture information (Dudani 76).

The boundary following algorithm used within the Visual Planner is that proposed by Dudani (Dudani 76). This algorithm has a number of desirable properties, such as producing the chain code of a boundary (see Chain Code below) which is used later for corner detection (see corner detection below), and area information.

The boundary following algorithm starts by scanning an edge detected image from left to right and top to bottom. Once an edge pixel is found it is used as the initial boundary point. The boundary point is then flagged as used, so it is not considered again. Each of the current pixels eight neighbours (0-7) are then searched to find the next boundary point, as shown in Figure C2.

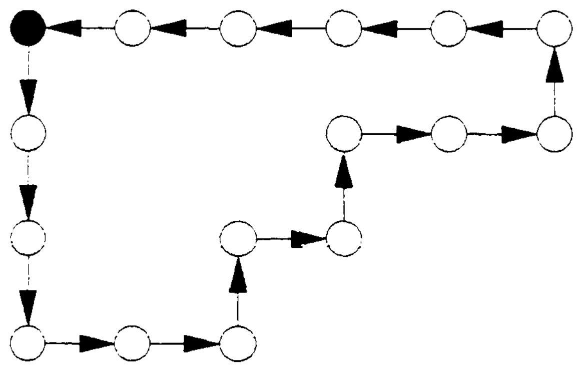
0 (i-1,j-1)	7 (i,j-1)	6 (i+1,j-1)
1 (i-1,j)	(i,j)	5 (i+1,j)
2 (i-1,j+1)	3 (i,j+1)	4 (i+1,j+1)

Figure C2 Order of search for Boundary Following.

The next boundary point found then becomes the current boundary point and is flagged as used. The whole process is repeated until no more joining boundary points exist. Each boundary point is saved in an array of points.

C3.5 Chain Code

Chain codes are used to represent a boundary by a connected sequence of straight line segments of specified length and direction (Gonzalez 87). The boundary following algorithm described above gives us the direction of the boundary, if we then count the number of adjacent points that are in the same direction we have the chain code of the object. An example is shown in Figure C3.



Chain Code: 3 3 3 5 5 7 5 7 5 5 7 1 1 1 1 1

Figure C3 Formation of Chain Code.

C3.6 Corner Detection

The corner detector used here is a heuristic detection method based on the chain code (Panayiotou 93) of an object. Initially we start by considering the first and last points in the chain code to be corners. If the object is closed then the first and last points will be the same. A corner is selected within the chain code if the direction of the succeeding values are in the same direction and their gradient magnitude is greater then some predefined threshold. The threshold selected depends on the average size of all the objects in the objects characteristics database. if the objects are small in terms of pixel values, then the threshold selected will be small, typically two to three pixels, if the objects are large and well defined, then the threshold values are increased, typically 20-30 pixels. Such a corner detector needs knowledge of object sizes. This is particularly suited for the Visual Planner as such information is readily available. Other algorithms such as the Plessy Algorithm (Noble 88) could have been used, as this works on chain codes, however, this algorithm is computationally expensive.

C4 Robust Segmentation Algorithms

A number of algorithms based on the work carried out by Nazif (Nazif 84) are also implemented within this prototype and have been extended by Panayiotou (Panayiotou 93). This was carried out as the segmentation algorithms presented above may not always work to their full potential. e.g. Due to some domain change in lighting we ave lost some line information. Even though Domain Rules are used to detect and correct such changes, information may have been irretrievably lost. A set of rules (Nazif 84) are used after different algorithms are selected which will join or delete lines if an edge detection algorithm has been selected, or refine regions by splitting or merging them if a region based segmentation technique is used.

C5 Algorithm Knowledge

This section describes which algorithm succeeds the current algorithm, along with their contextual information and parameters passed to them used for threshold values if applicable. The table below shows what knowledge is kept on each algorithm within the Visual Planner.

Algorithm	**Succeeding Algorithm List**	**Contextual Information**	**Threshold Parameters**
Thresholding Priority=1	Edge Detection Algorithms Region growing Region splitting	Linear Circular Geometric Blob	T_{min}, T_{min} = Min & Max intensity values of object to be segmented
Edge Detection Algorithms (Sobel, Prewitt and Laplacian) Priority=1	Mathematical Morphology Thinning Robust line rules Hough Transform	Linear Circular Geometric Blob	T = Threshold value depending on object intensity.
Thinning Priority=2	Mathematical Morphology Robust line rules Boundary following	Linear Geometric Blob	
Mathematical Morphology Priority=2	Robust line rules Boundary following Thinning	Linear Geometric Blob	
Hough Transform Priority=2		Circular	x,y radius = major axis/2 and minor axis/2
Robust Line Rules Priority=2	Boundary following Mathematical morphology Thinning	Linear Geometric Blob	
Boundary Following Priority=3	Chain code	Linear Geometric Blob	
Chain Code Priority=4	Corner point detector	Geometric	
Corner Point Detector Priority=5		Geometric	
Region growing Priority=1	Robust region rules	Blob	T_{max}, T_{min} = object max and min Intensity values.
Region splitting Priority=1	Robust region rules	Blob	T_{max}, T_{min} = object max and min Intensity values.
Robust Region Rules Priority=2		.Blob	